UMRR

Interface Documentation

Contents

1 Topology 4
1.1 Stationary Traffic Monitoring sensor system 5
1.2 Portable Traffic Monitoring sensor system 6
2 Radar System Components 8
2.1 Individual sensors 8
2.1.1 Sensor Hardware Identification 9
2.1.2 Sensor Software Identification 9
2.2 Sub Network controller (Central ECU) 10
2.2.1 Central ECU Hardware Identification 11
2.2.2 Central ECU Software Identification 11
3 Frequency Approval 12
4 Antennae 13
4.1 Type 26 14
5 Cables And Connectors 15
5.1 Sensor Connector 15
5.2 Central ECU Connectors 17
6 Mechanical Interface 18
6.1 Single Sensor Housing Version 5 18
6.2 Single Sensor Housing Version 5.5 19
6.3 Coverage of the sensor 20
6.4 Central Processor 21
7 Co-ordinate System 22
7.1 Object representation 23
7.2 Line Application (applicable for e.g. Elk warning project) 24
7.2.1 Horizontal line 24
7.2.2 Vertical line 25
7.3 Multi Bumper usage 26
8 CAN cable installation 27
9 Communication scheme 28
9.1 Internal CAN Bus Interface Specification 28
9.2 External CAN Bus Interface Specification 28
9.3 External Ethernet Interface Specification 28
10 Internal CAN Bus Interface Specification 29
10.1 General Interpretation of the CAN Data 29
10.2 Introduction 30
10.3 CAN-Settings 30
10.4 Specification of the sensor input data 30

10.4.1 General Description 30
10.4.2 Methods to send a command 32
10.4.3 Basic Commands 32
10.4.4 Special Commands 33
10.4.5 Software Download 33

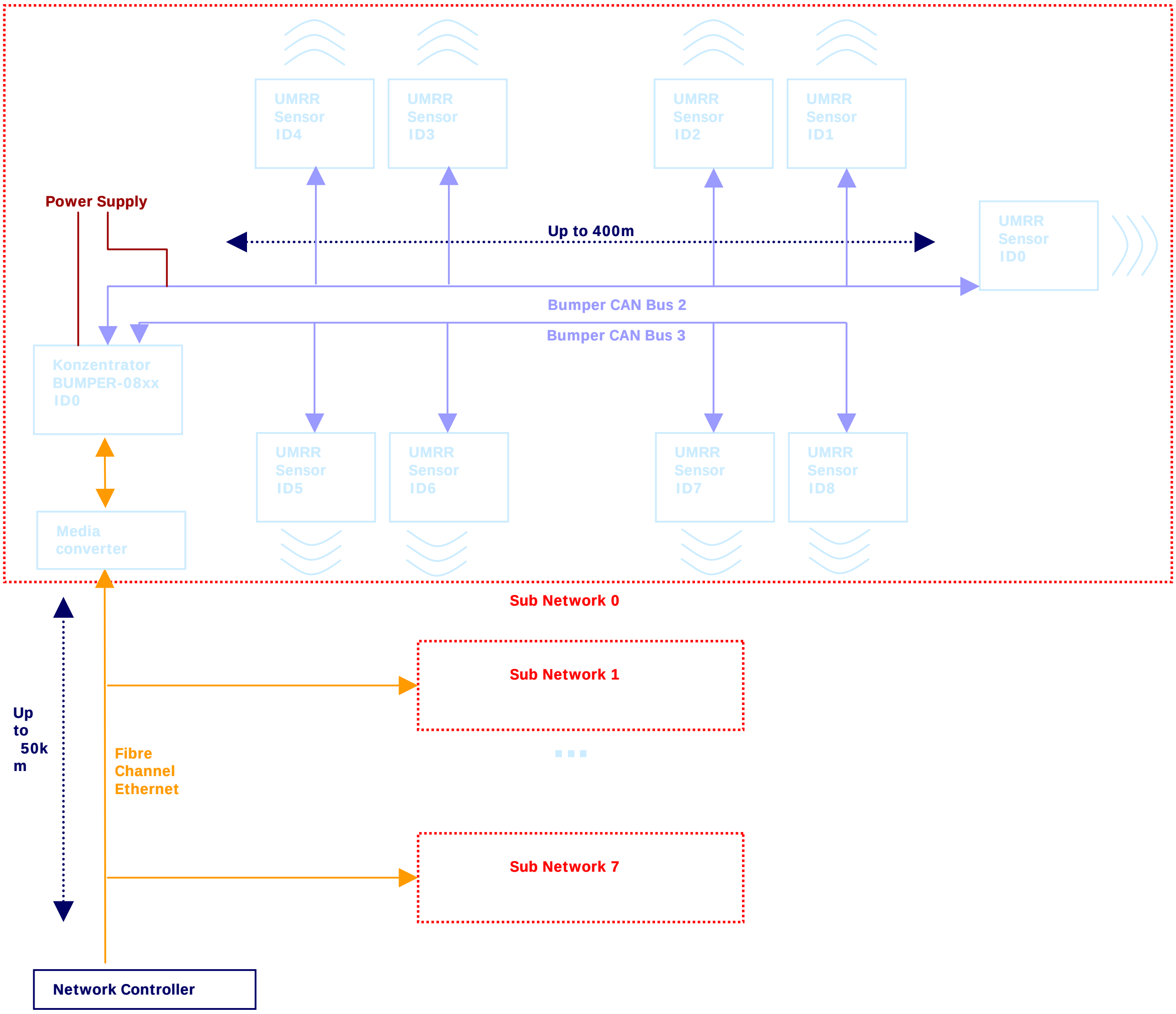
11 External CAN Bus Interface Specification 35

11.1 Specification of the output data 36
11.1.1 Status messages 36
11.1.2 Object data messages 38
11.2 Specification of the input data 40
11.2.1 Command message 40
Basic Commands 41
11.2.2 Command_Read message 42

12 External Ethernet Interface Specification 46

12.1 Introduction 46
12.2 Communication basics 46
12.3 Output data specification 47
12.3.1 Training sequence 47
12.3.2 Header 47
12.3.3 Encapsulated transmit data 48
12.3.4 End of List 48
12.4 Input data specification 49
12.4.1 Training sequence 49
12.4.2 Encapsulated receive data 50
12.4.3 End of List 51
12.5 Receive data specification 52
12.6 Receive data actions 53
12.7 Non-volatile parameters 54
12.7.1 Bumper software update considerations 55

1 Topology



Network topology

- Alarm Management is located in the "Network Controller"
- Each sub network is identified by an IP address
- Each Camera is identified by an IP address
- Each sub network is capable to cover up to 900m (determined by CAN cable length)
- Each sub network is controlled by a "sub network controller"
    - Ethernet 100Mbit/s interface
    - Connection oriented operation

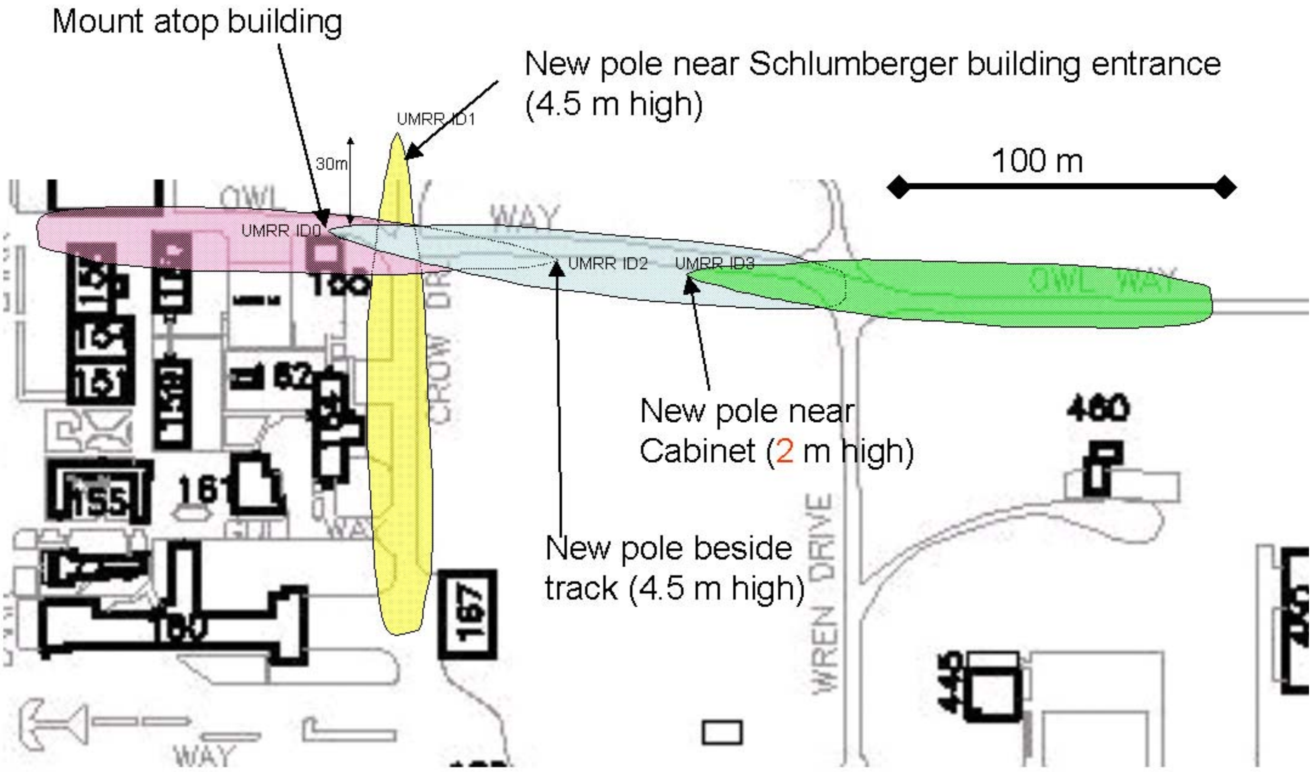
1.1 Stationary Traffic Monitoring sensor system

The sensor system consists of 4 UMRR sensors (ID0 .. ID3) and one tracking controller.
The sensors are connected to the Bumper as follows:

- ID0, ID1 on Bumper CAN2
- ID2, ID3 on Bumper CAN3

(These settings can be modified – if necessary – through Bumper firmware update).

The following image shows the sensor layout for the stationary Traffic Monitoring system:



Assuming Type 26 antenna, nominal 150 m range for automobile target

Figure 1: Sensor layout for stationary system

1.2 Portable Traffic Monitoring sensor system

The sensor system consists of 3 UMRR sensors (ID0 .. ID2) and one tracking controller.
The sensors are connected to the Bumper as follows:

- ID0, ID1 on Bumper CAN2
- ID2 on Bumper CAN3

(These settings can be modified – if necessary – through Bumper firmware update).

The following picture shows the sensor layout for the portable Traffic monitoring system.

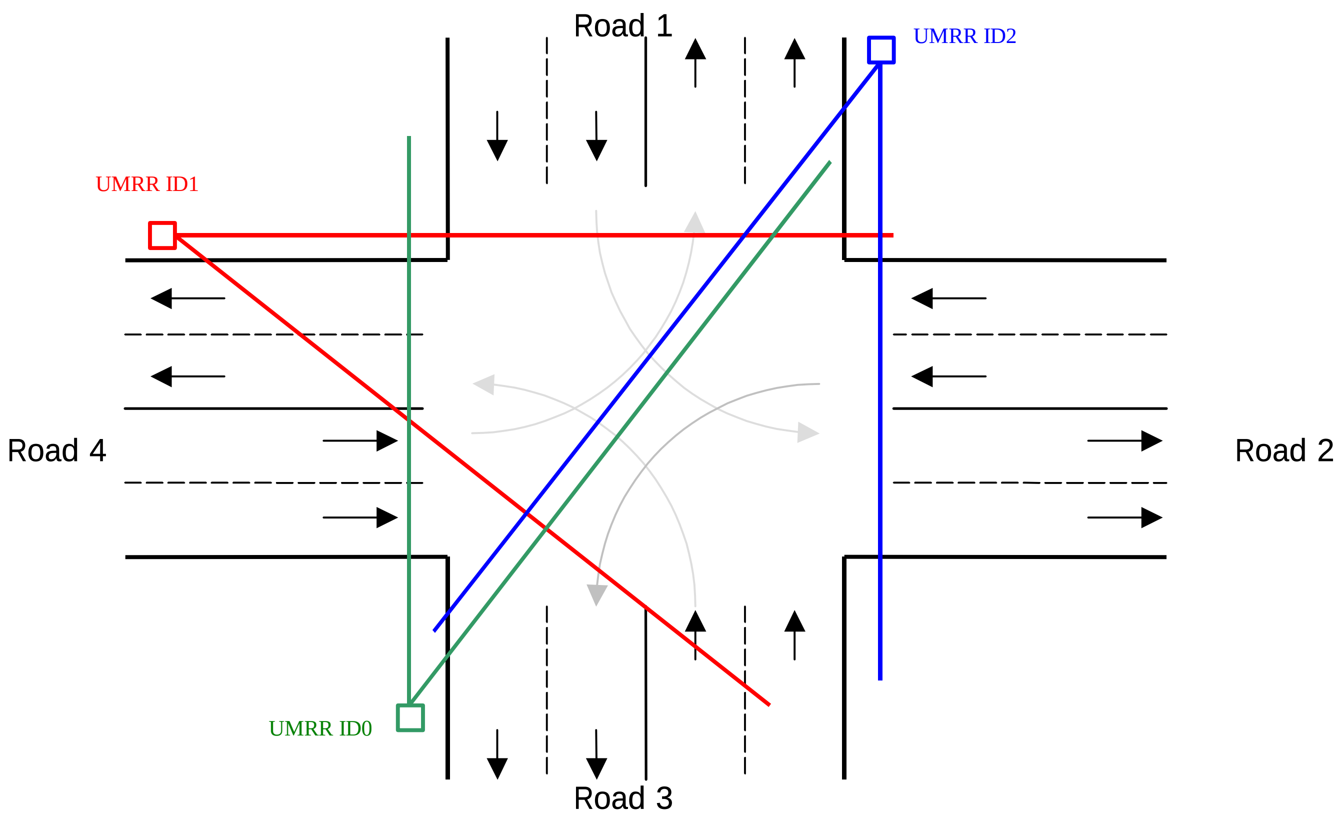


Figure 2: Sensor layout for portable system

In this configuration, oncoming traffic from road 3 is monitored for left turners, with vision on oncoming traffic from road 1.

Therefore, the sensors should always be placed in the portable system as follows:

- ID2 monitors the road where left turners are expected
- ID0 monitors the road where left turners depart
- ID1monitors the road where potentially dangerous traffic for the left turners is expected.

If another road is to be monitored for left turners, the above depicted sensor layout has to be angled by 90 / 180/ 270 degrees.

2 Radar System Components

2.1 Individual sensors

The main task of the sensor component is the **detection of any obstacles** in the field of view. Range, relative radial speed and angle of each object are measured. Its interface to the central processor is the object list reported cycle by cycle.

Figure 3: Photograph of the UMRR Sensor

Each sensor comprises a CAN port for communication and an SPI port for data logging.

2.1.1 Sensor Hardware Identification

The individual sensors are referred to as

UMRR-P-xxdd-yy-zz
-P Platform Sensor
-xxdd (DSP Generation xx, Derivative dd)
-yy (Microwave Module Generation yy)
-zz (Antenna Type zz)

Example:

UMRR-P-0708-14-32

where UMRR means the Universal Medium Range Radar platform developed by s.m.s GmbH.

2.1.2 Sensor Software Identification

The sensor software is given as follows:

Example:

Umrrflsh_ID0_RF1208_DSP0436_Rel005_2004-02-18_P_ACCSG_OUV

Umrrflsh	Software for Flash or RAM Download
ID0	Sensor ID in the network
RF1108	RF Module Serial Number
DSP0366	DSP Module Serial Number
Rel005	Software Release
P	Platform Software
ACCSG	Parameter Settings for ACC Stop&Go Application
OUV	Software for Upside Down Mounted Sensor

2.2 Sub Network controller (Central ECU)

The sensor system may consists of a number of individual sensors and a central processing unit. The latter controls the whole system, interprets the sensor data and communicates with the vehicle and the ECU (Electronic Control Unit).

Figure 4: Photograph of the Central ECU

The algorithms that run on the central processor may be moved to existing ECU(s) of any vehicle system (for example an other existing sensor system). The central processor unit will then become obsolete.

In the central processor, the tasks

- **Tracking**
- **Situation Analysis**
- **Decision Algorithms**
- **Synchronization**
- **Diagnose**
- **Communication**

are performed. It interfaces at its input all individual sensors, and at the output there are a number of possible sets of data available:

- a list of the reliably tracked objects and their parameters (mainly position and relative speed) in the overall field of view (i.e. the superposed fields of view of the individual sensors).
- customer specific output data.

2.2.1 Central ECU Hardware Identification

The central processor has the identification

BUMPER-0501 Serial #05.22
where again **05** represents the device generation and **01** the derivative version. A Serial number of the DSP is given.

2.2.2 Central ECU Software Identification

The central ECU software identification is given as follows:

Example:

bumperflsh_ID0_Rel122 _2004-02-18_P _ EXPRO

bumperflsh	Software for Flash or RAM Download
ID0	Sensor ID in the network
Rel122	Software Release
P	Platform Software
EXPRO	Abr. For Expro-File

3 Frequency Approval

Prototype sensors have a test license regarding frequency allocation.

Production-type sensors operate according to

European Standards:
- ERC Recommendation 70-03
- ETSI EN 300-440

US Standards:
- FCC part 15.209
- FCC part 15.245

Canadian Standards:
- RSS-210

Japan:
- Radiolocation
- Specified Low Power Radio Station

CETECOM

ICT Services GmbH

Confirmation of Test Results

Customer: Auftraggeber:	SMS GmbH Mittelweg 7 38106 Braunschweig Germany
Testing laboratory: Testlabor:	CETECOM ICT Services GmbH Laboratory: RSC - RLS Untertürkheimer Str. 6-10 D-66117 Saarbrücken
Object identification: Prüfgegenstand:	UMRR
Object description: Objektbeschreibung:	Speed and Distance Radar for Railway and Automotive application. 24.125 GHz
Test Report No.: Prüfberichtsnummer:	2–3072–01–01/02
Specifications: Prüfvorschrift(en):	ETSI EN 300 440 –1 V1.3.1 ETSI EN 300 440 –2 V1.1.1 ERC Recommendation 70–03
Statement: Ergebnis:	**No deviation in the frequency range > 1 GHz from technical specification(s)/Standard(s) have been ascertained in the course of the performed tests.** **Es wurden im Verlauf der durchgeführten Prüfung keine Abweichungen > 1 GHz von den, durch die genannte(n) Prüfvorschrift(en)/ Standard(s) vorgegebenen Anforderungen beobachtet.**

Saarbrücken, 06.11.2002

Place, date

(Manager of Laboratory)

accredited by Reg TP/DATEch/DEKITZ under accreditation Certificate TTI-P-G 166/98

CETECOM ICT Services GmbH Untertürkheimer Str. 6 - 10 D-66117 Saarbrücken

Figure 5: Frequency Approval Document.

See more: Frequency Approval.

4 Antennae

On the following pages, ISO-SNR diagrams are displayed that illustrate the field of view of different types of antennae.

ISO-SNR means that at a given fixed reflector size, the signal-to-noise ratio (SNR) is plotted over x and y. Hence, the shape of the radar beam including possible side lobes can be explained.

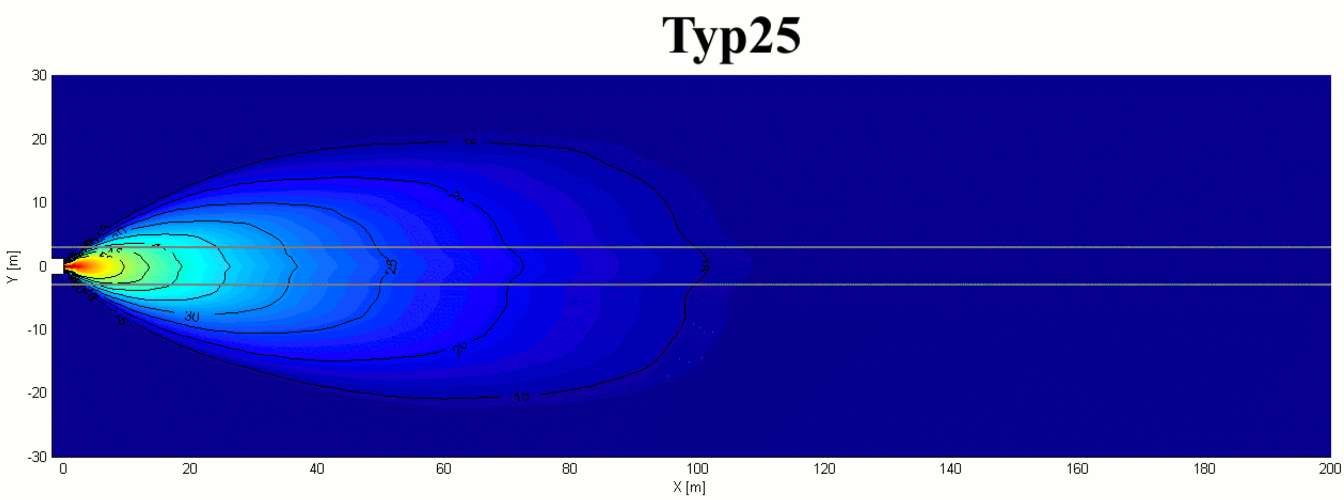
The numbers which are written as a legend into the diagrams, and which also determine the colors, specify the SNR for a fixed reflector size.

A translation in detection range can roughly be given as:

SNR Number [dB]	Corresponds to max detection range of:
15	Truck
20	Passenger car
25	Motorbike
30	Bicycle
35	Human

Those numbers are estimations for illustration only. In practice, the detection range also depends on the clutter level, presence of other reflectors etc.

The following animated graphic shows a comparison of the types 24-25-26.



4.1 Type 26

This setup is commonly used for long range applications like ACC or similar rear-looking functions (new version, follow-up model for type 14).

Parameter	Value
Type	26
Operational Mode(s)	FMSK
Maximum Range (Truck)	240m
Maximum Range (Car)	160m
Max. Range (Pedestrian)	60m
Azimuth 3dB Limits	+-6 degree
Elevation 3dB Limits	+-4 degree
Max. Az. Field of View	+-20 degree
Antenna Type	Patch Antenna
Housing Type	5.5

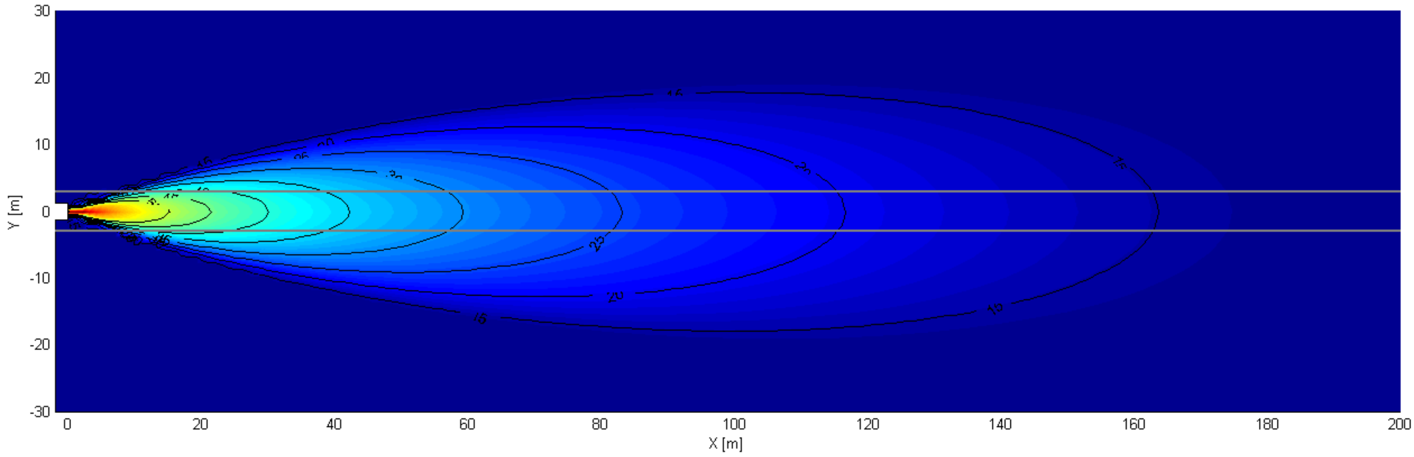


Figure 6: Type 26 antenna single sensor setup.

5 Cables And Connectors

The set of cables and connectors comes prepared with the sensor system to avoid damage due to pinout mismatches.

5.1 Sensor Connector

Mounted on the back side of the radar there is a 8 pin male circular connector (waterproof IP67 series 702, manufacturer Binder GmbH, Germany). You must connect a female counterpart (see drawing):

Figure 7: Sensor Connector

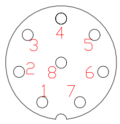


Figure 8: Female counterpart of sensor connector (rear view)

Sensor Version UMRR8:

Pin	UMRR-yyxx	Color (New Cables)
1	Internal	
2	GND	Blue = GND
3	Internal	
4	CAN_L	Yellow = CAN_L
5	CAN_H	Green = CAN_H
6	Internal	
7	+9V...+32V	Red = +9V...+32V
8	Internal	

Table 1: Sensor Connector Pinout

5.2 Central ECU Connectors

Mounted to the central ECU are four male D-SUB 9 connectors (prepared for waterproof connection) for the **CAN buses**.

See also Figure 12.

An appropriate female D-SUB 9 counterpart must be connected
CAN:

Pin	D-SUB9
1	n.c.
2	CAN_L
3	GND
4-6	n.c.
7	CAN_H
8	n.c.

Table 2: Central ECU CAN Connector Pinout

Mounted to the central ECU is also one female D-SUB 9 connector (prepared for waterproof connection) for the power supply.

An appropriate male D-SUB 9 counterpart must be connected to the Power:

Pin	D-SUB9
1-2	+12V
3	n.c.
4-5	GND
6	+12V
7-8	n.c.
9	GND

Table 3: Central ECU Power Connector Pinout

For the CAN bus handled here (internal CAN sensors == CAN2 of central ECU), one 120Ω resistor is integrated at the sensor end of the cable.

One additional 120Ω resistor must be integrated on the other end of the cable, i.e. at the central ECU connector, CAN2.

Ethernet interface: The Ethernet option hosts one RJ-45 female connector. You must connect a RJ-45 male counterpart.

6 Mechanical Interface

6.1 Single Sensor Housing Version 5

The overall size is (cable stump not included):

Sensor Housing:	**WxHxD: 105mm x 94mm x 34mm**
Weight:	**335g with aluminium body**
	270g with magnesium body

Figure 9: Photographs of Housing V5

The picture above shows the housing.

In the normal mounting position, the type sticker (containing model identification, serial number, date) is fixed on top of the sensor.

When looking at the front of the sensor (radome), the cable outlet is on the right side. It may be required that the sensor is mounted Bottom Up. In that case the cable outlet is on the left side.

6.2 Single Sensor Housing Version 5.5

The overall size is (connector not included):

Sensor Housing:	**WxHxD: 108mm x 97mm x 37mm**
Weight:	**410g (Al body)**

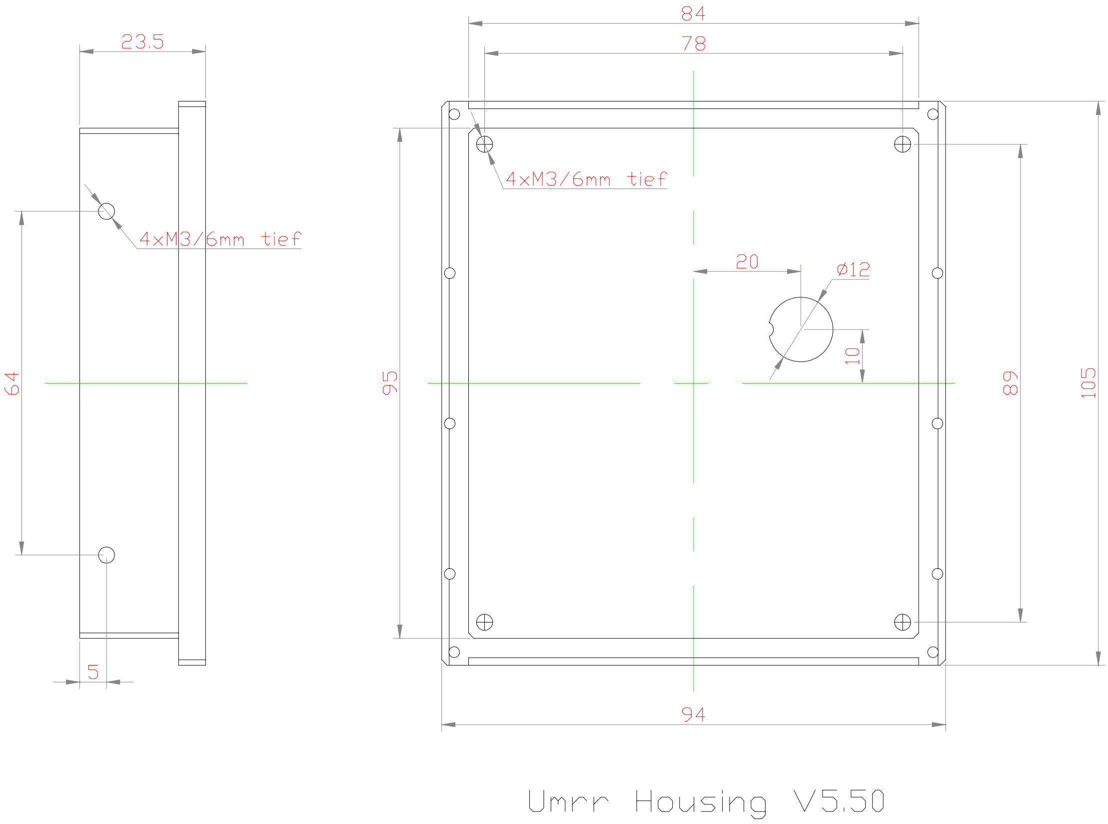


Figure 10: 2D Drawing of Housing V5.5

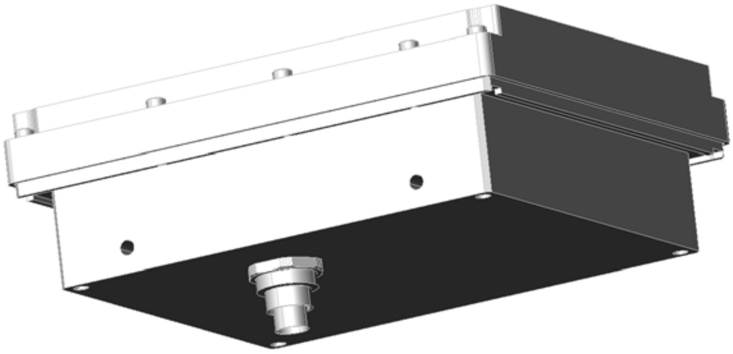


Figure 11: 3D Drawing of Housing V5.5

6.3 Coverage of the sensor

The sensors have a weather proof plastic radome, which is optimized along with the antenna. It is not necessary to protect the sensors by any additional means.

Hence, for an optimum detection performance, do not mount the radar behind bumper material or similar covers. The effective range will then be reduced depending on the transmission attenuation of the material.

If, however it is required to hide the sensors for design or other reasons, the may be mounted behind other non-conductive materials.

6.4 Central Processor

The overall size is (connectors not included):

WxHxD:	**17,3cm x 7,8cm x 5,7cm**
Weight:	**approx. 700g**

Figure 12: Photograph of the central ECU

The Ethernet interface is not included in this housing.

7 Co-ordinate System

Figure 13 illustrates the co-ordinate system of a multi sensor system using one bumper.
- the data are reported in Cartesian co-ordinates.

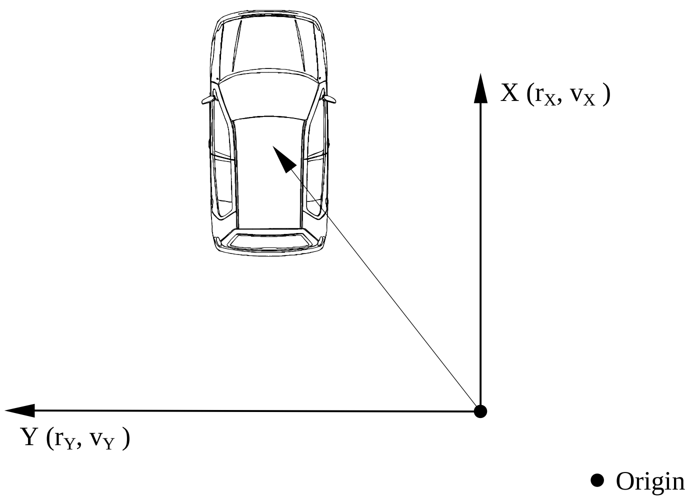


Figure 13: Drawing of co-ordinate system

Variables:

r_X – range in X-direction
r_Y – range in Y-direction
v_X – speed in X-direction
v_Y – speed in Y-direction

7.1 Object representation

Tracked object output is either **tracks** or **objects**.

Tracks are tracked (i.e. stabilized through tracking algorithms) reflection points with a Cartesian velocity vector. When a subject enters the vicinity of the sensor system, at first a track is built.
The track model is based on evanescent dimensions.

Objects are tracks with a width and length estimate and heading information. Objects are merged tracks with (almost) identical movement data. The length estimate is permanently adapted through raw radar target reflections.
The object model is based on dimensions from the width and length estimate. Therefore, a subject vehicle is abstracted with the shape of a rectangle, oriented with heading information. The heading can be extracted from the velocity vector.
When an object is built, the default length / width is 3.5m / 1.5m.

Object coordinates are reported from the center of the object.

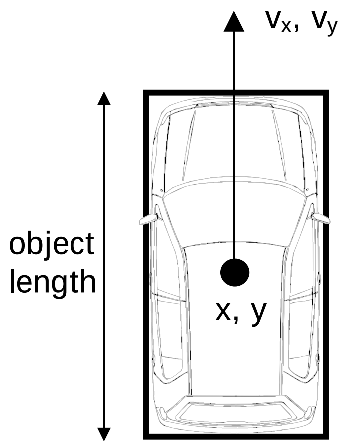


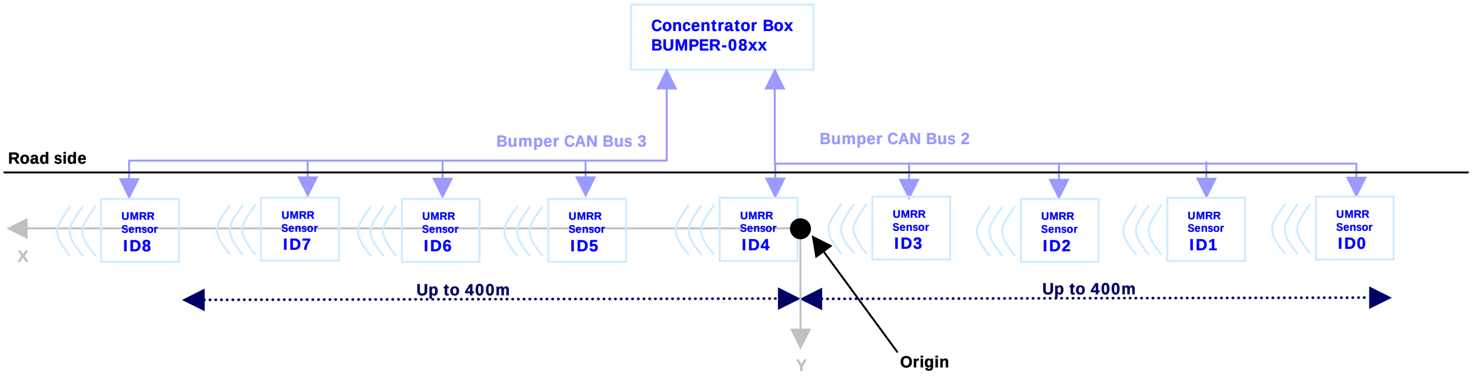
Figure 14: Object coordinate origin

7.2 Line Application (applicable for e.g. Elk warning project)

This section is not meant for the intersection project.

In a line application, the Bumper position is the origin. If the Bumper is remote, the position where the CAN bus(ses) enter the field are the origin.

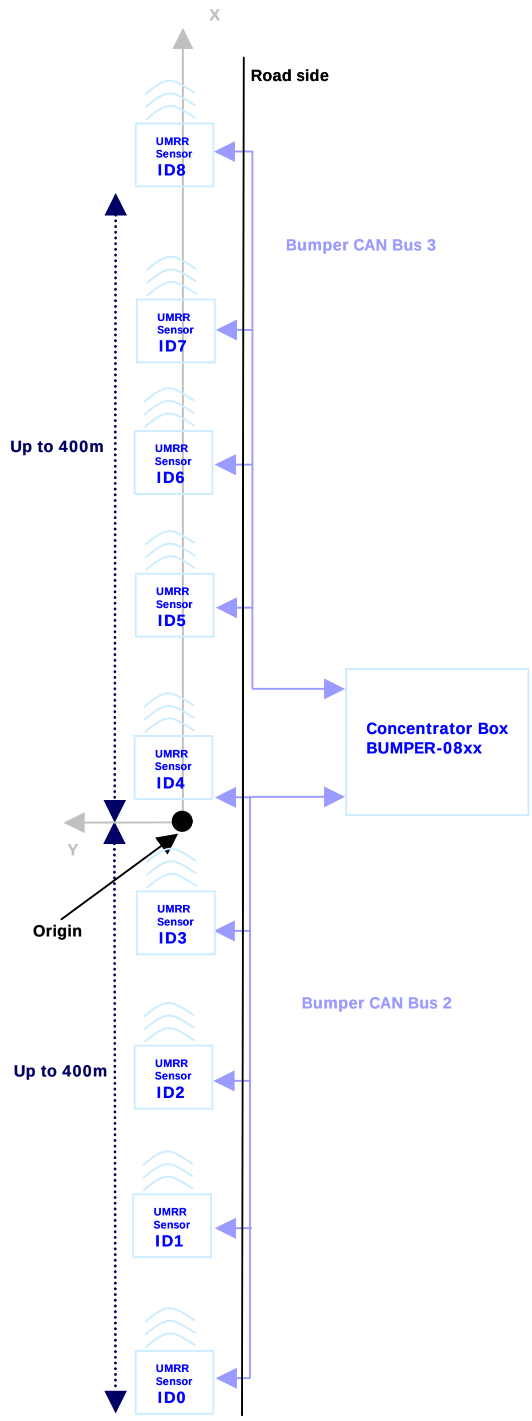
7.2.1 Horizontal line



Orientation:

- 0 degrees as default value for the straight line
- each sensor may be oriented by **+5 degrees** away from the road side. (Positive value means left turn).

7.2.2 Vertical line



Orientation:

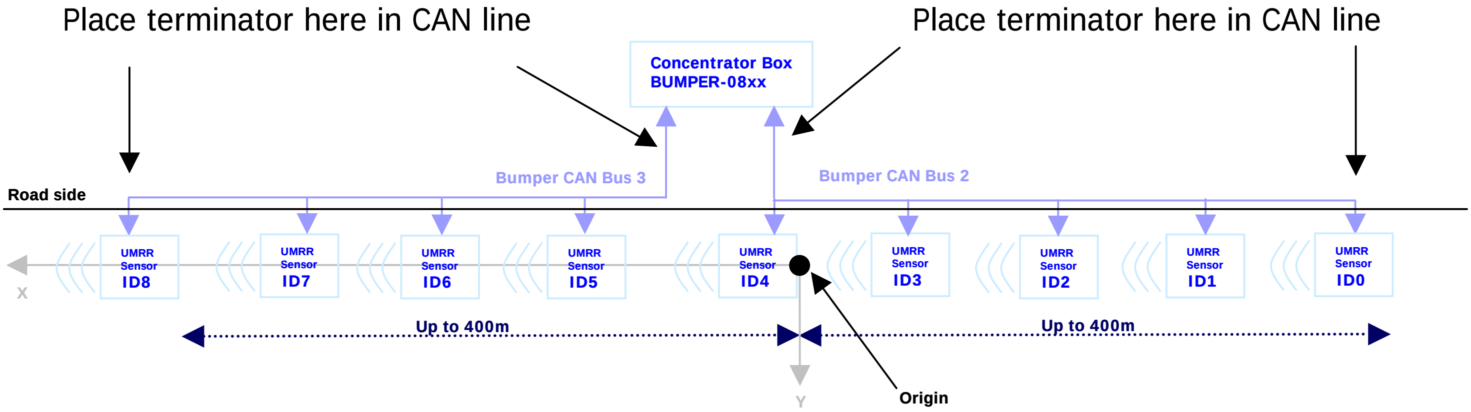
- -90 degrees as default value for the straight line
- each sensor may be oriented by +5 degrees away from the road side, resulting in **–85 degrees** for the vertical line. (Positive value means left turn).

7.3 Multi Bumper usage

If more than one bumper is used, each bumper has its individual origin.

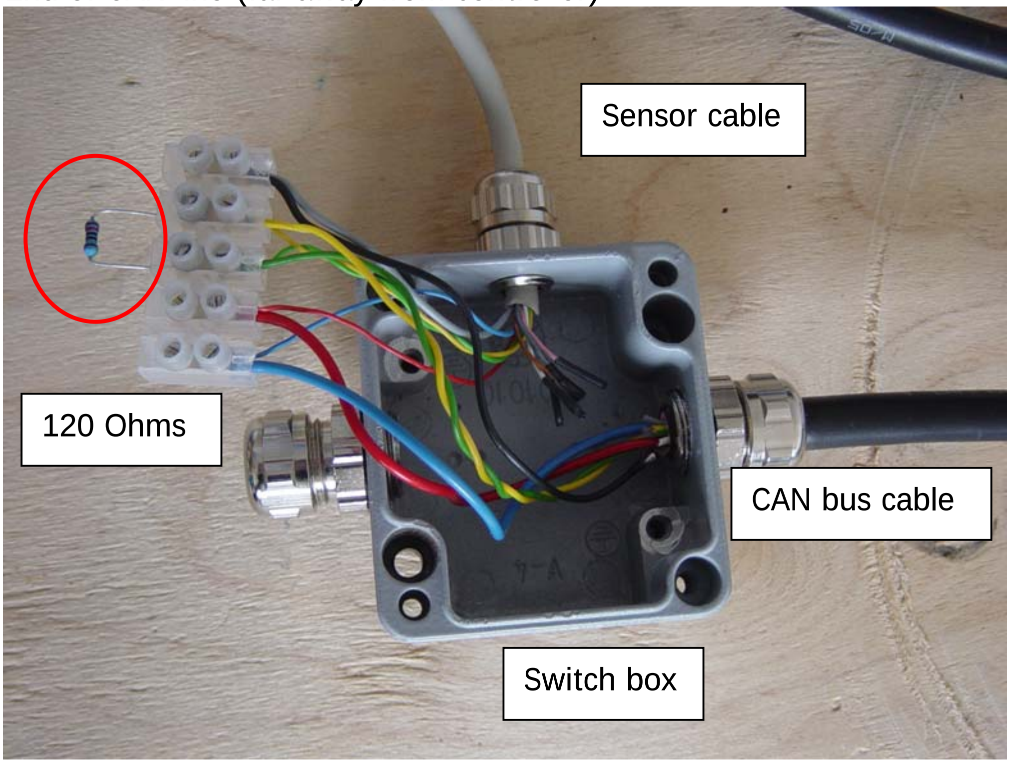
8 CAN cable installation

If installations cover several hundred meters, a **120 Ohms terminator resistor** is required at each end of the CAN cable.



The following image shows how and where the resistor is inserted:

1) End of CAN line (far away from controller)



9 Communication scheme

The following chapters describe system communication for a user.

9.1 Internal CAN Bus Interface Specification

That chapter describes how to address the radars for a software update or sending commands. In general, no user interaction is required.

9.2 External CAN Bus Interface Specification

That chapter describes how to address the sub network controller for a software update, how to send commands and how to retrieve object data from the tracking output. In general, no user interaction is required.

CAN communication is disabled by default, but may be activated by sending a command.

9.3 External Ethernet Interface Specification

That chapter describes how to address the sub network controller for a software update, how to send commands and how to retrieve object data from the tracking output.

Data message composition is identical to the External CAN Bus Specification. The Ethernet module encapsulates CAN messages in Ethernet packets, adding sequence control and training sequences.

10 Internal CAN Bus Interface Specification

10.1 General Interpretation of the CAN Data

A group of bits are described by
- First Bit and last Bit Number inside a message
- offset,
- resolution.

The bit group must be interpreted with the formula:

Value = (group of bits - Offset) * Resolution.

The offset must be subtracted first, then multiply the result by the resolution.

For example: In the CANalyzer software, you need to enter the following values:

Bits:	0...13
format	unsigned intel
offset:	–131.072m (negative value for the offset!)
minimum:	-131.072m
maximum:	131.056m
factor:	0.016m

Example:

Identifier:	**XXX**
DLC:	**8**
Format:	**Intel**
Type:	**unsigned**

Datenbyte0	Datenbyte1	Datenbyte2	Datenbyte3	Datenbyte4	Datenbyte5	Datenbyte6	Datenbyte7
0..13Bit X- Coordinate range Resolution: 16mm Offset: 8192 Interval: -131.072m...131.056m							

CAN message

Bit 63 0
Hex 00 00 00 00 00 00 00 E4
E4h = 228d
Value = (228 - 8192) * 0.016
Value = -127.424m

10.2 Introduction

This specification gives a detailed description of the CAN data communication used in the UMRR-xxyy based systems on the sensor CAN.

If a sub network controller is involved, the data exchange is
- **for UMRR ID0 to ID4:** between the sensors and the central ECU's CAN port 2 (CAN2).
- **for UMRR ID5 to ID8:** between the sensors and the central ECU's CAN port 3 (CAN3).

A system is assumed to consist of a set of radars (sensors), the signal processing part two (for example tracking) for this set is done in a central ECU who receives the described message scheme.

10.3 CAN-Settings

Baud Rate:	125kBit/s
T_{seg1}:	8
T_{seg2}:	7
T_{sjw}:	2

10.4 Specification of the sensor input data

10.4.1 General Description

The command message is used to transmit new parameters or initiate various actions in the sensors. It is sent broadcast but the destination sensor is the only one to execute the command (if the destination ID is set to 255 = broadcast, all sensors will execute the command).

Using the command messages, the radar may flexibly be switched in its detection behavior to adapt to different situations. It makes the radar a very powerful and flexible instrument.

Command messages may by initiated by test tools and / or the central ECU and are transmitted to the sensors.

Message Specification command:

Message Type: **command**
Identifier: **0x3F2**
DLC: **8**
Source: **central ECU / test tools**
Destination: **UMRR sensor**
Format: **Intel**
Type: **unsigned**

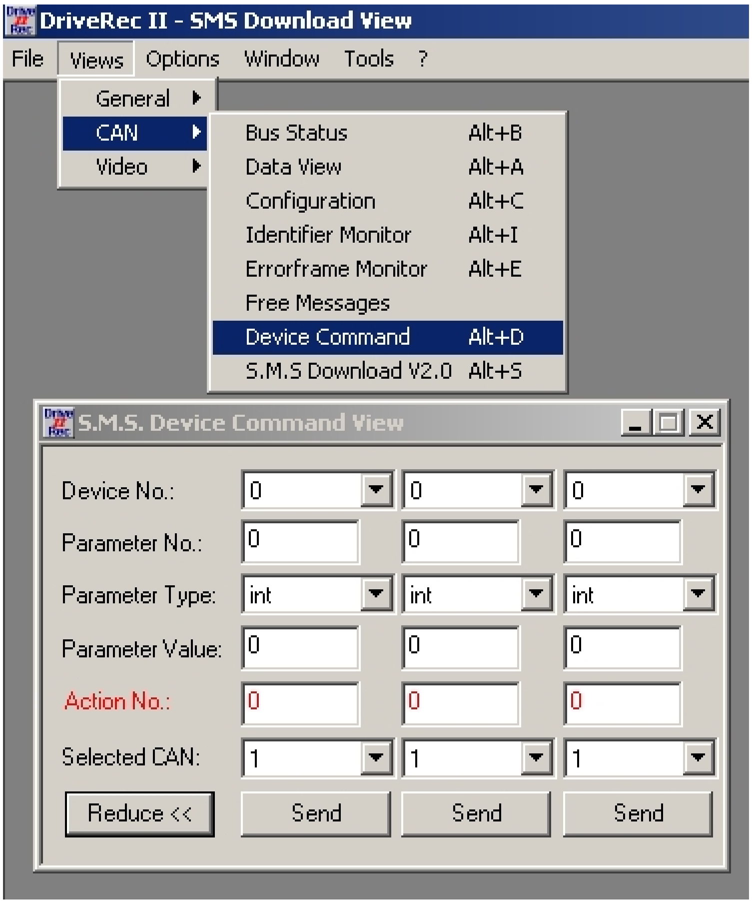
Data Byte 7	Data Byte 6	Data Byte 5	Data Byte 4	Data Byte 3	Data Byte 2	Data Byte 1	Data Byte 0
Parameter data byte 3 Hi-Byte	parameter data byte 2	parameter data byte 1	parameter data byte 0 Lo-Byte	Action	Parameter type	Parameter number	Destination device ID

Destination Device ID	Meaning
0..15	Sensor 0..15
255	Broadcast Command

Parameter Type	Meaning
0	Integer
1	Float (do not use)

10.4.2 Methods to send a command

Use the DriveRecorder2 software (http://www.smartmicro.de/html/drive_recorder.html) to comfortably transmit command messages to any sensor.



Alternatively, use any CAN based test tool and synthesize the command message as described.

10.4.3 Basic Commands

Name	Description	Number	Action	Default Value	Typical values
Interface mode		0	0	0	0 = normal output 1 = simulate targets
Sync mode		3	0	0	0 = free running 1 = synchronized operation 1 2 = synchronized operation 2
Nofsimobjects	Number of simulated targets in CAN target simulation mode	6	0	1	0...31
Max targets	Maximum number of targets to be transmitted	7	0	31	0...31
Can angle offset	A mechanical or electronical angle offset may be compensated [0.01deg]	11	0	0	-9000 ... 9000

10.4.4 Special Commands

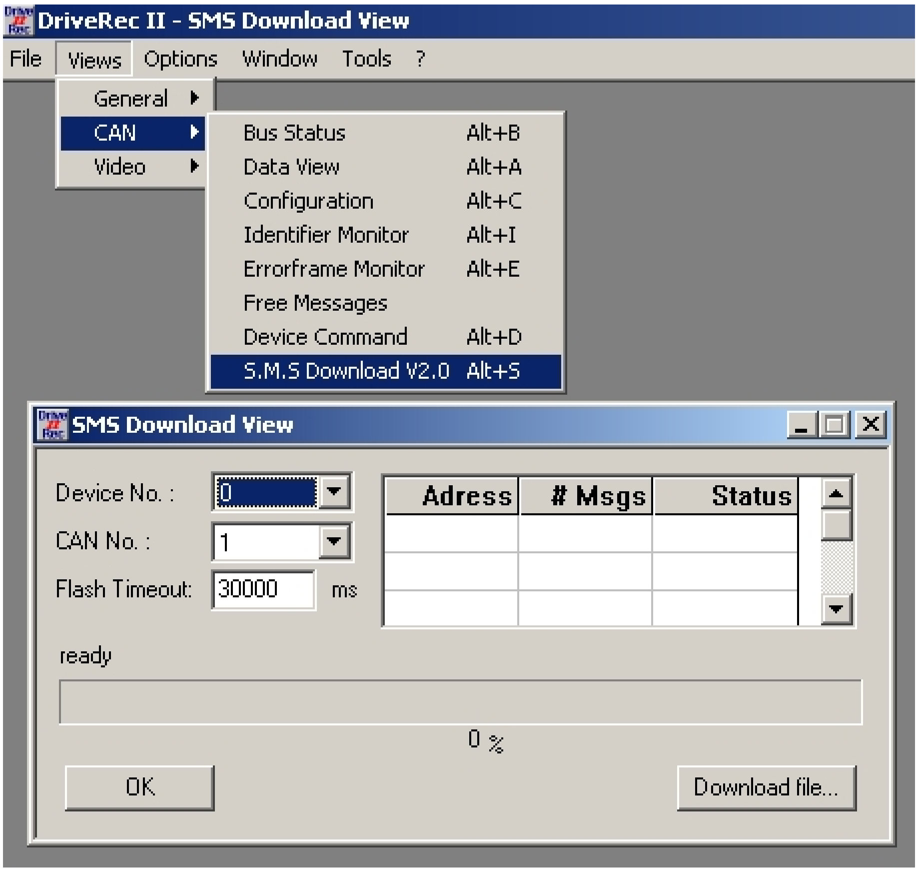
Description of the “Action” data field.

Common Actions:

Action	Meaning	Parameter
0	No Action	See above
1	RESET	Don’t care

10.4.5 Software Download

Use the DriveRecorder2 software (http://www.smartmicro.de/html/drive_recorder.html) to comfortably download software to any sensor.



The following message is used to download new software to the sensors. It contains pure data to be stored in the flash memory. The download data must be preceded by a command indicating the data stream. The addressed sensor then enters a data reception mode and stores the data in its flash memory. All other sensors remain silent. After the data transfer is finished, a new command is transmitted to the sensors, ordering them to continue normal operation. A software download is always initiated by the DriveRecorder2 and sent either to the central ECU or to the sensors.

Message Specification can_boot_data

Message Type: **can_boot_data**
Identifier: **0x3F4**
DLC: **8**
Source: **DriveRecorder2**
Destination: **UMRR sensor / central ECU**
Format: **Intel**
Type: **unsigned**

Data Byte 7	Data Byte 6	Data Byte 5	Data Byte 4	Data Byte 3	Data Byte 2	Data Byte 1	Data Byte 0
Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0

11 External CAN Bus Interface Specification

This specification gives a detailed description of the CAN data communication used on the central ECU's external CAN (CAN1).

Baud Rate:	500kBit/s
T_{seg1}:	8
T_{seg2}:	7
T_{sjw}:	1

Synchronization on one Edge only.

Other baud rates are available on request.

11.1 Specification of the output data

There are two categories of messages to be transmitted:

a) Status messages of the sensor
b) Data of detected objects.

11.1.1 Status messages

Part a) is used to transmit information on:

sensor_control

- Sensor Status (works correctly / errors)
- System Mode (sensor detection mode)
- Sensor ID in the network
- Field of view

The sensor control message is transmitted every cycle.

Message-Specification sensor_control:

Message Type: **sensor_control**
Identifier: **0x500**
DLC: **8**
Quelle: **Central ECU**
Destination: **Vehicle Systems**
Format: **Intel**
Type: **unsigned**
Resolution: **1**
Offset: **0**

Data Byte0	Data Byte1	Data Byte2	Data Byte3	Data Byte4	Data Byte5	Data Byte6	Data Byte7
Source Device (int) = ID in the network	Status CAN Communication	Sensors present (Lo-Byte)	Bits 0..3 Sensors present (Hi-Byte) Bits 4..7 Ethernet connection state	time _stamp_0 low-byte of unsigned long (in ms)	time_ stamp_1	time_ stamp_2	time_ stamp_3

Data field explanation:

- Source device:
  D_Net value of Bumper IP, value **10** = e.g. 192.168.0.**10**

- Status CAN Communication: bit coded error value
  0x00 no error
  0x01 Invalid UMRR on CAN
  0x02 unused
  0x04 Invalid Sensor ID
  0x08 unused
  0x10 Wrong target number
  0x20 CAN HW Reinit
  0x40 Sensor communication incomplete
  0x80 Processing duration longer than reserved timeframe

- Sensors present:
  Indication of present sensors in actual measurement cycle
  UMRR_ID15 = MSB
  UMRR_ID0 = LSB

- Ethernet connection state

ET_NOTCONNECT	0
ET_CONNECTING	1
ET_ISCONNECT	2
ET_RELIEVE_ERROR	3
ET_ERROR_STATE	4
ET_WAIT_OF_REPEAT_DATA	5

11.1.2 Object data messages

Part b) is used for targets, which can be represented by one point. It is again composed of two types of messages:

b 1) Control message used as a header for the object list *object_control*
b 2) Object data messages (lower priority) *object_data_1...3.*

Each individual Central Processor system prioritizes the detected objects and limits the number of transmitted objects, according to the commands sent to this system.

The maximum number of objects is limited by software to 240, but this value can be extended.

Object Control Message (Header for Object List)
In part b 1) the following data are transmitted with high priority:

object_control:

- Number of objects in the appending list *nofobjects*
- Number of messages for each object *nofmessages*
- Cycle time of the corresponding cycle the objects were detected *tscan*
- Current value of the cycle counter *cycle_count*

Message-Specification object_control:

Message Type: **object_control**
Identifier: **0x501**
DLC: **8**
Source: **Central ECU**
Destination: **Vehicle Systems**
Format: **Intel**
Type: **unsigned**
Resolution: **1**
Offset: **0**

Data Byte0	Data Byte1	Data Byte2	Data Byte3	Data Byte4	Data Byte5	Data Byte6	Data Byte7
nofobjects (int)	nof- messages (int)	t_scan in ms	Mode	cycle_count_0 low-byte of unsigned long	cycle_count_1	cycle_count_2	cycle_count_3

Mode	Meaning
0	**Tracking operation (default, normal)**
4	Output of simulation targets

Object Data for single point representation (Object List)

Part b 2) comprises (nofobjects x nofmessages) CAN messages. No more than 3 messages per object shall be used (data load on the bus). At least one message per object must be sent. In addition to the cartesian position and speed the following informations are transmitted:

- constant (over the lifetime) index for every tracked object; *Object ID*
- used dynamic model (0..stationary object; 12..moving object); *Object Type*

Message Specification object_data_1:

Message Type: **object_data_1**

Identifier:

Object #0: **0x510**
Object #1: **0x511**
Object #2: **0x512**
...
Object #239: **0x5FF**

DLC: **8**
Source: **Central ECU**
Destination: **Vehicle Systems**
Format: **Intel**
Type: **unsigned**

Item	Bit length	Start	End	Unit	Resolution	Offset
X coordinate range	14	Byte 0, Bit 0	Byte 1, Bit 5	m	0.1	8192
Y coordinate range	14	Byte 1, Bit 6	Byte 3, Bit 3	m	0.1	8192
X velocity component	11	Byte 3, Bit 4	Byte 4, Bit 6	m/s	0.1	1024
Y velocity component	11	Byte 4, Bit 7	Byte 6, Bit 1	m/s	0.1	1024
Object length	8	Byte 6, Bit 2	Byte 7, Bit 1	m	0.2	none
Object ID	6	Byte 7, Bit 6	Byte 7, Bit 7	-	-	none

All objects are reported in unsorted order.

Data explanation:

- **Object length -** determines track type: point track or object. Objects are cars, trucks etc. that are tracked with a object length and width within the tracking filter.
  - Object length == 0 represents a point track
  - Object length != 0 represents an object. For oncoming traffic, the center of the object front side is reported. For departing traffic, the center of the object rear side is reported.

- **Object ID –** ID from the tracking filter. IDs are retained as long as the track is present in the tracking filter.

11.2 Specification of the input data

11.2.1 Command message

The bumper command message is used to transmit new parameters or initiate various actions in the central ECU.

A great number of commands are available to modify the detection and/or tracking behavior of the central ECU. The commands that may be used in different applications are customized and will be specified individually.

Message Specification command:

Message Type: **command**
Identifier: **0x3F2**
DLC: **8**
Source: **Test tools**
Destination: **Central ECU**
Format: **Intel**
Type: **unsigned**

Message composition for Bumper Action 0 ..239:

Data Byte0	Data Byte1	Data Byte2	Data Byte3	Data Byte4	Data Byte5	Data Byte6	Data Byte7
Destination device ID	Parameter number	Parameter Type	Action	Parameter data byte 0 (low-byte)	Parameter data byte 1	Parameter data byte 2	Parameter data byte 3

Message composition for Bumper Action 240 .. 255:

Bumper Action values 240 .. 255 are used to **pass on a command to the sensor CAN** (#2). The destination Sensor ID is calculated by:

Sensor_ID = BumperAction – 240

whereas Bumper Action == 255 initiates Broadcast message on the sensor CAN.
The Sensor Action is stored in Parameter Byte 3 (Data Byte 7). Parameter Type, Parameter number and Parameter Value are passed on to the sensor.

Data Byte0	Data Byte1	Data Byte2	Data Byte3	Data Byte4	Data Byte5	Data Byte6	Data Byte7
Destination device ID	Parameter number	Parameter Type	Action	Parameter data byte 0 (low-byte)	Parameter data byte 1	Parameter data byte 2	Sensor Action

Destination Device ID	Meaning
0..15	Sensor or Bumper 0..15
255	Broadcast Command

Parameter Type	Meaning
0	Integer interval: -2147483648 2147483647
1	Float interval: -2147.483648 2147.483647 conversion: data byte * 1000000

negative value:

-2 hexadecimal 0xFF FF FF FE

2 = 0x00 00 00 02 negation
=> 0xFF FF FF FD +1
=> 0xFF FF FF FF FE = -2

Basic Commands

Name	Description	Number	Action	Default Value	Typical values
Interface mode		0	0	0	0 = normal output 4 = simulate targets
Can1 master tx		11	0	0	0 = CAN remains silent 1 = CAN transmission active
Nofsimobjects	Number of simulated targets in CAN target simulation mode	20	0	31	0...64

11.2.2 Command_Read message

The bumper command may be used to read parameter values from the central ECU. Parameter reads to UMRR sensors attached to the central ECU are not supported.

Message Specification command_read:

Message Type:	**command_read**
Identifier:	**0x3F2**
DLC:	**8**
Source:	**Test tools**
Destination:	**Central ECU**
Format:	**Intel**
Type:	**unsigned**

Message composition for Bumper Action 0 ..239:

Data Byte0	Data Byte1	Data Byte2	Data Byte3	Data Byte4	Data Byte5	Data Byte6	Data Byte7
Destination device ID	Parameter number	**Parameter Type**	Action	Don't care	Don't care	Don't care	Don't care

Destination Device ID	Meaning
0..15	Sensor or Bumper 0..15

Parameter Type	Meaning
4	Integer interval: -2147483648 2147483647
5	Float interval: -2147.483648 2147.483647 conversion: data byte * 1000000

The central ECU will respond with a **modified command** message, yielding the command value:

Message Specification parameter_value:

Message Type: **parameter_value**
Identifier: **0x3F2**
DLC: **8**
Source: **Central ECU**
Destination: **Test tools**
Format: **Intel**
Type: **unsigned**

Message composition for Bumper Action 0 ..239:

Data Byte0	Data Byte1	Data Byte2	Data Byte3	Data Byte4	Data Byte5	Data Byte6	Data Byte7
Source device ID	Parameter number	**Parameter Type**	Action	Parameter data byte 0 (low-byte)	Parameter data byte 1	Parameter data byte 2	Parameter data byte 3

The parameter data bytes contain the parameter value read from the central ECU's memory.

Source Device ID	Meaning
0..15	Sensor or Bumper 0..15

Parameter Type	Meaning
8	Integer interval: -2147483648 2147483647
9	Float interval: -2147.483648 2147.483647 conversion: data byte * 1000000

Program Example for float conversion:

```
unsigned char RCV_data[8];
int           itmp;
float         ftmp;
char          *BytePtr;


            // float
            ftmp  =(float)atof(ParamValueE2->Text.c_str() );
             // floats will be send as integer (*1 000 000)
            itmp = 1000000.0 * ftmp + 0.5;
            BytePtr = (char*)&itmp;
            CandisCommand.ParByte1 = *BytePtr++;
            CandisCommand.ParByte2 = *BytePtr++;
            CandisCommand.ParByte3 = *BytePtr++;
            CandisCommand.ParByte4 = *BytePtr++;
                            break;

RCV_data[7]= CandisCommand.ParByte4;
RCV_data[6]= CandisCommand.ParByte3;
RCV_data[5]= CandisCommand.ParByte2;
RCV_data[4]= CandisCommand.ParByte1;

RCV_data[3]= (char)CandisCommand.Action;
RCV_data[2]= (char)CandisCommand.ParType;
RCV_data[1]= (char)(CandisCommand.ParNumber);
RCV_data[0]= (char)(CandisCommand.Destination);

// send msg
// bumper_command: Id 0x3F2 ; Length 8
Cancard[0]->send_fifo_msg(sc,0x3F2,8,RCV_data);
```

The following message is used to download new software to the central ECU. It is only used by service personnel. Although the reprogramming of new software is locked by a specific command sequence, this Message Identifier should not be used in the CAN network in normal operation, for it is interpreted by the DSP every time it is received and therefore causes additional processor load.

Message Specification can_boot_data

Message Type:	**can_boot_data**
Identifier:	**0x3F4**
DLC:	**8**
Source:	**CANLoader**
Destination:	**Central ECU**
Format:	**Intel**
Type:	**unsigned**

Data Byte0	Data Byte1	Data Byte2	Data Byte3	Data Byte4	Data Byte5	Data Byte6	Data Byte7
data byte 0	data byte 1	data byte 2	data byte 3	data byte 4	data byte 5	data byte 6	data byte 7

12 External Ethernet Interface Specification

12.1 Introduction

This section describes communication with the Ethernet module of the sub network controller.

In general, **data message composition is identical to the External CAN Bus Specification.** This section describes all Ethernet related aspects only.

Ethernet speed rating is 100Mbit/s.

Data representation is unsigned Intel.

12.2 Communication basics

Each sub network controller is identified by an individual IP in the network.

Communication is connection oriented via TCP/IP. The Network Controller acts as a server. The sub network controllers act as clients. Each client establishes TCP/IP based connections to the server and acts responsible for the connection management:

- Connections are held until the server disconnects or the client runs into a Ethernet error condition.
- 6s after power up, the first connect attempt is initiated.
- every 20s, the client attempts to establish a connection.
- 5s after disconnect, next connect attempt is initiated.
- in Ethernet error condition, the client attempts to establish a connection every 120s.

The sub network controller runs in **measurement cycles**. The cycle time is determined by the UMRR sensor's measurement time. Default is 50ms. Sensor network error condition is indicated by a cycle time of 60ms.
Every cycle, status information and objects lists are sent over Ethernet (when connected to the server). If no objects lists are present due to absence of targets in the surveillance area, at least the status information is sent.

If the transmit data size exceeds the Ethernet data payload size, further Ethernet packets are appended.
If the data payload exceeds the transmit buffer, the end of list sequence is written to the end of the last Ethernet packet. All additional transmit data will be lost.

12.3 Output data specification

Transmit data consists of the following parts:
- training sequence for begin of data detection
- header
- encapsulated transmit data as to external CAN specification
- end of list sequence

12.3.1 Training sequence

The training sequence consists of the following bytes: “AFAFAFAFA”. This sequence of bytes will not appear in any other part of the data payload.

Byte	Content
0	‘A’
1	‘F’
2	‘A’
3	‘F’
4	‘A’
5	‘F’
6	‘A’
7	‘F’
8	‘A’

12.3.2 Header

Byte	Content	Description
9	1	Header Parameter set Version
10	20	Number of Header bytes
11	0 .. 255	Sub net controller IP A net: xxx.xxx.xxx.xxx
12	0 .. 255	Sub net controller IP B net: xxx.xxx.xxx.xxx
13	0 .. 255	Sub net controller IP C net: xxx.xxx.xxx.xxx
14	0 .. 255	Sub net controller IP D net: xxx.xxx.xxx.xxx
15	0 .. 255	Ethernet last error 0 – no error
16 .. 17	0 .. 65535	Ethernet error count
18	0 .. 255	UMRR network no sync count
19 .. 20	0 .. 65535	UMRR Reception diagnostics – see CAN 0x500 message description
21	0 .. 255	CAN2 error – see CAN 0x500 message description
22	0 .. 255	CAN3 error – see CAN 0x500 message description
23	0 .. 8	Last UMRR
24 .. 26	0	Unused
27 .. 28	0	Header CRC – TBD – filled with zero

12.3.3 Encapsulated transmit data

Transmit data contents may vary from this point. In general, tracking object output follows, but in accordance to communication extension and sub network controller parameterization, further data contents may be transmitted.
Data content is determined by the CAN ID followed by a fixed length eight byte data payload.

Encapsulated transmit data composition is as follows:

Byte	Content	Description
N+0	0 .. 255	CAN ID high byte
N+1	0 .. 255	CAN ID low byte
N+2	0 .. 255	Data Byte 0
N+3	0 .. 255	Data Byte 1
N+4	0 .. 255	Data Byte 2
N+5	0 .. 255	Data Byte 3
N+6	0 .. 255	Data Byte 4
N+7	0 .. 255	Data Byte 5
N+8	0 .. 255	Data Byte 6
N+9	0 .. 255	Data Byte 7

The server must determine by the CAN ID whether a 10 Byte CAN message is relevant and interpret the relevant messages. Please refer to section 11.1.

12.3.4 End of List

The end of List consists of the following bytes: “EDEDEDEDE”. This sequence of bytes will not appear in any other part of the data payload.

Byte	Content
0	‘E’
1	‘D’
2	‘E’
3	‘D’
4	‘E’
5	‘D’
6	‘E’
7	‘D’
8	‘E’

12.4 Input data specification

Receive data consists of the following parts:
- training sequence for begin of data detection
- encapsulated receive data as to external CAN specification
- end of list delimiter

No more than 120 Parameters shall be sent at a time.

12.4.1 Training sequence

The training sequence consists of the following bytes: 0xC4 0xC4 0xC4 0xC4 0xC4 0xC4 0xC4 0xC4 0xC4 . This sequence of bytes will not appear in any other part of the data payload.

Byte	Content
0	0xC4
1	0xC4
2	0xC4
3	0xC4
4	0xC4
5	0xC4
6	0xC4
7	0xC4
8	0xC4

12.4.2 Encapsulated receive data

By default, the sub net controller listens for commands. Message composition is identical to the CAN commands in section 11.2. The command message is composed as follows:

Byte	Content	Description
9	Number_of_packets	Number of packets sent in a burst **(usually == 1)**
10	Packet_number	Packet number within burst, **starting from 0**
11	0 .. 255	Bumper ECU IP A net: xxx.xxx.xxx.xxx
12	0 .. 255	Bumper ECU IP B net: xxx.xxx.xxx.xxx
13	0 .. 255	Bumper ECU IP C net: xxx.xxx.xxx.xxx
14	0 .. 255	Bumper ECU IP D net: xxx.xxx.xxx.xxx
15	0 .. 255	CAN number, **set to 0**
16	0x03	CAN ID high byte
17	0xF2	CAN ID low byte
18	0 .. 1	Command target: **0: Bumper ECU** 1: Sensor
19	0 .. 255	Data Byte 0
20	0 .. 255	Data Byte 1
21	0 .. 255	Data Byte 2
22	0 .. 255	Data Byte 3
23	0 .. 255	Data Byte 4
24	0 .. 255	Data Byte 5
25	0 .. 255	Data Byte 6
26	0 .. 255	Data Byte 7

If more than one command is to be sent, commands can be appended in a single ethernet packet:

27	0x03	CAN ID high byte
28	0xF2	CAN ID low byte
29	0 .. 1	Command target: **0: Bumper ECU** 1: Sensor
30	0 .. 255	Data Byte 0
31	0 .. 255	Data Byte 1
32	0 .. 255	Data Byte 2
33	0 .. 255	Data Byte 3
34	0 .. 255	Data Byte 4
35	0 .. 255	Data Byte 5
36	0 .. 255	Data Byte 6
37	0 .. 255	Data Byte 7

etc.

12.4.3 End of List

The end of List consists of the following bytes: 0xE7 0xE7 0xE7 0xE7 0xE7 0xE7 0xE7 0xE7 0xE7. This sequence of bytes will not appear in any other part of the data payload.

Byte	Content
0	0xE7
1	0xE7
2	0xE7
3	0xE7
4	0xE7
5	0xE7
6	0xE7
7	0xE7
8	0xE7

12.5 Receive data specification

The following parameters can be modified during runtime. Parameters in RED indicate the minimum required information per device. Command Actions are used to program the parameter set into a device or read out data.

Name	Parameter	Type	Action	Values	Default
par_bumper.mode	0	Int	64	0 Normal 4 Simulate	0 Normal
par_bumper.lastumrr	2	Int	64	0 .. 8	8
par_bumper.umrr_active	5	Int	64	Bit coded - UMRR0 = LSB - UMRR8 = MSB	Bin 111111111
par_bumper.Flash_Par_State	7	Int	64	Bit coded See section 12.7	Bin 00000000 00000000 For default parameters 11011111 11111000 For parameters read from FLASH
par_bumper.nofsimobjects	20	Int	64	0 .. 64	32
par_bumper.eth_communicate	23	Int	64	0..1	1
par_bumper.eth_tx_rawtargets	24	Int	64	0..1 1 bumper reports raw sensor targets	0
par_bumper.eth_bumper_ip	29	Int	64	MSB: A_net 192 B_net 168 C_net 0 D_net 10 LSB	192.168.0.200
par_bumper.eth_client_ip	31	Int	64	MSB: A_net 192 B_net 168 C_net 0 D_net 10 LSB	192.168.0.201
par_bumper.eth_port	30	Int	64	0 .. 65535	36

Name	Parameter	Type	Action	Values	Default
par_track.f_multiHypoAreaDiameter	10	float	65	0.0 .. 300.0 m	40.0m

For every sensor (UMRR0 .. UMRR9):

Name	Parameter	Type	Action	Values	Default
par_umrr.antenna_type	0	Int	72+ID	24, 26, 193	26
par_umrr.i32_mod	1	Int	72+ID	18, 20	20
par_umrr.fPos_x	2	float	72+ID	-819,2 .. +819 X coordinate	0.0m
par_umrr.fPos_y	3	float	72+ID	-819,2 .. +819 Y coordinate	ID*10.0m
par_umrr.fPos_orientation	4	float	72+ID	[rad] –pi .. +pi Sensor orientation	0
par_umrr.fPos_h	21	float	72+ID	1.0 ..30.0 [m] mounting height above ground level	1.0m
par_umrr.headingHypotheses	22	float	72+ID	[rad] –pi .. +pi Main direction of travel for objects monitored by this sensor; in world coordinate system; modifying this parameter requires low level software to activate setting	0

12.6 Receive data actions

Action	Description
1	High level Software Reset
2	Low level software Reset
64	Address Bumper parameters
65	Address Tracking parameters
72+ID	Address UMRR sensor parameter within tracking
100	Program parameters into flash
240+ID	Pass on a parameter to a UMRR sensor

12.7 Non-volatile parameters

All Bumper parameters can be programmed non volatile into the Bumper's FLASH memory.

The parameters can be changed during runtime but all parameter changes are volatile until a command is issued to the Bumper to perform programming into FLASH. Then, the complete parameter set is programmed into the Bumper's FLASH memory.

The programming sequence is initiated by a Bumper command message with Action 100.

The Bumper leaves the functional software restarts for programming and enters the functional software directly after programming.

At startup of the functional software, the parameters are initialized with default values. Then, the FLASH section is tested for valid parameters. Once the Bumper's FLASH memory has been written, the parameters from the FLASH section are applied.

The parameters stored in the FLASH memory are separated into sections, each verified with a CRC. Only if a section CRC proves valid at startup of the Bumper functional software, the parameters from that section are applied.

After startup, **par_bumper.Flash_Par_State variable** contains the result of reading the FLASH parameter sections. The variable is bit-coded for each section:

- Bit value == '0' means: invalid FLASH parameter section, default values applied
- Bit value == '1' means: valid FLASH parameter section, FLASH values applied

The par_bumper.Flash_Par_State variable sections are coded as shown in the table below:

Bit	FLASH section
0 (LSB)	Par_bumper
1	Par_tracking
2	Par_object_tracking
3	Par_umrr ID0
4	Par_umrr ID1
5	Par_umrr ID2
6	Par_umrr ID3
7	Par_umrr ID4
8	Par_umrr ID5
9	Par_umrr ID6
10	Par_umrr ID7
11	Par_umrr ID8
12	Par_umrr ID9
13-31 (MSB)	Unused section

12.7.1 Bumper software update considerations

When a bumper software update is performed, the parameter sections are cleared. The parameters are then reset to default. The par_bumper.Flash_Par_State will yield 0 until a new parameter set is written into the Bumper's FLASH.