Operating instructions

Series 09 – Rugged Keypad.

Operating instructions and safety instructions

www.eao.com

Your Expert Partner for **Human Machine Interfaces**

Operating instructions and safety instructions

Project	S09 Rugged Keypad
Document Number	1707900701
Document Description	Operating and safety instructions for series 09 Rugged Keypads and RCC with CAN Interface

EAO Automotive GmbH & Co. KG

Richard-Wagner-Straße 3

08209 Auerbach/Vogtl.

Change History					
Version	**Valid from SW-Version**	**Change Description**	**Status**	**Author**	**Date**
01		initial	intial	BEC	15.10.2019
02		CAN Open Safety added	Update	ESF	05.11.2019
04		Correction in Chapter 8.4	Update	SCI	15.11.2019
05		CAN communications extended	Update	GOM	05.02.2020
06	4.05.001	Document number corrected Chapter 8.2 updated Chapter 11.2 updated Review ESF, column for SW-Version in Change History added	Update	SCI	25.02.2020
07	4.05.001	Chapter 11.1.2 - Output current Wakeup Out changed - circuit diagrams for wakeup added - example messages J1939 fixed	Update	SCI/GOM	03.03.2020
08	4.05.001	- Chapter 0 added	Update	GOM	06.03.2020
09	6.03.000	- Chapter 7.4 use inside closed vehicles - Chapter 14 Cleaning updated	Update	ESF	18.01.2021
10	6.04.000	- Chapter 9.6 (Error handling) added - Chapter 13 (RCC) added - RCC application note (proportional-digital) added - Structure of lighting objects and interaction of objects explained - Differences between SW-Version 4.05.001 and 6.03.000 explained - Application Note for manually change the Baudrate, node-ID and stuck button time added	Update	MAR	02.06.2021

Referenced Dokuments			
Reference	**Version**	**Author**	**Date**
K-Matrix J1939	20	HEP	04.02.2021
K-Matrix CANopen/CANopen-Safety	20	HEP	04.02.2021
1707 product information	02	EBJ	12.05.2021
CiA CANopen® application layer and general communication profile „CAN poster“]			
CiA 301 “CANopen application and communication profile”	4.2.0		21.02.2011
CiA 320 “Services and protocols for sleep and wake-up handling”	1.0.0		14.03.2018
SAE J1939-73: Application Layer - Diagnostics			05.2017
SAE J1939-21: Data Link Layer			03.2016
SAE J1939-81: Network Management			03.2017
Series 09, Rugged CAN Keypads - CE-Certification - Compliancy of EAO Products	-	HDI	July,15, 2021

Table of contents

1. Safety warnings 12
1.1 Intended use 12
2. Proper environment 13
3. General description 13
4. Technical specification 15
5. Scope of delivery 16
6. Storage 16
7. Mechanical installation/mounting 17
7.1 Installation in a panel by means of retaining clamps (retaining clamp version) 17
7.2 Installation in a panel with self-locking nuts (screw-in version) 18
7.2.1 Mounting sequence 18
7.3 Installation of the symbol inserts 19
7.4 Use inside closed vehicles 20
8. Electrical installation and interface operation 21
8.1 Electrical installation 21
8.1.1 Pinning 21
8.2 Booting and resetting behaviour 21
9. SAE J1939 communication protocol 23
9.1 Composition of the CAN Identifier 23
9.2 Keypad specific values 24
9.3 Installation in a network 24
9.4 Service Data – Proprietary A 25
9.4.1 Read Data object Request 25
9.4.2 Read data object reply OK 26
9.4.3 Read data object reply NOK 26
9.4.4 Write data object request 27
9.4.5 Write data object reply OK 28
9.4.6 Write data object reply NOK 28
9.5 Process Data Proprietary A and B 29
9.5.1 Process Data Proprietary A – Rx configuration messages 29
9.5.2 Process Data Proprietary B – Tx process data 29
9.6 Diagnostics 30
9.6.1 Current Module Temperature 30
9.6.2 Current voltage 31
9.6.3 Error handling 31
9.6.3.1 Active diagnostic trouble codes (DM1) 31

9.6.3.2 Previously active diagnostic trouble codes (DM2) 33
9.6.3.3 Diagnostic data clear/reset of previously active DTCs (DM3) 35
9.7 Save settings to Volatile/Non-Volatile memory 35
9.8 Custom Layer Settings 36
9.8.1 Manually configure the Baudrate 36
9.8.2 Manually configure the Node-ID 37
9.9 Basic Functions 37
9.9.1 Button press data 37
9.9.1.1 Stuck Button Time 37
9.9.1.2 Reconfigure Stuck Button Time 37
9.9.2 Symbol Illumination 38
9.9.2.1 Activating Symbol Illumination 38
9.9.2.2 Choosing the active brightness setting for the symbol LEDs 39
9.9.2.3 Changing the brightness setting for the symbol LEDs 40
9.9.2.4 Changing the temporal patterns for the symbol LEDs (flash modes) 40
9.9.2.5 Adjusting the timings in flash modes 41
9.9.2.6 Changing the global brightness for the symbol LEDs 41
9.9.3 Halo-Ring Illumination 42
9.9.3.1 Activating Halo lighting 42
9.9.3.2 Choosing the active brightness setting for the halo LEDs 43
9.9.3.3 Changing the brightness setting for the Halo LED illumination 44
9.9.3.4 Choosing the temporal patterns for the Halo LED illumination 45
9.9.3.5 Changing the global brightness for the Halo LED illumination 46
9.9.3.6 Choosing the active colour setting for the Halo LED illumination 46
9.9.3.7 Changing the colour setting for the Halo LED illumination 47
9.9.3.8 Adjusting the timings in flash modes 48
9.10 Changes between Software version 04.05.001 and 06.03.000 or newer 49
10. CANopen communication protocol 50
10.1 Composition of the CAN Identifier 50
10.2 Standard communication parameters 50
10.3 Installation in a CAN-network 50
10.4 SDO communication 52
10.5 PDO communication 53
10.5.1 PDO communication parameter 54
10.5.2 Mapping of PDOs 54
10.5.3 Automatic COB-ID update 56
10.6 Diagnostics, Error Codes 57

10.7 Save settings to Volatile and Non-Volatile memory 58
10.8 Custom Layer Settings 59
10.8.1 Manually configure the baud rate 59
10.8.2 Manually configure the Node-ID 60
10.9 Basic functions 60
10.9.1 Pressed Key data 60
10.9.1.1 Stuck Button Time 60
10.9.1.2 Reconfigure Stuck Button Time 61
10.9.2 Symbol Illumination 61
10.9.2.1 Activating Symbol Illumination 61
10.9.2.2 Choosing the active brightness setting for the symbol LED's 62
10.9.2.3 Changing the brightness setting for the symbol LED's 62
10.9.2.4 Changing the temporal patterns for the symbol LED Illumination 63
10.9.2.5 Changing the global brightness for the symbol LED Illumination 63
10.9.3 Halo-Ring Illumination 63
10.9.3.1 Activating Halo Ring Illumination 63
10.9.3.2 Choosing the active brightness setting for the Halo LED's 65
10.9.3.3 Changing the brightness setting for the Halo LED's 65
10.9.3.4 Choosing the temporal patterns for the Halo LED's for flashing modes 66
10.9.3.5 Changing the global brightness for the Halo LED's 66
10.9.3.6 Choosing the active colour setting for the Halo LED's 67
10.9.3.7 Changing the colour setting for the Halo LED's 67
10.9.3.8 Adjusting the timings in flash modes 68
10.10 Changes between Software version 04.05.001 and 06.03.000 or newer 69
11. Special functions 70
11.1 Sleep/Wakeup, Power Saving 70
11.1.1 Sleep/Wakeup over CAN 70
11.1.2 Sleep/Wakeup over 2 wire hardware interface 70
11.1.3 Sleep/Wakeup summary 72
11.2 Structure of lightning objects 72
11.2.1 Depending objects 73
11.2.2 Linking the objects 74
11.3 Communication timeout 76
12. CANopen-Safety communication protocol 76
13. Rotary Cursor controller 76
13.1 Communication Objects 77
13.2 Working with the RCC 78

14. Operation 78
14.1 General information 78
15. Cleaning 79
16. Optional accessories 79
17. Liability for quality defects 79
18. Service, repair 80
19. Decommissioning, disposal 80
20. Declaration of Conformity 80

List of abbreviations and units

CA	Controller Application
CAN	Controller Area Network
CiA	CAN in Automation
CLS	Custom Layer Setting
COB	Communication Object
CS	Command Specific
DLC	Data Length Count
DM	Diagnostic Message
EDS	Electronic Data Sheet
EOL	End Of Line
EU	European Union
HB	High Byte
HMI	Human-Machine-Interface
IP	International Protection
ISO	International Organization for Standardization
kbps	Kilo bit per second
LB	Low Byte
LED	Light emitting diode
LSB	Least Significant Bit
mm	milli meter
MOT	Ministry of Transport
MSB	Most Significant Bit
NMT	Network Management
NVM	Non Volatile Memory
PDO	Process Data Object
PDU	Payload Data Unit
PGN	Parameter Group Number

Prop A	Proprietary A
Prop B	Proprietary B
RCC	Rotary Cursor Controller
RGB	Red-Green-Blue
Rx	Receive
SAE	Society of Automotive Engineers
SDO	Service Data Object
SPN	Suspect Parameter Number
Tx	Transmit

Used symbols

⚠	**Caution!** Indicates a hazardous situation which, if not avoided, may result in a minor or moderate injury
⚠	**Attention** Describes information on installation which, if ignored, can lead to malfunctions
⚠	**Note** Indicates a situation which, if not avoided, may result in property damage
➤	Indicates an executive activity
☞	Indicates an application tip

1. Safety warnings

The safe system handling requires knowledge of the operating instructions.

Caution!

Connect the power supply in accordance with the safety regulations for electrical equipment.

- Risk of injury
- Damage to the keypad

Note

Avoid shocks and impacts to the keypad during installation

- Damage to or destruction of the keypad

The supply voltage must not exceed the specified limit.

- Damage to or destruction of the keypad

Protect the cable and the connector from damages

- Damage to or destruction of the keypad

Never kink the cable, do not bend the cable in tight radii. Dynamic movement of the cable should be avoided as far as possible, as well as chafing of the cable on system components.

- Damage to the cable or connector

1.1 Intended use

The modules were developed for applications in vehicles with MOT approval within the EU and for intelligent control with CAN bus integration. The robust, modular design with a protection degree of up to IP6K7 and the possibility of customer-specific adaptations and the arrangement of keypad symbols are facts that make the devices the best choice for harsh use in heavy-duty and special vehicles.

The keypads may only be operated within the parameters specified in the technical data.

The keypads must be used in such a way that no persons are endangered or machines damaged in the event of failure or malfunction.

Commissioning must be carried out by qualified personnel.

2. Proper environment

See Technical specification.

⚠	**Note** If possible, avoid abrupt changes in the operating temperature of the keypad, cable and connector. – Damage to the keypad, cable or connector

⚠	**Caution!** Do not operate the keypad in: – Potentially explosive atmospheres – Applications where the keypad, cable and connector are completely or partly submerged for extended periods of time – Situations in which the keypad, cable and connector are subjected to harsh external shocks and impacts – Do not use for remaining in final customer installations due to validation not completed – Risk of injury – Damage to the keypad, cable or connector

☞ The design of the keyboard (keys can still be pressed) means that keys are protected from freezing as long as the keys remain accessible and can be pressed. This means that the keyboard can also be used in snow and ice.

3. General description

Series 09 Rugged Keypads offer high reliability: The modules are designed for an intelligent control with CAN bus integration. The robust, modular design with protection degrees of up to IP6K7 and the possibility of customer-specific adaptations and the arrangement of symbol inserts predestine the devices for harsh use in heavy-duty and special vehicles. The modules are as well designed for ECE certification.

High reliability and functional safety are indispensable for the control of safety-relevant applications in vehicles and machines – whether in construction machinery, construction vehicles, agricultural machinery or in various types of special and commercial vehicles. Harsh environmental conditions and low back panel depth require a robust and compact product design. In addition, the control and signalling devices must be precisely configurable both mechanically and electronically for the respective application. The high-quality Rugged Keypad meets these requirements with a cutting-edge system integration.

Typical applications

Special vehicles such as fire engines, road sweepers, cleaning vehicles, dustcarts, snow clearing vehicles and snow groomers. Heavy-duty vehicles such as construction and agricultural vehicles

Advantages

- Individual 4-segment and RGB Halo Ring Illumination
- Intelligent HMIs with CAN bus integration
- Robust, ergonomic and innovative design with a protection degree of up to IP6K7 (mounted state front: IP6K7; back: IP20 without plugged connector). Protection degree for assembled situation in responsibility of customer application.
- Interchangeable ISO 7000 symbols or customer-specific symbols

Robust and innovative design

The design of the Rugged Keypads is characterised by a robust and innovative construction. The control and signalling devices, which are protected up to IP6K7, function reliably at an operating temperature of – 40 °C to + 85 °C. The low back panel depth and robust clip-in or screw-in mounting allow a flexible and easy installation, either vertically or horizontally. The high-quality devices also offer an excellent haptic and, thanks to the bright RGB LED halo and LED symbol illumination, are clearly visible in daylight and at night. An attractive and configurable 4-segment halo button illumination is integrated as standard.

The customisable illumination provides the operator with excellent visual feedback and is combined with a unique, contemporary design.

Durability

The series 09 CAN modules are produced in our automotive competence centre located in Germany. This allows us to apply our many years of comprehensive experience as an original equipment manufacturer (OEM) in the automotive industry to the heavy duty and special vehicles markets. At the same time, this offers EAO customers high quality, durable products and services. The development and production process is aligned and executed according to automotive standards, including qualified suppliers. This requirement ensures EAO high quality products and solutions.

CAN bus integration

Thanks to the CAN bus integration, the devices are integrated intelligently and easily into a CAN system – as standard with a Deutsch DT series connector. The device controls its function according to the CAN command.

Designed for E1 applications and CAN bus integration

The robust control units with flexible illumination are ideally suited for use in heavy duty and special vehicle applications.

4. Technical specification

Validation ongoing and not completed:

Mechanical characteristics

– Actuation force:	6,5 N
– Overload:	250 N
– Service life – Rugged Keypad:	up to 1 million cycles of operation

Electrical characteristics

– Operating voltage range:	8-32 VDC

Illumination

– LED symbol illumination – colour:	white LED
– LED halo ring illumination – colour:	RGB

Symbols

– Symbols in accordance with ISO 7000
– Customer-specific symbols on request

Connections/interfaces

– CAN interface (ISO 11898)	
– CAN protocols:	CANopen (CiA 401), SAE J1939
– Baudrate:	250 (default), 500 kb/s (software configurable)

Ambient conditions (validation not yet completed)

– Operating temperature:	-40 °C … +85 °C
– Storage temperature:	-40 °C … +85 °C

Protection degree
IP6K7 protection (mounted state front: IP6K7; back: IP20 without plugged connector). Protection degree for assembled situation in responsibility of customer application.

Attention

The protection degree of up to IP67 to be achieved depends on the front panel and type of mounting and must be ensured by the customer.

5. Scope of delivery

1 Rugged CAN keypad, with cable and connector (type Deutsch 6 pin)

Mounting material, depending on version

- Retaining clamp version: 6 retaining clamps graduated according to mounting panel thickness for 1mm - 4mm
- Screw-in version: 4 nuts each with washers and spacers and 1 sleeve

⚠	**Attention** For symbol inserts: quantity according to order, separate scope of delivery.

6. Storage

See Technical specification.

⚠	**Note** If possible, avoid abrupt changes in the storage temperature of the keypad cable and connector. – Damage to the keypad, cable or connector Do not expose the open contacts of the unprotected connector to condensing air humidity. – Damage to the keypad, cable or connector

Check the delivery immediately after unpacking with regard to completeness and transport damages.

If any damage or incompleteness is found, please contact the supplier immediately.

Optional accessories can be found in the annex Optional accessories.

7. Mechanical installation/mounting

2 mounting versions are available: Installation in a panel by means of retaining clamps (retaining clamp version) or screws (screw-in version).

7.1 Installation in a panel by means of retaining clamps (retaining clamp version)

See drawing 1707940001 Product Information.

If the keypad is mounted or dismounted several times, new retaining clamps must be used each time. This prevents the retaining clamps from settling. To change the retaining clamps, press the clamp flat on the underside, then you can push out and dismount the retaining clamp.

The marking of the retaining clamp for the respective front panel thickness is imprinted as number 1 to 4 on the clamp, which is also visible when mounted. The number 1 on the retaining clamp means that it is suitable for a front panel thickness of 1 mm. A mixed use of different types of retaining clamps in a keypad is not permitted.

The keypad shall be pressed evenly into the panel with the mounted retaining clamps, while tilting should be avoided. Press evenly on the housing and not on the buttons.

⚠	**Note** Damage to the buttons

The plug-in connection must be established with a suitable connector. The plug-in connection can alternatively be established before or after mounting the keypad into the panel.

The keypad is automatically centred to the panel by the 6 retaining clamps, i.e. the exact positioning of the keypad depends on the accuracy of the installation opening.

7.2 Installation in a panel with self-locking nuts (screw-in version)

See drawing 1707940001 Product Information.

If the keypad is mounted or dismounted several times, new self-locking nuts must be used each time.

Caution!

Exceeding the maximum permissible torque inevitably leads to the destruction of the keypad. The keypad is no longer tight nor leak proof in this case.

Incorrect mounting of the keypad, e.g. incorrect number of rubber washers or without washers (see 7.2.1 Mounting sequence), may result in damages to the keypad.

- Risk of injury
- Damage to the keypad
- Electric shock

The specified overload force refers to the switches, not to the mounting situation. For Rotary Cursor Controller it is recommended to use the screw-in variant.

- Risk of Unfastening

7.2.1 Mounting sequence

First of all, the rubber washers should be mounted according to the front panel thickness. No rubber washers are necessary for a front panel thickness of 1 mm. Mount 1 rubber washer per screw bolt for a front panel thickness of 2 mm, mount 2 rubber washers per screw bolt for a front panel thickness of 3 mm and mount 3 rubber washers per screw bolt for a front panel thickness of 4 mm. Since these have a slightly smaller inner diameter, they clamp lightly on the thread of the bolt to make it more difficult for the washers to fall off. Afterwards, the keypad has to be installed in the front panel, while the cable incl. connector has to be pushed through the sleeve. The sleeve can be installed rotated by 180° in each case. Afterwards, the 4 washers have to be installed over the screw bolts and fixed with the nuts. The correct tightening torque is defined in the 1707940001 Product Information and must be observed. The last step is to connect the plug connector.

The keypad of the screw-in version must be aligned manually to the panel. There is no automatic centering, the keypad can be slightly pushed into the mounting panel (depending on the opening size of the panel).

Caution!

For both versions, make sure that the opening for pressure and humidity compensation on the underside of the keypad is not closed or covered while installed.

– Damage to the keypad

The cables must not be bent during installation and the transition to the potting area must not be damaged. The minimum bending radius of R 6 mm for static bending and R 16 mm for dynamic bending (bending 10 times max.) must not be undercut. The plug connector needs to be fixed (e.g. by a DT Deutsch assembly clip) on application side to prevent the connector plug from free movement!

– Damage to the keypad

7.3 Installation of the symbol inserts

The symbol inserts can be changed in a panel both in the installed and in the non-installed state of the keypad.

Each symbol insert can be installed rotated in a 90° grid. When mounting the symbol inserts, make sure that the position of the symbols has the desired orientation towards the keypad. The symbol inserts have a mechanical coding to the housing of the keypad. A slight twisting of the symbol inserts to the keypad is technically possible and does not constitute a defect.

The symbol inserts must be mounted / dismounted under the edge of the housing around the insert using a tool without sharp edges (similar to mounting a vehicle tyre on a rim). The insert tool (article number: 09-0A00.0001), which is available as an accessory, is best suited for this. There should be no visible gap between the housing of the keypad and the symbol insert after installation, because dirt may penetrate here during operation and negatively influence the lighting function of the keypad.

⚠	**Caution!** When mounting and dismounting the symbol inserts, make sure that the housing is not damaged. The silicone of the housing must not be pierced, otherwise the protection degree of the keypad is not fulfilled. Similarly, the coating of the housing must not be damaged, otherwise the chemical resistance of the keypad is no longer guaranteed. – Damage to the keypad – Loss of tightness of the keypad It must be always ensured that the symbol inserts are mounted in the correct position. – Risk of injury

⚠	**Note** When changing the symbol inserts, make sure that there are no liquids or impurities between the housing and the symbol insert. These would negatively affect the illumination of the symbols.

When using the version for snap-in mounting in a mounting panel, you need retaining clamps.

When using the version for screw-in mounting in a mounting panel, you need the sleeve, nuts, washers and spacers to bridge the distance between keypad and sleeve in the thickness of the mounting panel.

Do not forget the spacers during mounting. Only hand-tighten the self-locking nuts.

– Damage to the keypad and/ or sleeve

⚠	**Note** Only use original mounting material and optional accessories. – Damage to the keypad

7.4 Use inside closed vehicles

Due to the product is designed for outdoor applications and not for use in closed vehicle interior, odor testing according odor testing specifications like VDA270 is in customer's responsibility. The test should be performed to customer application needs.

8. Electrical installation and interface operation

8.1 Electrical installation

Electrical installation shall only be done when the system is under no power.

Connection of the keypad module is realized with the 6-pin sealed Deutsch connector DT04-6P which is mounted to the cable.

⚠	**Caution!** The vehicle electrical system has to be equipped with a sufficiently selected suppressor diode to clip possible incoming overvoltage peaks before they arrive at the keypad according to DIN EN 16750-2 (Load Dump Test B) if a keypad with no load dump protection is selected. If a keypad with integrated load dump protection circuit is selected, an extern mounted suppressor diode are omitted. – Damaging to the keypad

8.1.1 Pinning

Please refer to 1707940001 Product Information.

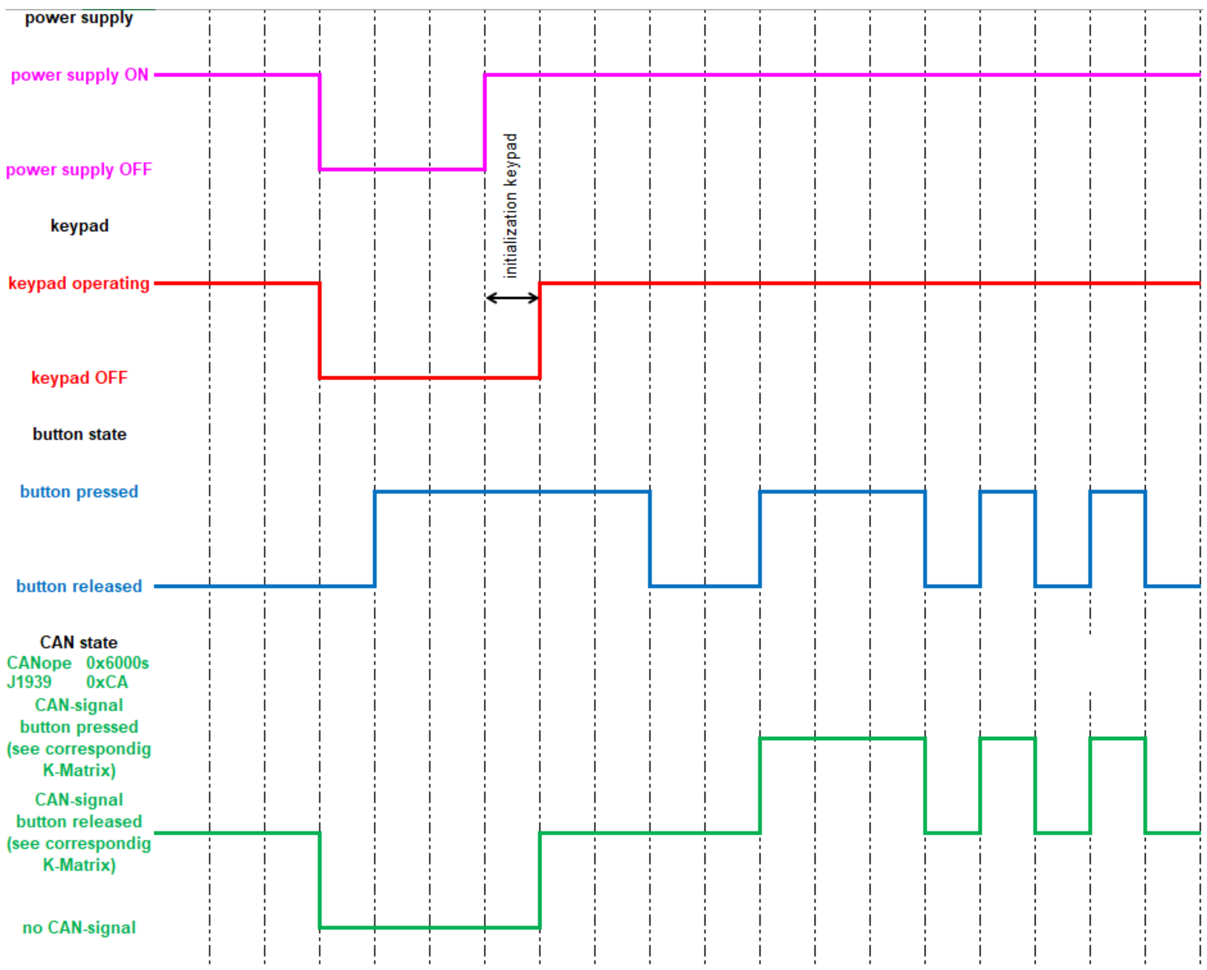
CAN bus termination on the keypad is not implemented by default. It has to be realized in the network externally according to ISO11898.

⚠	**Warning!** Cable must not be shortened. Connector must not be separated from the cable. – Damaging to the keypad – Loosing liability for defects

8.2 Booting and resetting behaviour

On first start-up and with a power reset the keypad will revert to its values that are saved in the non-volatile memory. All changes that were done to it and not saved in non-volatile memory will be reset. Certain objects of the keypad cannot be saved to non-volatile memory. That means that these values are reset with every power reset and will only be able to be activated by sending the corresponding CAN messages. All the objects that function in this way can be found in the K-Matrix. For example the objects that include the activation of the LEDs are always saved in volatile memory (after reset all LEDs will be turned off).

If a button is pressed before start-up or whilst a reset is performed. The button needs to be released and pressed again whilst the keypad is running for it to be detected.



Picture 1: Button State after Power-On Reset

9. SAE J1939 communication protocol

The most important document to understand the communication with the keypad is the K-Matrix. It includes a description of all available objects, how to address them and how to save them. To work with the K-Matrix please note that some objects have a ⊞ at the side. Which means that the row can be expanded and more information is then available.

This manual includes excerpts from the K-Matrix. All functions that the keypad has to offer can be found in the K-Matrix.

9.1 Composition of the CAN Identifier

In a J1939 network each CAN message has a 29-bit CAN identifier (CAN extended) which consists of a Parameter Group Number (PGN), a source address, a priority, a data page bit, an extended data page bit and a target address. It is composed as the following table represents:

	Bits 28 - 26	Bit 25	Bit 24	Bits 23 - 16	Bits 15 – 8	Bits 7 – 0
Explanation	Priority	Parameter Group Number (PGN)				Source address
		Reserved (EDP)	Data Page	PDU Format	PDU Specific (target address/ group extension)	
Peer-to-peer	See SAE J1939-21	See SAE J1939-21	See SAE J1939-21	0x00 – 0xEF	Target address	
Broadcast	See SAE J1939-21	See SAE J1939-21	See SAE J1939-21	0xF0 – 0xFF	Group extension	

The priority of the Can message is used to optimize traffic on the Bus-System. The lower the number, the higher the priority (000 → highest priority; 111 → lowest priority). Default values for customary or information based messages is 6 (0b110).

If the PDU format is less than 240, which means that the communication is set to peer-to-peer mode, the PDU specific field contains the target address.

If the PDU format is higher than 240, which means that the communication is set to broadcast mode, the PDU format field is forming the PGN together with the PDU specific field. The message is then sent to all members of the network.

The source address signals the address of the device that sent the CAN message.

The PGN uniquely defines the purpose of the message.

9.2 Keypad specific values

The default address of the keypad is set to 0x80 (0d128). It can be modified either by start up with the automatic Address claiming procedure (explained in 9.3), by Commanded Address message which is described in SAE J1939-81 or in our case with the proprietary Source Address Command (as described in 9.4.3).

The most important PGNs for communicating with the keypad are the Proprietary A and Proprietary B PGNs. All of the functions of the keypad are mapped to those. Basic functions are described in this manual. All details and all functions can be found in the corresponding K-Matrix.

9.3 Installation in a network

After connecting the keypad electronically to the network an address claim procedure is started automatically (as described in JAE J1939-81).

The software of a J1939 CAN device is the so called Controller Application (CA). Every CA is equipped with a unique address and an associated device name. To determine which CA sends a message each message contains a source address. There are predefined ranges for those addresses and they range from 0 to 255.

In the case of the Rugged Keypad a range of the source address is limited to the range between 128 and 247. Default value for the address is 128.

The address claim procedure sends a message with the corresponding parameter group number as specified by the J1939 standard with the desired source address and a 64 bit device name. This device name contains information about the Application and describes the main function. If there are other devices in the J1939 CAN network that already use the desired source address the device name with the higher priority claims the address.

Example of default address claim message as sent by the keypad module with default values:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EEFF80	8	0xFF	0xFF	0x1F	0x74	0x00	0x87	0x00	0x80

For the explanation which data byte contains which information please refer to the corresponding K-matrix or the SAE document SAE J1939-81. For practical use of the keypad it should be avoided to use the address claim functionality to manage the addresses of two or more keypads on the bus. It has to be taken into account that the keypads don’t keep their addresses assigned by address claim after a power-on-Reset. That means that the procedure of address claim starts after every power-on-reset and the keypads can be get different addresses each time they are powered on. The best way to use the keypad is to configure the node-ID as explained in 9.8.2.

If there is no address conflict detected the keypad will start with its normal communication and claims the source address 128 (0x80).

9.4 Service Data – Proprietary A

All access that is needed can be realized through the Prop A PGN. The composition of the CAN message to interact with the keypad accordingly is described in the following. How a message is composed also relies on the object the user wants to address. If the object is defined as an 8-bit data object it looks different than interacting with a 16- or 32- bit object. The definition of each object can be found in the K-Matrix.

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EF8000	8	CMD Byte		Index of Data object		Subindex of Data object		Content specific to addressed Data object	

9.4.1 Read Data object Request

- Keypad action on Rx:
  - If INDEX and SUBINDEX are readable objects, reply with a "Read data object reply OK" message.
  - If INDEX and SUBINDEX are not readable objects, reply with a "Read data object reply NOK" message.
- Transmitted by:
  - User Application Master
- Data bytes content:

BYTE #	8-bit Data	16-bit Data	32-bit Data
0	COMMAND (**0x00**)		
1	INDEX of returned Data Object		
2	SUBINDEX of returned Data Object		
3	Not used	Not used	Not used
4	Not used	Not used	Not used
5	Not used	Not used	Not used
6	Not used	Not used	Not used
7	Not used	Not used	Not used

- Example message read request for variant of the keypad (Index 0x08, Subindex 0x00):

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EF8000	8	**0x00**	0x08	0x00	0x00	0x00	0x00	0x00	0x00

9.4.2 Read data object reply OK

- Keypad action on Rx:
  - Ignore
- Transmitted by:
  - Keypad
- Data bytes content:

BYTE #	8-bit Data	16-bit Data	32-bit Data
0	COMMAND (**0x02**)		
1	INDEX of returned Data Object		
2	SUBINDEX of returned Data Object		
3	DATA	DATA[0] (LSB)	DATA[0] (LSB)
4	Not used (0xFF)	DATA[1] (MSB)	DATA[1]
5	Not used (0xFF)	Not used (0xFF)	DATA[2]
6	Not used (0xFF)	Not used (0xFF)	DATA[3] (MSB)
7	Not used (0xFF)	Not used (0xFF)	Not used (0xFF)

- Example message response to read request for variant of the keypad (Index 0x08, Subindex 0x00):

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EF0080	8	**0x02**	0x08	0x00	0xAB	0x06	0xFF	0xFF	0xFF

- Information in Databytes 3 and 4 ordered and put together:
  - 0x06AB → decimal: 1707 → the example is a keypad without RCC

9.4.3 Read data object reply NOK

- Keypad action on Rx:
  - Ignore
- Transmitted by:
  - Keypad
- Data bytes content:

BYTE #	8-bit Data	16-bit Data	32-bit Data
0	COMMAND (**0x03**)		
1	INDEX of returned Data Object		
2	SUBINDEX of returned Data Object		
3	Reject Reason		
4	Not used (0xFF)	Not used (0xFF)	Not used (0xFF)
5	Not used (0xFF)	Not used (0xFF)	Not used (0xFF)
6	Not used (0xFF)	Not used (0xFF)	Not used (0xFF)
7	Not used (0xFF)	Not used (0xFF)	Not used (0xFF)

– Example message response to read request for variant of the keypad (Index 0x08, Subindex 0x04):

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EF0080	8	**0x03**	0x08	0x04	0x09	0xFF	0xFF	0xFF	0xFF

The description for the reject reason code are defined in the K-Matrix.

9.4.4 Write data object request

– Keypad action on Rx:
  - If INDEX and SUBINDEX are writeable objects, reply with a "Write data object reply OK" message and write data into the indexed data object.
  - If INDEX and SUBINDEX are not writeable objects, reply with a "Write data object reply NOK" message.
– Transmitted by:
  - User Application Master
– Data bytes content:

BYTE #	8-bit Data	16-bit Data	32-bit Data
0	COMMAND (**0x04**)		
1	INDEX of returned Data Object		
2	SUBINDEX of returned Data Object		
3	DATA	DATA[0] (LSB)	DATA[0] (LSB)
4	Not used	DATA[1] (MSB)	DATA[1]
5	Not used	Not used	DATA[2]
6	Not used	Not used	DATA[3] (MSB)
7	Not used	Not used	Not used

– Example message write request for J1939 Node-ID (Index 0x1B, Subindex 0x00, new Node Address 0x85):

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EF8000	8	**0x04**	0x1B	0x00	0x85	0x00	0x00	0x00	0x00

9.4.5 Write data object reply OK

- Keypad action on Rx:
  - Ignore
- Transmitted by:
  - Keypad
- Data bytes content:

BYTE #	8-bit Data	16-bit Data	32-bit Data
0	COMMAND (**0x06**)		
1	INDEX of returned Data Object		
2	SUBINDEX of returned Data Object		
3	DATA	DATA[0] (LSB)	DATA[0] (LSB)
4	Not used (0xFF)	DATA[1] (MSB)	DATA[1]
5	Not used (0xFF)	Not used (0xFF)	DATA[2]
6	Not used (0xFF)	Not used (0xFF)	DATA[3] (MSB)
7	Not used (0xFF)	Not used (0xFF)	Not used (0xFF)

Example message response to write request for J1939 Node Address (Index 0x1B, Subindex 0x00, new Node Address 0x85):

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EF0080	8	**0x06**	0x1B	0x00	0x85	0xFF	0xFF	0xFF	0xFF

9.4.6 Write data object reply NOK

- Keypad action on Rx:
  - Ignore
- Transmitted by:
  - Keypad
- Data bytes content:

BYTE #	8-bit Data	16-bit Data	32-bit Data
0	COMMAND (**0x07**)		
1	INDEX of returned Data Object		
2	SUBINDEX of returned Data Object		
3	Reject Reason		
4	Not used (0xFF)	Not used (0xFF)	Not used (0xFF)
5	Not used (0xFF)	Not used (0xFF)	Not used (0xFF)
6	Not used (0xFF)	Not used (0xFF)	Not used (0xFF)
7	Not used (0xFF)	Not used (0xFF)	Not used (0xFF)

- protection register (Index 0x1B, Subindex 0x00, new Node Address 0x85):

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EF0080	8	**0x07**	0x1B	0x00	0x0B	0xFF	0xFF	0xFF	0xFF

9.5 Process Data Proprietary A and B

To make the setting of Lighting parameters and the reading of button states and so on easier, those basic functions are mapped by specific command bytes. All possibilities can be found in the K-Matrix.

9.5.1 Process Data Proprietary A – Rx configuration messages

To easily control all lighting capabilities of the keyboard, all of the basic functions can be accessed through specific command bytes on Prop A. All capabilities and how to address them can be found in the K-Matrix. The command bytes are stated in the following:

- 0x10 → basic lighting settings (Global brightness halos and symbols, activate halos and symbols)
- 0x11 → active colour setting halos
- 0x12 → active brightness setting halos
- 0x13 → active brightness setting symbols
- 0x14 → temporal pattern halos
- 0x15 → temporal pattern symbols
- 0x16 → activate halo LEDs

- Example message setting the global brightness of all LEDs to max (250) and activating all LEDs

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EF8000	8	**0x10**	0xFA	0xFA	0x3F	0x3F	0x00	0x00	0x00

0x3F → the LEDs for the 6 buttons are mapped bitwise, so 0x3F means all buttons on because in binary it is a 0b00111111

9.5.2 Process Data Proprietary B – Tx process data

The keypad is cyclically transmitting messages on the PGN Prop B to make the reading of button states, the current temperature and a message counter with Checksum as easy as possible.

How this message is composed is explained in the following. The K-Matrix specifies exactly the values in each byte and what they stand for.

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18FF0080	8	Button pressed state for Buttons 1-4	Button pressed state for Buttons 5-6	Button stuck error state for Buttons 1-4	Button stuck error state for Buttons 5-6	Not available 0xFF	Current temper-ature	Not available 0xFF	4-bit Message Counter and 4- Bit Checksum

9.6 Diagnostics

The keypad supports the Diagnostic Trouble Codes as specified in SAE J1939-73. The corresponding diagnostic message PGNs (DM1, DM2, DM3) can be found in the K-Matrix and are specified if they need to be read via read command or are submitted cyclically. The error codes are listed in the K-matrix too and are shown in the table below:

Error code	Name	Description
0x7F000	No error	
0x7F250	Overvoltage	Input voltage exceeds 33 volts
0x7F250	Undervoltage	Input voltage falls below 8 volts
0x7F030	Overtemperature warning	Keypad temperature exceeds 100°C
0x7F040	Overtemperature error	Keypad temperature exceeds 125 °C
0x7F050	Temperature sensor defect	Implausible temperature values are measured
0x7F070	Button stuck	Button is pressed longer than the time value defined in object 0x2101 (default value: 10s)
0x7F110	Button pressed at startup	One or more buttons are pressed during power-on-reset
0x7F080	Button unstable	Switching element bounces excessively; reason: switching system worn out
0x7F100	Button out of range	Resistance of the switching system reach upper limit; reason: switching system worn out
0x7F120	Button crosstalk	Button was detected as pressed although a other button should be pressed; detection over cyclically detuning the voltage divider of each button; reason: incoming moisture in the keypad
0x7F130	Button test low	Cyclically detuning of voltage divider of each button returns implausible values; reason: incoming moisture in the keypad, switching system worn out

The current module temperature and the current voltage is measured by the keypad and can be read via a read request according to chapter 9.4.

9.6.1 Current Module Temperature

The object for storing the temperature is object number 200 (0xC8) and it is also cyclically transmitted by the Process Data Prop B as described in chapter 9.5.2. The value transmitted is a hexadecimal value. To calculate the actual temperature it has to be transferred into decimal and 40 needs to be subtracted.

Example for calculating the current temperature value:

Given value hexadecimal	Given value decimal	Subtract by	Actual temperature
0x49	73	40	33 °C

9.6.2 Current voltage

The object for storing the voltage is object number 201 (0xC9). The value is a 16 Bit hexadecimal value LSB first in data bytes 3 and 4. The value needs to be transferred into decimal and then shows the current voltage in mV.

Example for calculating the current Voltage:

Given value hex byte 3	Given value hex byte 4	Value ordered and in decimal	Actual voltage
0xEB	0x34	13547 mV	13,547 V

9.6.3 Error handling

9.6.3.1 Active diagnostic trouble codes (DM1)

For the active diagnostic trouble codes two ways of communication are possible:

– Only one active diagnostic trouble code:

The DM1-message is send as shown:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18FECA80	8	0x00	0x00	TC1	TC2	TC3	TC4	0xFF	0xFF

The data field contains the trouble Code in data Byte [2-5]. The data Byte [0, 1, 6, 7] are not used by the keypad and send as 0x00 [0, 1] or [6, 7]. A detailed structure of the Trouble Code is shown below:

Bit	TC1	TC2	TC3	TC4
0	SPN[0]	SPN[8]	FMI[0]	OC[0]
1	SPN[1]	SPN[9]	FMI[1]	OC[1]
2	SPN[2]	SPN[10]	FMI[2]	OC[2]
3	SPN[3]	SPN[11]	FMI[3]	OC[3]
4	SPN[4]	SPN[12]	FMI[4]	OC[4]
5	SPN[5]	SPN[13]	SPN[16]	OC[5]
6	SPN[6]	SPN[14]	SPN[17]	OC[6]
7	SPN[7]	SPN[15]	SPN[18]	0

TC1 contains the Low Byte of the SPN and TC2 the second Byte. The Bit 16, 17 and 18 of the SPN are Bit 5, 6 and 7 of TC3. The SPN values of all possible errors are shown in the J1939 K-matrix for the rugged keypad. TC3 Bit 0-4 contains the Failure Mode Identification Code (FMI). The 32 possible FMI codes are described in SAE J1939-73. TC4 contains the occurrence count of the active error. The maximum value of occurrence count is 127. Bit 7 of TC4 contains the SPN conversion method and is set to 0.

This message is cyclically send every second from the keypad. The cyclic time is a protocol standard and can't be changed. If no error is active, the data bytes [0-7] of the message contains 0x00.

Example 1: Undervoltage Error (0x7F260)

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18FECA 80 (DM1)	8	0x00	0x00	0x60	0xF2	0xE4	0x01	0xFF	0xFF

The FMI for the example contains the value 0x04. According to SAE J1939-73 the value 0x04 stands for: "voltage below normal". The occurrence count is 0x01. If the error occurs for a second time, the occurrence count will be incremented.

Example 2: Stuck Button Error (0x7F070); Button 4 Stuck

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18FECA 80 (DM1)	8	0x00	0x00	0x74	0xF0	0xE2	0x05	0xFF	0xFF

The Button Number is added to the general stuck button SPN 0x7F070. The result is SPN 0x7F074 for button 4 stuck. The FMI is 0x02, that means: "data erratic, intermittent or incorrect". The error occurs for the 5th time.

– Two or more active diagnostic trouble codes:

If two or more diagnostic trouble codes are active, the trouble codes will be send over the transport protocol. The following table shows the messages:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x1CECFF80 (TP.CM_BAM)	8	0x20	Byte Count LSB	Byte Count MSB	Message Count	0xFF	0xCA	0xFE	0x00
0x1CEBFF80 (TP.DT)	8	Message number	Message Data 0	Message Data 1	Message Data 2	Message Data 3	Message Data 4	Message Data 5	Message Data 6
0x1CEBFF80 (TP.DT)	8	Message number	Message Data 7	Message Data 8	Message Data 9	Message Data 10	Message Data 11	Message Data 12	Message Data 13

The data transmission starts with a broadcast announce message send by the keypad to signalize an upcoming transmission of DM1 (data byte 5-7 of TP.CM_BAM contains the PGN of DM1). The message contains the byte count of DM1 (16-bit value) and a message count (8-bit value). After the broadcast announce message, data transfer messages are send by the keypad. The first data byte of TP.DT contains the message number, the following 7 byte are payload.

Example: Undervoltage Error (0x7F260), Button 3 Stuck Error (0x7F073), Button 4 Stuck Error (0x7F074), Button 5 Stuck Error (0x7F075)

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x1CECFF80 (TP.CM_BAM)	8	0x20	0x12	0x00	0x03	0xFF	0xCA	0xFE	0x00
0x1CEBFF80 (TP.DT)	8	0x01	0x00	0x00	0x73	0xF0	0xE2	0x03	0x74
0x1CEBFF80 (TP.DT)	8	0x02	0xF0	0xE2	0x07	0x75	0xF0	0xE2	0x03
0x1CEBFF80 (TP.DT)	8	0x03	0x60	0xF2	0xE4	0x03	0xFF	0xFF	0xFF

The 18 data bytes (data byte 1-2 in TP.CM_BAM) are packed in three messages (data byte 3 in TP.CM_BAM). The trouble codes are packed in the following format:

– 0x00, 0x00, X, X, X, X, Y, Y, Y, Y, …, 0xFF, 0xFF, …

The data starts with two bytes 0x00 followed by the trouble codes. X and Y are symbolic for a single trouble code. After the last trouble code, the payload of the last data transfer message are filled up with 0xff to get an 8 byte data field.

9.6.3.2 Previously active diagnostic trouble codes (DM2)

The keypad saves every occurred error in an internal memory that can be read out over a request. Same as in case DM1 it is necessary to decide between two possible ways of communication:

– Only one previously active diagnostic trouble code:

If one error are stored in the internal memory, the data transmission are showed in the following table:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EA80FF (Request)	3	0xCB	0xFE	0x00	---	---	---	---	---
0x18FECB80 (DM2)	8	0x00	0x00	TC1	TC2	TC3	TC4	0xFF	0xFF

The DM2 message (PGN = 0x18FECB80) takes the same payload as DM1 message but they are not send automatically. The DM2 message must be requested with a request message.

Example: Undervoltage Error (0x7f260, Count = 3)

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EA80FF (Request)	3	0xCB	0xFE	0x00	---	---	---	---	---
0x18FECB80 (DM2)	8	0x00	0x00	0x60	0xF2	0xE4	0x03	0xFF	0xFF

– Two or more previously active diagnostic trouble codes:

If two or more error are stored in the error memory, the data transmission occurs in the same way as a DM1 transmission with two or more errors over the data transfer protocol:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EA80FF (Request)	3	0xCB	0xFE	0x00	---	---	---	---	---
0x1CECFF80 (TP.CM_RTS)	8	0x10	Byte Count LSB	Byte Count MSB	Message Count	0xFF	0xCB	0xFE	0x00
0x1CEC80FF (TP.CM_CTS)	8	0x11	Packet Count	Next Package	0xFF	0xFF	0xCB	0xFE	0x00
0x1CEBFF80 (TP.DT)	8	Message number	Message Data 0	Message Data 1	Message Data 2	Message Data 3	Message Data 4	Message Data 5	Message Data 6
0x1CEBFF80 (TP.DT)	8	Message number	Message Data 7	Message Data 8	Message Data 9	Message Data 10	Message Data 11	Message Data 12	Message Data 13
0x1CEC80FF (TP.CM_End OfMsgACK)	8	0x13	Byte Count LSB	Byte Count MSB	Message Count	0xFF	0xCB	0xFE	0x00

The transmission must be requested over a request message with PGN of DM2 in the payload. Then the keypad send a ready to send answer with the byte count and the message count of the transmission. Before the keypad starts transmitting DM2 it expects a clear to send message. After the data transfer the keypad expects an end of message acknowledgment. Without an acknowledgment the keypad sends a connection abort message, shown in the following table:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x1CECFF80 (TP.Conn_Abort)	3	0xFF	0x03	0xFF	0xFF	0xFF	0xCB	0xFE	0x00

The first data byte of TP.Conn_Abort identify the connection abort. The data byte 1 describe the reason of the connection abort and is specified in the document SAE J1939-21. In case that the TP.CM_EndOfMsgACK are not received by the keypad, the reason code is 0x03 (a timeout occurred and this is the connection abort to close the session).

Example: Overvoltage Error (0x7F250, Count = 1), Button 2 Stuck Error (0x7F072, Count = 2), Button 0 Stuck Error (0x7F070, Count = 2)

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EA80FF (Request)	3	0xCB	0xFE	0x00	---	---	---	---	---
0x1CECFF80 (TP.CM_RTS)	8	0x10	0x0E	0x00	0x02	0x04	0xCB	0xFE	0x00
0x1CEC80FF (TP.CM_CTS)	8	0x11	0x01	0x01	0xFF	0xFF	0xCB	0xFE	0x00
0x1CEBFF80 (TP.DT)	8	0x01	0x00	0x00	0x50	0xF2	0xE3	0x01	0x72
0x1CEBFF80 (TP.DT)	8	0x02	0xF0	0xE2	0x02	0x70	0xF0	0xE2	0x02
0x1CEC80FF (TP.CM_End OfMsgACK)	8	0x13	0x0E	0x00	0x02	0xFF	0xCB	0xFE	0x00

The FMI of the first trouble code (overvoltage error) are 0x03. That means: “voltage above normal, or shorted to high source”. The payload are exactly 14 byte and the second data transfer message are not filled up with 0xFF at the end.

9.6.3.3 Diagnostic data clear/reset of previously active DTCs (DM3)

For resetting all stored error the following messages are used:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EA80 FF (Request)	3	0xCC	0xFE	0x00	---	---	---	---	---
0x18E8FF 80(ACKM)	8	0x00	0x00	0x00	0x00	0x00	0xCC	0xFE	0x00

The request message contains the PGN 0x00FECC (DM3) in the data field. After the stored errors are cleared, the keypad sends an acknowledgment message with the PGN of DM3.

9.7 Save settings to Volatile/Non-Volatile memory

By default all changes written in the objects like lighting, node address and so on are stored in a volatile memory after writing. This means that with a voltage reset these changes will not take any effect any more. The keypad has the ability to store settings in the non-volatile memory to implement them as fully available after a reset of the module.

Saving into the non-volatile memory is done by writing into the object number 5 (0x05). The value written into the object depends on the group to which the setting in question belongs to. Please refer to the K-Matrix to see which setting belongs to which save group.

How to write in the Object is described in chapter 9.4.3.

Example message for saving all data belonging to the Network Save Group (e.g. Node Address).

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EF8000	8	0x04	0x05	0x00	0xA4	0x00	0x00	0x00	0x00

Some changes (e.g. Node Address) take effect after voltage reset of the keypad.

9.8 Custom Layer Settings

The Baudrate and the Node Address can be changed by the user. The default values are as follows:

Object	Default value
baud rate (0x09)	0x03 (250 kbps)
Node-ID (0x1B)	0x80 (128)

The objects with their respective description including the corresponding safe/clear group can be found in the K-Matrix. To change the values the user has to send a write data object request (see 9.4.3) to the keypad with the corresponding object. After the keypad accepted the change and responded with a write data object reply OK message the values have to be stored in non-volatile memory (see 9.7). After the values are stored correctly a power reset has to be performed. After that the keypad will be set to the new settings and communication will be able only by using the changed values.

9.8.1 Manually configure the Baudrate

As an example the following table shows the J1939 messages that are necessary for changing the Baudrate from 250 kbps (default value) to 500 kbps.

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x09	0x00	0x02	0x00	0x00	0x00	0xFF
0x18EF0080	8	0x06	0x09	0x00	0x02	0x00	0x00	0x00	0xFF
0x18EFFF00	8	0x04	0x05	0x00	0xA4	0x00	0x00	0x00	0x00
0x18EF0080	8	0x06	0x05	0x00	0xA4	0x00	0x00	0x00	0xFF

First step for changing the baud rate is to send the new baud rate to the keypad with a Prop A message defined in J1939-21. Data byte 3 represents the new baud rate (0x02 = 500kbps; 0x03 = 250kbps), data byte 1 the index of the baud rate and data byte 0 the command (Write data object request). After the keypad receives a valid baud rate setting, the keypad send a “write data object reply OK”- message (data byte 0 = 0x06).

Second step is to save the new baud rate to the non-volatile memory. The index of the store-NVM-object is 0x05 (data byte 1). The data byte 3 contains the save group. Referenced to the J1939 K-Matrix of the keypad the baud rate uses the save group “Network” (0xA4). After the write request the keypad sends a “write data object reply OK”- message.

Third step is to disconnect the keypad from the power supply and do a power-on-reset. After that the keypad uses the new baud rate 500kbps for communication.

9.8.2 Manually configure the Node-ID

As an example the following table shows the J1939 messages that are necessary for changing the Node-ID from 0x80 (default value) to 0x85.

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x1B	0x00	0x85	0x00	0x00	0x00	0xFF
0x18EF0080	8	0x06	0x1B	0x00	0x85	0x00	0x00	0x00	0xFF
0x18EFFF00	8	0x04	0x05	0x00	0xA4	0x00	0x00	0x00	0x00
0x18EF0080	8	0x06	0x05	0x00	0xA4	0x00	0x00	0x00	0xFF

First step for changing the Node-ID is to send the new ID to the keypad with a Prop A message defined in J1939-21. Data byte 3 represents the new Node-ID, Data byte 1 the Index of the Node-ID and data byte 0 the command (write data object request). After the keypad receives a valid Node-ID, the keypad send a “write data object reply OK”- message (Data byte 0 = 0x06).

Second step is to save the new node-ID to the non-volatile memory. The index of the store-NVM-object is 0x05 (data byte 1). The data byte 3 contains the save group. Referenced to the J1939 K-Matrix of the keypad the Node-ID uses the save group “Network” (0xA4). After the write request the keypad sends a “write data object reply OK”- message.

Third step is to disconnect the keypad from the power supply and do a Power-On-Reset. After that the keypad uses the new Node-ID 0x85 for communication.

9.9 Basic Functions

9.9.1 Button press data

To read from the device in case a button is pressed, the device sends the Prop B message cyclically as described in chapter 9.5.2.

The state for each button can also be read by a read command as described in 9.4. The information needed can be found in the object with the index 202 (0xCA). Sub-Indexes 0 to 5 correspond to the Buttons 1 – 6.

For further information please refer to the corresponding K-Matrix.

9.9.1.1 Stuck Button Time

The Keypad provide the option to detect a continuous pressed button and send an error message after a defined time. This so called stuck button detection will be controlled via the stuck button detection time object (Object index: 0xA2)

9.9.1.2 Reconfigure Stuck Button Time

The keypads with software version 04.05.001 have no possibility to reconfigure the stuck button time. The values is set fixed to 10 seconds and can only be changed during production process.

Keypads with software version 06.03.000 or newer provide the possibility to reconfigure the stuck button time in field. The following table shows an example for reconfigure the stuck button time to 3 seconds and save the new stuck button time:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0xA2	0x00	0xB8	0x0B	0x00	0x00	0xFF
0x18EF0080	8	0x06	0xA2	0x00	0xB8	0x0B	0x00	0x00	0xFF
0x18EFFF00	8	0x04	0x05	0x00	0xA0	0x00	0x00	0x00	0x00
0x18EF0080	8	0x06	0x05	0x00	0xA0	0x00	0x00	0x00	0xFF

Data byte 3 contains the low byte and data byte 4 the high byte of the 16-bit value stuck button time.

9.9.2 Symbol Illumination

The symbols of the keypad are illuminated with white LEDs. In the following the corresponding Objects will be described to turn them on and how to use all the functions the keypad has to offer. Writing of values will not be described as it is already stated in chapter 9.4.3. All changes done in the object will be reset after voltage reset if not stored in non-volatile memory according to chapter 9.7.

9.9.2.1 Activating Symbol Illumination

There are 2 ways to activate the lighting of the symbols. In both variants it is mandatory to know that the LEDs for each button can be activated or deactivated exclusively. This is managed by bitcoding the LEDs for the buttons. Writing a 1 to the desired bit-position the LED will be turned on, writing a 0 will turn it off. By adding the bit-values of the buttons together it is possible to activate any desired pattern of the symbols.

Button 1	Button 2	Button 3
0b000001 (0x01)	0b000010 (0x02)	0b000100 (0x04)
Button 4	Button 5	Button 6
0b001000 (0x08)	0b010000 (0x10)	0b100000 (0x20)

1. Activation through Process data Prop A config message
   This is described in chapter 9.5.1. The corresponding message identifier is 0x10. And the Data byte containing the information for the symbol LEDs is Byte 4. This can be found in the corresponding K-Matrix as well.
   Example data field for turning on the symbol LEDs of the first and third button:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x10	0x00	0x00	0x00	0x05	0x00	0x00	0x00

2. <u>Activation through writing in the corresponding object with a write request</u>
   There is an object described in the K-Matrix that stores the information for the activated/deactivated symbol LEDs. The object in question is number 108 (0x6C). Writing the corresponding bit-value into this object will lead to the same result as described in chapter 9.4.3.
   Example data field for turning on the symbol LEDs of the first and third button:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x6C	0x00	0x05	0x00	0x00	0x00	0x00

9.9.2.2 Choosing the active brightness setting for the symbol LEDs

There are 3 predefined brightness settings available to choose from in the keypad. The K-Matrix defines the object and which value stands for which setting. Subindex 0 to 5 stands for the buttons 1 to 6.

Value	Setting name	Setting description
0x00	Setting 0 (even)	Brightness for LEDs button wise
0x01	Setting 1 (even)	Brightness for LEDs button wise
0x02	Setting 2 (individual)	Brightness for LEDs individually (corresponds to Halo LEDs, as the symbol is illuminated with only one LED, see also chapter 9.9.3)

There are 2 ways to set the active brightness setting for the buttons.

1. <u>Activation through Process data Prop A config message</u>
   This is described in 9.5.1. The corresponding message identifier is 0x13. The data bytes 1 through 6 are corresponding to buttons 1 through 6. This can be found in the K-matrix as well.
   Example data field for setting the active brightness setting of all button symbol LEDs to setting 1:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x13	0x01	0x01	0x01	0x01	0x01	0x01	0x00

2. <u>Activation through writing in the corresponding object with a write request</u>
   There are objects described in the K-Matrix that store the information for the active brightness setting. The object in question is number 101 (0x65). The subindexes 0 through 5 correspond with the buttons 1 to 6. Writing an object is described in 9.4.3.
   Example data field for setting the active brightness setting for button 2 to Setting 1:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x65	0x01	0x01	0x00	0x00	0x00	0x00

9.9.2.3 Changing the brightness setting for the symbol LEDs

Each of the aforementioned brightness settings can be accessed through objects as described in the K-Matrix.

Object	Setting name	Setting description
59 (0x3B)	Setting 0 (even)	Brightness for LEDs button wise
60 (0x3C)	Setting 1 (even)	Brightness for LEDs button wise
61 (0x3D)	Setting 2 (individual)	Brightness for LEDs individually (corresponds to Halo LEDs, as the symbol is illuminated with only one LED)

The subindex for Setting 0 and 1 are mapped from 0 to 5 corresponding to the buttons 1 to 6. You can find the mapping for Setting 2 in the K-Matrix. Writing an object is described in chapter 9.4.3.

Example data field for setting the brightness in Setting 1 for button 2 to 20% (0x32):

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x3C	0x01	0x32	0x00	0x00	0x00	0x00

9.9.2.4 Changing the temporal patterns for the symbol LEDs (flash modes)

There are 4 predefined temporal patterns available for the symbol LEDs.

Value	Setting name	Setting description
0x00	Steady	LED always on
0x01	Flash slow	LED flashing on and off with default value times
0x02	Flash fast	LED flashing on and off with default value times faster than 0x01
0x03	Pulsate	LEDs slowly turning off and on with default value times

There are 2 ways to set the temporal pattern setting for the buttons.

1. Activation through Process data Prop A config message
   This is described in 9.5.1. The corresponding message identifier is 0x15. The data bytes 1 through 6 are corresponding to buttons 1 through 6. This can be found in the K-Matrix as well.
   Example data field for setting the temporal pattern of all button symbol LEDs to setting 1:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x15	0x01	0x01	0x01	0x01	0x01	0x01	0x00

2. Activation through writing in the corresponding object with a write request
   There is an object described in the K-Matrix that stores the information for the temporal pattern setting for the symbol LEDs. The object in question is number 107 (0x6B). The subindexes 0 through 5 correspond with the buttons 1 to 6. Writing an object is described in 9.4.3.
   Example data field for setting the temporal pattern for button 2 to Setting 1:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x6B	0x01	0x01	0x00	0x00	0x00	0x00

9.9.2.5 Adjusting the timings in flash modes

The 2 flashing modes described in chapter 9.9.2.4 can be adjusted by the user. This means that the timings of how long the LED stays on and the period when it is turned on can be adjusted. For this there are 4 objects defined in the K-Matrix:

Object	Object name	Object description
41 (0x29)	Flash_Slow_On_Time	Timing for how long the LED is on in Flash mode slow
42 (0x2A)	Flash_Slow_Period_ms	Time after which the LED is turned on in Flash mode slow
43 (0x2B)	Flash_Fast_On_Time	Timing for how long the LED is on in Flash mode fast
44 (0x2C)	Flash_Fast_Period_ms	Time after which the LED is turned on in Flash mode fast

There are default values defined for each object in the K-Matrix. But each value can be changed by hand. The values can be edited by means of a Prop A config message as described in 9.5.1.

Example data fields for setting the timing of the fast mode to 20ms LED on after each 50ms:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x2B	0x00	0x14	0x00	0x00	0x00	0x00
0x18EFFF00	8	0x04	0x2C	0x00	0x32	0x00	0x00	0x00	0x00

9.9.2.6 Changing the global brightness for the symbol LEDs

The global brightness setting is a setting that dims the LEDs above all other settings. So to get realistic values in the aforementioned settings the global brightness needs to be set to 100%. The brightness is coded in 250 steps from 0 (0%) to 250 (100%). The setting of global brightness always affects all Symbol LEDs at once.

There are 2 ways to set the global brightness for the symbol LEDs.

1. Activation through Process data Prop A config message
   This is described in 9.5.1. The corresponding message identifier is 0x10. The data byte that accesses this information is Byte number 2.
   Example data field for setting the global brightness of the symbol LEDs to 20%:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x10	0x00	0x32	0x00	0x00	0x00	0x00	0x00

2. Activation through writing in the corresponding object with a write request
   There is an object described in the K-Matrix that stores the information for the global brightness setting for the symbol LEDs. The object in question is number 103 (0x67). Writing an object is described in 9.4.3.
   Example data field for setting the global brightness for the symbol LEDs to 20%:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x67	0x00	0x32	0x00	0x00	0x00	0x00

9.9.3 Halo-Ring Illumination

The Halo-Rings of the keypad are illuminated with 4 RGB LEDs for each button. In the following the corresponding objects will be described to turn them on and how to use all the functions the keypad has to offer. Writing of values will not be described as it is already stated in chapter 9.4.3.

All changes done in the object will be reset after voltage reset if not stored in non-volatile memory according to chapter 9.7.

9.9.3.1 Activating Halo lighting

As there are 4 LEDs per button it is important to know that there are always 2 steps needed to activate the halo LEDs. In the first step it is set which of the 4 LEDs are supposed to be turned on. And in the second step the halo for the button in question is activated.

1. First step: setting the LEDs that need to be turned on per button
   This is done through writing in the corresponding area in Process data Prop A. How this is done is described in 9.5.1. The corresponding message identifier is 0x16. The data bytes 1 through 6 correspond with the buttons 1 to 6. The values of each data byte are bit coded for the 4 LEDs. The value written will either turn the LED on or off. 1 means on and 0 means off. By adding the Bit-Values of the LEDs together it is possible to activate any desired pattern of the LEDs.
   They are coded as follows:

Buttons 1 - 6		
Bit 3		Bit 0
Bit 2		Bit 1

Example data field for activating all four LEDs of the halo of button 2:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x16	0x00	0x0F	0x00	0x00	0x00	0x00	0x00

2. Second step: activating the halo for the button in question
   There are now 2 ways of achieving this.
   a. Activation through Process data Prop A config message
      This is described in 9.5.1. The corresponding message identifier is 0x10. And the Data byte containing the information for the symbol LEDs is Byte 3. This can be found in the K-Matrix as well. The value written in the corresponding byte is bit-coded similarly to the symbol activation described in 9.9.2.1.
      Example data field for turning on the halo for button 2:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x10	0x00	0x00	0x02	0x00	0x00	0x00	0x00

   b. Activation through writing in the corresponding object over a write request
      There is an object described in the K-Matrix that stores the information for the activated/deactivated halo LEDs. The object in question is number 104 (0x68). Writing the corresponding bit-value into this object will lead to the same result as described in a. the subindex 0 to 5 correspond with the buttons 1 – 6. The value written in the corresponding byte is bit-coded similarly to the symbol activation described in chapter 9.9.2.1.
      Writing an object is described in chapter 9.4.3.
      Example data field for turning on the complete halo for button 2:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x68	0x01	0x0F	0x00	0x00	0x00	0x00

9.9.3.2 Choosing the active brightness setting for the halo LEDs

There are 3 predefined brightness settings available to choose from in the keypad. The K-Matrix defines the object and which value stands for which setting. Subindex 0 to 5 stand for the buttons 1 to 6.

Value	Setting name	Setting description
0x00	Setting 0 (even)	Brightness for LEDs button wise
0x01	Setting 1 (even)	Brightness for LEDs button wise
0x02	Setting 2 (individual)	Brightness for LEDs individually

There are 2 ways to set the active brightness setting for the buttons.

1. Activation through Process data Prop A config message
   This is described in 9.5.1. The corresponding message identifier is 0x12. The data bytes 1 through 6 are corresponding to buttons 1 through 6. This can be found in the K-Matrix as well.
   Example data field for setting the active brightness setting of all button halo LEDs to setting 1:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x12	0x01	0x01	0x01	0x01	0x01	0x01	0x00

2. Activation through writing in the corresponding object with a write request
   There is an object described in the K-Matrix that stores the information for the active brightness setting. The object in question is number 100 (0x64). The subindex 0 through 5 correspond with the buttons 1 to 6. Writing an object is described in 9.4.3.
   Example data field for setting the active brightness setting for button 2 to Setting 1:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x64	0x01	0x01	0x00	0x00	0x00	0x00

9.9.3.3 Changing the brightness setting for the Halo LED illumination

Each of the aforementioned brightness settings can be accessed through objects as described in the K-Matrix.

Object	Setting name	Setting description
57 (0x39)	Setting 0 (even)	Brightness for LEDs button wise
58 (0x3A)	Setting 1 (even)	Brightness for LEDs button wise
61 (0x3D)	Setting 2 (individual)	Brightness for LEDs individually

The subindex for Setting 0 and 1 are mapped from 0 to 5 corresponding to the buttons 1 to 6. You can find the mapping for Setting 2 in the K-Matrix. Writing an object is described in chapter 9.4.3.

Example data field for setting the brightness in Setting 1 for button 2 to 50% (0x7D):

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x3A	0x01	0x7D	0x00	0x00	0x00	0x00

The individual setting number 2 provides the possibility to adjust brightness values for each LED alone. The LEDs are mapped to the sub index of the corresponding object (0x3D) as described in the K-Matrix.

Example data field for setting the brightness in Setting 2 for the top left LED of button 2 to 20% (0x32):

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x3D	0x0C	0x32	0x00	0x00	0x00	0x00

9.9.3.4 Choosing the temporal patterns for the Halo LED illumination

There are 5 predefined temporal patterns available for the halo LEDs.

Value	Setting name	Setting description
0x00	Steady	LED always on
0x01	Flash slow	LED flashing on and off with default value times
0x02	Flash fast	LED flashing on and off with default value times faster than 0x01
0x03	Pulsate	LEDs slowly turning off and on with default value times
0x04	Rotate	4 halo LEDs are slowly turning off one by one to visualize a rotating ring

There are 2 ways to set the temporal pattern setting for the buttons.

1. <u>Activation through Process data Prop A config message</u>
   This is described in 9.5.1. The corresponding message identifier is 0x14. The data bytes 1 through 6 are corresponding to buttons 1 through 6. This can be found in the K-Matrix as well.
   Example data field for setting the temporal pattern of all button halo LEDs to setting 1:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x14	0x01	0x01	0x01	0x01	0x01	0x01	0x00

2. <u>Activation through writing in the corresponding object with a write request</u>
   There is an object described in the K-Matrix that stores the information for the temporal pattern setting for the halo LEDs. The object in question is number 106 (0x6A). The subindex 0 through 5 correspond with the buttons 1 to 6. Writing an object is described in chapter 9.4.3.
   Example data field for setting the temporal pattern for button 2 to Setting 1:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x6A	0x01	0x01	0x00	0x00	0x00	0x00

9.9.3.5 Changing the global brightness for the Halo LED illumination

The global brightness setting is a setting that dims the LEDs above all other settings. So to get realistic values in the aforementioned settings the global brightness needs to be set to 100%. The brightness is coded in 250 steps from 0 (0%) to 250 (100%). The setting of global brightness always affects all halo LEDs at once.

There are 2 ways to set the global brightness for the Halo LEDs.

1. <u>Activation through Process data Prop A config message</u>
   This is described in 9.5.1. The corresponding message identifier is 0x10. The data byte that accesses this information is Byte number 1.
   Example data field for setting the global brightness of the Halo LEDs to 20%:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x10	0x32	0x00	0x00	0x00	0x00	0x00	0x00

2. <u>Activation through writing in the corresponding object with a write request</u>
   There is an object described in the K-Matrix that stores the information for the global brightness setting for the halo LEDs. The object in question is number 102 (0x66). Writing an object is described in 9.4.3.
   Example data field for setting the global brightness setting of the halos to 20%:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x66	0x00	0x32	0x00	0x00	0x00	0x00

9.9.3.6 Choosing the active colour setting for the Halo LED illumination

There are 4 predefined colour settings available to choose from in the keypad. The K-Matrix defines the object and which value stands for which setting. Subindex 0 to 5 stand for the buttons 1 to 6.

Value	Setting name	Setting description
0x00	Setting 0 (even)	Colour for LEDs button wise
0x01	Setting 1 (even)	Colour for LEDs button wise
0x02	Setting 2 (even)	Colour for LEDs button wise
0x03	Setting 3 (individual)	Colour for LEDs individually (mapping of LEDs see K-Matrix)

There are 2 ways to set the active colour setting.

3. <u>Activation through Process data Prop A config message</u>
   This is described in 9.5.1. The corresponding message identifier is 0x11. The data bytes 1 through 6 are corresponding to buttons 1 through 6. This can be found in the K-Matrix as well.

Example data field for setting the active colour setting of all button halo LEDs to setting 1:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x11	0x01	0x01	0x01	0x01	0x01	0x01	0x00

1. <u>Activation through writing in the corresponding object with a write request</u>
   There is an object described in the K-Matrix that stores the information for the active colour setting. The object in question is number 109 (0x6D). The subindex 0 through 5 correspond with the buttons 1 to 6. Writing an object is described in 9.4.3.
   Example data field for setting the active colour setting for button 2 to Setting 1:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x6D	0x01	0x01	0x00	0x00	0x00	0x00

9.9.3.7 Changing the colour setting for the Halo LED illumination

Each of the aforementioned colour settings can be accessed through objects as described in the K-Matrix. By default the keypad has preset values in each setting. Each colour of the RGB LEDs can be set with values from 0 to 250 to mix colours according to the user's application.

Object	Setting name	Setting description
45 (0x2D)	Colour Setting Halo Red Even 0	Red colour channel of colour setting 0 Even
46 (0x2E)	Colour Setting Halo Green Even 0	Green colour channel of colour setting 0 Even
47 (0x2F)	Colour Setting Halo Blue Even 0	Blue colour channel of colour setting 0 Even
48 (0x30)	Colour Setting Halo Red Even 1	Red colour channel of colour setting 1 Even
49 (0x31)	Colour Setting Halo Green Even 1	Green colour channel of colour setting 1 Even
50 (0x32)	Colour Setting Halo Blue Even 1	Blue colour channel of colour setting 1 Even
51 (0x33)	Colour Setting Halo Red Even 2	Red colour channel of colour setting 2 Even
52 (0x34)	Colour Setting Halo Green Even 2	Green colour channel of colour setting 2 Even
53 (0x35)	Colour Setting Halo Blue Even 2	Blue colour channel of colour setting 2 Even

Object	Setting name	Setting description
54 (0x36)	Colour Setting Halo Red Individual 3	Red colour channel of colour setting 3 Individual
55 (0x37)	Colour Setting Halo Green Individual 3	Green colour channel of colour setting 3 Individual
56 (0x38)	Colour Setting Halo Blue Individual 3	Blue colour channel of colour setting 3 Individual

The subindex for Setting 0, 1 and 2 are mapped from 0 to 5 corresponding to the buttons 1 to 6. You can find the mapping for Setting 2 in the K-Matrix. Writing an object is described in chapter 9.4.3.
Example data field for setting the red channel in Setting 1 for button 2 to 50 (0x32):

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x30	0x01	0x32	0x00	0x00	0x00	0x00

The individual setting number 3 provides the possibility to adjust colour values for each LED alone. The LEDs are mapped to the subindex of the corresponding objects (0x36, 0x37 and 0x38) as described in the K-Matrix.
Example data field for setting the blue channel in Setting 3 for the top left LED of button 2 to 50(0x32):

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x18EFFF00	8	0x04	0x38	0x0C	0x32	0x00	0x00	0x00	0x00

9.9.3.8 Adjusting the timings in flash modes

The 2 flashing modes described in chapter 9.9.2.4 and 9.9.3.4 can be adjusted by the user. This means that the timings of how long the LED stays on and the period when it is turned on can be adjusted. The timings affect flash modes of symbols and halos alike. For this there are 4 objects defined in the K-Matrix:

Object	Object name	Object description
41 (0x29)	Flash_Slow_On_Time	Timing for how long the LED is on in Flash mode slow
42 (0x2A)	Flash_Slow_Period_ms	Time after which the LED is turned on in Flash mode slow
43 (0x2B)	Flash_Fast_On_Time	Timing for how long the LED is on in Flash mode fast
44 (0x2C)	Flash_Fast_Period_ms	Time after which the LED is turned on in Flash mode fast

There are default values defined for each object in the K-Matrix. But each value can be changed by hand. The values can be edited by means of a Prop A config message as described in chapter 9.5.1.
Example data fields for setting the timing of the fast mode to 20ms LED on after each 50ms:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x18EFFF00	8	0x04	0x2B	0x00	0x14	0x00	0x00	0x00	0x00
0x18EFFF00	8	0x04	0x2C	0x00	0x32	0x00	0x00	0x00	0x00

9.10 Changes between Software version 04.05.001 and 06.03.000 or newer

During further development it was necessary to implement some changes in the software. The changes for J1939 communication are shown in the table below:

Type	Index	Change	04.05.001	06.03.000
Object	0x01	NVM group has changed	EOL	EOL identification
Object	0x05	New NVM groups added (0xA5 = Load Dump; 0xA6 = EOL identification)	---	new groups
Object	0x06	New NVM groups added (0xA5 = Load Dump; 0xA6 = EOL identification)	---	New groups
Object	0x16	NVM group has changed	EOL	EOL identification
Object	0x17	NVM group has changed	EOL	EOL identification
Object	0x1As0	Adjust Undervoltage limit	8000	7000
Object	0x1As3	Adjust hysteresis time	80	500
Object	0x1Ds0-s3	New development object added	---	New Object
Object	0x1Es0-s2	New development object added	---	New Object
Object	0x1Fs0-s12	New development object added	---	New Object
Error Codes	---	The error codes were reduced to the essential groups	---	---
Error Codes	0x18FECA (PGN)	Button indices changed	B1 -> 0, B2 -> 2, B3 -> 4, B4 -> 1, B5 -> 3, B6 -> 5	B1 -> 0, B2 -> 1, B3 -> 2, B4 -> 3, B5 -> 4, B6 -> 5

10. CANopen communication protocol

The most important document to understand the communication with the keypad is the K-Matrix. It includes a description of all available objects, how to address them and how to save them. It is the equivalent to the object dictionary. To work with the K-Matrix please note that some objects have a ⊞ at the side. Which means that the row can be expanded and more information is then available.

This manual includes excerpts from the K-Matrix. All functions that the keypad has to offer can be found in the K-Matrix. All addressable objects are also defined in the corresponding EDS (electronical data sheet) file. To interface correctly with the keypad the use of the latest EDS is mandatory.

10.1 Composition of the CAN Identifier

The CAN-IDs are composed according to the CAN open standard which is defined in the official CAN in Automation (CiA) Document CiA301 “CANopen application layer and communication profile”.

The Identifier consists of 11 Bit. The first 7 Bits are the Node-ID which identifies the addressed device. The other 4 Bits contain the function code which identifies the type of message is sent and what it does.

The definitions of the function code can be found in the corresponding document CiA301.

10.2 Standard communication parameters

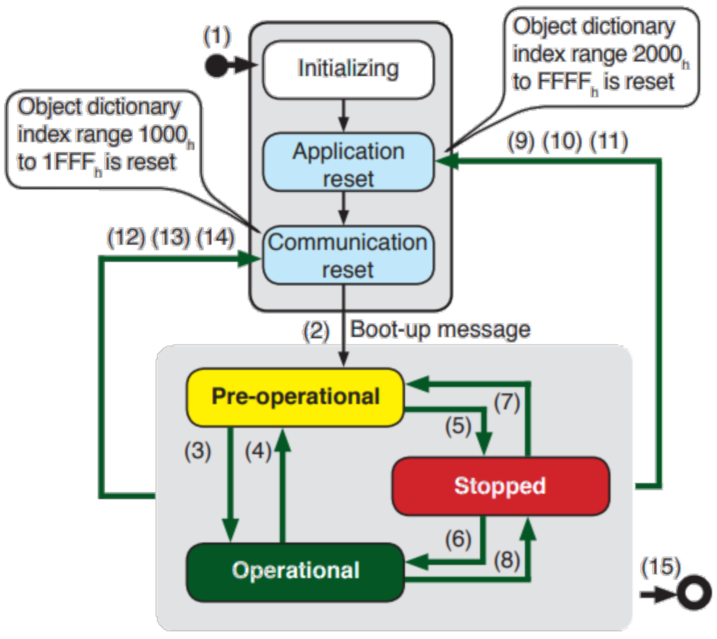
The default Node-ID of the keypad is set to 0x0B (0d11) by default. It can be modified either by using the LSS master commands or by a specific CAN message to the corresponding object as described in 10.4.

The default baud rate of the Keypad is set to 250kbps. It can also be changed by means of LSS master or a specific CAN message according to 10.4.

10.3 Installation in a CAN-network

After connecting the keypad electronically to the network it signifies its status to the master by sending a network management message according to CiA301. It is sent with the function code 0x700 + node ID. It contains 1 data byte with the content 0x00. After that the keyboard signals its current status by means of a heartbeat. By default it will send a message with the ID 0x700+node ID with 1 databyte containing 0x7F which signals that the keypad is in pre-operational state. The timing of the heartbeat can be adjusted according to the CANopen standard CiA301.

Switching in between the NMT states is defined in the official CANopen documentation CiA301. The following is an excerpt to visualize the different states.



(1) Power on
(2) Automatic switch to Pre-operational
(3) and (6) NMT switch to Operational
(4) and (7) NMT switch to Pre-operational
(5) and (8) NMT switch to Stopped
(9), (10) and (11) NMT switch to Application reset
(12), (13) and (14) NMT switch to Communication reset
(15) Power-off or hardware reset

[source: CiA CANopen® application layer and general communication profile „CAN poster“]

Example of boot up sequence and setting the keypad into operational state right after boot-up with default values as seen by the CAN master (only SDOs):

CAN-ID	Rx/Tx	DLC	Data Byte 0	Data Byte 1	Description
0x70b	Rx	1	0x00	-	Boot-Up
0x70b	Rx	1	0x7F	-	Heartbeat pre-operational
0x000	Tx	2	0x01	0x0b	NMT command to set device with node ID 0x0b to Operational state
0x70b	Rx	1	0x05	-	Heartbeat Operational

Different access to objects is possible dependent on the NMT state the keypad is currently in. The following table visualizes the possibilities,

Object	Operating modes			
	Initialisation	Pre-Operational	Operational	Stopped
PDO			X	
SDO		X	X	
SYNC		X	X	
Emergency		X	X	X
NMT		X	X	X
Node Guard (Heart Beat)		X	X	X
Boot-Up	X			

10.4 SDO communication

Communication to the keypad is defined by the CANopen standard. All access can be realized by sending and receiving Service Data objects (SDOs).

An SDO message is composed defined by CiA 301 like the following:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0xXXX + Node ID	8	Command specifier (CS)	Index LB	Index HB	Subind ex	LLB (D0)	LHB (D1)	HLB (D2)	HHB (D3)

Identifier:
- It is composed as described in 10.1. The function code of a SDO being received by the keypad is 0x600 (client to server) and a response will be sent with the function code 0x580 (server to client).

Command specifier:
- The CS in data byte 0 specifies the transfer type of the message and the access type (read or write) it is defined in detail by the CANopen standard.
- Excerpt of the standard with some commonly used CSs

CS	Description	Data bytes	Function
0x22	Download Request	Not specified	Send data to the keypad
0x23	Download Request	4 (D0 – D3)	
0x27	Download Request	3 (D0 – D2)	
0x2B	Download Request	2 (D0 – D1)	
0x2F	Download Request	1 (D0)	
0x60	Download Response	Not specified	Response of the keypad that data was received
0x40	Upload Request	Not specified	Read data from the keypad
0x43	Upload Response	4 (D0 – D3)	Response of the keypad with the read data
0x47	Upload Response	3 (D0 – D2)	
0x4B	Upload Response	2 (D0 – D1)	
0x4F	Upload Response	1 (D0)	
0x80	Abort Domain Transfer	4 (D0 – D3)	Keypad signifies transmission Error (data bytes contain error code)

Index:
- This is the Index of the addressed object. Note that it is written least significant byte first in data bytes 1 and 2.

Subindex:
- This describes the subindex of the addressed object. Whether or not an object does have addressable subindexes which can be found in the K-Matrix and the corresponding EDS file.

D0 – D3:
- The data that shall be written or read to or from the object is written in data bytes 4 to 7. Again it is ordered least significant byte first.

Example of CAN message transfer requesting the data that is stored in Object 0x1000 (Device Type) as seen by the CAN master:

11-Bit Identifier	DLC	Rx/Tx	Data field							
			Byte 0	Byte 1	Byt 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	Tx	0x40	0x00	0x10	0x00	0x00	0x00	0x00	0x00
0x58B	8	Rx	0x43	0x00	0x10	0x00	0x91	0x01	0x03	0x00

10.5 PDO communication

The CANopen protocol implemented in the keypad offers the communication with Process data objects (PDO). This offers the possibility to customize the communication to the users' needs and also the exchange of information over the CAN bus without protocol overhead. The communication with PDOs is defined in the CANopen standard CiA301 and the possible PDOs can be found in the K-Matrix. The keypad offers the use of TPDO1 to TPDO4 and RPDO1 to RPDO7.

By default TPDO1 and RPDO1 are active and objects are mapped to the PDOs can be found in the K-matrix. Default setting are defined in the K-matrix, too.

10.5.1 PDO communication parameter

For each PDO there exists a communication parameter object in which the data how the PDO will be transmitted or is addressed is stored.

It holds the COB-ID which is the address of the PDO with which it will be sent or received.

It also holds the Transmission type which is defined by the CANopen standard. In the case of the keypad the Transmission Type is not changeable. The transmission type of every first PDO is set to event driven, profile-specific. Every other transmission type is set to event-driven, manufacturer-specific.

The inhibit time of the PDOs is optional it specifies the minimum time between transmissions of the PDO. Once the PDO is transmitted no other transmissions of the PDO will take place in the duration of the inhibit time. The Event Time specifies the time period in which the PDO is transmitted. This affects TPDOs.

The Event Time is specified in multiples of milliseconds. Each time the PDO is transmitted the event timer will be reset so that after it expires the PDO will be transmitted again. It will however not be transmitted more often than the inhibit time specifies.

Each object that is allowed to be mapped to the TPDO can be transmitted periodically by the TPDO. Some objects are able to trigger the transmission of the TPDO independently from the event timer. This means that when one of the objects is mapped to the PDO the PDO will be transmitted once the value of the object changes. Those objects are:

- 0x2014 (button error)
- 0x2015 (button stuck error)
- 0x2400 (RCC tilt X and Y)
- 0x2401 (RCC rotation)
- 0x2402 (RCC button pressed)
- 0x6000s1 (keypad button pressed)

10.5.2 Mapping of PDOs

It is possible for the user of the keypad to map objects to PDOs to use the PDO communication for interfacing with the keypad. Which objects can be mapped to PDOs is defined by the EDS file. For each PDO there are 2 objects of interest for mapping the parameters. The first object are the communication parameters. In this object the COB-ID, the Transmission Type, the inhibit Timer and the Event Timer are stored. The second object holds the information which objects are mapped to the PDO. These are called PDO Mapping parameters. How to map objects to a PDO will be described in the following.

The PDO mapping/ remapping is only possible in state pre-operational of the keypad. If the keypad is set to operational state, PDO mapping cannot be changed.

1. Deactivating the PDO in question
   To deactivate a PDO the most significant Bit of the corresponding COB-ID of the PDO must be set to 1. This is done in the corresponding PDO Communication Parameter object Example message for disabling RPDO2:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x23	0x01	0x14	0x01	0x00	0x00	0x00	0x80

2. Setting thehighest sub-index supported to 0
   Once the PDO is deactivated the count of how many objects the PDO holds must be set to 0. This information is stored in the sub-index 0 of the PDO Mapping parameter object.
   Example message for setting the highest sub-index supported for RPDO2 to 0:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x01	0x16	0x00	0x00	0x00	0x00	0x00

3. Mapping the desired objects to the PDO
   After the first 2 steps are done it is possible to map up to 8 objects to the desired RPDO. Please note that not all objects can be mapped to PDOs.
   Example message for mapping the object 0x2207 (global brightness Halos) to the first sub-index of RPDO2:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x23	0x01	0x16	0x01	0x08	0x00	0x07	0x22

   Please note that Byte 4 holds the information of the expected data length of the mapped object in bits. Byte 5 holds the sub-index of the mapped object. And Bytes 6 and 7 hold the address of the mapped object written with LSB first.

4. Setting the highest sub-index supported to the count of mapped objects in the PDO
   Sub-Index 0 of the PDO mapping parameter object must now be set to the count of objects mapped to the PDO.
   Example message for setting the count of objects in RPDO2 to 1.

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x01	0x16	0x00	0x01	0x00	0x00	0x00

5. Activating the PDO
   To activate the PDO, the COB-ID must be written in the PDO Communication Parameter. It is possible to set the COB-ID to a desired value. The range in which the COB-ID can be set according to the CANopen standard is described in the document CiA301. Important to notice is that the MSB of the COB-ID must be set to 0 to activate it.
   Example message for setting the COB-ID of the RPDO2 to 0x300:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x23	0x01	0x14	0x01	0x00	0x03	0x00	0x00

After these steps are executed successfully the PDO (if it is a TPDO) will start to transmit periodically or can be accessed (RPDO) by CAN messages with the given COB-ID.

Example of CAN message transfer writing a value into the mapped object in RPDO2 (as defined in the examples see above) as seen by the CAN master:

11-Bit Identifier	DLC	Rx/Tx	Data field
			Byte 0
0x300	1	Tx	0xFA

10.5.3 Automatic COB-ID update

By default the COB-ID of the Keypad does not change its value if the Node-ID of the Keypad is changed. If this functionality is needed the keypad offers this function though. To activate this function the object 0x2005 needs to be changed. By default it is set to 0x00 which means no automatic change of the COB-IDs. To activate the automatic COB-ID update the customer will have to change the value of this object to 0x01.

For further information please refer to the current K-Matrix.

Please note that the changing of the COB-ID for the EMCY-message is not affected by this object. The COB-ID of the EMCY-message will always be updated automatically.

10.6 Diagnostics, Error Codes

The keypad offers ways of diagnostics. For one there are the error handling objects according to CiA301, which are namely object 0x1001 and 0x1003.

Object 0x1001 is the Error register. It will display the currently existing error. The values which the object can display are defined by the CANopen standard and the interpretation can be found in CiA301. In case that are more than one errors are active, the error register contains the newest error. All error codes supported by keypad are defined in the K-matrix and shown in the table below:

Error code	Name	Description
0x0071	Sleep objection by application	Button is pressed state Prepare Sleep (see CiA-320 for more information)
0x1xxx	Generic error	See CiA-301
0x3101	Overvoltage	Input voltage exceeds 33 volts
0x3102	Undervoltage	Input voltage falls below 8 volts
0x4201	Overtemperature warning	Keypad temperature exceeds 100°C
0x4203	Overtemperature error	Keypad temperature exceeds 125°C
0x4204	Temperature sensor defect	Implausible temperature values are measured
0x5001	Button stuck	Button is pressed longer than the time value defined in object 0x2101 (default value: 10s)
0x5002	Button pressed at startup	One or more buttons are pressed during power-on-reset
0x7002	Button unstable	Switching element bounces excessively; reason: switching system worn out
0x7004	Button out of range	Resistance of the switching system reach upper limit; reason: switching system worn out
0x7006	Button crosstalk	Button was detected as pressed although a other button should be pressed; detection over cyclically detuning the voltage divider of each button; reason: incoming moisture in the keypad
0x7007	Button test low	Cyclically detuning of voltage divider of each button returns implausible values; reason: incoming moisture in the keypad, switching system worn out

The other predefined error handling object is object 0x1003. This is the Pre-defined error field. It holds up to 20 error codes of errors that have occurred in the past. Subindex 0 holds the number of stored error messages. Subindexes 1 through 20 hold the error codes of the occurred errors. They will be stored in the field in that way that the most current error is always on top. So “older” errors will be moved to higher Subindexes if a new error is stored. To encode the stored errors please see the K-Matrix. All possible error codes are described in the sheet “error codes”. This error field can be reset by the user by writing a value of 0 to subindex 0 of the object. All stored error codes will be erased by this action. This object will be saved to non-volatile memory automatically.

In addition to these standardized diagnostic handling tools, the keypad offers more functionality. There is the button error object (0x2014) which displays possible errors for each button. For more information please refer to the K-Matrix.

Next to the error register end the error field the keypad offers the possibility to send emergency messages described in CIA301. Every time an error occurred an emergency message are send over the CAN-bus. The structure of the message are shown in the following table:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x80 + Node-ID	8	Error Code LSB	Error Code MSB	Error Regist er	Error Data 0	Error Data 1	Error Data 2	Error Data 3	Error Data 4

The emergency message are only send once after an error occurred without a repeat. Two examples (undervoltage error, occurred three times; button 2 stuck error; occurred two times)

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x8b	8	0x02	0x31	0x05	0x00	0x00	0x03	0x00	0x00
0x8b	8	0x01	0x50	0x21	0x00	0x00	0x02	0x00	0x00

The current temperature (0x2008) and the current voltage (0x2009) is also monitored by the keypad and can be read from the corresponding objects. How the values can be interpreted can be found in the K-Matrix.

10.7 Save settings to Volatile and Non-Volatile memory

By default all changes written in the objects like lighting, Node Address and so on are stored in volatile memory after writing. This means that with a voltage reset these changes will not take any effect any more. The keypad has the ability to store settings in the non-volatile memory to implement them fully available after module reset.

Saving into the non-volatile memory is done by writing into the object 0x1010. To save the word “save” has to be written in Hex (0x65766173) to the corresponding sub-index which corresponds with the save group of the desired object. Please refer to the K-Matrix to see which sub-index belongs to which save group and which object is saved via which save group.

Example message for saving all data belonging to the CLS Save Group (e.g. Node Address).

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x23	0x10	0x10	0x06	0x73	0x61	0x76	0x65

Some changes (e.g. Node-ID) take effect after voltage reset of the keypad.

10.8 Custom Layer Settings

The baud rate and the Node-ID can be changed by the user by means of writing in the corresponding sub-index of object 0x2004. The default values are as follows:

Object	Default value
Baud rate (subindex 2)	0x03 (250 kbps)
Node-ID (subindex 1)	0x0B (Node-ID 11)

The objects with their respective description including the corresponding save/clear Block can be found in the K-Matrix. To change the values the user has to first write 0xAA to sub-index 3 of Object 0x2004. This is the protection register to make sure no accidental change occurs. Then the desired value can be written to the correct subindex. After the keypad accepted the change the values have to be stored in non-volatile memory (see chapter 10.7). After the values are stored correctly a power reset has to be performed. After that the keypad will be set to the new settings and communication will be able only by using the changed values.

10.8.1 Manually configure the baud rate

As an example the following table shows the CANopen messages that are necessary for changing the baud rate from 250 kbps (default value) to 500 kbps.

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x60B	8	0x2F	0x04	0x20	0x03	0xAA	0x00	0x00	0x00
0x58B	8	0x60	0x04	0x20	0x03	0x00	0x00	0x00	0x00
0x60B	8	0x2F	0x04	0x20	0x02	0x02	0x00	0x00	0x00
0x58B	8	0x60	0x04	0x20	0x02	0x00	0x00	0x00	0x00
0x60B	8	0x23	0x10	0x10	0x06	0x73	0x61	0x76	0x65
0x58B	8	0x60	0x10	0x10	0x06	0x00	0x00	0x00	0x00

First step is to unlock the baud rate object. The Node-ID and the baud rate can't be written before the value 0xAA is written to the protection object.

Second step for changing the baud rate is to send the new baud rate to the keypad. Data byte 4 represents the new baud rate (0x02 = 500kbps; 0x03 = 250kbps), data byte 1 the low byte and data byte 2 the high byte of the index, data byte 3 the subindex and data byte 0 the command (Download request; 1 byte). After the keypad receives a valid baud rate, the keypad send a "Download Response"- message (Data byte 0 = 0x60).

Third step is to save the new baud rate to the non-volatile memory. The index of the store-NVM-object is 0x1010 (data byte 1 and 2). The data byte 3 contains the save group. Referenced to the CANopen K-Matrix of the keypad the baud rate uses the save group "Custom Layer Settings". The data bytes 4 – 8 contains the ASCII-Coded word "SAVE" (0x73; 0x61; 0x76; 0x65). After the write request the keypad sends a "Download Response"-message.

4th step is to disconnect the keypad from the power supply and do a Power-On-Reset. After that the keypad uses the new baud rate 500kbps for communication.

10.8.2 Manually configure the Node-ID

As an example the following table shows the CANopen messages that are necessary for changing the Node-ID from 0x0B (default value) to 0x0E.

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x60B	8	0x2F	0x04	0x20	0x03	0xAA	0x00	0x00	0x00
0x58B	8	0x60	0x04	0x20	0x03	0x00	0x00	0x00	0x00
0x60B	8	0x2F	0x04	0x20	0x01	0x0E	0x00	0x00	0x00
0x58B	8	0x60	0x04	0x20	0x01	0x00	0x00	0x00	0x00
0x60B	8	0x23	0x10	0x10	0x06	0x73	0x61	0x76	0x65
0x58B	8	0x60	0x10	0x10	0x06	0x00	0x00	0x00	0x00

First step is to unlock the Node-ID object. The Node-ID and the baud rate can’t be written before the value 0xAA is written to the protection object.

Second step for changing the Node-ID is to send the new Node-ID to the keypad. Data byte 4 represents the new Node-ID (0x0E), data byte 1 the low byte and data byte 2 the high byte of the index, data byte 3 the subindex and data byte 0 the command (Download request; 1 byte). After the keypad receives a valid Node-ID, the keypad send a “Download Response”- message (Data byte 0 = 0x60).

Third step is to save the new Node-ID to the non-volatile memory. The index of the store-NVM-object is 0x1010 (data byte 1 and 2). The data byte 3 contains the save group. Referenced to the CANopen K-Matrix of the keypad the Node-ID uses the save group “Custom Layer Settings”. The data bytes 4 – 8 contains the ASCII-Coded word “SAVE” (0x73; 0x61; 0x76; 0x65). After the write request the keypad sends a “Download Response”- message.

4th step is to disconnect the keypad from the power supply and do a Power-On-Reset. After that the keypad uses the new Node-ID 0x0E for communication.

10.9 Basic functions

10.9.1 Pressed Key data

The information whether or not a button is pressed can be found in a defined object 0x6000 of the keypad. The sub-index correspond with the buttons of the keypad. And the content visualizes whether or not the button is pressed. If subindex 1 is 2, button 2 is pressed (Bitwise coded).

For further information please refer to the corresponding K-Matrix.

10.9.1.1 Stuck Button Time

The Keypad provide the option to detect a continuous pressed button and send an error message after a defined time. This so called stuck button detection will be controlled via the stuck button detection time object (Object index: 0x2101s0).

10.9.1.2 Reconfigure Stuck Button Time

The keypads with software version 04.05.001 have no possibility to reconfigure the stuck button time. The values is set fixed to 10 seconds and can only be changed during production process.

Keypads with software version 06.03.000 or newer provide the possibility to reconfigure the stuck button time in field. The following table shows an example for reconfigure the stuck button time to 3 seconds and save the new stuck button time:

CAN-ID	DLC	Data Byte 0	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Data Byte 5	Data Byte 6	Data Byte 7
0x60B	8	0x2B	0x01	0x21	0x00	0xB8	0x0B	0x00	0x00
0x58B	8	0x60	0x01	0x21	0x00	0x00	0x00	0x00	0x00
0x60B	8	0x23	0x10	0x10	0x07	0x73	0x61	0x76	0x65
0x58B	8	0x60	0x10	0x10	0x07	0x00	0x00	0x00	0x00

Data byte 4 contains the low byte and data byte 5 the high byte of the 16-bit value stuck button time.

10.9.2 Symbol Illumination

The symbols of the keypad are illuminated with white LEDs. In the following the corresponding Objects will be described to turn them on and how to use all the functions the keypad has to offer. All changes done in the object will be reset after voltage reset if not stored in non-volatile memory according to chapter 10.7.

10.9.2.1 Activating Symbol Illumination

The LEDs for each button can be activated or deactivated exclusively. This is managed by bitcoding the LED's of the buttons. Writing a 1 to the desired Bit-position the LED will be turned on, writing a 0 will turn it off. By adding the Bit-Values of the buttons together it is possible to activate any desired pattern of the symbols.

Button 1	Button 2	Button 3
0b000001 (0x01)	0b000010 (0x02)	0b000100 (0x04)
Button 4	Button 5	Button 6
0b001000 (0x08)	0b010000 (0x10)	0b100000 (0x20)

There is an object described in the K-Matrix that stores the information for the activated/deactivated symbol LEDs. The object in question is 0x6200 and the corresponding subindex is S1. Writing the corresponding bit-value into this object will lead to activating the Symbol LEDs with the configured brightness and pattern.

Example message for turning on the symbol LEDs of the first and third button:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x00	0x62	0x01	0x05	0x00	0x00	0x00

10.9.2.2 Choosing the active brightness setting for the symbol LED’s

There are 3 predefined brightness settings available to choose from in the keypad. The K-Matrix defines the object and which value stands for which setting. Subindex 1 to 6 stand for the buttons 1 to 6.

Value	Setting name	Setting description
0x00	Setting 0 (even)	Brightness for LEDs button wise
0x01	Setting 1 (even)	Brightness for LEDs button wise
0x02	Setting 2 (individual)	Brightness for LEDs individually (corresponds to Halo LEDs, as the symbol is illuminated with only one LED)

There is an object described in the K-Matrix that stores the information for the active brightness setting. The object in question is 0x2200. The subindex 1 through 6 correspond with the buttons 1 to 6.

Example message for setting the active brightness setting for button 2 to Setting 1:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x00	0x22	0x02	0x01	0x00	0x00	0x00

10.9.2.3 Changing the brightness setting for the symbol LED’s

Each of the aforementioned brightness settings can be accessed through objects as described in the K-Matrix.

Object	Setting name	Setting description
0x211F	Setting 0 (even)	Brightness for LEDs button wise
0x2121	Setting 1 (even)	Brightness for LEDs button wise
0x2123	Setting 2 (individual)	Brightness for LEDs individually (corresponds to Halo LEDs, as the symbol is illuminated with only one LED)

The subindex for Setting 0 and 1 are mapped from 1 to 6 corresponding to the buttons 1 to 6. You can find the mapping for Setting 2 in the K-Matrix. The value for the lighting can be found in the tab “value definition” in the K-Matrix.

Example message for setting the brightness in Setting 1 for button 2 to 50% (0x7D):

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x21	0x21	0x02	0x7D	0x00	0x00	0x00

10.9.2.4 Changing the temporal patterns for the symbol LED Illumination

There are 4 predefined temporal patterns available for the symbol LEDs.

Value	Setting name	Setting description
0x00	Steady	LED always on
0x01	Flash slow	LED flashing on and off with default value times
0x02	Flash fast	LED flashing on and off with default value times faster than 0x01
0x03	Pulsate	LEDs slowly turning off and on with default value times

There is an object described in the K-Matrix that stores the information for the temporal pattern setting for the symbol LEDs. The object in question is number 0x2204. The subindex 1 through 6 correspond with the buttons 1 to 6.

Example message for setting the temporal pattern for button 2 to Setting 1 (Flash slow):

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x04	0x22	0x02	0x01	0x00	0x00	0x00

10.9.2.5 Changing the global brightness for the symbol LED Illumination

The global brightness setting is a setting that is dimming the LEDs above all other settings. So to get realistic values in the aforementioned settings the global brightness needs to be set to 100%. The value for the brightness can be found in the tab “value definition” in the K-Matrix. The setting of global brightness always affects all Symbol LEDs at once.

There is an object described in the K-Matrix that stores the information for the global brightness setting for the symbol LEDs. The object in question is number 0x2206.

Example message for setting the global brightness to 20%:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x06	0x22	0x00	0x32	0x00	0x00	0x00

10.9.3 Halo-Ring Illumination

The Halo-Rings of the keypad are illuminated with 4 RGB LEDs for each button. In the following the corresponding objects will be described to turn them on and how to use all the functions the keypad has to offer.

All changes done in the object will be reset after voltage reset if not stored in non-volatile memory according to 10.7.

10.9.3.1 Activating Halo Ring Illumination

As there are 4 LEDs per button it is important to know that there are always 2 steps needed to activate the halo LEDs. In one step it is set which of the 4 LEDs are supposed to be turned on. And in the second step the halo for the button in question is activated.

By default the halo LED’s are activated and can be switched on with object 0x6200 subindex 2.

1. <u>First step: setting the LEDs that need to be turned on per button</u>
   There exists an object in the K-Matrix which stores the configuration of Halo LEDs per button. The object in question is 0x2208. The subindexes 1 – 6 of this object correspond with the buttons 1 – 6 of the keypad. The values of each subindex are bit coded for the 4 LEDs. The value written will either turn the LED on or off. 1 means on and 0 means off. By adding the Bit-Values of the LEDs together it is possible to activate any desired pattern of the LEDs.

   They are coded as follows:

Buttons 1 - 6		
Bit 3		Bit 0
Bit 2		Bit 1

Example message for activating LED 0 and LED 1 of the halo-ring of button 2:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x08	0x22	0x02	0x03	0x00	0x00	0x00

2. <u>Second step: activating the halo for the button in question</u>

   This is done by writing on the corresponding Lighting activation object. The corresponding object is 0x6200 and the subindex is 2. This can be found in the K-Matrix as well. The value written in the corresponding byte is bitcoded similarly to the symbol activation described in chapter 10.9.2.1.

   Example message for turning on the halo-ring of button 2:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x00	0x62	0x02	0x02	0x00	0x00	0x00

10.9.3.2 Choosing the active brightness setting for the Halo LED’s

There are 3 predefined brightness settings available to choose from in the keypad. The K-Matrix defines the object and which value for the setting is stored.

Value	Setting name	Setting description
0x00	Setting 0 (even)	Brightness for LEDs button wise
0x01	Setting 1 (even)	Brightness for LEDs button wise
0x02	Setting 2 (individual)	Brightness for LEDs individually

There are 2 ways to set the active brightness setting for the buttons.

There is an object described in the K-Matrix that stores the information for the active brightness setting. The object in question is number 0x2201. The subindex 1 through 6 correspond with the buttons 1 to 6.

Example message for setting the active brightness setting for button 2 to Setting 1:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x01	0x22	0x02	0x01	0x00	0x00	0x00

10.9.3.3 Changing the brightness setting for the Halo LED’s

Each of the aforementioned brightness settings can be accessed through objects as described in the K-Matrix.

Object	Setting name	Setting description
0x2120	Setting 0 (even)	Brightness for LEDs button wise
0x2122	Setting 1 (even)	Brightness for LEDs button wise
0x2121	Setting 2 (individual)	Brightness for LEDs individually

The subindex for setting 0 and 1 are mapped from 1 to 6 corresponding to the buttons 1 to 6. You can find the mapping for Setting 2 in the K-Matrix.

Example message for setting the brightness in setting 1 for button 2 to 20%:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x22	0x21	0x02	0x32	0x00	0x00	0x00

The individual setting number 2 provides the possibility to adjust brightness values for each LED alone. The LEDs are mapped to the sub index of the corresponding object as described in the K-Matrix.

Example message for setting the brightness in Setting 2 for the top left LED of button 2 to 20% (0x32):

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x23	0x21	0x0D	0x32	0x00	0x00	0x00

10.9.3.4 Choosing the temporal patterns for the Halo LED's for flashing modes

There are 5 predefined temporal patterns available for the halo LEDs.

Value	Setting name	Setting description
0x00	Steady	LED always on
0x01	Flash slow	LED flashing on and off with default value times
0x02	Flash fast	LED flashing on and off with default value times faster than 0x01
0x03	Pulsate	LEDs slowly turning off and on with default value times
0x04	Rotate	4 halo LEDs are slowly turning off one by one to visualize a rotating ring

There is an object described in the K-Matrix that stores the information for the temporal pattern setting for the halo LEDs. The object in question is 0x2205. The subindex 1 through 6 correspond with the buttons 1 to 6. Subindex 7 controls the RCC.

Example message for setting the temporal pattern for button 2 to Setting 1:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x05	0x22	0x02	0x01	0x00	0x00	0x00

10.9.3.5 Changing the global brightness for the Halo LED's

The global brightness setting is a setting that is dimming the LEDs above all other settings. So to get realistic values in the aforementioned settings the global brightness needs to be set to 100%. The value for the brightness can be found in the tab “value definition” in the K-Matrix. The setting of global brightness always affects all halo LEDs at once.

There is an object described in the K-Matrix that stores the information for the global brightness setting for the halo LEDs. The object in question is number 0x2207.

Example message for setting the global brightness setting to 20%:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x07	0x22	0x00	0x32	0x00	0x00	0x00

10.9.3.6 Choosing the active colour setting for the Halo LED’s

There are 4 predefined colour settings available to choose from in the keypad. The K-Matrix defines the object and which value stands for which setting. Subindex 1 to 6 correspond to the buttons 1 to 6.

Value	Setting name	Setting description
0x00	Setting 0 (even)	Colour for LEDs button wise
0x01	Setting 1 (even)	Colour for LEDs button wise
0x02	Setting 2 (even)	Colour for LEDs button wise
0x03	Setting 3 (individual)	Colour for LEDs individually (mapping of LEDs see K-Matrix)

There is an object described in the K-Matrix that stores the information for the active colour setting. The object in question is 0x2203. The subindex 1 through 5 correspond with the buttons 1 to 6. Subindex 7 controls the RCC.

Example message for setting the active colour setting for button 2 to Setting 1:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x03	0x22	0x02	0x01	0x00	0x00	0x00

10.9.3.7 Changing the colour setting for the Halo LED’s

Each of the aforementioned colour settings can be accessed through objects as described in the K-Matrix. By default the keypad has preset values in each setting. Each colour of the RGB LEDs can be set with values from 0 to 250 to mix colours according to the user’s application.

Object	Setting name	Setting description
0x210D	Colour Setting Halo Red Even 0	Red colour channel of colour setting 0 Even
0x210E	Colour Setting Halo Green Even 0	Green colour channel of colour setting 0 Even
0x210F	Colour Setting Halo Blue Even 0	Blue colour channel of colour setting 0 Even
0x2113	Colour Setting Halo Red Even 1	Red colour channel of colour setting 1 Even
0x2114	Colour Setting Halo Green Even 1	Green colour channel of colour setting 1 Even
0x2115	Colour Setting Halo Blue Even 1	Blue colour channel of colour setting 1 Even
0x2119	Colour Setting Halo Red Even 2	Red colour channel of colour setting 2 Even
0x211A	Colour Setting Halo Green Even 2	Green colour channel of colour setting 2 Even

0x211B	Colour Setting Halo Blue Even 2	Blue colour channel of colour setting 2 Even
0x211C	Colour Setting Halo Red Individual 3	Red colour channel of colour setting 3 Individual
0x211D	Colour Setting Halo Green Individual 3	Green colour channel of colour setting 3 Individual
0x211E	Colour Setting Halo Blue Individual 3	Blue colour channel of colour setting 3 Individual

The subindexes for setting 0, 1 and 2 are mapped from 1 to 6 corresponding to the buttons 1 to 6. Subindex 7 controls the RCC. You can find the mapping for setting 2 in the K-Matrix.

Example message for setting the red channel in setting 1 for button 2 to 50:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x13	0x21	0x02	0x32	0x00	0x00	0x00

The individual setting number 3 provides the possibility to adjust colour values for each LED individually. The LEDs are mapped to the subindex of the corresponding objects as described in the K-Matrix.

Example message for setting the blue channel in Setting 3 for the top left LED of button 2 to 50:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2F	0x1E	0x21	0x0D	0x32	0x00	0x00	0x00

10.9.3.8 Adjusting the timings in flash modes

The 2 flashing modes described in chapter 10.9.2.4 and 10.9.3.4 can be adjusted by the user. This means that the timings of how long the LED stays on and the period when it is turned on can be adjusted. The timings for these modes affect symbols and halos alike. For this there are 4 objects defined in the K-Matrix:

Object	Object name	Object description
0x2104	Flash_Slow_On_Time	Timing for how long the LED is on in Flash mode slow
0x2105	Flash_Slow_Period_ms	Time after which the LED is turned on in Flash mode slow
0x2106	Flash_Fast_On_Time	Timing for how long the LED is on in Flash mode fast
0x2107	Flash_Fast_Period_ms	Time after which the LED is turned on in Flash mode fast

There are default values defined for each object in the K-Matrix. But each value can be changed by hand.

Example messages for setting the timing of the fast mode to 20ms LED on after each 50ms:

11-Bit Identifier	DLC	Data field							
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x60B	8	0x2B	0x06	0x21	0x00	0x14	0x00	0x00	0x00
0x60B	8	0x2B	0x07	0x21	0x00	0x32	0x00	0x00	0x00

10.10 Changes between Software version 04.05.001 and 06.05.001 or newer

During further development it was necessary to implement some changes in the software. The changes for CANopen/CANopenSafety communication are shown in the table below:

Type	Index	Change	04.05.001	06.05.001
Object	0x1009s0	NVM block changed	EAO-only	EOL-ID
Object	0x1010s0	Increased supported subindex to 7	6	7
Object	0x1010s8	New sub-object was added	---	EOL-ID
Object	0x1011s0	Increased supported subindex to 7	6	7
Object	0x1011s8	New sub-object was added	---	EOL-ID
Object	0x2001s0	NVM block changed	EAO-only	EOL-ID
Object	0x2007s1	Adjust Undervoltage limit	8000	7000
Object	0x2007s4	Adjust hysteresis time	80	500
Object	0x2017s0	NVM block changed	EAO-only	EOL-ID
Object	0x2125s0-s3	New object added	---	New object
Object	0x2996s1-s3	New development object added	---	New object
Object	0x2997s0-s3	New development object added	---	New object
Object	0x2999s0-sD	New development object added	---	New object
Error Codes	---	The error codes were reduced to the essential groups	---	---
Object	0x1001s0; 0x1003	Button indices changed	B1 -> 0, B2 -> 2, B3 -> 4, B4 -> 1, B5 -> 3, B6 -> 5	B1 -> 0, B2 -> 1, B3 -> 2, B4 -> 3, B5 -> 4, B6 -> 5

11. Special functions

The keypad offers some special functions which will be described in the following in a more detailed way. Please make sure to refer to the corresponding K-Matrix as well, as these descriptions will just be excerpts of the functionality.

11.1 Sleep/Wakeup, Power Saving

To save power, the keypad offers a sleep mode. There is more than one way to set the keypad into sleep-mode. It can be achieved either through a specific Sleep timer on CAN or by setting an external hardware signal. Please refer to the K-Matrix to find the object in which the Sleep/Wakeup source can be chosen and what value stands for which option. If the use of the sleep mode is not wanted in an application, the user has to make sure that it is configured to the 2-wire hardware interface mode and the wakeup-in pin is either left open or pulled reliably to GND.

11.1.1 Sleep/Wakeup over CAN

This function is implemented according to CiA320 for CANopen, for J1939 the protocol is manufacturer specific on the basis of the CANopen implementation and is specified in the corresponding K-Matrix.

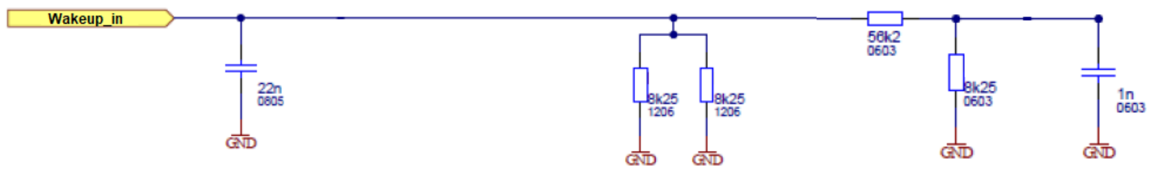
If the keypad is configured to sleep/ wakeup over CAN it will enter sleep mode if no CAN message is received for 5 seconds. After that the keypad can be woken up by sending an arbitrary CAN message (or keep alive message) or by pressing a button. Please refer to the K-Matrix to see how each button can be configured to enable wakeup or disable it.

The wakeup out pin is disabled in this mode.

11.1.2 Sleep/Wakeup over 2 wire hardware interface

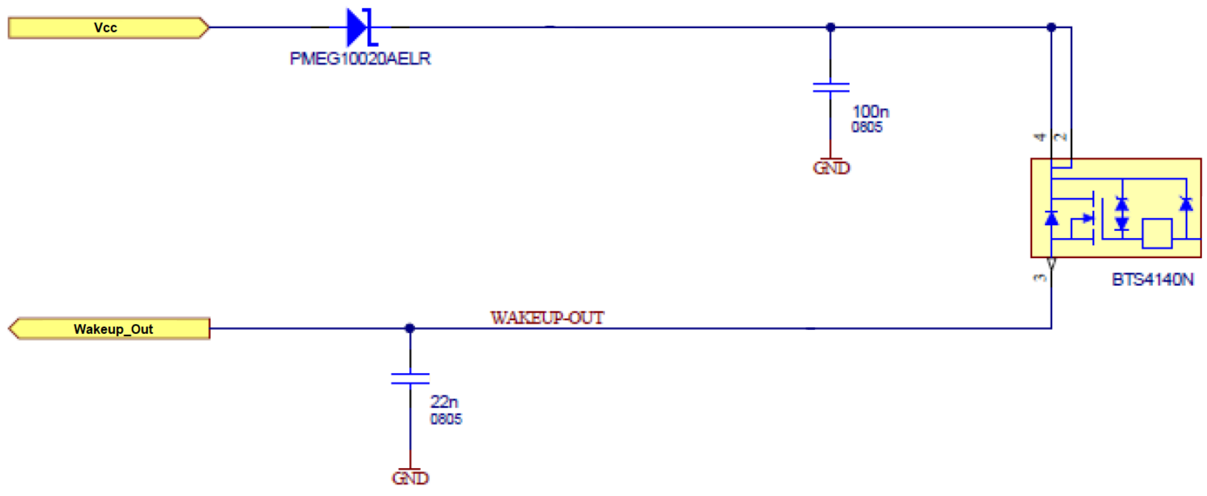
The keypad offers a 2-wire interface to enable Sleep mode. For this it offers a wakeup-in pin with which the keypad can be set to sleep mode and can be woken up respectively. The pin can draw between ~2mA and 7,5mA depending on the voltage on the wakeup-in pin.

This pin is read by the microcontroller analogical. After a reset of the keypad the pin needs to be pulled high to activate the wakeup-in. If this does not happen the port will stay deactivated. To activate the sleep mode a falling edge must be generated on the pin. The voltage must switch from high to low. This can be done by switching to a low level or by turning the operating port on the controlling device to high-z. The voltage must be held at low level to keep the keypad in sleep mode. To wake the keypad back up the voltage at the wakeup pin must switch from low to high and must then be held at high to keep the keypad awake.



11-1 circuit diagram Wake-up-in

The 2nd wire is the wakeup output. This offers the functionality for the keypad to set other devices into sleep mode or to wake them up. As long as the keypad is woken up the output is set to high. Once the device enters sleep-mode the output is turned off and set to open.



11-2 circuit diagram wake-up-out

In the following the voltage levels to enter sleep mode and to wake up the keypad are explained.

Max. input current wakeup in	Ca. 8 mA (@ 32 V)
Max. output current wakeup out	200 mA
Voltage level low	< 30 % * T30
Voltage level high	> 70 % * T30

11.1.3 Sleep/Wakeup summary

RKP state		State Awake	State Sleep
Sleep/Wakeup Source	Sleep/Wakeup interface		
Hardware two-wire interface	Wakeup_Out Pin	On = High [~ (T30 – 2,3V)]	OFF = Low [~ 0V]
	Wakeup_In Pin	VIN = > 70% * T30	VIN = < 30% * T30
	CAN Rx	Ignored (regarding sleep state) Other CAN messages are received normally	Ignored
	CAN Tx	No CiA320 messages are sent	No communication
	Buttons	Button state is interpreted safely	Cyclic button read → keypad wakes with button push Button state is not interpreted safely
CAN Bus	Wakeup_Out Pin	"High-Z" (Low)	"High-Z" (Low)
	Wakeup_In Pin	Ignored	Ignored
	CAN Rx	If there are no "stay awake" messages sent or the command "go to sleep" is sent the keypad changes it's state to sleep	With communication on the CAN bus the keypad changes its state to awake
	CAN Tx	Responds to CiA320 messages	No communication
	Buttons	Button state is interpreted safely	Cyclic button read → keypad wakes with button push Button state is not interpreted safely

11.2 Structure of lightning objects

For understanding how the lightning setup works, knowledge over the internal dependency between the lightning objects is necessary. For the communication protocols J1939 and CANopen/CANopenSafety the lightning setup uses the same topology. The objects which are used for the lightning setup are listed in the following section.

11.2.1 Depending objects

Index CANopen	Index J1939	Object Name
Colour settings Halo		
0x210D:1-7	0x2D:0-6	Colour Setting 0 Even Halo Red
0x210E:1-7	0x2E:0-6	Colour Setting 0 Even Halo Green
0x210F:1-7	0x2F:0-6	Colour Setting 0 Even Halo Blue
0x2113:1-7	0x30:0-6	Colour Setting 1 Even Halo Red
0x2114:1-7	0x31:0-6	Colour Setting 1 Even Halo Green
0x2115:1-7	0x32:0-6	Colour Setting 1 Even Halo Blue
0x2119:1-7	0x33:0-6	Colour Setting 2 Even Halo Red
0x211A:1-7	0x34:0-6	Colour Setting 2 Even Halo Green
0x211B:1-7	0x35:0-6	Colour Setting 2 Even Halo Blue
0x211C:1-31	0x36:0-30	Colour Setting 3 Indiv Halo Red
0x211D:1-31	0x37:0-30	Colour Setting 3 Indiv Halo Green
0x211E:1-31	0x38:0-30	Colour Setting 3 Indiv Halo Blue
Brightness settings Halo		
0x2120:1-7	0x39:0-6	Brightness Setting 0 Even Halo
0x2122:1-7	0x3A:0-6	Brightness Setting 1 Even Halo
0x2206:0	0x66:0	Global Brightness Halos
Brightness settings Symbol		
0x211F:1-6	0x3B:0-5	Brightness Setting 0 Even Symbol
0x2121:1-6	0x3C:0-6	Brightness Setting 1 Even Symbol
0x2207:0	0x67:0	Global Brightness Symbols
Brightness settings Halo + Symbol		
0x2103:0	0x28:0	Minimum Brightness Value
0x2123:1-31	0x3D:0-30	Brightness Setting 2 Indiv

Lightning management

0x2200:1-6	0x65:0-5	Active Brightness Setting Symbols
0x2201:1-7	0x64:0-4	Active Brightness Setting Halos
0x2203:1-7	0x6D:0-6	Active Colour Setting Halos
0x2104:0	0x29:0	Flash slow on time ms
0x2105:0	0x2A:0	Flash slow period ms
0x2106:0	0x2B:0	Flash fast on Time ms
0x2107:0	0x2C:0	Flash fast period ms
0x2204:1-6	0x6B:0-5	Temporal Pattern Symbols
0x2205:1-7	0x6A:0-6	Temporal Pattern Halos
0x2208:1-6	0x68:0-6	Activate Halo LEDs
0x6200:1	0x6C:0	Activate Symbol LEDs
0x6200:2	0x69:0	Activate Halo LEDs

11.2.2 Linking the objects

The interaction of the individual objects for colour and brightness adjustment can be described most easily using the following formula:

$$Output\ Brightness = \ Global\ Brightness\ Setting * \frac{Brightness\ Setting\ X}{250} * \frac{Colour\ Setting\ X}{250}$$

For the Halo LEDs. The same results for the brightness of the symbol LEDs:

$$Output\ Brightness = \ Global\ Brightness\ Setting * \frac{Brightness\ Setting\ X}{250}$$

The output brightness value resulting from the formulas is influenced by the following additional objects and at last send to the LED driver.

<u>Minimum brightness value:</u>

This object determines the minimum brightness of the LEDs.

$Output\ Brightness\ New = Output\ Brightness$, if output brightness > minimum brightness value,

Else:

$Output\ Brightness\ New = Minimum\ Brightness\ Value.$

<u>Temporal pattern symbols and temporal pattern values:</u>

A special lighting function can be selected for the symbol and halo ring lighting using the two objects temporal pattern symbols and halos. The following functions are available:

0 = continuous lightning:

$Output\ Brightness\ New = Output\ Brightness$

1 = flash slow:

$Output\ Brightness\ New = Output\ Brightness$, for T_{On} ms,

(T_{On} = value from object 0x2104)

$Output\ Brightness\ New = 0$, for T_{Off} ms,

(T_{Off} = value from object 0x2105 – value from object 0x2104)

2 = flash fast:

$Output\ Brightness\ New = Output\ Brightness$, for T_{On} ms,

(T_{On} = value from object 0x2106)

$Output\ Brightness\ New = 0$, for T_{Off} ms,

(T_{Off} = value from object 0x2107 – value from object 0x2106)

3 = pulsate:

$Output\ Brightness\ New = Output\ Brightness * f_{pulsate}(t)$

4 = rotate:

$Output\ Brightness\ New = Output\ Brightness * f_{rotate}(t)$

<u>Activate halo LED’s (Object 6200:2), Activate symbol LED’s (Object 6200:1), Activate halo LED’s (Object 2208:1-6):</u>

The PWM output can be activated via the objects if the bit belonging to the LED is set or deleted in the objects (see K-Matrix CANopen). The difference between the objects 0x6200: 2 and 0x2208 is that with the object 0x2208: 1-6 each of the four individual RGB LEDs of the halo rings can be activated or deactivated, whereas with object 0x6200: 2 the complete halo ring is activated or deactivated . A haloring must first be activated via object 0x6200: 2 and then individual LEDs must be deactivated via object 0x2208. Expressed in formula, this means:

$Output\ Brightness\ New = Output\ Brightness$, if Bit in 0x6200:2 and 0x2208 is set,

Otherwise:

$Output\ Brightness\ New = 0.$

The symbol lighting is activated in the same way as the halo rings via object 0x6200: 1.

Further objects with influence on the symbol and halo ring lighting are: active brightness setting symbols (0x2200), active brightness setting halo (0x2201), active colour setting halo (0x2203). The objects 0x2200, 0x2201, 0x2203 can be used to choose between the various settings for the brightness and colour settings. You can choose between 4 memory positions for the colour setting of the halo rings. You can choose between three storage positions for adjusting the brightness of the halo rings and symbol illumination.

11.3 Communication timeout

This function is available in the J1939 version of the keypad only.

The keypad offers the functionality to signalize a communication timeout. If no “Prop A”-message is received for a specified time the halo LEDs of the keypad will start to flash white. Communication can be started again by sending “Prop A”-messages. Once in communication timeout mode the LED settings will revert back to the off state. Other values that were stored in volatile memory will not be lost. The LEDs will just be turned off.

The timer for when the timeout mode will be entered can be configured in the corresponding lost communication timeout object (0xA3). The default value of this object are 2 seconds. Please refer to the K-Matrix to see which values can be chosen.

12. CANopen-Safety communication protocol

- For detailed information, please have a look at the corresponding K-matrix.

13. Rotary Cursor controller

Next to the 6 button keypad 1707, a keypad with 2 buttons and a rotary cursor controller expand the S09 series. The RCC keypad 1708 is set up on the hard- and software platform of the 6 button. The communication with the RCC keypad are totally the same as for the 6 button keypad and the protocol J1939 or CANopen/CANopenSafety. The objects and messages that are used especially for the RCC are described in the following sections.

13.1 Communication Objects

Object name	Index	Subindex	Description	Data Type
RCC Direction	0xB4 (J1939) 0x2400 (CANopen)	0x00 (J1939) 0x01 (CANopen)	RCC Direction X + = left - = right	unsigned int 8 (J1939) signed int 8 (CANopen)
		0x01 (J1939) 0x02 (CANopen)	RCC Direction Y + = up - = down	unsigned int 8 (J1939) signed int 8 (CANopen)
RCC Rotation	0xB5 (J1939) 0x2401 (CANopen)	0x00	Rotation (notch) position of RCC 0x00: reserved 0x01 – 0x14: position 0x15 – 0xFC: reserved 0xFE: error 0xFF: not available	unsigned int 8
RCC Button State	0xB6 (J1939) 0x2402 (CANopen)	0x00	Pressed state of the RCC 0x00: not pressed 0x01: pressed 0x02 – 0xFC: reserved 0xFE: error 0xFF: not available	unsigned int 8
RCC Temperature	0xB7 (J1939) 0x2403 (CANopen)	0x00 (j1939) 0x01 (CANopen)	RCC Temperature Base	unsigned int 8 (J1939) signed int 16 (CANopen)
		0x01 (J1939) 0x02 (CANopen)	RCC Temperature Handle	unsigned int 8 (J1939) signed int 16 (CANopen)
RCC Unique ID	0xB8 (J1939) 0x2405 (CANopen)	0x00	Unique ID Byte 0 (J1939) Unique ID (CANopen)	unsigned int 8 (J1939) unsigned int 32 (CANopen)
		0x01	Unique ID Byte 1	unsigned int 8 (J1939)
		0x02	Unique ID Byte 2	unsigned int 8 (J1939)
		0x03	Unique ID Byte 3	unsigned int 8 (J1939)
		0x04	Unique ID Byte 4	unsigned int 8 (J1939)
		0x05	Unique ID Byte 5	unsigned int 8 (J1939)
RCC Version	0xB9 (J1939) 0x2404 (CANopen)	0x00 (J1939) 0x02 (CANopen)	RCC Hardware Version	unsigned int 8
	0xBA (J1939) 0x2404 (CANopen)	0x00 (J1939) 0x01 (CANopen)	RCC Software Version	unsigned int 8

13.2 Working with the RCC

The lightning functions of the RCC are equal to the normal buttons except two points:

- The RCC don’t have a symbol and no symbol illumination
- The halo-ring of the RCC are illuminated with only one led and not 4 as the buttons

Normally, the RCC are referenced with the index 7 for configure the lightning parameters. Due to point one no index for RCC are implemented for the symbol lightning. Due to point two the halo-ring of the RCC does not support the rotate function.

The object RCC direction provides the values of the joystick function of the RCC. The interpretation of the values are different between the protocol J1939 and CANopen. A J1939 keypad sends the position of the joystick as an unsigned 8-bit value with an offset of 125 that means 125 are the neutral position. The range of the value is ± 7 that for maximal positive deflection 132 and for maximal negative deflection 118 are send. There is no difference between the X and the Y direction.

A CANopen keypad sends the X and Y deflection as a signed 8-bit value with a value range of ± 7. The maximal negative deflection are reached at -7, the maximal positive deflection with +7.

The RCC keypad provide a proportional signal from the RCC and uses the whole value range from -7 to +7 (or 118 to 132 for J1939 keypads) to represent the deflection. With these version it is possible for the customer to define a specific switching threshold for every application.

The rotation position of the RCC can be read out by the RCC rotation object. The value are in the range of 1 – 20 and are an absolute value of the RCC. After a Power-On-Reset of the RCC the same value for RCC rotation are read out as before.

The remaining objects for RCC communication are described sufficient under point 13.1. Additionally the information are documented in the K-matrix of the keypad.

14. Operation

14.1 General information

After switching on the supply voltage, the keypad will boot automatically. If the keypad uses J1939-protocol and the keypad receive no messages on the bus it will start blinking after 2 seconds. The operating temperature inside the keypad is continuously monitored. When a limit temperature of 85°C is exceeded, measures to lower the internal operating temperature are initiated automatically, starting with reducing the button illumination brightness.

A functional status of the application can be realised by a defined colouring of the illumination (colour change) or by the illumination segments of the halo ring illumination which are switched on and off successively (e.g. flashing in a defined frequency, sequentiallight, etc.). Combinations of colour and flashing, sequential etc. are also possible (e.g. rotating in different colours).

☞ The button illumination of the keypad can be used for function and fault indication of the application e.g. by flashing or colour change.

15. Cleaning

Cleaning with water and commercially available soft cloths is possible. Do not use solvents. Please ensure that the distance between the nozzle and the product is not less than 0.5 m when using a high pressure water jet.

⚠	**Caution!** Cleaning with a high-pressure water jet is not permitted with a distance nozzle to product less than 0.5m. – Damage to the keypad – Electrical shock due to leakage of the keypad

16. Optional accessories

Symbol inserts with customer-specific symbols or alternatively according to ISO 7000 can be procured. Symbol inserts without symbols (blanks) can also be procured.

A insert tool for changing the symbol inserts are available (article number: 09-0A00.0001).

17. Liability for quality defects

The general function of the keypad has been tested at the factory before delivery. However, if errors occur despite the careful quality control, they must be reported immediately to EAO or the dealer.

The liability for quality defects is 12 months for the delivered products. Within this period of time, faulty parts, except wear parts, will be repaired or replaced free of charge if the keypad is returned to EAO, free of charge.

Damages caused by improper use or the use of force are excluded from the liability for quality defects. Damages caused by repairs or modifications to the keypad are also excluded. EAO is exclusively responsible for repairs to the keypad.

Further claims cannot be asserted. Claims arising from the purchase contract remain unaffected.

EAO is not liable for consequential damages. The right to design changes, in particular in the sense of product improvement, is reserved.

18. Service, repair

In case of a defective keypad, cable or connector, please contact your dealer. In case of malfunctions, the cause of which you cannot clearly identify, please send the defective keypad to the following address:

EAO Automotive GmbH & Co. KG
Service
Richard-Wagner-Strasse 3
08209 Auerbach/ Vogtl.
Tel: +49 (0)3744/ 8264-0
Email: service.esa@eao.com
www.eao.com

19. Decommissioning, disposal

Disconnect the system from the power supply before the disconnection.

Do not pull at the cable when removing the connector.

Dispose of the device, components and accessories, packaging materials and documentation in accordance with the country-specific waste treatment and disposal regulations in the area of use.

20. Declaration of Conformity

The declaration of conformity certificate is available for download on the EAO-homepage www.eao.com within the download section. The document valid for the product listed in this manual is:

Series 09, Rugged CAN Keypads - CE-Certification - Compliancy of EAO Products,
from July,15, 2021

In case of further certificates are required, please ask your dealer.