



# O3M CAN Interface

This document explains the CAN interface of the O3Mxxx sensor family. It describes the mainly the J1939 messages and the UDS features (e.g. online parametrization and graphic primitives (O3M2xx only)). The CANopen specifics are briefly mentioned as well.

V07 – 2020-05-19

## 1. Content

2.	Output Messages .....	3
2.1.	The DBC File .....	3
2.2.	A note on message ID's .....	3
2.3.	Messages common to all O3M variants .....	3
2.3.1.	Version Information .....	3
2.3.2.	Global Information .....	4
2.3.3.	Virtual Switches .....	5
2.3.4.	2D camera calibration data.....	6
2.4.	Messages exclusive to DI variant.....	7
2.4.1.	Basic function parameters .....	7
2.4.2.	Basic function results.....	10
2.5.	Messages exclusive to OD variant .....	12
2.6.	Messages exclusive to LG variant.....	17
3.	Input Messages .....	22
3.1.	Messages common to all O3M variants .....	22
3.1.1.	Ego data .....	22
3.1.2.	Virtual switches.....	23
3.1.3.	A note on the UDS transfer protocol .....	23
3.1.4.	2D Overlay input.....	25
3.1.5.	Online parametrization .....	33
3.2.	Description of the Messages .....	36
3.3.	Messages exclusive to DI variant.....	37
3.3.1.	Online parameters specific to DI variant .....	37
3.4.	Messages exclusive to OD variant .....	39
3.4.1.	Online parameters specific to OD variant.....	39
3.5.	Messages exclusive to LG variant.....	45
3.5.1.	Online parameters specific to LG variant .....	45
3.6.	Standby Mode.....	50
3.6.1.	Enter Standby Mode.....	50
3.6.2.	Leave Standby Mode.....	50



3.7.	Teach.....	51
3.7.1.	Teach command.....	51
3.7.2.	Unteach command.....	51
3.8.	O3M Reset.....	51
3.9.	UDS Response Codes .....	51
3.10.	UDS Diagnostic Information .....	51
4.	CANopen .....	53
5.	Version History .....	54



## 2. Output Messages

### 2.1. The DBC File

Every firmware release for O3M contains a file with the ending “.dbc”. This file describes all messages and signals used in CAN communication including encoding and descriptions. This document's intention is to explain how to use the information contained in the signals. For convenience, the encoding of the signals is briefly mentioned. For more information on the signal encoding, please refer to the file provided with the firmware release.

### 2.2. A note on message ID's

The O3M message ID's are generated according to J1939 ID specifications. Therefore, byte 0 of the ID of each output message is the O3M's J1939 ID and byte 1 of each input message is the O3M's J1939 ID. In this document, the part of the message ID that depends on the O3M's J1939 ID is highlighted in red. To communicate with an O3M with a different J1939 ID, please adjust the message IDs accordingly. The letter “x” at the end of the ID indicates the use of an extended identifier.

**Example:** Message name: Global\_Information (O3M **Output** message)

ID according to DBC file (database definition): 0x4FF01EFx (extended identifier)

Byte3	Byte2	Byte1	Byte0
0x04	0xFF	0x01	0xEF (J1939 ID of O3M. Default is 0xEF)

**Example:** Message name: VS\_MixedInput (O3M **Input** message)

ID according to DBC file (database definition): 0x4EFEFFEx (extended identifier)

Byte3	Byte2	Byte1	Byte0
0x04	0xEF	0xEF (J1939 ID of O3M. Default is 0xEF)	FE

### 2.3. Messages common to all O3M variants

#### 2.3.1. Version Information

The message **DBC\_File\_Version** can be used to query the version of the O3M's communication matrix. The version is part of every DBC file, where it is present in the first line of the file content, e.g. Version “DI 3.8.0”. To ensure correct communication to O3M it is best practice to verify that the communication protocol version on O3M matches the one expected by the communication function.

The message DBC\_File\_Version is output only at request. To query the communication protocol version used by O3M, please follow these steps:

1) **Send** request message to O3M

Send a message using the following specification:

Name (.dbc-file)	RQST
ID	0x18EAEFF1x
DLC	3



Payload	Byte0	Byte1	Byte2
	0xA0	0xFF	0x00

## 2) **Receive** response from O3M

Receive the response message

Name (.dbc-file)	DBC_File_Version		
ID	0x4FFA0EFx		
DLC	3		
Payload	Byte0	Byte1	Byte2
	Variant	DBC Version Major	DBC Version Minor

There is no output of the patch number of the protocol version (eg DI\_4.21.7). Patch version increments are only used for firmware patches or documentation updates and therefore do not change the actual protocol and do not effect communication.

The variant byte indicates the O3M variant:

Variant Byte	O3M Variant
0x00	DI
0x01	OD
0x02	LG

Note: The O3M also outputs a message with ID 18E8FFEfx to acknowledge the RQST message. If the first byte of that message is unequal to 0, the request was unsuccessful and the DBC\_File\_Version message is not output.

### 2.3.2. Global Information

The global information message contains O3M status information and should be continuously monitored to ensure that any problems encountered by O3M are noticed and reacted to.

In particular, use the signal SwCtrl\_OpMode to monitor the O3M's operational mode and ensure that it is 0x22 (RUN) unless in case of reset or standby mode.

In addition, the message counter can be used to verify that communication is still running.

Name (.dbc-file)	Global_Information
ID	0x4FF01EFx
DLC	7

Signals of the global information message:

Position	Name	Unit	Range	Meaning
Bit 0 – 31	Glob_master_time	µs	[0-4294967295]	Current timestamp in µs. Wraps around after 71 minutes.
Bit 32 – 39	Glob_sensor_available	Bit-field	-	Every bit indicates the status of one feature that might affect the quality of O3M output. The output will still be available in these cases, but the quality might be diminished. The bit "TRACKING_ERROR" is only used in OD variant of O3M.  INTERFERENCE_DETECTED (1u) SPRAY_DETECTION (2u) TRACKING_ERROR (4u) INVALID_CAM_ORIENTATION (8u) SIGNAL_PATH_MONITORING (16u)



				INTERNAL_ERROR (32u) BLOCKAGE_DETECTED (64u) FORCE_CALIBRATION_RESET (128u)
Bit 40 – 47	Blockage_Stat us	%	[0 .. 100]	Percentage of the O3M's 3D camera window that appears to be covered by dirt, fog, or similar. Please use vision assistant to set-up blockage detection. The value 0xFE indicates that blockage detection is not available on O3M. The value 0xFF indicates a general error.
Bit 48 – 53	SwCtrl_OpMo de	-	Enumeration	Current status of O3M operational mode. 0x11: INIT 0x12: STARTUP 0x13: DSP_BOOT 0x14: SELFTEST 0x15: WIAT_DSP_BOOTED 0x17: PARAMETERIZING 0x21: LIMITED RUN 0x22: RUN 0x23: STANDBY <b>0x31: EMERGENCY</b> The operational mode should be 0x22 (RUN) during normal operation. When the O3M is set to standby mode using diagnostic service (see below), the operation mode should be 0x23 (STANDBY). 0x31 is an emergency mode from which O3M cannot automatically recover.
Bit 54 – 55	Global_Inform ation_cnt	-	[0 .. 3]	Message counter; cycles through the values 0, 1, 2, 3. Changes every time the message is sent.

### 2.3.3. Virtual Switches

O3M has the ability to perform simple logic operations using result values as well as values input over CAN. For details on setting up the logic, please refer to ifm Vision Assistant.

To send values over CAN to O3M to use as inputs to the logic, please use the message VS\_Mixed\_Input described in the section “O3M input messages”

The results of the logic operations are output using three CAN messages: VS\_MixedOutput, VS\_AnalogOutput and VS\_DigitalOutput. Use ifm Vision Assistant to configure the sending of these messages.

The message VS\_MixedOutput offers 2 analog outputs as well as 38 digital outputs and should be sufficient for most applications. If you need to use more outputs, 4 additional analog outputs are available in the message VS\_AnalogOutput and 62 additional digital outputs in the message VS\_DigitalOutput.

All digital outputs are encoded as single bits in the CAN message. All analog outputs are encoded using 12 bits. The values 0x000 through 0xFA0 are used to encode the values 0 to 1 with a resolution of 1/4000 per bit. 0xFFD indicates that the result mapped to the analog output is smaller than 0, 0xFFE indicates that it is greater than 1, 0xFFF means that no value is mapped to the output.

The output messages have the following layout:

Name (.dbc-file)	VS_MixedOutput	
ID	0x4FF52 <b>EF</b> x	
DLC	8	
Payload	Bit 0 – 11	Analog output 00
	Bit 12 – 23	Analog output 01



	Bit 24 – 61	Digital output 0 through 37, one digital output per bit
	Bit 62 – 63	Message counter; cycles through the values 0, 1, 2, 3. Changes every time the message is sent.

Name (.dbc-file)	VS_AnalogOutput	
ID	0x4FF50EFx	
DLC	8	
Payload	Bit 0 – 11	Analog output 02
	Bit 12 – 23	Analog output 03
	Bit 24 – 35	Analog output 04
	Bit 36 – 47	Analog output 05
	Bit 48 – 49	Message counter; cycles through the values 0, 1, 2, 3. Changes every time the message is sent.

Name (.dbc-file)	VS_DigitalOutput	
ID	0x4FF51EFx	
DLC	8	
Payload	Bit 0 – 61	Digital output 38 through 99, one digital output per bit
	Bit 62 – 63	Message counter; cycles through the values 0, 1, 2, 3. Changes every time the message is sent.

### 2.3.4. 2D camera calibration data

For O3M25x and O3M 26x, the calibration information of the O3M's 2D camera can be output on CAN. This behavior can be configured using the ifm Vision Assistant.

The 2D camera calibration information contains all information which is needed to calibrate the 3D values output by O3M to the 2D image.

The message **Constant\_2D\_Calib\_Data** contains the optical model of the 2D camera lens using the Bouguet model. To preserve bandwidth, only one parameter is sent per operating cycle. Since there are a total of 13 parameters, it takes 13 operating cycles to collect all parameters in order to have a complete optical model of the O3M's 2D camera.

Name (.dbc-file)	Constant_2D_Calib_Data				
ID	0x4FF0AEFx				
DLC	5				
Payload	Byte0	Byte1	Byte2	Byte3	Byte4
	Multiplex Signal	One parameter of the optical model as float32 datatype using little-endian (intel) byte order.			
		Multiplex Signal Value		Parameter	
		0x00		Fx	
		0x01		Fy	
		0x02		Mx	
		0x03		My	
		0x04		Alpha	
		0x05		K1	
		0x06		K2	
		0x07		K5	
		0x08		K3	
		0x09		K4	
		0x0A		Center offset X	
		0x0B		Center offset Y	
		0x0C		Center offset Z	



A description of these parameters can be found here:

[http://www.vision.caltech.edu/bouquetj/calib\\_doc/htmls/parameters.html](http://www.vision.caltech.edu/bouquetj/calib_doc/htmls/parameters.html)

The message **Dynamic\_2D\_Calib\_Data** contains information about the position (center and rotation angles) of the 2D camera in O3M world coordinates. The ID of the message is 0x4FF09EF. The values can change as the O3M position and rotation is changed using dynamic calibration or online parameter services. Therefore, the values are output continuously. The byte order of the message is little endian (intel).

The message layout is as follows:

Position	Name	Unit	CAN to physical	Range	Special values	Meaning
Bit 0 – 11	ExtrCalib_2D_rot_x	rad	$P = 0.1 * \pi / 180 * C - 1.1 * \pi$	$[-1.1 * \pi .. 1.1 * \pi]$	*	Rotation angle around x-Axis between 2D camera and world coordinate system
Bit 12 – 19	ExtrCalib_2D_delta_tx	m	$P = 0.01 * C - 1.2$	$[-1.2 .. 1.2]$	**	Additional offset between 2D camera center and world coordinate system in x-Direction. Static offset is signal "Center offset X" in message "Constant_2D_Calib_Data"
Bit 20 – 31	ExtrCalib_2D_rot_y	rad	$P = 0.1 * \pi / 180 * C - 1.1 * \pi$	$[-1.1 * \pi .. 1.1 * \pi]$	*	Rotation angle around y-Axis between 2D camera and world coordinate system
Bit 32 – 39	ExtrCalib_2D_delta_ty	m	$P = 0.01 * C - 1.2$	$[-1.2 .. 1.2]$	**	Additional offset in y-direction. See ExtrCalib_2D_delta_tx.
Bit 40 – 51	ExtrCalib_2D_rot_z	rad	$P = 0.1 * \pi / 180 * C - 1.1 * \pi$	$[-1.1 * \pi .. 1.1 * \pi]$	*	Rotation angle around z-Axis between 2D camera and world coordinate system
Bit 52 – 59	ExtrCalib_2D_delta_tz	m	$P = 0.01 * C - 1.2$	$[-1.2 .. 1.2]$	**	Additional offset in z-direction. See ExtrCalib_2D_delta_tx.
Bit 62 – 63	Dynamic_2D_Calib_Data_cnt	-	-	$[0 .. 3]$	-	Message counter; cycles through the values 0, 1, 2, 3

\*: 0xFFD = Value smaller than minimum, 0xFFE = Value larger than maximum, 0xFFF = Error

\*\*: 0xFD = Value smaller than minimum, 0xFE = Value larger than maximum, 0xFF = Error

Please check O3M documentation for more information on coordinate systems and rotation angles.

## 2.4. Messages exclusive to DI variant

On O3MXX1, it is possible to define regions of interest (ROI) where the minimum, maximum, median, or a certain percentile of the 3D values is calculated for every region. The regions are defined using ifm vision assistant.

### 2.4.1. Basic function parameters

To make it easier to understand the output values of the defined regions, the region definitions themselves are also output on CAN. To minimize load, only one message is used and there are different layouts to send all parameters. The receiver has to collect all the information and combine it to know the exact definition of the ROIs.



There are 4 different message layout. Layout 0 contains the general information on basic function configuration and is sent once at the beginning. Layouts 1 through 3 contain the information about the configuration of 1 ROI and are sent consecutively for each defined ROI. After layout 3 is sent for the last ROI, the process starts again with layout 0. So, in total, the layout order is 0 – (1 – 2 – 3) – (1 – 2 – 3) – (1 – 2 – 3) – ... – 0 – (1 – ... with (1 – 2 – 3) sent for every defined ROI.

Since the properties of the ROIs are configured using ifm vision assistant, it is not always necessary to receive and interpret this information. But sometimes, it is convenient to automatically react to changes in the ROI definition in the application, so that the application does not have to be adjusted when the ROI definitions are changed on O3M. Therefore, the ROI definition values are always sent on CAN.

Name (.dbc-file)	BF_Global_Parameters		
ID	0x4FF02EFx		
DLC	7		
Payload	Bit 0 – 2	ROI_Param_mode: Layout indicator	
	Bit 3 – 55	Payload depending on ROI_Param_mode value	

This layout is sent once at the beginning and contains the general configuration of the basic function feature.

ROI_Param_mode = 0	Definitions common to all ROIs
Bit 2 – 4	OutputMode Defines which values are sent on CAN for each ROI. Selecting an output mode with fewer output values reduces load on the CAN. 0 = x 1 = y 2 = z 3 = xyz 4 = ampl 5 = xyz+ampl 6 = output not available
Bit 8 – 14	Number_of_groups Number of ROI groups defined. When ROIs are combined to groups, only one output is created for each group. When ROIs are not combined, one group is used for each ROI.
Bit 16 – 22	Number_of_rois Total number of ROIs defined. This number also indicates how often the (1 – 2 – 3) ROI_Param_mode cycle will be sent since the definition is sent for each ROI.
Bit 24 – 29	TemporalWindowSize Indicates the setting of the filter for the output values. Please refer to ifm vision assistant for details on this parameter.

The following layouts are sent once per defined ROI and contain the detailed parametrization of each ROI.

ROI_Param_mode = 1	ROI configuration values – Part 1	
Bit 2 – 7	ROI_number Indicates which ROI the values in the message belong to. This signal is repeated in each layout message.	
Bit 8 – 14	ROI_group Indicates which ROI group the ROI belongs to.	
Bit 16 – 30	ROI_minimum_cartesian_x Indicates the minimum x-coordinate for 3D ROIs. For 2D ROIs, the value is -10000	
	Unit	cm
	Range	[-10000 to 10000]





	CAN to physical	$P = C - 10000$
	Special values	0x7FFD = smaller than minimum
		0x7FFE = greater than maximum
		0x7FFF = error
Bit 32 – 46	ROI_maximum_cartesian_x Indicates the maximum x-coordinate for 3D ROIs. For 2D ROIs, the value is 10000	
	Unit	cm
	Range	[-10000 to 10000]
	CAN to physical	$P = C - 10000$
	Special values	0x7FFD = smaller than minimum
		0x7FFE = greater than maximum
		0x7FFF = error
Bit 47 – 49	ROI_Grp_reference_domain Indicates the reference domain value for the ROI group. Please refer to ifm vision assistant for details. Values:	
	0x0	All values independent
	0x1	X
	0x2	Y
	0x3	Z
	0x4	Amplitude
Bit 50 – 55	ROI_upper_left_x Indicates the pixel column of the upper left corner for 2D ROIs. The columns are counted from 0 to 63 with 0 being the left-most column. For 3D ROIs, the value is 0.	

ROI_Param_mode = 2	ROI configuration values – Part 2													
Bit 2 – 7	ROI_number Indicates which ROI the values in the message belong to. This signal is repeated in each layout message.													
Bit 8 – 9	ROI_Grp_output_value_type Type of the output value for the ROI group. Values: <table><tr><td>0x0</td><td>Minimum</td></tr><tr><td>0x1</td><td>Maximum</td></tr><tr><td>0x2</td><td>Mean</td></tr><tr><td>0x3</td><td>Percentile</td></tr></table>		0x0	Minimum	0x1	Maximum	0x2	Mean	0x3	Percentile				
0x0	Minimum													
0x1	Maximum													
0x2	Mean													
0x3	Percentile													
Bit 10 – 15	ROI_lower_right_x Indicates the pixel column of the lower right corner for 2D ROIs. The columns are counted from 0 to 63 with 0 being the left-most column. For 3D ROIs, the value is 63.													
Bit 16 – 30	ROI_minimum_cartesian_y Indicates the minimum y-coordinate for 3D ROIs. For 2D ROIs, the value is -10000 <table><tr><td>Unit</td><td>cm</td></tr><tr><td>Range</td><td>[-10000 to 10000]</td></tr><tr><td>CAN to physical</td><td><math>P = C - 10000</math></td></tr><tr><td>Special values</td><td>0x7FFD = smaller than minimum</td></tr><tr><td></td><td>0x7FFE = greater than maximum</td></tr><tr><td></td><td>0x7FFF = error</td></tr></table>		Unit	cm	Range	[-10000 to 10000]	CAN to physical	$P = C - 10000$	Special values	0x7FFD = smaller than minimum		0x7FFE = greater than maximum		0x7FFF = error
Unit	cm													
Range	[-10000 to 10000]													
CAN to physical	$P = C - 10000$													
Special values	0x7FFD = smaller than minimum													
	0x7FFE = greater than maximum													
	0x7FFF = error													
Bit 32 – 46	ROI_maximum_cartesian_y Indicates the maximum y-coordinate for 3D ROIs. For 2D ROIs, the value is 10000 <table><tr><td>Unit</td><td>cm</td></tr><tr><td>Range</td><td>[-10000 to 10000]</td></tr><tr><td>CAN to physical</td><td><math>P = C - 10000</math></td></tr></table>		Unit	cm	Range	[-10000 to 10000]	CAN to physical	$P = C - 10000$						
Unit	cm													
Range	[-10000 to 10000]													
CAN to physical	$P = C - 10000$													



	Special values	0x7FFD = smaller than minimum
		0x7FFE = greater than maximum
		0x7FFF = error

ROI_Param_mode = 3	ROI configuration values – Part 3	
Bit 2 – 7	ROI_number Indicates which ROI the values in the message belong to. This signal is repeated in each layout message.	
Bit 8 – 11	ROI_lower_right_y Indicates the pixel row of the lower right corner for 2D ROIs. The rows are counted from 0 to 15 with 0 being the top-most row. For 3D ROIs, the value is 15.	
Bit 12 – 15	ROI_upper_left_y Indicates the pixel row of the upper left corner for 2D ROIs. The rows are counted from 0 to 15 with 0 being the top-most row. For 3D ROIs, the value is 0.	
Bit 16 – 30	ROI_minimum_cartesian_z Indicates the minimum z-coordinate for 3D ROIs. For 2D ROIs, the value is -10000	
	Unit	cm
	Range	[-10000 to 10000]
	CAN to physical	P = C – 10000
	Special values	0x7FFD = smaller than minimum
		0x7FFE = greater than maximum
		0x7FFF = error
Bit 32 – 46	ROI_maximum_cartesian_z Indicates the maximum z-coordinate for 3D ROIs. For 2D ROIs, the value is 10000	
	Unit	cm
	Range	[-10000 to 10000]
	CAN to physical	P = C – 10000
	Special values	0x7FFD = smaller than minimum
		0x7FFE = greater than maximum
		0x7FFF = error

## 2.4.2. Basic function results

The results of the basic function feature are output on CAN for each defined ROI group.

The output values may include x, y, z, and amplitude values, depending on the selected OutputMode value. Like the ROIs and ROI groups themselves, the output mode is also configured using ifm vision assistant.

To reduce load when not all outputs are selected, the output messages are not arranged by ROI group but by output value. This way, when only one output value is configured, the other messages need not be sent.

On the other hand, if all output values for the same ROI group are needed, it is necessary to receive all messages and combine the matching values.

Whenever values from several messages are combined, it is important that all values are from the same O3M cycle. Otherwise, the x, y, and z values might not even belong to the same pixel.

For this end, every message has a message counter signal (Bits 62 – 63). When 2 newly received messages have the same message counter value, they belong to the same O3M cycle and the information from both messages can be combined.



It is best to buffer the last two received messages of each type and compare the message counters to match them up before combining the information. Here are some examples:

Example 1:

	Latest message counter	Previous message counter
Message1	2	1
Message2	2	1
Action: Use latest for both		

Example 2:

	Latest message counter	Previous message counter
Message1	2	1
Message2	3	2
Action: Use message with message counter 2 for both. Keep latest Message2 data for next time.		

Example 3:

	Latest message counter	Previous message counter
Message1	2	1
Message2	0	3
This can't happen. Messages are getting lost in this case. It is necessary to improve the receiver.		

Example 4:

	Latest message counter	Previous message counter
Message1	0	3
Message2	0	3
Action: Use latest for both. The message counter goes from 3 directly back to 0.		

When information from 3 or more messages has to be combined, there can only be 2 different message counter values in all messages, because they can be from at most 2 different O3M cycles. Therefore, it is always sufficient to save the last 2 received data for all messages. To combine the information, use the message counter value that is available for all messages.

Every message contains result values of 4 ROI groups. If not all ROI groups are defined, the values will be invalid. Please don't use the output values unless the ROI groups are defined.

Messages for which no ROI groups are defined will not be output by O3M.

For example, when there are 5 ROI groups defined and output mode is set to 3 (x,y,z), the following messages are sent:

Message	ID	Content
BF_x_output_0	0x4FF10EFx	X value for ROI groups 1 through 4
BF_y_output_0	0x4FF11EFx	Y value for ROI groups 1 through 4
BF_z_output_0	0x4FF12EFx	Z value for ROI groups 1 through 4
BF_x_output_1	0x4FF14EFx	X value for ROI group 5 (other bits invalid)
BF_y_output_1	0x4FF15EFx	Y value for ROI group 5 (other bits invalid)
BF_z_output_1	0x4FF16EFx	Z value for ROI group 5 (other bits invalid)

Here is a complete list off all available messages and ID's:

Groups	X	Y	Z	Amplitude
Groups 1 – 4	0x4FF10EFx	0x4FF11EFx	0x4FF12EFx	0x4FF13EFx



Groups 5 – 8	0x4FF14EFx	0x4FF15EFx	0x4FF16EFx	0x4FF17EFx
Groups 9 – 12	0x4FF18EFx	0x4FF19EFx	0x4FF1AEFx	0x4FF1BEFx
Groups 13 – 16	0x4FF1CEFx	0x4FF1DEFx	0x4FF1EEFx	0x4FF1FEFx
Groups 17 – 20	0x4FF20EFx	0x4FF21EFx	0x4FF22EFx	0x4FF23EFx
Groups 21 – 24	0x4FF24EFx	0x4FF25EFx	0x4FF26EFx	0x4FF27EFx
Groups 25 – 28	0x4FF28EFx	0x4FF29EFx	0x4FF2AEFx	0x4FF2BEFx
Groups 29 – 32	0x4FF2CEFx	0x4FF2DEFx	0x4FF2EEFx	0x4FF2FEFx
Groups 33 – 36	0x4FF30EFx	0x4FF31EFx	0x4FF32EFx	0x4FF33EFx
Groups 37 – 40	0x4FF34EFx	0x4FF35EFx	0x4FF36EFx	0x4FF37EFx
Groups 41 – 44	0x4FF38EFx	0x4FF39EFx	0x4FF3AEFx	0x4FF3BEFx
Groups 45 – 48	0x4FF3CEFx	0x4FF3DEFx	0x4FF3EEFx	0x4FF3FEFx
Groups 49 – 52	0x4FF40EFx	0x4FF41EFx	0x4FF42EFx	0x4FF43EFx
Groups 53 – 56	0x4FF44EFx	0x4FF45EFx	0x4FF46EFx	0x4FF47EFx
Groups 57 – 60	0x4FF48EFx	0x4FF49EFx	0x4FF4AEFx	0x4FF4BEFx
Groups 61 – 64	0x4FF4CEFx	0x4FF4DEFx	0x4FF4EEFx	0x4FF4FEFx

The message layout is the same for all messages:

Bits 0 – 14	Signal for first group in message
Bits 15 – 29	Signal for second group in message
Bits 32 – 46	Signal for third group in message
Bits 47 – 61	Signal for fourth group in message
Bits 62 – 63	Message Counter Signal

To obtain the physical value from the CAN value, use the following formulas:

Value type	Formula	Special values
X (unit is meters)	$P = 0.01 * C - 100$	0x7FFD = value smaller than -100m 0x7FFE = value greater than 100m 0x7FFF = error
Y (unit is meters)	$P = 0.01 * C - 100$	0x7FFD = value smaller than -100m 0x7FFE = value greater than 100m 0x7FFF = error
Z (unit is meters)	$P = 0.01 * C - 100$	0x7FFD = value smaller than -100m 0x7FFE = value greater than 100m 0x7FFF = error
Amplitude	$P = 2 * C$	0x0 = error

## 2.5. Messages exclusive to OD variant

On O3MXX1, it is possible to install a firmware optimized for detecting objects and reflectors. This software variant also includes a function for predicting crash situations called the Crash Predictor (CP).

The OD software is capable of detecting and tracking up to 20 objects and reflectors. Each object has a unique identifier by which it is tracked. The objects are arranged by relevant meaning that the object that is most likely to interact with the defined vehicle is in index 0. When the mode is set to “reflector detection”, reflector objects are sorted before regular objects in the array. The number of objects which are output on CAN as well as the definition of the vehicle and the parameterization of the crash predictor function can be defined using ifm Vision Assistant.

The O3M uses a 2D-line model of the object's nearest edge to estimate object position. Therefore, an object position is defined using two points (x1,y1) and (x2,y2) and the minimum and maximum extent of the object (zMin and zMax). Please refer to the ifm Vision Assistant for details on object definition.



The results of the crash predictor function is output on CAN using the CAN message  
Crash\_Predictor\_Info (ID 0x4FF02EFx):

Name (.dbc-file)	Crash_Predictor_Info			
ID	0x4FF02EFx			
DLC	8			
Payload	Signal name	unit	CAN to physical conversion	Description
Bit 0 – 7	CP_object_id	-	P = C	ID of the object which is predicted to cause the crash
Bit 8 – 15	CP_ttc	s	P = 0.04*C	Predicted time to collision. Only available in intelligent mode. See ifm Vision Assistant documentation for details.
Bit 16 – 24	CP_impact_velocity	m/s	P = 0.1*C	Predicted impact velocity
Bit 25 – 27	CP_crash_predicted	enum	P = C -2	Information about whether or not a crash is predicted: 0x-2 = crash predictor is disabled 0x-1 = crash predictor is not available (after crash) 0x0 = crash predictor is enabled; no crash is predicted 0x1 = crash predicted 0x2 = error
Bit 28 – 29	CP_criticality	enum	P = C	Describes the criticality of a predicted crash detected by the intelligent/dynamic crash predictor. The values describe in which 'time zone' is the object, according to the time distance parameters. 0x0 = No crash predicted 0x1 = crash predicted for time zone 1 0x2 = crash predicted for time zone 2 0x3 = crash predicted for time zone 3
Bit 30	CP_Minimum_zone_triggered	-	P = C	For zone based mode: Describes the minimum value of all zones in which an object is detected. For example, if objects are detected in zones 1 and 3, the value will be 1. For intelligent/dynamic crash prediction mode: Indicates objects are detected in the minimum zone, means there are objects standing closer than minimum allowed distance. The minimum zone must be distinct from zone of the zone-based crash prediction, it's different definition. See ifm Vision Assistant documentation about the parameter "minimum allowed distance" for details.
Bit 32 – 39	CP_obj_ID_zone1	-	P = C	ID of object detected in zone 1. Only valid if there is an object in that zone (signal CP_Zone_1 P=1 or C = 2).
Bit 40 – 47	CP_obj_ID_zone2	-	P = C	ID of object detected in zone 2. Only valid if there is an object in that zone (signal CP_Zone_2 P=1 or C = 2).
Bit 48 – 55	CP_obj_ID_zone3	-	P = C	ID of object detected in zone 3. Only valid if there is an object in that zone (signal



				CP_Zone_3 P=1 or C = 2).
Bit 56 – 57	CP_Zone_1	-	P = C - 1	0x0 = Invalid, zone not defined 0x1 = Zone not occupied 0x2 = Zone is occupied 0x3 = reserved
Bit 58 – 59	CP_Zone_2	-	P = C - 1	0x0 = Invalid, zone not defined 0x1 = Zone not occupied 0x2 = Zone is occupied 0x3 = reserved
Bit 60 – 61	CP_Zone_3	-	P = C - 1	0x0 = Invalid, zone not defined 0x1 = Zone not occupied 0x2 = Zone is occupied 0x3 = reserved
Bit 62 – 63	Crash_Predictor_Info_cnt	-	P = C	Message counter. Range is 0..3. Use this value to combine the messages for each O3M cycle. Please refer to “Basic Function Result” section for details.

O3M sends out the detailed information for up to 20 objects. The number of objects which are output on CAN is configured using ifm Vision Assistant.

The information for each object is divided into two CAN messages (A and B). It is important to combine only messages with matching message counters as described in the “Basic Function Result” section because since the objects are ordered by criticality, the ordering of messages can change with each cycle causing the object in an array index to change from one cycle to the next. Even if both messages describe the same object, it is not desirable to combine information of different age again making it necessary to be very careful about message re-combination.

The message IDs are as follows:

Object index	ID message A	ID message B
0	0x4FF10EFx	0x4FF11EFx
1	0x4FF12EFx	0x4FF13EFx
2	0x4FF14EFx	0x4FF15EFx
3	0x4FF16EFx	0x4FF17EFx
4	0x4FF18EFx	0x4FF19EFx
5	0x4FF1AEFx	0x4FF1BEFx
6	0x4FF1CEFx	0x4FF1DEFx
7	0x4FF1EEFx	0x4FF1FEFx
8	0x4FF20EFx	0x4FF21EFx
9	0x4FF22EFx	0x4FF23EFx
10	0x4FF24EFx	0x4FF25EFx
11	0x4FF26EFx	0x4FF27EFx
12	0x4FF28EFx	0x4FF29EFx
13	0x4FF2AEFx	0x4FF2BEFx
14	0x4FF2CEFx	0x4FF2DEFx
15	0x4FF2EEFx	0x4FF2FEFx
16	0x4FF30EFx	0x4FF31EFx
17	0x4FF32EFx	0x4FF33EFx
18	0x4FF34EFx	0x4FF35EFx
19	0x4FF36EFx	0x4FF37EFx

After combining both messages, all relevant information about the object is available.

Information from A-type message (\* stands for the index of the object):



Payload	Signal name	unit	CAN to physical conversion	Description
Bit 0 – 6	Obj_*_vx	m/s	P = C*0.5 - 30	Relative velocity of the object along the x-axis. Special values: 0x7D = less than -30 m/s 0x7E = more than 30 m/s 0x7F = error
Bit 7	Obj_*_Type	enum	P = C	Type identifier. 0x0 = regular object 0x1 = retroreflector
Bit 8 – 14	Obj_*_vy	m/s	P = C*0.5 - 30	Relative velocity of the object along the y-axis. Special values: 0x7D = less than -30 m/s 0x7E = more than 30 m/s 0x7F = error
Bit 15	Obj_*_Measured	enum	P = C	Flag indicating if the object was measured (or extrapolated) in the current frame. 0x0 = extrapolated 0x1 = measured
Bit 16 – 20	Obj_*_ay	m/s <sup>2</sup>	P = C – 10	Relative acceleration of the object along the y-axis. Special values: 0x1D = Less than -10 m/s <sup>2</sup> 0x1E = Greater than 10 m/s <sup>2</sup> 0x1F = Error
Bit 21 – 23	Obj_*_ep	enum	P = C	Existence probability, provided as an enumeration. Higher values indicate a higher confidence about the existence of the object. 0x0: [0.00..0.25]% 0x1: [0.25..0.50]% 0x2: [0.50..0.75]% 0x3: [0.75..0.85]% 0x4: [0.85..0.90]% 0x5: [0.90..0.95]% 0x6: [0.95..1.0]%
Bit 24 – 28	Obj_*_ax			Relative acceleration of the object along the x-axis. Special values: 0x1D = Less than -10 m/s <sup>2</sup> 0x1E = Greater than 10 m/s <sup>2</sup> 0x1F = Error
Bit 29 – 31	Obj_*_qvx	enum	P = C	Quality of velocity along the x-axis, provided as an enumeration. Higher values indicate a higher confidence about the velocity value. 0x0: [0.00..0.25]% 0x1: [0.25..0.50]% 0x2: [0.50..0.75]% 0x3: [0.75..0.85]% 0x4: [0.85..0.90]% 0x5: [0.90..0.95]% 0x6: [0.95..1.0]%
Bit 32 – 35	Obj_*_az	m/s <sup>2</sup>	P = C – 5	Relative acceleration of the object along the z-axis. Special values: 0xD = Less than -5 m/s <sup>2</sup> 0xE = Greater than 5 m/s <sup>2</sup> 0xF = Error
Bit 36 – 37	Obj_*_TrackAge	enum	P = C	Number of frames for which the object has been tracked by O3M. 0x0 = [0..2] frames 0x1 = [3..12] frames



				0x2 = [13..25] frames 0x3 = more than 25 frames
Bit 38 – 45	Obj_*_ld	-	P = C	Unique ID of the object. 0 indicates, that no object was detected. Valid values range from 1 to 255. Use this value to recognize the same object in the following frames.
Bit 46 – 56	Obj_*_zMin	m	P = 0.02*C-10	Minimum z-coordinate of the object. Special values: 0x7FD = less than -10 m 0x7FE = greater than 30 m 0x7FF = error
Bit 57 – 61	Obj_*_vz	m/s	P = C*0.5 – 6	Relative velocity of the object along the z-axis Special values: 0x1D = less than -6 m/s 0x1E = more than 6 m/s 0x1F = error
Bit 62 – 63	Obj_*_A_cnt	-	P = C	Message counter. Range is 0..3. Use this value to combine the messages for each O3M cycle. Please refer to “Basic Function Result” section for details.

Information from B-type message:

Payload	Signal name	unit	CAN to physical conversion	Description
Bit 0 – 7	Obj_*_dz	m	P = 0.02*C	Height of the object (zMax = zMin + dz). Special values: 0xFD = less than 0 0xFE = more than 5m 0xFF = error
Bit 8 – 19	Obj_*_dy	m	P = 0.02*C – 40	Distance between both object endpoints along the y-axis: y2 = y1 + dy Special values: 0xFFD = less than -40 m 0xFFE = more than 40 m 0xFFF = error
Bit 20 – 31	Obj_*_dx	m	P = 0.02*C – 40	Distance between both object endpoints along the x-axis: x2 = x1 + dx Special values: 0xFFD = less than -40 m 0xFFE = more than 40 m 0xFFF = error
Bit 32 – 44	Obj_*_x1	m	P = 0.02*C – 80	x-position of the first object corner coordinate. Special values: 0x1FFD = less than -80 m 0x1FFE = more than 80 m 0x1FFF = error
Bit 45 – 57	Obj_*_y1	m	P = 0.02*C – 80	y-position of the first object corner coordinate. Special values: 0x1FFD = less than -80 m 0x1FFE = more than 80 m 0x1FFF = error
Bit 58 – 60	Obj_*_qvy	enum	P = C	Quality of velocity along the y-axis, provided as an enumeration. Higher values indicate a higher confidence about the velocity value. 0x0: [0.00..0.25]% 0x1: [0.25..0.50]% 0x2: [0.50..0.75]% 0x3: [0.75..0.85]%





				0x4: ]0.85..0.90]% 0x5: ]0.90..0.95]% 0x6: ]0.95..1.0]%
Bit 61	Obj_*_History	-	P = C	Since there are only 255 different object ID's, at some point O3M has to start re-using the same unique identifier for a new object. Whenever this happens, the history bit is changed. Therefore, if an object with ID 0x10 and hist = 0 has been seen and was lost, at some later time, a new object will be given ID 0x10 and hist = 1. At that time, the track for the old object is discarded.
Bit 62 – 63	Obj_*_B_cnt	-	P = C	Message counter. Range is 0..3. Use this value to combine the messages for each O3M cycle. Please refer to “Basic Function Result” section for details.

## 2.6. Messages exclusive to LG variant

On O3MXX1, it is possible to install a firmware optimized for detecting crop edges and longish heaps on the ground. Both objects are referred to as “lines” by O3M. Therefore, both names “heap” and “line” are used to describe the detected object in the CAN interface. Please refer to the O3M documentation for details about this feature.

The line guidance (LG) firmware is capable of detecting and tracking up to 5 heaps. With default settings, only 3 lines are used. But the communication on CAN interface is designed for up to 5 heaps, which is why the following description includes all 5 heaps, even so the sensor output is currently limited to 3.

Again, it is important to note that only messages from the same O3M cycle should be combined to generate the heap information. For a mechanism to combine the received messages, please refer to the chapter “Basic function results”.

### Reference point:

All positions in line guidance output are relative to a line guidance reference point which O3M generates dynamically. Therefore, the most important message is the message containing the reference point information because all other positions are relative to that.

Name (.dbc-file)	Reference_point_information		
ID	0x4FF02EFx		
DLC	5		
Payload	Bit 0 – 10	Reference point X-Value, Unit m (meters)	
	Bit 11 – 21	Reference point Y-Value, Unit m (meters)	
	Bit 22 – 30	Reference point Z-Value, Unit m (meters)	
	Bit 38 – 39	Message counter. Use this value to combine the messages	
CAN to Physical value conversion	X- and Y -Value: P = C*0.01 – 10 Z- Value: P = C*0.01 – 2		
Special values	X- and Y-Value: 0x7FF = Error 0x7FE = Value greater than 10 m 0x7FD = Value smaller than -10 m  Z-Value: 0x1FF = Error 0x1FE = Value greater than 2 m 0x1FD = Value smaller than -2 m		



## Heaps:

The properties of the heaps detected by O3M are output using 2 messages per heap. Please use the mechanism described in the chapter “Basic function results” to combine both messages, because it is very important that the information in both messages belongs to the same measurement when interpreting the line position.

To describe the exact properties of the heap, the following signals are used:

Name	CAN to Physical value conversion	Description	Special Values
Line_Id	$P = C$	Unique identifier of the line	
Line_alpha	$P = (0.1 * C - 25) * \pi / 180$	Orientation of the center line of the line in radians	0x1FD = smaller than -25/180* $\pi$ 0x1FE = greater than 25/180* $\pi$ 0x1FF = error
Line_curvature	$P = 0.0005 * C - 0.05$	Radius of curvature of the heap's center line. Unit is 1/m. 0 indicates a straight line, high absolute values indicate a tightly curved line. Positive values indicate a curvature direction to the left, negative values to the right.	0xFD = smaller than -0.05 0xFE = greater than 0.05 0xFF = error
Line_quality	$P = C$	Quality of the center line detection. Values range from 0 to 100. Higher values indicated a better quality.	0x7D = smaller than 0 0x7E = greater than 100 0x7F = error
Line_zStepDetectionHeight	$P = 0.01 * C$	Height of the cut edge	0x1D = smaller than 0 0x1E = greater than 5 0x1F = error
Line_zStepDetectionHeightValid	-	Valid flag for z step detection height signal	-
Line_foresight	$P = C$	Foresight range of the detection. Unit is m	0x1D = smaller than 0 0x1E = greater than 20 0x1F = error
Line_offset	$P = 0.01 * C - 10$	Offset of the heap's center line from the reference point in y-direction. Unit is m.	0x7FD = smaller than -10 0x7FE = greater than 10 0x7FF = error
Line_Type	-	Type of the line 0: heap 1: cut-edge	-
Line_Measured	-	Flag indicating that the line has been measured (rather than extrapolated) in the current frame.	-
Line_History	-	Toggle bit for newly created line with same id. Whenever a new line gets the same unique identifier as one that was detected previously, this bit is changed to indicate that the previous line is now discarded.	-



Line_centerOfGravityOffset	$P = 0.05 * C - 5$	Lateral offset of heap's center of gravity to heap's center line. Unit is m.	0xFD = smaller than -5 0xFE = greater than 5 0xFF = error
Line_centerOfGravityOffsetValid	-	Valid flag for center of gravity offset signal	-
Line_maxHeightOffset	$P = 0.05 * C - 5$	Lateral offset of maximum height position to center line of heap. Unit is m.	0xFD = smaller than -5 0xFE = greater than 5 0xFF = error
Line_maxHeightOffsetValid	-	Valid flag for max height offset signal	-
Line_heapHeight	$P = 0.01 * C$	Height of detected heap. Unit is m.	0x1FD = smaller than 0 0x1FE = greater than 5 0x1FF = error
Line_heapHeightValid	-	Valid flag for heap height signal	-
Line_heapWidth	$P = 0.01 * C$	Width of detected heap. Unit is m.	0x3FD = smaller than 0 0x3FE = greater than 10 0x3FF = error
Line_heapWidthValid	-	Valid flag for heap with signal	-
Line_heapArea	$P = 0.01 * C$	Area on yz-plane covered by heap. Unit is m <sup>2</sup> .	0x3FD = smaller than 0 0x3FE = greater than 10 0x3FF = error
Line_heapAreaValid	-	Valid flag for heap area signal	-

The properties of each heap are divided up into two messages using the following layout:

Message\_Line\_A (DLC = 8):

Bit position	Signal
Bit 0 – 6	Line_Id
Bit 7 – 15	Line_alpha
Bit 16 – 23	Line_curvature
Bit 24 – 30	Line_quality
Bit 31 – 39	Line_zStepDetectionHeight
Bit 40 – 44	Line_foresight
Bit 45 – 55	Line_offset
Bit 56	Line_Type
Bit 57	Line_Measured
Bit 58	Line_zStepDetectionHeightValid
Bit 59	Line_History
Bit 62 – 63	Message counter. Range is 0..3. Use this value to combine the messages for each O3M cycle. Please refer to “Basic Function Result” section for details.

Message\_Line\_B (DLC = 7):

Bit position	Signal
Bit 0 – 7	Line_centerOfGravityOffset
Bit 8 – 15	Line_maxHeightOffset
Bit 16 – 24	Line_heapHeight
Bit 25 – 34	Line_heapWidth



Bit 35 – 44	Line_heapArea
Bit 45	Line_maxHeightOffsetValid
Bit 46	Line_heapHeightValid
Bit 47	Line_heapWidthValid
Bit 48	Line_heapAreaValid
Bit 49	Line_centerOfGravityOffsetValid
Bit 54 – 55	Message counter. Range is 0..3. Use this value to combine the messages for each O3M cycle. Please refer to “Basic Function Result” section for details.

The message ID's of the two messages for each heap are the following:

Line Number	ID of Message A	ID of Message B
Line 0	0x4FF10EFx	0x4FF11EFx
Line 1	0x4FF12EFx	0x4FF13EFx
Line 2	0x4FF14EFx	0x4FF15EFx
Line 3	0x4FF16EFx	0x4FF17EFx
Line 4	0x4FF18EFx	0x4FF19EFx

### Curvature Command:

In addition to the properties of the detected heaps / crop edges, O3M also outputs the curvature command necessary to follow each line. The curvature command indicates the radius necessary to follow the selected line. It can be considered an additional heap property, but since it's output using a separate CAN message, it will be described as a separate object here.

Name	CAN to Physical value conversion	Description	Special Values
Curvature_Command	$P = C * 0.025 - 803.2$	Curvature radius necessary to follow the line. Unit is 1/km	0xFFFC = Curvature command not available 0xFFFD = smaller than -803.2 0xFFFE = greater than 803.2 0xFFFF = error

The curvature command signals for all 5 heaps are output using 5 messages. If less than 5 heaps are output, the curvature command messages for the unused heaps are not sent by O3M.

The curvature command messages have the following layout:

Name (.dbc-file)	MoCa_Curvature_A		
ID	0x4FF20EFx		
DLC	7		
Payload	Bit 0 – 15	Curvature_Command_0, Curvature command for line 0	
	Bit 16 – 22	Line_0_Id, Unique ID of line 0	
	Bit 24 – 30	Line_1_Id, Unique ID of line 1	
	Bit 32 – 47	Curvature_Command_1, Curvature command for line 1	
	Bit 54 – 55	Message counter; cycles through the values 0, 1, 2, 3. Changes every time the message is sent. Please use this counter to combine the message with other O3M output messages	

Name (.dbc-file)	MoCa_Curvature_B		
ID	0x4FF21EFx		
DLC	7		
Payload	Bit 0 – 15	Curvature_Command_2, Curvature command for line 2	
	Bit 16 – 22	Line_2_Id, Unique ID of line 2	



	Bit 24 – 30	Line_3_Id, Unique ID of line 3
	Bit 32 – 47	Curvature_Command_3, Curvature command for line 3
	Bit 54 – 55	Message counter; cycles through the values 0, 1, 2, 3. Changes every time the message is sent. Please use this counter to combine the message with other O3M output messages

Name (.dbc-file)	MoCa_Curvature_C	
ID	0x4FF22EFx	
DLC	7	
Payload	Bit 0 – 15	Curvature_Command_4, Curvature command for line 4
	Bit 16 – 22	Line_4_Id, Unique ID of line 4
	Bit 24 – 30	Reserved
	Bit 32 – 47	Reserved
	Bit 54 – 55	Message counter; cycles through the values 0, 1, 2, 3. Changes every time the message is sent. Please use this counter to combine the message with other O3M output messages



## 3. Input Messages

### 3.1. Messages common to all O3M variants

#### 3.1.1. Ego data

In OD and LG modes, O3M can use information on vehicle ego data to improve the quality of detected object motion and line estimates.

Use of ego data must be configured in O3M using ifm vision assistant. There are 3 ego data modes:

1. No use of ego data (default)
2. Use velocity from CAN only
3. Use velocity and yaw rate from CAN

To use ego data mode 2 or 3, it is necessary to provide the necessary input information on CAN. Velocity and yaw rate are available on many vehicles and can be forwarded to O3M using a gateway. If only steering angle is available, a yaw rate estimate can be calculated from the steering angle and the vehicle dimensions using geometric relations.

Velocity and yaw rate are sent to O3M using standard J1939 messages.

To send velocity information to O3M, two J1939 standard CAN messages are needed: The message EBS21 (ID 0x1803FFEx) containing the signal Wheel\_BasedVehicleSpeed, which is used to transfer the current speed of the vehicle, and the message TCO1 (ID 0xCFE6CFEx) containing the signal DirectionIndicator, which indicates whether the driving direction is forward or backward.

Both messages should be sent as often as new values are available.

Name (.dbc-file)	EBS21	
ID	0x1803FFEx	
DLC	8	
Payload	Bit 16 – 31	Wheel_BasedVehicleSpeed Unit: km/h Conversion: $C = P * 256$ (this means there is a resolution of 8 bits per km/h or 0.0039 km/h) Value range: [0 .. 251] Description according to J1939: <i>Actual speed of the vehicle (positive value for forward and backward speed) calculated as the average of the wheel speeds of one axle influenced by slip and filtered by a frequency range of 5 Hz to 20 Hz.</i> Actual speed of the vehicle (positive value for forward and backward speed) calculated as the average of the wheel speeds of one axle influenced by slip and filtered by a frequency range of 5 Hz to 20 Hz.

Name (.dbc-file)	TCO1	
ID	0xCFE6CFEx	
DLC	8	
Payload	Bit 30 – 31	DirectionIndicator Value range: [0 .. 3] Meanings: 0x0 = forward 0x1 = backward 0x2 = error 0x3 = not available



		Description according to J1939: <i>Indicates the direction of the vehicle.</i>
--	--	--

To send yaw rate information to O3M, the yaw rate signal of the J1939 standard CAN message VDC2 (ID 0x18F009FE) is used. Again, the message should be sent as often as new values are available.

Name (.dbc-file)	VDC2	
ID	0x18F009FEx	
DLC	8	
Payload	Bit 24 – 39	YawRate Unit: rad/s Conversion: $C = (P + 3.92) * 8192$ Please be sure to use the unit rad/s. To convert from deg/s to rad/s, divide by 180 and multiply by $\pi$ . Value range: [-3.92 .. 3.92] Description according to J1939: <i>Indicates the rotation about the vertical axis.</i>

### 3.1.2. Virtual switches

O3M has the ability to perform simple logic operations using result values as well as values input over CAN. For details on setting up the logic, please refer to ifm Vision Assistant.

The message VS\_Mixed\_Input is used to send values over CAN to O3M to use as inputs to the logic. It uses two different message layouts. In the first layout, 4 analog inputs and 14 digital inputs are available; the second layout contains an additional 2 analog inputs. Both layouts can be used alternately, but care should be taken to leave at least 5 ms time between any two consecutive messages.

All digital inputs are encoded as single bits in the CAN message. All analog outputs are encoded using 12 bits. The values 0x000 through 0xFA0 are used to encode the values 0 to 1 with a resolution of 1/4000 per bit. 0xFFF indicates that the signal is not available. All values from 0xFA1 to 0xFFE are reserved and should not be used.

Name (.dbc-file)	VS_MixedInput															
ID	0x4EFEFFEx															
DLC	8															
Payload	<p>Bit 0 – 1 is the input ID selector. If this signal has the value “0”, the analog inputs 0 through 3 and digital inputs 0 through 13 are expected in the message. If the value is “1”, the analog inputs 4 through 5 are expected.</p> <p><b>Input ID selector 0:</b></p> <table><tr><td>Bit 2 – 13</td><td>Analog input 00</td></tr><tr><td>Bit 14 – 25</td><td>Analog input 01</td></tr><tr><td>Bit 26 – 37</td><td>Analog input 02</td></tr><tr><td>Bit 38 – 49</td><td>Analog input 03</td></tr><tr><td>Bit 50 – 63</td><td>Digital input 0 through 13, one digital input per bit</td></tr></table> <p><b>Input ID selector 1:</b></p> <table><tr><td>Bit 2 – 13</td><td>Analog input 04</td></tr><tr><td>Bit 14 – 25</td><td>Analog input 05</td></tr></table>		Bit 2 – 13	Analog input 00	Bit 14 – 25	Analog input 01	Bit 26 – 37	Analog input 02	Bit 38 – 49	Analog input 03	Bit 50 – 63	Digital input 0 through 13, one digital input per bit	Bit 2 – 13	Analog input 04	Bit 14 – 25	Analog input 05
Bit 2 – 13	Analog input 00															
Bit 14 – 25	Analog input 01															
Bit 26 – 37	Analog input 02															
Bit 38 – 49	Analog input 03															
Bit 50 – 63	Digital input 0 through 13, one digital input per bit															
Bit 2 – 13	Analog input 04															
Bit 14 – 25	Analog input 05															

### 3.1.3. A note on the UDS transfer protocol

To send long messages to O3M, the UDS transfer protocol as defined in ISO14229 is used. A complete description of the UDS transfer protocol cannot be accomplished here. Therefore, this



section can only briefly describe what is necessary to send parameters to O3M. Please excuse the incompleteness and refer to the ISO14229 for more details.

Please always wait for the response messages sent by O3M before sending the next message to make sure that all commands are processed.

### Message IDs

The UDS protocol uses 2 messages: One request message (RQST) sent to O3M and a response message (RSP) sent from O3M.

Message	ID
RQST	0x18DAEF1x
RSP	0x18DAF1EFx

### Message content

A UDS message consists of a 1 byte service identifier and payload. The payload length is limited to 4096 bytes. All parameters used by O3M are much shorter than that.

### Protocol

There are two different protocols used: A single message protocol with a payload of 6 bytes (plus service identifier) and a multiple message protocol with the longer payload.

#### Single message protocol

When the payload is 6 bytes or less, a single message transfer is used.

Request message (sent to O3M)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Data size in bytes including service identifier	Service identifier	Payload – Up to 6 bytes. The DLC should be adjusted to transfer only the bytes that are used.					

Response message (sent by O3M)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Data size in bytes	Service identifier of the request. If the response is positive, 0x40 is added.	Additional information (e.g. error code) If the last byte of information (byte as indicated by data size in byte0) has the value 0x78, O3M cannot complete the command at once. In this case, please wait for another message.					

#### Multiple message protocol

When the payload is more than 6 bytes, several messages must be used to transfer it. For this purpose, the first message sent to and received from O3M is used to establish the transfer and the consecutive messages are used to transfer the data.

First request message (sent to O3M)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x10 + upper byte of data size Since data size can be up to 4095 bytes, the upper byte of the data size can be 0xF at the most.	Lower byte of data size	Service identifier	First 5 bytes of data				





First response message, flow control (sent by O3M)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x30 means ok, continue sending. Other values are errors.	Timing information						

Please wait for the first response message before sending consecutive request messages and do not send request messages faster than with 5 ms spacing in between two messages.

Consecutive message (sent to O3M)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Part identifier. First consecutive message has 0x21. The next messages count up to 0x2F and wrap around to 0x20.	Up to 7 bytes of data						

A response message is sent from O3M only after the complete data has been received. The final response message is the same as for single message transfer.

The UDS transfer protocol is used for the following O3M features:

- 2D overlay input
- Online parametrization

Please refer to these topics for more details on their payload content.

### 3.1.4. 2D Overlay input

With O3M2xx, it is possible to send commands to draw certain geometric features called *primitives* as an overlay into the 2D video image. The UDS transfer protocol is used for these commands.

For each primitive, it is necessary to follow the following steps:

- Create the command necessary for O3M to draw the item
- Create the UDP message structure to send the command to O3M
- Handle the UDP communication

#### *General information*

Every primitive needs a primitive number. O3M manages all primitives by their primitive number. The primitive number is necessary to turn off a primitive. Therefore, it is advisable to use a different primitive number for each primitive.

Depending on the type of the primitive, various properties of the primitive must be set. The properties of the available primitive types are described below.

All positions are counted from the top left corner of the image (position (0/0)) with x as the column position, y as the row position.

All primitives use the **UDS service identifier 0x2E**.

#### *Icon primitive*

The icon primitive commands displays icons already in the O3M memory at a certain place. Every icon must be loaded to O3M using the ifm vision assistant tool in advance. Please refer to the ifm vision assistant for details on how to load an icon to O3M.



Properties of the icon primitive:

Name	Data type	Description
primitiveNumber	UInt8	Unique identifier of the primitive. Please use a different identifier for every primitive sent.
iconID	UInt16	ID of the icon image to be displayed. The ID is set by vision assistant when loading the icons
posX	UInt12	Column of the icon display in pixels
posY	UInt12	Row of the icon display in pixels

Command layout:

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte0	0xFD							
Byte1	0x23							
Byte2	primitiveNumber							
Byte3	iconID low byte							
Byte4	iconID high byte							
Byte5	posX, low 8 bits							
Byte6	posY, low 4 bits				posX, high 4 bits			
Byte7	posY, high 8 bits							

UDS command flow:

1. Request message to O3M (ID 0x18DAEFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x10	0x09	0x2E	Byte0 – Byte4 of command				

2. Response message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x30	Timing information (0x00, 0x0A, 0x78, 0x00, 0x00, 0x00, 0x00)						

3. Request message to O3M (ID 0x18DAEFF1x, DLC = 4)

Byte0	Byte1	Byte2	Byte3
0x21	Byte4 – Byte7 of command		

4. Response message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x03	0x6E	0xFD	0x23	0x00	0x00	0x00	0x00

The icon is now displayed on the O3M 2D image at the specified position.

### *Ellipse primitive*

The ellipse primitive commands displays an ellipses with specified edge and fill color at a specified position within the 2D image. To find out the desired settings for colors and positions, it is best to use the ifm vision assistant overlay editor.

Properties of the ellipse primitive:

Name	Data type	Description
primitiveNumber	UInt8	Unique identifier of the primitive. Please use a different identifier for every primitive sent.
LineWidth	UInt8	Width of the line of the ellipses (pixels)
LineColorRed	UInt4	Red color value of line (0x00 .. 0xFF)
LineColorGreen	UInt4	Green color value of line (0x00 .. 0xFF)
LineColorBlue	UInt4	Blue color value of line (0x00 .. 0xFF)
LineColorAlpha	UInt4	Alpha value of line (0x00 .. 0xFF). 0x00 is transparent, 0xFF is opaque.
FillColorRed	UInt4	Red color value of face (0x00 .. 0xFF)
FillColorGreen	UInt4	Green color value of face (0x00 .. 0xFF)



FillColorBlue	Uint4	Blue color value of face (0x00 .. 0xFF)
FillColorAlpha	Uint4	Alpha value of face (0x00 .. 0xFF). 0x00 is transparent, 0xFF is opaque.
posX_leftUpCorner	Uint12	Column position of upper left corner
posY_leftUpCorner	Uint12	Row position of upper left corner
posX_rightDownCorner	Uint12	Column position of lower right corner
posY_rightDownCorner	Uint12	Row position of lower right corner

Command layout:

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte0	0xFD							
Byte1	0x22							
Byte2	primitiveNumber							
Byte3	LineWidth							
Byte4	LineColorGreen				LineColorRed			
Byte5	LineColorAlpha				LineColorBlue			
Byte6	FillColorGreen				FillColorRed			
Byte7	FillColorAlpha				FillColorBlue			
Byte8	posX_leftUpCorner, low 8 bits							
Byte9	posY_leftUpCorner, low 4 bits				posX_leftUpCorner, high 4 bits			
Byte10	posY_leftUpCorner, high 8 bits							
Byte11	posX_rightDownCorner, low 8 bits							
Byte12	posY_rightDownCorner, low 4 bits				posX_rightDownCorner, high 4 bits			
Byte13	posY_rightDownCorner, high 8 bits							

UDS command flow:

1. Request message to O3M (ID 0x18DAEFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x10	0x0F	0x2E	Byte0 – Byte4 of command				

2. Response message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x30	Timing information (0x00, 0x0A, 0x78, 0x00, 0x00, 0x00, 0x00)						

3. Request message to O3M (ID 0x18DAEFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x21	Byte5 – Byte11 of command						

4. Request message to O3M (ID 0x18DAEFF1x, DLC = 3)

Byte0	Byte1	Byte2
0x22	Byte12 – Byte13 of command	

5. Response message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x03	0x6E	0xFD	0x22	0x00	0x00	0x00	0x00

The ellipse is now displayed on the O3M 2D image with the specified colors and styles and at the specified position.

### Polyline primitive

The polyline primitive commands displays a line with specified line color and up to 22 points (vertices) within the 2D image. To find out the desired settings for colors and positions, it is best to use the ifm vision assistant overlay editor.

To optimize performance, only the positions of the used points are transferred. Therefore, the size of the payload and the number of messages to send to O3M depends on the desired number of vertices of the line.



Properties of the polyline primitive:

Name	Data type	Description
primitiveNumber	UInt8	Unique identifier of the primitive. Please use a different identifier for every primitive sent.
LineWidth	UInt8	Width of the line of the ellipses (pixels)
NumberOfPoints	UInt8	Number of points (vertices) of the line. At least 2 are needed to make a line. Up to 22 (0x16) are supported.
LineColorRed	UInt4	Red color value of line (0x00 .. 0xFF)
LineColorGreen	UInt4	Green color value of line (0x00 .. 0xFF)
LineColorBlue	UInt4	Blue color value of line (0x00 .. 0xFF)
LineColorAlpha	UInt4	Alpha value of line (0x00 .. 0xFF). 0x00 is transparent, 0xFF is opaque.
posX_01	UInt12	Column position of the first point
posY_01	UInt12	Row position of the first point
...	UInt12	Column and row position of points 2 through 22

Command layout:

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte0	0xFD							
Byte1	0x20							
Byte2	primitiveNumber							
Byte3	LineWidth							
Byte4	NumberOfPoints							
Byte5	LineColorGreen				LineColorRed			
Byte6	LineColorAlpha				LineColorBlue			
Byte7	posX_01, low 8 bits							
Byte8	posY_01, low 4 bits				posX_01, high 4 bits			
Byte9	posY_01, high 8 bits							
Byte10	posX_02, low 8 bits							
Byte11	posY_02, low 4 bits				posY_02, low 4 bits			
Byte12	posY_02, high 8 bits							
...	3 bytes per point for the X and Y position as described for posX_01 and posY_01							

The total number of bytes of the command is  $7 + 3 * (\text{NumberOfPoints})$ .

UDS command flow (example for 10 points):

The number of bytes of the command is  $7 + 3 * 10 = 37$  (Byte0 through Byte36). 5 bytes are transferred in the first request message and up to 7 in every consecutive message. Therefore, a total of 6 request messages is needed. The payload of the last message is  $37 - 5 - 4 * 7 = 4$ , the DLC is therefore 5 because the first byte is needed for the part identifier.

If more request messages are needed, please increment the part identifier for every message.

The O3M response will be sent by O3M after the last request message has been completed.

1. Request message to O3M (ID 0x18DAFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x10	0x26 (data size = 37 bytes payload + 1 byte service identifier)		0x2E	Byte0 – Byte4 of command			

2. Response message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x30	Timing information (0x00, 0x0A, 0x78, 0x00, 0x00, 0x00, 0x00)						

3. Request message to O3M (ID 0x18DAFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
-------	-------	-------	-------	-------	-------	-------	-------



	0x21	Byte5 – Byte11 of command						
4.	Request message to O3M (ID 0x18DAEFF1x, DLC = 8)							
	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
	0x22	Byte12 – Byte18 of command						
5.	Request message to O3M (ID 0x18DAEFF1x, DLC = 8)							
	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
	0x23	Byte19 – Byte25 of command						
6.	Request message to O3M (ID 0x18DAEFF1x, DLC = 8)							
	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
	0x24	Byte26 – Byte32 of command						
7.	Request message to O3M (ID 0x18DAEFF1x, DLC = 5)							
	Byte0	Byte1	Byte2	Byte3	Byte4			
	0x25	Byte33 – Byte36 of command						
8.	Response message from O3M (ID 0x18DAF1EFx)							
	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
	0x03	0x6E	0xFD	0x20	0x00	0x00	0x00	0x00

The line is now displayed on the O3M 2D image with the specified colors and styles and with the specified number of corners and corner positions.

#### Polygon primitive

The polygon primitive commands displays a filled or unfilled polygon with specified line color and up to 22 points (vertices) within the 2D image. To find out the desired settings for colors and positions, it is best to use the ifm vision assistant overlay editor.

To optimize performance, only the positions of the used points are transferred. Therefore, the size of the payload and the number of messages to send to O3M depends on the desired number of vertices of the polygon.

The polygon primitive is very similar to the polyline primitive. There are two major differences

1. The line is automatically closed to return to the first point after the last point.
2. The polygon is filled in the specified color.

Properties of the polygon primitive:

Name	Data type	Description
primitiveNumber	UInt8	Unique identifier of the primitive. Please use a different identifier for every primitive sent.
LineWidth	UInt8	Width of the line of the ellipses (pixels)
NumberOfPoints	UInt8	Number of points (vertices) of the line. At least 2 are needed to make a line. Up to 22 (0x16) are supported.
LineColorRed	UInt4	Red color value of line (0x00 .. 0xFF)
LineColorGreen	UInt4	Green color value of line (0x00 .. 0xFF)
LineColorBlue	UInt4	Blue color value of line (0x00 .. 0xFF)
LineColorAlpha	UInt4	Alpha value of line (0x00 .. 0xFF). 0x00 is transparent, 0xFF is opaque.
FillColorRed	UInt4	Red color value of inside (0x00 .. 0xFF)
FillColorGreen	UInt4	Green color value of inside (0x00 .. 0xFF)
FillColorBlue	UInt4	Blue color value of inside (0x00 .. 0xFF)
FillColorAlpha	UInt4	Alpha value of inside (0x00 .. 0xFF). 0x00 is transparent, 0xFF is opaque.
posX_01	UInt12	Column position of the first point
posY_01	UInt12	Row position of the first point
...	UInt12	Column and row position of points 2 through 22



Command layout:

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte0	0xFD							
Byte1	0x21							
Byte2	primitiveNumber							
Byte3	LineWidth							
Byte4	NumberOfPoints							
Byte5	LineColorGreen				LineColorRed			
Byte6	LineColorAlpha				LineColorBlue			
Byte7	FillColorGreen				FillColorRed			
Byte8	FillColorAlpha				FillColorBlue			
Byte9	posX_01, low 8 bits							
Byte10	posY_01, low 4 bits				posX_01, high 4 bits			
Byte11	posY_01, high 8 bits							
Byte12	posX_02, low 8 bits							
Byte13	posY_02, low 4 bits				posY_02, low 4 bits			
Byte14	posY_02, high 8 bits							
...	3 bytes per point for the X and Y position as described for posX_01 and posY_01							

The total number of bytes of the command is  $9 + 3 \cdot (\text{NumberOfPoints})$ .

UDS command flow (example for 4 points):

The number of bytes of the command is  $9 + 3 \cdot 4 = 19$  (Byte0 through Byte18). 5 bytes are transferred in the first request message and up to 7 in every consecutive message. Therefore, a total of 3 request messages is needed. The payload of the last message is  $19 - 5 - 1 \cdot 7 = 7$ , the DLC is therefore 8 because the first byte is needed for the part identifier.

If more request messages are needed, please increment the part identifier for every message.

The O3M response will be sent by O3M after the last request message has been completed.

1. Request message to O3M (ID 0x18DAEFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x10	0x14 (data size = 19 bytes payload + 1 byte service identifier)		0x2E	Byte0 – Byte4 of command			

2. Response message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x30	Timing information (0x00, 0x0A, 0x78, 0x00, 0x00, 0x00, 0x00)						

3. Request message to O3M (ID 0x18DAEFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x21	Byte5 – Byte11 of command						

4. Request message to O3M (ID 0x18DAEFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x22	Byte12 – Byte18 of command						

5. Response message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x03	0x6E	0xFD	0x21	0x00	0x00	0x00	0x00

The polygon is now displayed on the O3M 2D image with the specified colors and styles and with the specified number of corners and corner positions.

#### Text primitive

The text primitive commands displays a text polygon with specified text and background color and size within the 2D image. The text can have up to 40 characters encoded using utf-16 encoding. To find out the desired settings for colors and positions, it is best to use the ifm vision assistant overlay editor.



To optimize performance, only the used characters. Therefore, the size of the payload and the number of messages to send to O3M depends on the number of characters in the text.

Properties of the text primitive:

Name	Data type	Description
primitiveNumber	Uint8	Unique identifier of the primitive. Please use a different identifier for every primitive sent.
LineColorRed	Uint4	Red color value of text (0x00 .. 0xFF)
LineColorGreen	Uint4	Green color value of text (0x00 .. 0xFF)
LineColorBlue	Uint4	Blue color value of text (0x00 .. 0xFF)
LineColorAlpha	Uint4	Alpha value of text (0x00 .. 0xFF). 0x00 is transparent, 0xFF is opaque.
FillColorRed	Uint4	Red color value of background (0x00 .. 0xFF)
FillColorGreen	Uint4	Green color value of background (0x00 .. 0xFF)
FillColorBlue	Uint4	Blue color value of background (0x00 .. 0xFF)
FillColorAlpha	Uint4	Alpha value of background (0x00 .. 0xFF). 0x00 is transparent, 0xFF is opaque.
NumberOfChars	Uint8	Number of characters of the text (0x11 through 0x40)
FontSize	Uint8	Index of the font size. Usually, 1 through 4 are used.
TextPosX	Uint12	Column position of the text display
TextPosY	Uint12	Row position of the text display
TextChar01	Int16	Text character 1, utf-16 encoded
...	Int16	Additional text characters

Command layout:

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte0	0xFD							
Byte1	0x24							
Byte2	primitiveNumber							
Byte3	LineColorGreen				LineColorRed			
Byte4	LineColorAlpha				LineColorBlue			
Byte5	FillColorGreen				FillColorRed			
Byte6	FillColorAlpha				FillColorBlue			
Byte7	NumberOfChars							
Byte8	FontSize							
Byte9	TextPosX, low 8 bits							
Byte10	TextPosY, low 4 bits				TextPosX, high 4 bits			
Byte11	TextPosY, high 8 bits							
Byte12	TextChar01, low 8 bits							
Byte13	TextChar01, high 8 bits							
...	Additional text characters, 2 bytes per character							

The total number of bytes of the command is  $12 + 2 \times \text{NumberOfChars}$ .

UDS command flow (example for 8 characters, 'O3M text'):

The number of bytes of the command  $12 + 2 \times 8 = 28$  (Byte0 through Byte27). 5 bytes are transferred in the first request message and up to 7 in every consecutive message. Therefore, a total of 5 request messages is needed. The payload of the last message is  $28 - 5 \times 7 = 3$ , the DLC is therefore 3 because the first byte is needed for the part identifier.

If more request messages are needed, please increment the part identifier for every message.

The O3M response will be sent by O3M after the last request message has been completed.





1. Request message to O3M (ID 0x18DAEFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x10	0x1D (data size = 28 bytes payload + 1 byte service identifier)		0x2E	Byte0 – Byte4 of command			

2. Response message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x30	Timing information (0x00, 0x0A, 0x78, 0x00, 0x00, 0x00, 0x00)						

3. Request message to O3M (ID 0x18DAEFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x21	Byte5 – Byte11 of command						

4. Request message to O3M (ID 0x18DAEFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x22	Byte12 – Byte18 of command						

5. Request message to O3M (ID 0x18DAEFF1x, DLC = 8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x23	Byte19 – Byte25 of command						

6. Request message to O3M (ID 0x18DAEFF1x, DLC = 3)

Byte0	Byte1	Byte2
0x24	Byte26 – Byte27 of command	

7. Response message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x03	0x6E	0xFD	0x24	0x00	0x00	0x00	0x00

The text is now displayed on the O3M 2D image with the specified colors and styles and at the specified position.

### Changing a primitive

To change parameters of an existing primitive, e.g. color or position, re-send the primitive using the same primitive number and the new desired values.

### Removing a primitive

To remove an existing primitive, the primitive command “clear primitive” is used.

Properties of the clear primitive command:

Name	Data type	Description
primitiveNumber	UInt8	Unique identifier of the primitive to remove. Please use the same value that was used to create the primitive.

Command layout:

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Byte0	0xFD							
Byte1	0x27							
Byte2	primitiveNumber							

UDS command flow:

1. Request message to O3M (ID 0x18DAEFF1x, DLC = 5)

Byte0	Byte1	Byte2	Byte3	Byte4
0x04	0x2E	Byte0 – Byte2 of command		

2. Response message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x03	0x6E	0xFD	0x27	0x00	0x00	0x00	0x00





The primitive with the specified primitive number is now removed from the O3M 2D image.

### 3.1.5. Online parametrization

The online parametrization allows the modification of some parameters immediately without writing them permanently.

The parameters have to be sent and then applied.

Please be aware that some parameters have to be sent in little endian byte order and some parameters in big endian byte order.

#### *Set short parameter without apply*

This command is used to change parameters with a size of up to four bytes. It does not include an apply command.

Request message to O3M (ID 0x18DAEFF1x)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Length	Command 0x4B	ID high byte	ID low byte	Payload byte 1	Payload byte 2	Payload byte 3	Payload byte 4

The length is 3+number of payload bytes. So it's 4 for a parameter of one byte and 7 for a parameter with four bytes.

#### *Set short parameter with apply*

This command is used to change parameters with a size of up to four bytes. It does contain an apply command. This apply also applies all previous parameters that were sent without the apply.

Request message to O3M (ID 0x18DAEFF1x)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Length	Command 0x4D	ID high byte	ID low byte	Payload byte 1	Payload byte 2	Payload byte 3	Payload byte 4

The length is 3+number of payload bytes. So it's 4 for a parameter of one byte and 7 for a parameter with four bytes.

The list of parameter IDs can be found below.

#### *Large parameter block without apply*

This command is used to change parameters of arbitrary size. A single command only allows to change up to 4080 byte, so for larger parameters several commands are needed. This command does not contain an apply command.

Request to O3M (ID 0x18DAEFF1x), first message

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
bits 7..4=0001 bits 3..0 = length bits 11..8	Length bits 7..0	Command 0x4c	ID high byte	ID low byte	Offset bits 31..24	Offset bits 23..16	Offset bits 15..8



## Second message

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x21	Offset bits 7..0	Size payload bits 31..24	Size payload bits 23..16	Size payload bits 15..8	Size payload bits 7..0	First byte of the parameter	Second byte of the parameter

The next message starts with 0x22 and contains 7 bytes of the parameter. In the following messages the first byte counts up to 0x2f and then restarts with 0x20.

For the parameter ID please see below.

The offset is the offset in the parameter. This is usually 0 except if the size is >4080 byte then it's necessary to send several large parameter block commands and change the offset accordingly.

### *Large parameter block with apply*

This command is used to change parameters of arbitrary size. A single command only allows to change up to 4080 byte, so for larger parameters several commands are needed. This command does contain an apply command.

The content of the messages is the same as "Large parameter block without apply", except the command is now 0x4E.

### *Apply command*

This command applies the changes that were sent previously.

Request message to O3M (ID 0x18DAEFF1x, DLC = 3)

Byte0	Byte1	Byte2
0x02	Command 0x4A	0x02

### *Clear command*

This command clears all changes that are not yet applied.

Request message to O3M (ID 0x18DAEFF1x, DLC = 3)

Byte0	Byte1	Byte2
0x02	Command 0x4A	0x01

### *Get dirty status command*

The dirty status tells if there are changes not yet applied and if there are changes that are applied.

Request message to O3M (ID 0x18DAEFF1x, DLC = 3)

Byte0	Byte1	Byte2
0x02	Command 0x4A	0x06

The answer from the O3M (ID 0x18DAF1EFx, DLC = 8):

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
-------	-------	-------	-------	-------	-------	-------	-------



0x03	0x8A	State working copy	State Shadow copy	Fill byte	Fill byte	Fill byte	Fill byte
------	------	--------------------	-------------------	-----------	-----------	-----------	-----------

State working copy:

0 = no changes applied

1 = There were changes applied

State shadow copy:

0 = no unapplied changes

1 = there are unapplied changes

### *Return to flash command*

This command reverts the changes done by online parametrization and restores the original settings from the permanent parameter memory.

Request message to O3M (ID 0x18DAEFF1x, DLC = 3)

Byte0	Byte1	Byte2
0x02	Command 0x4A	0x05

### *Online parametrization answer messages*

**Response OK** message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x01	Command + 0x40	0x00	0x00	0x00	0x00	0x00	0x00

If the command was 0x4A then byte1 will be 0x8A.

**Please wait** message from O3M (ID 0x18DAF1EFx)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x03	0x7F	Command	0x78	0x00	0x00	0x00	0x00

The command is the one from the request.

This message means the O3M is still processing the command. It may send several of these messages.

**Error** response message from O3M (ID 0x18DAF1EFx, DLC=8)

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x03	0x7F	Command	Errorcode	0x00	0x00	0x00	0x00

Errorlist:



Errorcode	Description
0x10	General Reject
0x11	Service Not Supported
0x12	Subfunction Not Supported
0x13	Invalid Format
0x14	Response Too Long
0x21	BusyRepeat Request
0x22	Conditions Not Correct. This means online parametrization is disabled.
0x24	Request Sequence Error
0x31	Request Out Of Range
0x33	Access Denied
0x35	Invalid Key
0x36	Exceeded Number Of Attempts
0x37	Time Delay Not Expired
0x70	Upload Download Not Accepted
0x71	Transfer Data Suspended
0x72	General Programming Failure
0x73	Wrong Block Sequence Counter
0x7E	Subfunction Not Supported In Active Session
0x7F	Service Not Supported In Active Session

### Flow Control Message

The flow control message is part of the underlying TP protocol. For details see [https://en.wikipedia.org/wiki/ISO\\_15765-2](https://en.wikipedia.org/wiki/ISO_15765-2)

The flow control is sent by the O3M after it gets the first message of an multi-part transmission. It tells the sender how fast and in which block sizes it may send.

A TP message from online parametrization looks like this:

30 00 0A 00 01 00 00 00

It tells the sender it may send all the data without another flow control message and it should keep the messages 10ms apart

## 3.2. Description of the Messages

Explanation:

Name: The name of the parameter

Factory Value: The content at delivery or after a firmware update

Data type: uint=unsigned integer / sint = signed integer / float = IEEE754 floating point. The number indicates the number of bits used.

Min: The minimum allowed value.

Max: The maximum allowed value.

Apply needs special action: A "y" means there's internally some reinit happening on apply. If this takes some time the runmode will change during this time and the sensor will not be available for a short time. A "n" here means the parameter is applied without any reinit happening.

Parameter Number: The number to identify the parameter

Description: Description of the parameter

Byte order: The byte order. It means that the value 0x12 34 56 78 has to be given as 78 56 34 12 (Little endian) or 12 34 56 78 (Big endian)



### 3.3. Messages exclusive to DI variant

#### 3.3.1. Online parameters specific to DI variant

Name	Factory Value	Data type	Min	Max	Apply needs special action	Parameter Number	Description	Byte order
EthernetOutputConfiguration	0	uint8	0	1	n	0x4805	0 is standard output, 1 debug output	little endian
PMDExtrCalib_camCal_transX	0	float32	-30	30	y	0x4807	Extrinsic calibration of camera: X translation [m]	little endian
PMDExtrCalib_camCal_transY	0	float32	-30	30	y	0x4808	Extrinsic calibration of camera: Y translation [m]	little endian
PMDExtrCalib_camCal_transZ	1	float32	-30	30	y	0x4809	Extrinsic calibration of camera: Z translation [m]	little endian
PMDExtrCalib_camCal_rotX	1,570796327	float32	-3,1	3,14	y	0x480A	Extrinsic calibration of camera: Y rotation [rad]	little endian
PMDExtrCalib_camCal_rotY	1,570796327	float32	-3,1	3,14	y	0x480B	Extrinsic calibration of camera: Y rotation [rad]	little endian
PMDExtrCalib_camCal_rotZ	0	float32	-3,1	3,14	y	0x480C	Extrinsic calibration of camera: Z rotation [rad]	little endian
PMDExtrCalib_IlluCal_transX	-0,085	float32	-30	30	y	0x480D	Extrinsic calibration of camera: X translation [m]	little endian
PMDExtrCalib_IlluCal_transY	0,052	float32	-30	30	y	0x480E	Extrinsic calibration of camera: Y translation [m]	little endian
PMDExtrCalib_IlluCal_transZ	0,047	float32	-30	30	y	0x480F	Extrinsic calibration of camera: Z translation [m]	little endian
PMDExtrCalib_IlluCalibIsRelative	1	uint8	0	1	y	0x4810	Flag indicating whether illu calibration is given relative to camera or absolute in world coordinates.	little endian
DistImageCust_sprayRemovalSensitivity	0	uint8	0	3	n	0x4811	Spray Removal customization	little endian
DistImageCust_pixelPlausibilizationThresholds	1	uint8	0	3	n	0x4812	Pixel plausibilization customization	little endian
DistImageCust_blockageSensitivity	0	uint8	0	3	n	0x4813	Blockage customization	little endian
DistImageCust_spatialFilterX Min	-100	float32	-100	100	n	0x4814	Spatial filter on the Cartesian coordinates, minimum X	little endian
DistImageCust_spatialFilterX Max	100	float32	-100	100	n	0x4815	Spatial filter on the Cartesian coordinates, maximum X	little endian
DistImageCust_spatialFilterY Min	-100	float32	-100	100	n	0x4816	Spatial filter on the Cartesian coordinates, minimum Y	little endian
DistImageCust_spatialFilterY Max	100	float32	-100	100	n	0x4817	Spatial filter on the Cartesian coordinates, maximum Y	little endian
DistImageCust_spatialFilterZ Min	-100	float32	-100	100	n	0x4818	Spatial filter on the Cartesian coordinates, minimum Z	little endian



DistImageCust_spatialFilterZMax	100	float32	-100	100	n	0x4819	Spatial filter on the Cartesian coordinates, maximum Z	little endian
DistImageCust_reflectorThresholdValue	0,1	float32	0	1	n	0x481A	Value for setting the reflectivity threshold to detect retroreflectors	little endian
DistImageCust_reflectorCloseRange	0	uint8	0	1	n	0x481B	Reflector Close Range	little endian
DistImageCust_bfActive	1	sint32	0	1	n	0x481C	Flag for switching BF function on/off, default: on	little endian
DistImageCust_bfROIs_numberOfROIs	1	sint32	1	64	n	0x4822	number of used ROIs	little endian
DistImageCust_bfTemporalWindowSize	1	sint32	1	50	n	0x482B	Filtering window over time	little endian
DistImageCust_bfOutputMode	5	sint32	0	5	n	0x482C	Output mode for CAN output 0: x only; 1: y only; 2: z only; 3: xyz only; 4: ampl only; 5: xyz+ampl	little endian
AutoCalibParam_numberOfPatterns	0	uint8	0	8	n	0x482D	Number of patterns to be used for autocalibration (0,1 : autocalibration disabled)	little endian
triggeredStreetCalibration	0	uint8	0	1	n	0x4834	Flag indicating if the triggered calibration based on the street plane estimation is active	little endian
num_Frames_Averaging	3	sint32	0	50	n	0x4836	the window size for temporal filtering of pmd raw data (0: no filtering)	little endian
pixelPlausiReflectivityThreshold	2.1	float32	0	20	n	0x4837	reflectivity threshold for pixel plausibilization (0: no filtering)	little endian
Modulation_Frequency_Mode	3	uint8	0	3	y	0x4838	0 = random 13 frequency hopping (default) 1 = fixed triple 1 2 = fixed triple 2 3 = fixed triple 3	little endian
amplThresholdFactor	1.0	float32	0.1	10	n	0x483B	Amplitude factor for compensation of unusual camera/illu combinations	little endian
ROICanOutputOnOff	1	uint8	0	1	n	0x440B	Switch on or off the ROI CAN messages.	big endian
RotateImage90	0	uint8	0	1	y	0x483D	Rotate the Algo output by 90 degrees for displaying in 2D overlay	little endian
Output2D_OnOff	1	int8_t	0	1	n	0x5C03	1: 2D output shall be on 0: 2D output shall be off	little endian
<b>RotateImage90</b>	0	uint8_t	0	1	y	0x5C06	Rotate the analog video by 90 degree clockwise	little endian
Overlay3DOnOff	1	uint8	0	1	n	0x6001	0: No Overlay sent to GP 1: calculate overlay and send overlay data to the GP	big endian
FixedGraphicPrimitivesOnOff	1	uint8	0	1	n	0x6003	Show FixedGraphicPrimitives =1, or do not show =0	big endian



max_number_of_algo_primitives_for_short_primitives	96	uint8	0	96	n	0x6004	sum of all "max number of * short primitives"	big endian
max_number_of_algo_primitives_for_long_primitives	0	uint8	0	10	n	0x6005	sum of all "max number of * long primitives"	big endian
max_number_of_CAN_input_primitives_for_short_primitives	10	uint8	0	10	n	0x6006	sum of all "max number of * short primitives"	big endian
max_number_of_CAN_input_primitives_for_long_primitives	4	uint8	0	10	n	0x6007	sum of all "max number of * long primitives"	big endian
max_number_of_fixed_primitives_for_short_primitives	10	uint8	0	24	n	0x6008	sum of all "max number of * short primitives"	big endian
max_number_of_fixed_primitives_for_long_primitives	8	uint8	0	10	n	0x6009	sum of all "max number of * long primitives"	big endian
ROIGraphicPrimitivesOnOff	1	uint8	0	1	n	0x600C	Show ROIGraphicPrimitives =1, or do not show =0	big endian
Display3DFOV	0	uint32	0	3	n	0x600D	0 - Do not show 3D FOV overlay 1 - Show 3D FOV overlay rectangle 2 - Show 3D FOV overlay rectangle with "3D FOV" text 3 - Show 3D FOV overlay rectangle and Clip Algo primitives outside with grey overlay	big endian
BlinkingOnDuration	10	uint16	1	100	n	0x600F	for the blinking of graphic primitives, this is the "on" duration.  Unit is 100ms.	big endian
BlinkingOffDuration	10	uint16	1	100	n	0x6010	for the blinking of graphic primitives, this is the "off" duration.  Unit is 100ms.	big endian

### 3.4. Messages exclusive to OD variant

#### 3.4.1. Online parameters specific to OD variant

Name	Factory Value	Data type	Min	Max	Apply needs special action	Parameter Number	Description	Byte order
EthernetOutputConfiguration	0	uint8	0	1	n	0x8805	0 is standard output, 1 debug output	little endian
VehicleDim_xMin	-1	float32	-20	20	y	0x8808	Vehicle dimension description (in world coordinates), axis-	little endian



							parallel box on the ground plane	
VehicleDim_xMax	1	float32	-20	20	y	0x8809	Vehicle dimension description (in world coordinates), axis-parallel box on the ground plane	little endian
VehicleDim_yMin	-1	float32	-20	20	y	0x880A	Vehicle dimension description (in world coordinates), axis-parallel box on the ground plane	little endian
VehicleDim_yMax	1	float32	-20	20	y	0x880B	Vehicle dimension description (in world coordinates), axis-parallel box on the ground plane	little endian
VehicleDim_zMax	2	float32	0	10	y	0x880C	Vehicle dimension description (in world coordinates), axis-parallel box on the ground plane	little endian
PMDExtrCalib_cam Cal_transX	0	float32	-30	30	y	0x880D	Extrinsic calibration of camera: X translation [m]	little endian
PMDExtrCalib_cam Cal_transY	0	float32	-30	30	y	0x880E	Extrinsic calibration of camera: Y translation [m]	little endian
PMDExtrCalib_cam Cal_transZ	1	float32	0.5	30	y	0x880F	Extrinsic calibration of camera: Z translation [m]	little endian
PMDExtrCalib_cam Cal_rotX	-1,570796327	float32	-3,1	3,14	y	0x8810	Extrinsic calibration of camera: Y rotation [rad]	little endian
PMDExtrCalib_cam Cal_rotY	1,570796327	float32	-3,1	3,14	y	0x8811	Extrinsic calibration of camera: Y rotation [rad]	little endian
PMDExtrCalib_cam Cal_rotZ	0	float32	-3,1	3,14	y	0x8812	Extrinsic calibration of camera: Z rotation [rad]	little endian
PMDExtrCalib_IlluCal_transX	-0,085	float32	-30	30	y	0x8813	Extrinsic calibration of camera: X translation [m]	little endian
PMDExtrCalib_IlluCal_transY	0,052	float32	-30	30	y	0x8814	Extrinsic calibration of camera: Y translation [m]	little endian
PMDExtrCalib_IlluCal_transZ	0,047	float32	-30	30	y	0x8815	Extrinsic calibration of camera: Z translation [m]	little endian
PMDExtrCalib_IlluCaliblsRelative	1	uint8	0	1	y	0x8816	Flag indicating whether illu calibration is given relative to camera or absolute in world coordinates.	little endian





ObjectListCust_sprayRemovalSensitivity	0	uint8	0	3	n	0x8817	Spray Removal customization	little endian
ObjectListCust_pixelPlausibilizationThresholds	1	uint8	0	2	n	0x8818	Pixel plausibilization customization	little endian
ObjectListCust_blockageSensitivity	0	uint8	0	3	n	0x8819	Blockage customization	little endian
ObjectListCust_spatialFilterXMin	-100	float32	-100	100	n	0x881A	Spatial filter on the Cartesian coordinates, minimum X	little endian
ObjectListCust_spatialFilterXMax	100	float32	-100	100	n	0x881B	Spatial filter on the Cartesian coordinates, maximum X	little endian
ObjectListCust_spatialFilterYMin	-100	float32	-100	100	n	0x881C	Spatial filter on the Cartesian coordinates, minimum Y	little endian
ObjectListCust_spatialFilterYMax	100	float32	-100	100	n	0x881D	Spatial filter on the Cartesian coordinates, maximum Y	little endian
ObjectListCust_spatialFilterZMin	-100	float32	-100	100	n	0x881E	Spatial filter on the Cartesian coordinates, minimum Z	little endian
ObjectListCust_spatialFilterZMax	100	float32	-100	100	n	0x881F	Spatial filter on the Cartesian coordinates, maximum Z	little endian
ObjectListCust_reflectorThresholdValue	0,1	float32	0	1	n	0x8820	Value for setting the reflectivity threshold to detect retroreflectors	little endian
ObjectListCust_autoCalibrationMode	0	uint8	0	42	n	0x8821	Autocalibration Mode (0: disabled, 1: enabled, 42: test mode)	little endian
ObjectListCust_objectDetectionVariant	1	uint8	0	1	n	0x8822	Object detection variant selection (0: normal object list + crash predictor, 1: Retroreflector mode)	little endian
ObjectListCust_ObjectDetectionZMin	0,5	float32	-10	10	n	0x8823	Minimum z coordinate for normal object detection	little endian
ObjectListCust_ObjectDetectionZMax	2	float32	-10	10	n	0x8824	Maximum z coordinate for normal object detection	little endian
ObjectListCust_CrashPredictorSensitivity	1	uint8	0	2	n	0x8825	Customization of crash predictor - 0: low sensitivity (optimized for low false positive rates) - 1: medium sensitivity - 2: high sensitivity (optimized for high true positive rates)	little endian



ObjectListCust_Ego MotionDynamics	2	uint8	0	2	n	0x8826	Parametrization of the sidestepping capabilities for crash avoidance calculations. - 0: no sidestepping modelled (e.g., railway) - 1: low sidestepping dynamics (e.g., trucks, 0.1 rad/s) - 2: high sidestepping dynamics (e.g., cars, 0.3 rad/s)	little endian
ObjectListCust_acc Brake	5	float32	0	30	n	0x8827	Definition of brake deceleration for crash predictor.	little endian
ObjectListCust_delayBrake	1	float32	0	3	n	0x8828	Definition of brake delay for crash predictor ObjectListCust_delayBrake3 >= ObjectListCust_delayBrake2 >= ObjectListCust_delayBrake	little endian
ObjectListCust_ego VMin	0	float32	-40	40	n	0x8829	Definition of minimal ego velocity for calculating crash events	little endian
ObjectListCust_ego VMax	18	float32	-40	40	n	0x882A	Definition of maximal ego velocity for calculating crash events	little endian
ObjectListCust_Ego DataMode	2	uint8	0	2	y	0x882B	Enables ego velocity calculation and yaw rate input 0: ego velocity and Yaw Rate are taken into account; 1: only ego velocity; 2: no data used.	little endian
ObjectListCust_cpDeactivateTimeAfterTrigger	10	float32	0	60	n	0x882C	Minimal time between two predicted crashes.	little endian
ObjectListCust_cpMaxCrashObjectDistance	25	float32	0	50	n	0x882D	Maximum distance for relevant objects.	little endian
ObjectListCust_cpMinDistAllowed	0,5	float32	0	5	n	0x882E	Objects with a distance below this value lead to a predicted collision.	little endian
ObjectListCust_cpActivate	0	uint8	0	1	n	0x882F	Enable flag for collision detection.	little endian
AutoCalibParam_numberOfPatterns	0	uint8	0	8	n	0x8830	Number of patterns to be used for autocalibration (0,1 : autocalibration disabled)	little endian
triggeredStreetCalibration	0	uint8	0	1	n	0x8837	Flag indicating if the triggered calibration based on the street plane estimation is active	little endian



num_Frames_Averaging	0	sint32	0	50	n	0x8838	the window size for temporal filtering of pmd raw data (0: no filtering)	little endian
pixelPlausiReflectivityThreshold	2.1	float32	0	20	n	0x8839	reflectivity threshold for pixel plausibilization (0: no filtering)	little endian
Modulation_Frequency_Mode	3	uint8	0	3	y	0x883A	0 = random 13 frequency hopping (default) 1 = fixed triple 1 2 = fixed triple 2 3 = fixed triple 3	little endian
amplThresholdFactor	1.0	float32	0.1	10	n	0x883B	Amplitude factor for compensation of unusual camera/illu combinations	little endian
ObjectListCust_delayBrake2	0	float32	0	3	n	0x883D	brake delay for crash predictor for criticality level 2 in seconds validity condition ObjectListCust_delayBrake3 >= ObjectListCust_delayBrake2 >= ObjectListCust_delayBrake	little endian
ObjectListCust_delayBrake3	0	float32	0	3	n	0x883E	brake delay for crash predictor for criticality level 3 in seconds validity condition ObjectListCust_delayBrake3 >= ObjectListCust_delayBrake2 >= ObjectListCust_delayBrake	little endian
CANMaxNumberOfObjects	8	uint8	0	20	n	0x840A	Configuration of maximum number of objects in Object List on CAN	big endian
RotateImage90	0	uint8	0	1	y	0x8843	Rotate the output by 90 degree clockwise for presentation in 2D	little endian
Output2D_OnOff	1	int8_t	0	1	n	0x9C03	1: 2D output shall be on 0: 2D output shall be off	little endian
RotateImage90	0	uint8_t	0	1	y	0x9C06	Rotate the analog video by 90 degree clockwise	little endian
Overlay3DOnOff	1	uint8	0	1	n	0xA001	0: No Overlay sent to GP 1: calculate overlay and send overlay data to the GP	big endian
FixedGraphicPrimitivesOnOff	1	uint8	0	1	n	0xA003	Display the FixedGraphicPrimitives =1 or do not display =0	big endian
max_number_of_algo_primitives_for_short_primitives	96	uint8	0	96	n	0xA004	Max algo short primitives that can be	big endian



							generated from one algo result frame	
max_number_of_algo_primitives_for_long_primitives	0	uint8	0	10	n	0xA005	Max algo short primitives that can be generated from one algo result frame	big endian
max_number_of_CAN_input_primitives_for_short_primitives	10	uint8	0	10	n	0xA006	Max CAN short primitives that can be displayed in one frame	big endian
max_number_of_CAN_input_primitives_for_long_primitives	4	uint8	0	10	n	0xA007	Max CAN long primitives that can be displayed in one frame	big endian
max_number_of_fixed_primitives_for_short_primitives	10	uint8	0	24	n	0xA008	Max Fixed short primitives that can be displayed in one frame	big endian
max_number_of_fixed_primitives_for_long_primitives	8	uint8	0	10	n	0xA009	Max Fixed long primitives that can be displayed in one frame	big endian
MaxAlgoObjectsToDraw	10	uint8	0	20	n	0xA00A	Max number of objects that can be processed for primitives for Algo objects overlay	big endian
CAOnOff	1	uint8	0	1	n	0xA00C	Display the CAGraphicPrimitives =1 or do not display =0	big endian
OD3DOnOff	1	uint8	0	1	n	0xA00E	Display the OD3DGraphicPrimitives =1 or do not display =0	big endian
ODReflectiveOnOff	1	uint8	0	1	n	0xA010	Display the OD3DGraphicPrimitives =1 or do not display =0	big endian
displayOnlyMostCriticalZone	0	uint8	0	1	n	0xA011	0 - Show Crash Predicted objects in each zone independently 1 - Show Crash Predicted object in the most critical zone only. Other two zones are not shown	big endian
Display3DFOV	0	uint32	0	3	n	0xA012	0 - Do not show 3D FOV overlay 1 - Show 3D FOV overlay rectangle 2 - Show 3D FOV overlay rectangle with "3D FOV" text 3 - Show 3D FOV overlay rectangle and Clip Algo primitives outside with grey overlay	big endian



BlinkingOnDuration	10	uint16	1	100	n	0xA014	for the blinking of graphic primitives, this is the "on" duration. Unit is 100ms.	big endian
BlinkingOffDuration	2	uint16	1	100	n	0xA015	for the blinking of graphic primitives, this is the "off" duration. Unit is 100ms.	big endian

### 3.5. Messages exclusive to LG variant

#### 3.5.1. Online parameters specific to LG variant

Name	Factory Value	Data type	Min	Max	Apply needs special action	Parameter Number	ID	Description	Byte order
EthernetOutputConfiguration	0	uint8	0	1	n	0xC805	5	0 is standard output, 1 debug output	little endian
PMDExtrCalib_cameraCal_transX	0	float32	-30	30	y	0xC808	8	Extrinsic calibration of camera: X translation [m]	little endian
PMDExtrCalib_cameraCal_transY	0	float32	-30	30	y	0xC809	9	Extrinsic calibration of camera: Y translation [m]	little endian
PMDExtrCalib_cameraCal_transZ	1	float32	0.5	30	y	0xC80A	10	Extrinsic calibration of camera: Z translation [m]	little endian
PMDExtrCalib_cameraCal_rotX	-1,570796327	float32	-3,1	3,14	y	0xC80B	11	Extrinsic calibration of camera: Y rotation [rad]	little endian
PMDExtrCalib_cameraCal_rotY	1,570796327	float32	-3,1	3,14	y	0xC80C	12	Extrinsic calibration of camera: Y rotation [rad]	little endian
PMDExtrCalib_cameraCal_rotZ	0	float32	-3,1	3,14	y	0xC80D	13	Extrinsic calibration of camera: Z rotation [rad]	little endian
PMDExtrCalib_IlluCal_transX	-0,085	float32	-30	30	y	0xC80E	14	Extrinsic calibration of camera: X translation [m]	little endian
PMDExtrCalib_IlluCal_transY	0,052	float32	-30	30	y	0xC80F	15	Extrinsic calibration of camera: Y translation [m]	little endian



PMDExtrCalib_Illu Cal_transZ	0,047	float32	-30	30	y	0xC810	16	Extrinsic calibration of camera: Z translation [m]	little endian
PMDExtrCalib_Illu CalibIsRelative	1	uint8	0	1	y	0xC811	17	Flag indicating whether illu calibration is given relative to camera or absolute in world coordinates.	little endian
LineGuidanceCus t_sprayRemovalS ensitivity	0	uint8	0	3	n	0xC812	18	Spray Removal customization	little endian
LineGuidanceCus t_pixelPlausibiliza tionThresholds	1	uint8	0	2	n	0xC813	19	Pixel plausibilization customization	little endian
LineGuidanceCus t_blockageSensiti vity	0	uint8	0	3	n	0xC814	20	Blockage customization	little endian
LineGuidanceCus t_spatialFilterXMi n	-100	float32	-100	100	n	0xC815	21	Spatial filter on the Cartesian coordinates, minimum X	little endian
LineGuidanceCus t_spatialFilterXMa x	100	float32	-100	100	n	0xC816	22	Spatial filter on the Cartesian coordinates, maximum X	little endian
LineGuidanceCus t_spatialFilterYMi n	-100	float32	-100	100	n	0xC817	23	Spatial filter on the Cartesian coordinates, minimum Y	little endian
LineGuidanceCus t_spatialFilterYMa x	100	float32	-100	100	n	0xC818	24	Spatial filter on the Cartesian coordinates, maximum Y	little endian
LineGuidanceCus t_spatialFilterZMi n	-100	float32	-100	100	n	0xC819	25	Spatial filter on the Cartesian coordinates, minimum Z	little endian
LineGuidanceCus t_spatialFilterZMa x	100	float32	-100	100	n	0xC81A	26	Spatial filter on the Cartesian coordinates, maximum Z	little endian
LineGuidanceCus t_reflectorThresho ldValue	0,1	float32	0	1	n	0xC81B	27	Value for setting the reflectivity threshold to detect retroreflectors	little endian
LineGuidanceCus t_lineGuidanceMo de	0	uint8	0	1	n	0xC81C	28	kind of line to be detected: 0 - heap detection, 1 - cut edge	little endian
LineGuidanceCus t_EgoDataMode	2	uint8	0	2	y	0xC81D	29	Ego Data Usage Mode - 0: Integrate velocity and yawrate input - 1: Use only velocity information	little endian



								- 2: No velocity and yawrate information	
LineGuidanceCust_steeringXMin	10	float32	0,3	30	n	0xC81E	30	Minimum x coordinate for steering computation	little endian
LineGuidanceCust_steeringXMax	20	float32	0,3	30	n	0xC81F	31	Maximum x coordinate for steering computation	little endian
LineGuidanceCust_steeringTTC	2,5	float32	0,3	10	n	0xC820	32	Prediction time [s] steering computation	little endian
LineGuidanceCust_maxLineAngle	0,34906585	float32	0	3,14	n	0xC821	33	maximum orientation (slope) of line in rad	little endian
LineGuidanceCust_xMin	0	float32	-100	100	n	0xC822	34	only lines which intersect with this area remain and are given as output	little endian
LineGuidanceCust_xMax	10	float32	-100	100	n	0xC823	35	only lines which intersect with this area remain and are given as output	little endian
LineGuidanceCust_yMin	-5	float32	-100	100	n	0xC824	36	only lines which intersect with this area remain and are given as output	little endian
LineGuidanceCust_yMax	5	float32	-100	100	n	0xC825	37	only lines which intersect with this area remain and are given as output	little endian
LineGuidanceCust_minHeight	0,3	float32	0,2	2	n	0xC826	38	min height of heap/cut edge/... [m]	little endian
LineGuidanceCust_minWidth	0,25	float32	0	10	n	0xC827	39	minimum width of heap/cut edge/... [m]	little endian
LineGuidanceCust_maxWidth	3	float32	-1	10	n	0xC828	40	maximum width of heap/cut edge/... [m] (negative value <=> unlimited)	little endian
LineGuidanceCust_skipStreetPlane Estimation	0	uint8	0	1	n	0xC829	41	determines if street plane estimation process is skipped <=> planeValid flag is set to 0	little endian
LineGuidanceCust_referencePointX Pos	0	float32	-100	100	n	0xC82A	42	x value of the reference point location.	little endian



LineGuidanceCus t_referencePointY Pos	0	float32	-100	100	n	0xC82B	43	y value of the reference point location.	little endian
LineGuidanceCus t_referencePointZ Pos	0	float32	-100	100	n	0xC82C	44	z value of the reference point location.	little endian
AutoCalibParam_ numberOfPattern s	0	uint8	0	8	n	0xC82D	45	Number of patterns to be used for autocalibration (0,1 : autocalibration disabled)	little endian
triggeredStreetCal ibration	0	uint8	0	1	n	0xC834	52	Flag indicating if the triggered calibration based on the street plane estimation is active	little endian
lineQualityThresh old	0	float32	0	1	n	0xC835	53	Applied thresholding value on the line quality measure	little endian
num_Frames_Av eraging	3	sint32	0	50	n	0xC836	54	the window size for temporal filtering of pmd raw data (0: no filtering)	little endian
pixelPlausiReflecti vityThreshold	2.1	float32	0	20	n	0xC837	55	reflectivity threshold for pixel plausibilization (0: no filtering)	little endian
Modulation_Frequ ency_Mode	3	uint8	0	3	y	0xC838	56	0 = random 13 frequency hopping (default) 1 = fixed triple 1 2 = fixed triple 2 3 = fixed triple 3	little endian
amplThresholdFa ctor	1.0	float32	0.1	10	n	0xC839	57	Amplitude factor for compensation of unusual camera/illu combinations	little endian
LineGuidanceCus t_lineOutputAvera geFilter	0	float32	0	1	n	0xC83D	61	Parameter shall enable the user to smooth the line description output	little endian
CANMaxNumber OfLines	1	uint8	0	5	n	0xC40A	10	Configuration of maximum number of objects in Object List on CAN	big endian
<b>RotateImage90</b>	0	uint8	0	1	y	0xC83E	62	Rotate the output by 90 degree clockwise for presentation at 2D	little endian
Output2D_OnOff	1	int8_t	0	1	n	0xDC03	3	1: 2D output shall be on	little endian





								0: 2D output shall be off	
RotateImage90	0	uint8_t	0	1	y	0xDC06	6	Rotate the analog video by 90 degree clockwise	little endian
Overlay3DOnOff	1	uint8	0	1	n	0xE001	1	0: No Overlay sent to GP 1: calculate overlay and send overlay data to the GP	big endian
StartupIconOnOff	1	uint8	0	1	n	0xE002	2	1: show an Icon and Backgroundcolor (Splash screen) at startup 0: Do not show anything at startup	big endian
StartupDisplayDuration	2000	uint32	0	0xFFFF	n	0xE003	3	Time to show the Icon and Backgroundcolor (Splash screen) at startup	big endian
StartupIconID	1	uint8	0	40	n	0xE004	4	Which Icon to show at startup (Initial: ifm logo)	big endian
FixedGraphicPrimitivesOnOff	1	uint8	0	1	n	0xE006	6	Display the FixedGraphicPrimitives =1 or do not display =0	big endian
max_number_of_algo_primitives_for_short_primitives	96	uint8	0	96	n	0xE008	8	Max algo short primitives that can be generated from one algo result frame	big endian
max_number_of_algo_primitives_for_long_primitives	0	uint8	0	10	n	0xE009	9	Max algo short primitives that can be generated from one algo result frame	big endian
max_number_of_CAN_input_primitives_for_short_primitives	10	uint8	0	10	n	0xE00A	10	Max CAN short primitives that can be displayed in one frame	big endian
max_number_of_CAN_input_primitives_for_long_primitives	4	uint8	0	10	n	0xE00B	11	Max CAN long primitives that can be displayed in one frame	big endian
max_number_of_fixed_primitives_for_short_primitives	10	uint8	0	24	n	0xE00C	12	Max Fixed short primitives that can be displayed in one frame	big endian
max_number_of_fixed_primitives_for_long_primitives	8	uint8	0	10	n	0xE00D	13	Max Fixed long primitives that can be displayed in one frame	big endian



DisplayEgoSpeedOnOff	1	uint8	0	1	n	0xE00E	14	formatting of crash predicted objects: 1: write the own velocity 0: Do not write	big endian
DisplayEgoYawrateOnOff	1	uint8	0	1	n	0xE00F	15	formatting of crash predicted objects: 1: write the own yaw rate 0: do not write	big endian
Display3DFOV	0	uint32	0	3	n	0xE010	16	0 - Do not show 3D FOV overlay 1 - Show 3D FOV overlay rectangle 2 - Show 3D FOV overlay rectangle with "3D FOV" text 3 - Show 3D FOV overlay rectangle and Clip Algo primitives outside with grey overlay	big endian
BlinkingOnDuration	10	uint16	1	100	n	0xE012	18	for the blinking of graphic primitives, this is the "on" duration. Unit is 100ms.	big endian
BlinkingOffDuration	10	uint16	1	100	n	0xE013	19	for the blinking of graphic primitives, this is the "off" duration. Unit is 100ms.	big endian

## 3.6. Standby Mode

The O3M sensor has a standby mode. In this mode it disables the illumination and reduces the power consumption. It's not possible to use the online calibration in this mode.

The possible response codes are listed in "UDS Response Codes".

### 3.6.1. Enter Standby Mode

Request message to O3M (ID 0x18DAEFF1x, DLC = 3)

Byte0	Byte1	Byte2
0x02	Command 0x4A	0x03

### 3.6.2. Leave Standby Mode

Request message to O3M (ID 0x18DAEFF1x, DLC = 3)

Byte0	Byte1	Byte2
0x02	Command 0x4A	0x04



## 3.7. Teach

When the O3M receives a teach command the user defined logic saves data values into the non-volatile memory. When it receives an unteach command it erases them. The possible response codes are listed in “UDS Response Codes”.

### 3.7.1. Teach command

Request message to O3M (ID 0x18DAEFF1x, DLC = 3)

Byte0	Byte1	Byte2
0x02	Command 0x4A	0x07

### 3.7.2. Unteach command

Request message to O3M (ID 0x18DAEFF1x, DLC = 3)

Byte0	Byte1	Byte2
0x02	Command 0x4A	0x08

## 3.8. O3M Reset

Request message to O3M (ID 0x18DAEFF1x), DLC = 8

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
2	Command 0x11	1	00 Fill bytes	00 Fill bytes	00 Fill bytes	00 Fill bytes	00 Fill bytes

## 3.9. UDS Response Codes

OK

Byte0	Byte1
0x01	0x8A

Please wait

Byte0	Byte1	Byte2	Byte3
0x03	0x7F	0x4A	0x78

Error

Byte0	Byte1	Byte2	Byte3
0x03	0x7F	0x4A	Error code

## 3.10. UDS Diagnostic Information

Use UDS service 0x22 (ReadDataByIdentifier).



Use identifier 0x07 0x01 to read temperature and voltages of the camera unit in degree celsius / volt.

- Response: 0x62 0x07 0x01 plus 52 data bytes:

[0:0] Supply Voltage

[44:0] Temperature on PCB

[48:0] Frontend Temperature

Use identifier 0x07 0x02 to read temperature and voltages of the illumination unit in degree celsius / volt.

- Positive Response: 0x62 0x07 0x02 plus 12 data bytes:

[0:0] Umin (voltage during modulation)

[4:0] Unom (voltage outside modulation)

[8:0] Temperature



## 4. CANopen

In the firmware release in the subdirectory "Specs" there is an EDS file that contains the interface specification of O3M in CANopen mode.

But The payload of the data messages is the same as it is for J1939. It is not possible to receive only single signals, ie you have to receive the whole 8 bytes of the crash\_predictor\_info message and to get the CP\_object\_id. The data structure of the payload those messages is not specified in the EDS file. Please refer to this document (J1939 section) or the DBC file for the data structure (bit offsets, length, etc) of the messages.



## 5. Version History

Version	Date	Changes
V01	2017-09-07	Initial Release
V02	2017-12-04	<ul style="list-style-type: none"><li>- Fixed wrong offset in the Crash Predictor Info message</li><li>- added short chapter about CAN/Open</li></ul>
V03	2018-01-11	<ul style="list-style-type: none"><li>- Fixed wrong ID for leave standby command</li><li>- added teach/unteach</li><li>- added UDS response codes</li></ul>
V04	2018-03-14	<ul style="list-style-type: none"><li>- Several Fixes in the crash_predictor_info</li></ul>
V05	2018-10-08	<ul style="list-style-type: none"><li>- Fixed binary format of "Large parameter block without apply"</li></ul>
V06	2019-01-28	<ul style="list-style-type: none"><li>- General format changes</li><li>- Fixes in the CANopen section</li></ul>
V07	2020-05-19	<ul style="list-style-type: none"><li>- Review with small corrections</li><li>- Fixes table of contents</li><li>- Added UDS diagnostic chapter</li></ul>