



**Commercial
Vehicles**

Handbook

Customer-Specific Functional Control Unit (Kundenspezifisches Funktionssteuergerät, KFG)

Order no.: 100174

Revision: 0.5

Version dated: March 2017



Commercial
Vehicles

VW Commercial Vehicles

Legal notice

Volkswagen AG

Berliner Ring 2

38440 Wolfsburg

Copyright

© 2017 Volkswagen AG, Wolfsburg



Technical System Description	5
1 Introduction	5
1.1 Variants	6
1.1.1 Basic variant	6
1.1.2 Max variant	7
1.2 Free programming	8
2 Function description	10
2.1 Connections	10
2.2 Supply	15
2.3 Signal concept	16
2.4 Description of input filters	17
2.5 Inputs and outputs	18
2.5.1 Multifunction input (MFE)	18
2.5.2 Multifunction outputs (MFA)	27
2.6 Application functions	45
2.6.1 Configurable logic module	45
2.6.2 Configurable arithmetic modules (AU)	55
2.6.3 Permanently coded functions	63
2.6.4 Permanently coded functions (safety)	74
2.6.5 Enhanced functions	77
2.7 Communication interfaces	82
2.7.1 CAN	83
2.7.2 Communication module (CM) (max variant)	86
3 Safety concept	112
3.1 Initialisation	112
3.2 Error detection	112
3.3 Signal flow	113
3.4 Application	113
3.5 Watchdog	114
3.6 Signal pool	114
3.7 Secure discreet input	114
3.8 Secure CAN signal	115
3.9 Secure function	115
3.10 Secure SPS	115
3.11 Secure output	116
3.12 Secure parametrisation	116
3.13 Detection of main controller failure	116
3.14 Detection of safety controller failure	116
4 Specifications	117
4.1 Absolute thresholds	117
4.2 Technical data	117

Developer manual	118
5 The interface library for the KFG app interface	118
5.1. Introduction	118
5.2. Planned scope of application	118
5.3. KFG app interface	118
5.3.1. Data types	118
5.4. Architecture of KFG client and KFG server	119
5.4.1. KFG client communication	119
5.4.2. Functions	122
5.5. KFG client – general methods	123
5.5.1. Synchronous data requests	123
5.5.2. Subscribing to data objects	123
5.5.3. Removing a subscription	124
5.5.4. Accessing functions	124
6 Example application	125
6.1. Setting up Android Studio	125
6.2. Opening an example application project in Android Studio	125
6.3. Creating an Android project	128
6.4. Adding files from the interfaces library for the KFG App interface to an Android Project	131
6.5. Setting up the data connection to the KFG	132
6.5.1. App development for WLAN connection	132
6.5.2. App development for Bluetooth connection	134
6.6. Functions	136
6.6.1. Searching for KFG devices and displaying the devices found in the app	136
6.6.2. Displaying the device information and establishing a connection to the device from the app	137
6.6.3. Requesting the KFG identifier and testing for the expected unique number by the app	138
6.6.4. Displaying the engine speed in the app	140
6.6.5. Displaying the door locking status in the app	142
6.6.6. Locking and unlocking the doors using the app	143
6.6.7. Displaying the number of fastened seatbelts in the app	145
6.7. Overview of classes	146

Technical System Description

1 Introduction

This document includes a technical overview of the KFG variants and offers a detailed description of the individual KFG functions. These functions can be parametrised using a configuration tool.



Abb. 1.1 View of KFG max variant

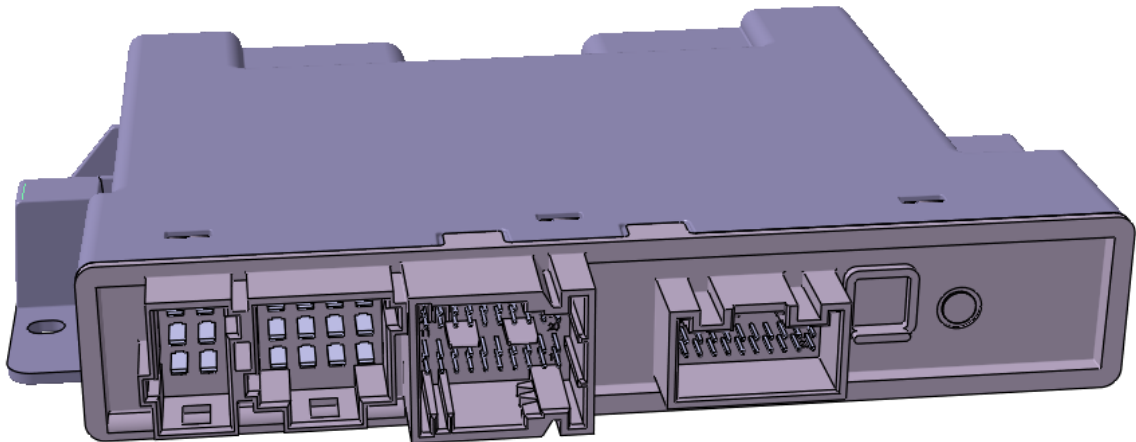


Abb. 1.2 View of KFG basic variant

1.1 Variants

1.1.1 Basic variant

The basic variant comprises the basic functions and extended functions from the specifications. For output/input various interfaces are available:

- 8 discreet wake-up-capable low active inputs
- 8 discreet wake-up-capable high active inputs
- 8 discreet wake-up-capable analogue inputs
- 4 discreet half bridge outputs with a nominal current of 5A and up to 300Hz PWM
- 6 discreet high-side outputs with a nominal current of 5 A
- 2 discreet high-side outputs with a nominal current of 10 A
- 6 discreet high-side outputs with a nominal current of 5 A
- 2 discreet low-side outputs with a nominal current of 1 A and up to 5 kHz PWM
- 4 discreet high-side outputs with a nominal current of 0.5 A and up to 300 Hz PWM
- 4 discreet low-side outputs with a nominal current of 0.5 A and up to 300 Hz PWM
- 2 discreet outputs with relay switches with a nominal current of 1 A, galvanically isolated work circuit
- 3 CAN connections
- 1 LIN connection

Special features:

- Temperature monitoring
- Overvoltage and undervoltage shutoff
- Polarity reversal protection in term. 30 cable
- Type of protection IP5K1

The following basic functions are available:

- Configurable logic- and calculation blocks
- Outputs and inputs with flexible configuration
- Light control
- Window regulators
- Taxi and special vehicle functions

The following extended functions are available:

- Working speed governor
- Power take-off
- Speed limiter
- Remote engine start/stop

1.1.2 Max variant

For the max variant, the basic variant is enhanced with a connectivity module. This module facilitates the connection of additional interfaces:

- WLAN access point
- Bluetooth
- USB host with charging function up to 1.5 A

1.1.2.1 Radio range and framework conditions

Due to the installation position (behind the glove compartment, suspended, connector pointing down), it is possible to provide WLAN coverage for the vehicle interior and the area surrounding the vehicle up to a maximum radius of 10m around the KFG. As an option, an external aerial can be connected for larger ranges.

For multiple use of radio interfaces in other devices as well (additional WLAN access points or Bluetooth service), the following limitations apply: The spacing from each KFG edge to the next transmitter aerial must be at least 12 cm.

WLAN:

If the channel assignment is specified as either automatic or fixed (with 3 WLANS e.g. 1-5-9), no impairments of the KFG are expected.

Bluetooth:

The coexistence of additional Bluetooth devices and connections does not result in a fundamental impairment of the KFG. However, as the density increases, limitations in the data transfer rate are likely.

The start time of the connectivity module is approx. 5-7 seconds, depending on the activated software modules. This value includes the whole boot process from reset until external devices can attempt to make a connection.

All functions of the connectivity module can be made available up to a term. 30 voltage of 6.5 V.

For vehicles with backup battery:

The connectivity module can retain full functionality during all start voltage drops.

For vehicles without backup battery:

The connectivity module can retain full functionality during a warm start.

During a cold start, however, the function of the connectivity module is no longer assured. Depending on the extent of the voltage drop below 6 V, the system responds as follows:

- The USB charging function fails. A connected device reconnects itself at higher voltage, a previously active charging process will resume.
- The radio connection (WLAN + BT) fails. A connected device can reconnect at higher voltage.
- The micro-controller of the connectivity module performs a reset. At higher voltage, the module will restart.

1.2 Free programming

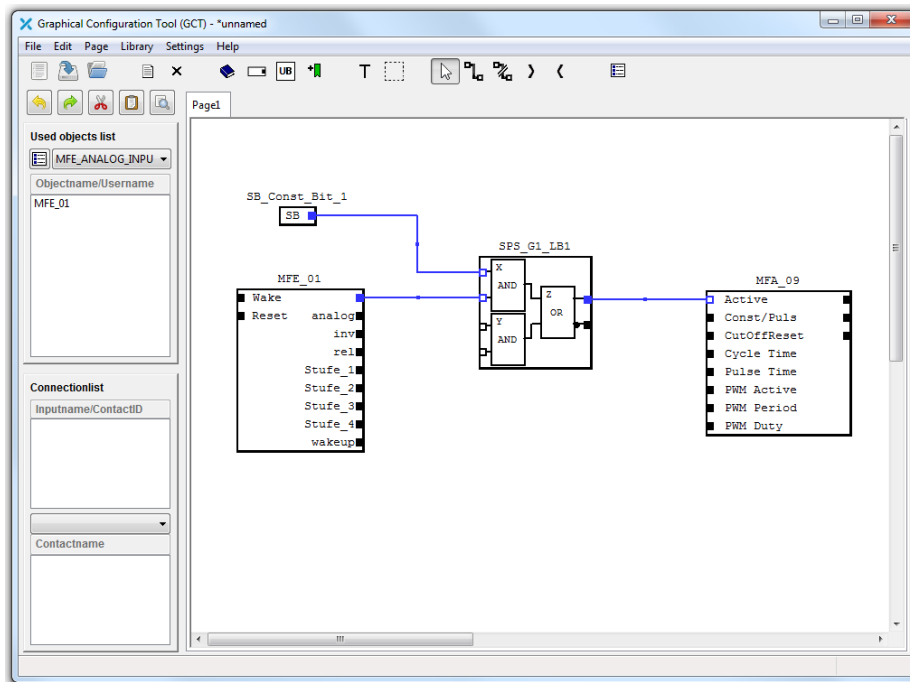


Abb. 1.3 Graphical configuration tool

The GCT software is the graphical solution for free programmability of the KFG. Handling is simplified for various functions such as inputs and outputs, application functions and communication modules of the KFG. The functions can be selected easily from a library using "drag & drop" and connected on the user interface with a few mouse clicks. The possibility of distributing the connected function blocks over different drawing areas means that even complex correlations can be represented well. In addition to the graphical connection, the application offers the means of parametrising the function blocks.

Also, the software allows for adding "user-defined" function blocks to the library based on functions which were generated by connecting and parametrising blocks. The creator of a block determines which inputs and outputs are routed outwards and defines the parameters which can be parametrised by the end user of the function block. These "user-defined" functions are protected from unauthorised inspection. Only the owner of the master file can view this function and change it retrospectively.

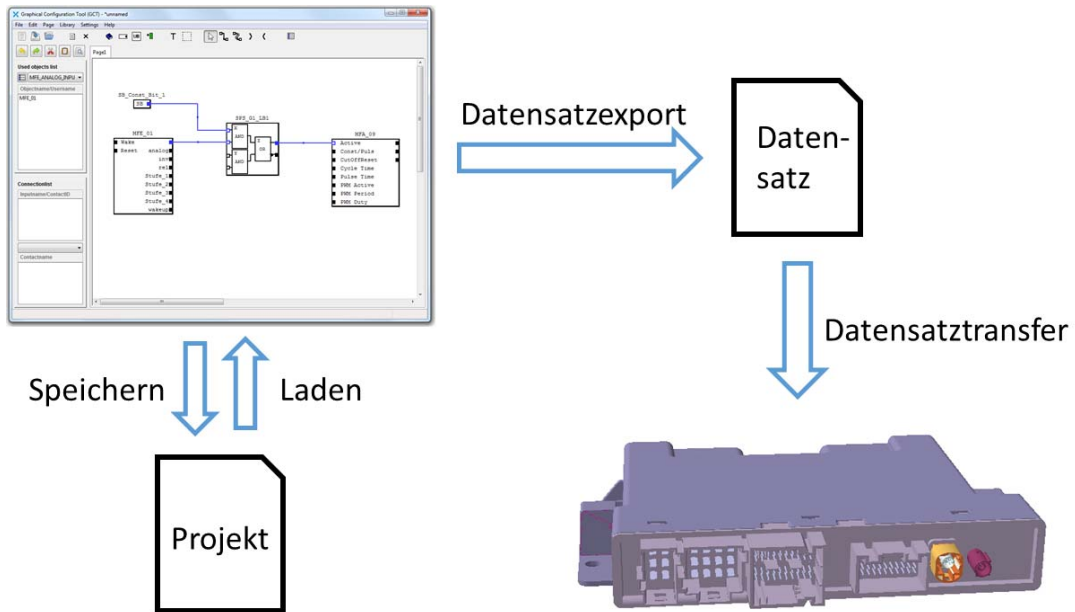


Abb. 1.4 Export of a configuration

The created configurations can be saved and reloaded. In addition, the configurations can be exported and then be transferred to the KFG via a data record transfer.

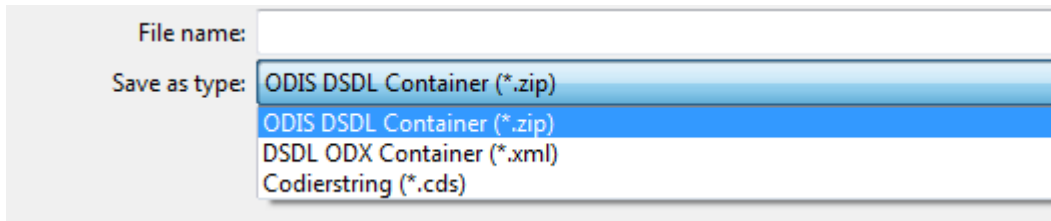


Abb. 1.5 Data record types



Operation of this software is not covered by this document. Details are available in a separate user handbook.

2 Function description

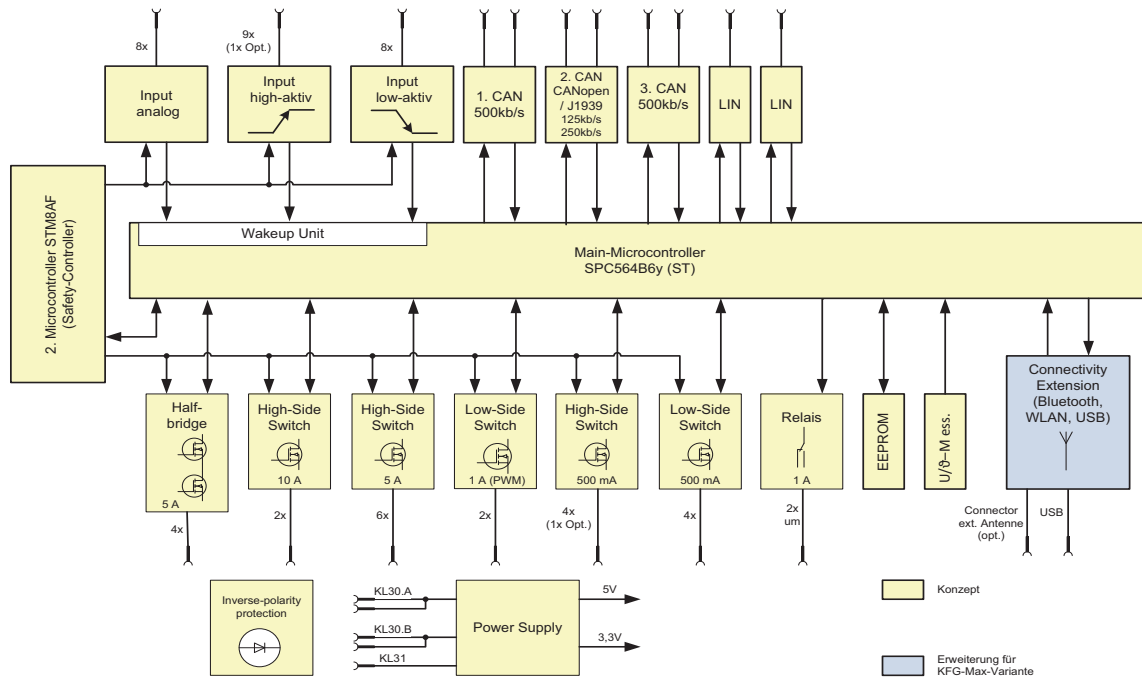


Abb. 2.1 Block circuit diagram

The KFG comprises the main controller, the end stages, the input stages and the communication interfaces. A second micro-controller (safety controller) serves as an I/O extension of the main controller and as intelligent watchdog. This safety controller is also responsible for actuation of the outputs should the main controller completely fail.

For the radio interfaces (WLAN and Bluetooth) and the USB connection, a second circuit board (connectivity module CM) is mounted on the main circuit board that can communicate via an SPI connection with the main controller.

Data are exchanged via an internal interface between the main controller and the safety controller. Flashing of the safety controller is also possible in this way from the main controller.

2.1 Connections

For the inputs, signal outputs to 1A and the CAN and LIN interfaces, MQS connectors are used. The cable outputs and supply connections are assigned to an MCP connector with 2.8 mm pins.

For a USB interface, an HSD connection is fitted.

Distribution of the control unit connections to the connectors and their type is as follows:

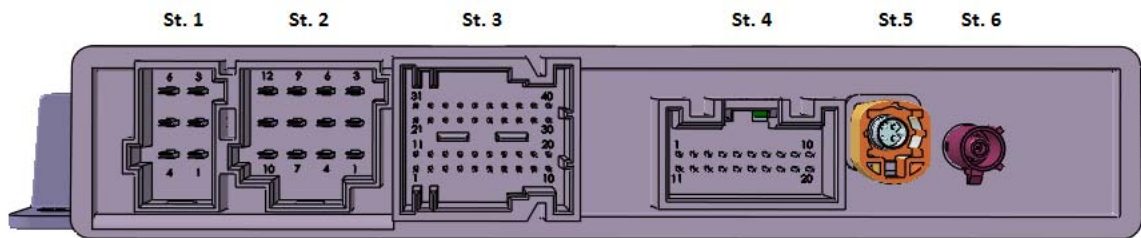


Abb. 2.2 KFG connector view

ST1 supply connection:

Equipment mounting WSK 013 040 CL with 6 pins, flat contact housing 4F0 972 706, coding B-C connection for 4 x term. 30 and 2 x term. 31 via 2.8 mm flat connector

Connector 1		
Pin no.	6	3
Signal	Terminal 30_1	Terminal 30_1
Pin no.	5	2
Signal	Terminal 30_2	Terminal 30_2
Pin no.	4	1
Signal	GND	GND

ST2 cable outputs:

Device mounting WSK 013 040 CL with 12 pins, flat contact housing 4F0 973 712, coding B-C connection for 4 x 5 A half bridge, 6 x 5 A output and 2 x 10 A output via 2.8 mm flat connector

Connector 2				
Pin no.	12	9	6	3
Signal	MFA_2	MFA_19	MFA_1	MFA_6
Pin no.	11	8	5	2
Signal	MFA_21	MFA_20	MFA_4	MFA_5
Pin no.	10	7	4	1
Signal	MFA_22	MFA_3	MFA_8	MFA_7

ST3 control outputs, discreet inputs:

Equipment mounting WSK 013 070 BC with 40 pins, flat contact housing 4H0 906 231 connection for 4 x 0.5A high-side, 4 x 0.5A low-side, 2 x 1A PWM low-side, 6 x relay contacts, 8 x HS input, 8 x LS input, 8 x analogue input via 0.64mm MQS

Connector 3										
Pin no.	31	32	33	34	35	36	37	38	39	40
Signal	MFA_9	MFA_10	MFE_10	MFE_12	MFE_14	MFE_16	MFE_2	MFE_4	MFE_6	MFE_8
Pin no.	21	22	23	24	25	26	27	28	29	30
Signal	MFA_11	MFA_12	MFE_9	MFE_11	MFE_1	MFE_15	MFE_1	MFE_3	MFE_5	MFE_7
Pin no.	11	12	13	14	15	16	17	18	19	20
Signal	MFA_17	MFA_18	MFA_14	REL_NO2	REL_COM2	REL_NC2	MFE_18	MFE_20	MFE_22	MFE_24
Pin no.	1	2	3	4	5	6	7	8	9	10
Signal	MFA_15	MFA_16	MFA_13	REL_NO1	REL_COM1	REL_NC1	MFE_17	MFE_19	MFE_21	MFE_23

ST4 communication:

Equipment mounting WSK 013 070 C with 20 pins, flat contact housing 8W0 972 420, coding A connection for 4 x CAN high, 4 x CAN low, 2 x LIN, 1 x HW term.15, 2 x Ethernet (TRD0+, TRD0-) via 0.64mm MQS.

The necessary distance between the Ethernet pins and other communication pins is adhered to. Of the 20 pins, 4 are not equipped. They serve as a reserve for possible later enhancements.

Connector 4 (only max variant)										
Pin no.	1	2	3	4	5	6	7	8	9	10
Signal	CAN1_H	CAN1_L	CAN3_H	CAN3_L	NC	NC	LIN1	KL15	NC	NC
Pin no.	11	12	13	14	15	16	17	18	19	20
Signal	CAN2_H	CAN1_L	NC	NC	GND_A	GND_B	NC	NC	NC	NC

ST5 USB interface:

HSD connector, coding Z (neutral), water blue, RAL 5021

Connector 5 (only max variant)		
Pin no.	1	2
Signal	USB_GND	D-
Pin no.	3	4
Signal	USB +5V	D+

ST6 connector for external aerial:

FAKRA coaxial connector 90° angled, D coding, bordeaux, RAL 4004

Signals up to a frequency of 20 Hz can be picked up at discrete inputs. Higher frequencies are suppressed by HW and SW filters.



Table with pin assignment and description

Plug	Pin	Signal (current)	Description
ST1			Power supply connector
ST1	1	GND	Terminal 31, connection (2 pins)
ST1	2	KL30_2	Terminal 30, connection 2 (2 pins)
ST1	3	KL30_1	Terminal 30, connection 1 (2 pins)
ST1	4	GND	Terminal 31, connection (2 pins)
ST1	5	KL30_2	Terminal 30, connection 2 (2 pins)
ST1	6	KL30_1	Terminal 30, connection 1 (2 pins)
ST2			Power output connector
ST2	1	MFA_7	Output, high-side, 5 A, KL30_2
ST2	2	MFA_5	Output, high-side, 5 A, KL30_1
ST2	3	MFA_6	Output, high-side, 5 A, KL30_1
ST2	4	MFA_8	Output, high-side, 5 A, KL30_2
ST2	5	MFA_4	Output, high-side, 10A, KL30_2
ST2	6	MFA_1	Output, high-side, 5 A, KL30_1
ST2	7	MFA_3	Output, high-side, 5 A, KL30_2
ST2	8	MFA_20	Output, half bridge 2, 5 A, KL30_1
ST2	9	MFA_19	Output, half bridge 1, 5 A, KL30_1
ST2	10	MFA_22	Output, half bridge 2, 5 A, KL30_2
ST2	11	MFA_21	Output, half bridge 1, 5 A, KL30_2
ST2	12	MFA_2	Output, high-side, 10A, KL30_1
ST3			Power output connector
ST3	1	MFA_15	Output, low-side, 300 Hz PWM, 0.5 A
ST3	2	MFA_16	Output, low-side, 300 Hz PWM, 0.5 A
ST3	3	MFA_13	Output, low-side, 5kHz PWM, 1A
ST3	4	REL_NO1	N/O relay contact, potential-free
ST3	5	REL_COM1	Changeover relay contact, potential-free
ST3	6	REL_NC1	N/C relay contact, potential-free
ST3	7	MFE_17	Input, high-side, wakeup-capable, digital
ST3	8	MFE_19	Input, high-side, wakeup-capable, digital
ST3	9	MFE_21	Input, high-side, wakeup-capable, digital
ST3	10	MFE_23	Input, high-side, wakeup-capable, digital
ST3	11	MFA_17	Output, low-side, 300 Hz PWM, 0.5 A
ST3	12	MFA_18	Output, low-side, 300 Hz PWM, 0.5 A
ST3	13	MFA_14	Output, low-side, 5kHz PWM, 1A
ST3	14	REL_NO2	N/O relay contact, potential-free
ST3	15	REL_COM2	Changeover relay contact, potential-free
ST3	16	REL_NC2	N/C relay contact, potential-free
ST3	17	MFE_18	Input, high-side, wakeup-capable, digital
ST3	18	MFE_20	Input, high-side, wakeup-capable, digital
ST3	19	MFE_22	Input, high-side, wakeup-capable, digital
ST3	20	MFE_24	Input, high-side, wakeup-capable, digital
ST3	21	MFA_11	Output, high-side, 300 Hz PWM, 0.5 A, KL30_1
ST3	22	MFA_12	Output, high-side, 300 Hz PWM, 0.5 A, KL30_1
ST3	23	MFE_9	Input, low-side, wakeup-capable, digital
ST3	24	MFE_11	Input, low-side, wakeup-capable, digital

Plug	Pin	Signal (current)	Description
ST3	25	MFE_13	Input, low-side, wakeup-capable, digital
ST3	26	MFE_15	Input, low-side, wakeup-capable, digital
ST3	27	MFE_1	Input, low-side, wakeup-capable, analogue
ST3	28	MFE_3	Input, low-side, wakeup-capable, analogue
ST3	29	MFE_5	Input, low-side, wakeup-capable, analogue
ST3	30	MFE_7	Input, low-side, wakeup-capable, analogue
ST3	31	MFA_9	Output, high-side, 300 Hz PWM, 0.5 A, KL30_1
ST3	32	MFA_10	Output, high-side, 300 Hz PWM, 0.5 A, KL30_1
ST3	33	MFE_10	Input, low-side, wakeup-capable, digital
ST3	34	MFE_12	Input, low-side, wakeup-capable, digital
ST3	35	MFE_14	Input, low-side, wakeup-capable, digital
ST3	36	MFE_16	Input, low-side, wakeup-capable, digital
ST3	37	MFE_2	Input, low-side, wakeup-capable, analogue
ST3	38	MFE_4	Input, low-side, wakeup-capable, analogue
ST3	39	MFE_6	Input, low-side, wakeup-capable, analogue
ST3	40	MFE_8	Input, low-side, wakeup-capable, analogue
ST4			Communication connector
ST4	1	CAN1_H	Convenience CAN bus (high)
ST4	2	CAN1_L	Powertrain CAN bus (low)
ST4	3	CAN3_H	Convenience CAN bus (high)
ST4	4	CAN3_L	Powertrain CAN bus (low)
ST4	5	NC	Not connected
ST4	6	NC	Not connected
ST4	7	LIN1	LIN1
ST4	8	KL15	Input, high-side, wakeup-capable, digital
ST4	9	NC	Not connected
ST4	10	NC	Not connected
ST4	11	CAN2_H	Vehicle body manufacturer CAN bus, CANopen (high)
ST4	12	CAN2_L	Vehicle body manufacturer CAN bus, CANopen (low)
ST4	13	NC	Not connected
ST4	14	NC	Not connected
ST4	15	GND_A	Earth (500 mA max.)
ST4	16	GND_B	Earth (500 mA max.)
ST4	17	NC	Not connected
ST4	18	NC	Not connected
ST4	19	NC	Not connected
ST4	20	NC	Not connected
ST5			HSD_USB connector
ST5	1	USB GND	USB earth
ST5	2	D -	USB D- (TX-)
ST5	3	USB +5 V	5 V USB power supply
ST5	4	D+	USB D+ (TX+)
ST6			FAKRA aerial connector
ST6	Inside	ANT	External aerial
ST6	Shield	ANT GND	External aerial earth

2.2 Supply

- Power comes from 2 separate term.30 supplies
- Basic system chip for supply of 5 V for CAN interfaces and for supply of 3.3 V for MCU and EEPROM
- Voltage measurements for
 - term. 30A and term. 30B
 - Term.30_internal
 - 5 V-CAN

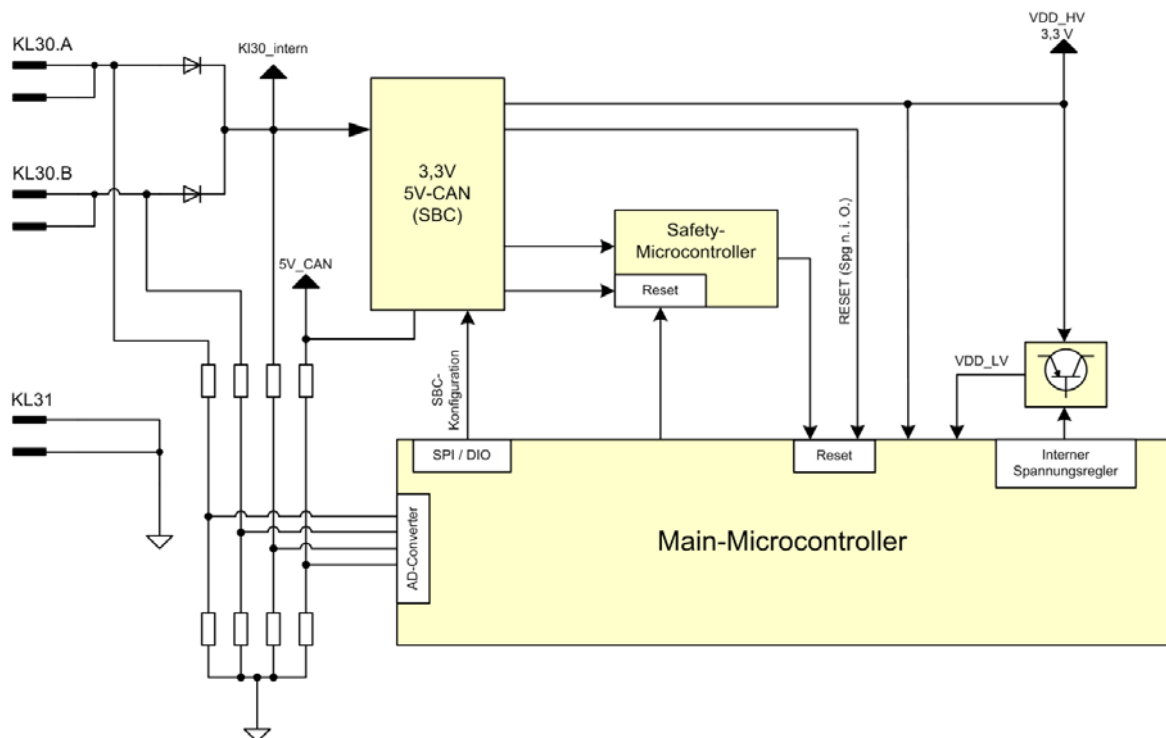


Abb. 2.3 Block circuit diagram voltage supply

KL30

Term. 30A and term. 30B are supplied by two 2.8mm MCP pins respectively. The power outputs (0.5, 1, 5 and 10A) are distributed evenly over both supply cables. Supply of the micro controller unit and the connectivity circuit board is from both term. 30 cables, which means that in the event that one cable should fail, the KFG itself can continue to function.

Term.31

Term.31 is connected to two 2.8mm MCP pins. Both cables have to be routed to a common earth point. No potential equalisation currents may flow via the KFG.

2.3 Signal concept

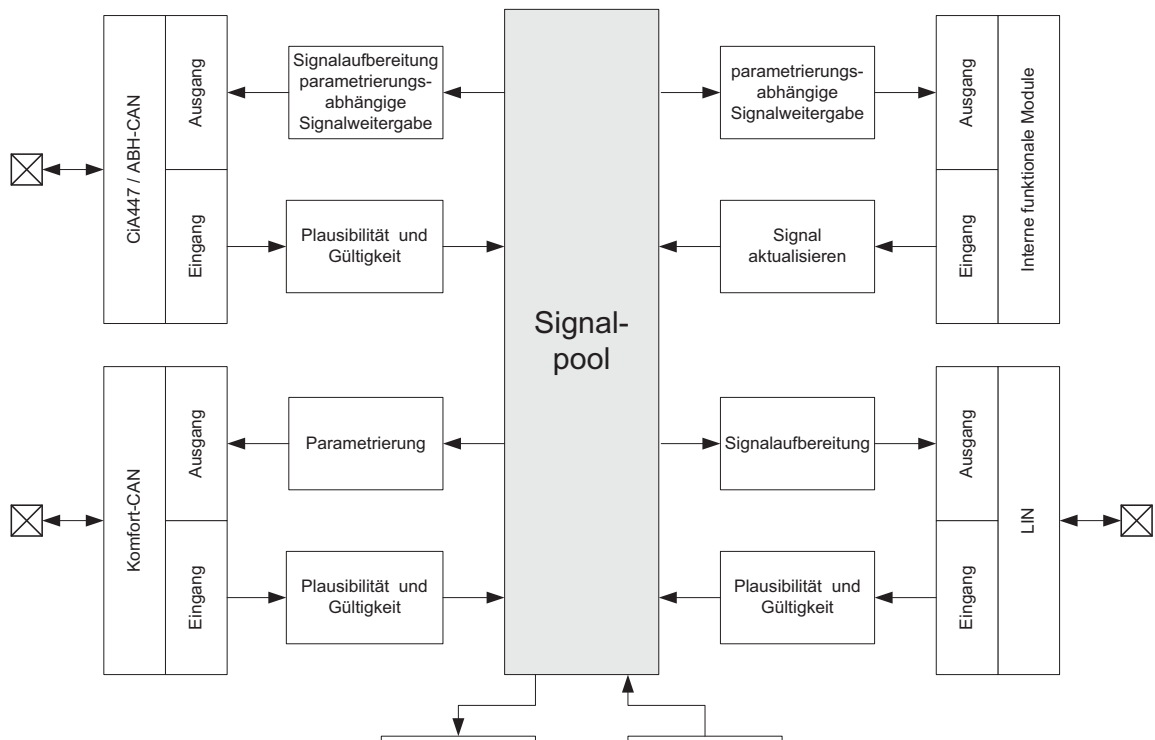


Abb. 2.4 Signal concept

The central element in the software is the signal pool. It records all signals from the signal sources: communication, measured values, discrete inputs, output status, etc. Through a standardised interface, the application modules can access random signals from the signal pool either directly or via so-called parametrisable function inputs. The allocation as to which signal is evaluated by a function input can be determined freely through diagnosis (variant coding).

CAN reception signals are written to the signal pool by means of an internal interface.

2.4 Description of input filters

The filter functions of certain inputs of the function blocks are identical in structure but can be parametrised individually.

A filter block is generally active. If one or more filter parameter bits are placed (status unequal to "0"), the respective filter signal is evaluated in relation to the input signal as well (logically "AND")

The filter signals 1-3 have a special position as they do not represent inverted outputs of logic blocks, which means that they can be configured variably. If filter bit 8 is set, the input signal is used logically inverted.

NR	Filter signal	Description
1	Group 12 logic block 1 output A	Takes account of the status of output A (non-inverted output) from group 12 logic block 1
2	Group 12 logic block 2 output A	Takes account of the status of output A (non-inverted output) from group 12 logic block 2
3	Group 12 logic block 3 output A	Takes account of the status of output A (non-inverted output) from group 12 logic block 3
4	CiA447 awake	Bit is set if CiA447 CAN is active.
5	Terminal change term.15R to term.15	Bit is set with terminal change from term.15R to term.15, reset is automatic after 3 SPS call-up cycles.
6	No over-temperature	Bit is set as long as no over-temperature is detected in KFG.
7	Terminal 58	Bit is set as long as term.58 signal is unequal to 0.
8	Input signal inverted	The input signal is evaluated inverted
9	ICAN error	Bit is set as long as control unit is in limp home mode.
10	No under-voltage	No under-voltage present.
11	Terminal 61	Function with "engine running"
12	Terminal 15	Function from "ignition ON"
13	Terminal 15R	Function from "ignition key in radio position"
14	Terminal 15C	Function from "ignition key inserted"
15	Central locking open	CL status is unequal to "externally locked"
16	Run-on time active	Function as long as at least one reason to keep ECU awake (inc. run-on time) is active.

2.5 Inputs and outputs

2.5.1 Multifunction input (MFE)

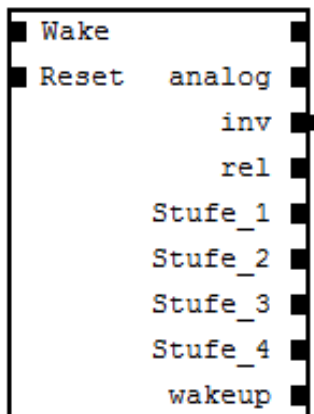
The KFG has a total of 24 discreet inputs. These MFEs can be parametrised individually using a configuration tool. It should be noted that not all MFEs have the same capability. The available inputs are listed as follows

- 8 positive active (high-side) digital inputs
- 8 earth active (low-side) digital inputs
- 8 earth active (low-side) analogue inputs

Essentially, the inputs can be assigned to two categories. These categories are also the same in the configuration tool.

- MFE_ANALOG
- MFE_DIGITAL

2.5.1.1 Function block type MFE_ANALOGUE



MFE_ANALOG

There are a total of 8 analogue inputs (MFE_Analog). These are designated MFE_01 – MFE_08 in the configuration tool.

The MFEs consist of:

- 8 earth active (low-side) analogue inputs

The analogue inputs are subject to digital and analogue evaluation and their input values are therefore available as digital, analogue and also 4 configurable thresholds.

The inputs can be used as buttons or switches.

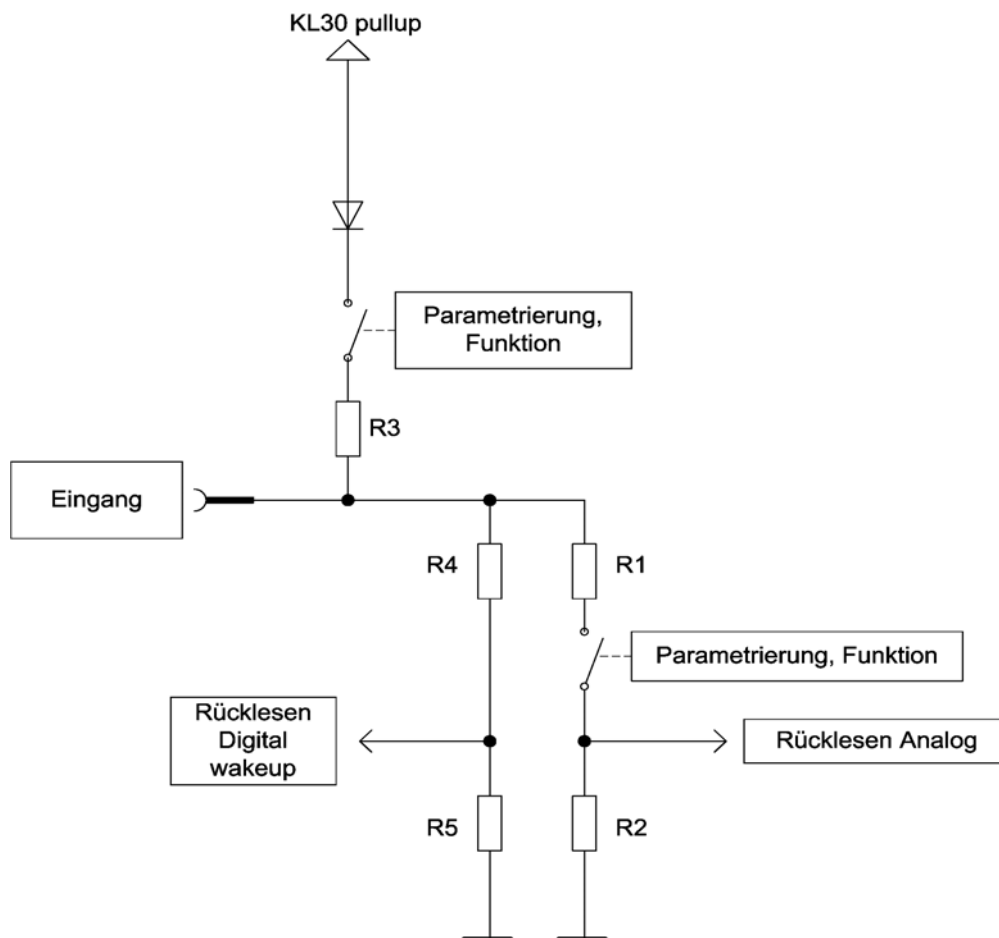
Circuit diagram and special features

Block	Description
MFE_01	Pull-up resistance, configurable via evaluation
MFE_02	Pull-up resistance, configurable via evaluation
MFE_03	Pull-up resistance, configurable via evaluation
MFE_04	Pull-up resistance, configurable via evaluation
MFE_05	Pull-up resistance, configurable via evaluation
MFE_06	Pull-up resistance, configurable via evaluation
MFE_07	Pull-up resistance, configurable via evaluation
MFE_08	Pull-up resistance, configurable via evaluation

The debounce time of the inputs with active control unit is: 20 ms±5 ms



Circuit diagram



Resistance values:

- R1 = 100 kOhm
- R2 = 18.7 kOhm
- R3 = 1.82 kOhm
- R4 = 100 kOhm
- R5 = 64.9 kOhm

	Digital input	Threshold values	Alarm button	Reset button	Paddle	Voltage measurement	Thresholds, configurable	Description
Wake-up	x	x	x	x	x	x	x	Input type for which wake-up function is available (configurable via "evaluation" parameter)
Pull-up	x	x	x	x	x	-	x	Input type for which a pull-up resistor (R3) is activated (configurable via "evaluation" parameter) The pull-up resistor is activated in the sleep mode of the control unit, if the input is configured as wakeup-capable. If the input is not configured as wakeup-capable, the pull-up resistor is deactivated in the sleep mode.

Electrical properties

Current flow convention: positive currents flow into the device, negative currents flow out. Unless otherwise defined, voltages are specified against local earth (term. 31) of the device.

Parameters	Conditions	Symbol	min	Type	max	Unit
Maximum input voltage	$U_{\text{Bat}} = 9 - 16\text{V}$ $T_{\text{amb}} = -40 - 85^{\circ}\text{C}$	U_{in}	-1		21	V
Scale:	$U_{\text{Bat}} = 9 - 16\text{V}$ $T_{\text{amb}} = -40 - 85^{\circ}\text{C}$	$U_{\text{in_mess}}$	0		20	V
Input current	$U_{\text{Bat}} = U_{\text{in}} = 14\text{V}$ 25°C, pull-up off	$I_{\text{in_14V}}$	70		250	μA
Input current	$U_{\text{Bat}} = 14\text{V}$, $U_{\text{in}}=0\text{V}$, 25°C pull-up on	$I_{\text{in_Pull}}$	-8.0		-6.0	mA

The inputs are measured by analogue means. Digital switching thresholds for the evaluation in awake state can be configured via diagnosis.

MFE1 to MFE8 can be parametrised as wakeup-capable. In this case the control unit will wake up upon an edge change on the wakeup-capable input. The relevant voltage levels for the wakeup function cannot be parametrised and are as follows:

Parameters	Conditions	Symbol	min	Type	max	Unit
Maximum low-level voltage	$U_{\text{Bat}} = 9 - 16\text{V}$ $T_{\text{amb}} = -40 - 85^{\circ}\text{C}$	U_{L}			2.2	V
Maximum high-level voltage	$U_{\text{Bat}} = 9 - 16\text{V}$ $T_{\text{amb}} = -40 - 85^{\circ}\text{C}$	U_{H}	6.2			V
Minimum hysteresis	$U_{\text{Bat}} = 9 - 16\text{V}$ $T_{\text{amb}} = -40 - 85^{\circ}\text{C}$	U_{hyst}	0.5			V

Inputs

Block	Width	Filter	Description
Wake	1 bit	no	Capable of waking up control unit (0) OFF (1) ON
Reset	1 bit	no	If the input is parametrised to the button (see parameter "Type"), an edge change (0) → (1) sets the output signal "MFE_xy" to 0. Static operation is not possible. With parameters set to "switch", the reset input has no function.

Parameters

Name	Description
Type	Selection: (0) Push-button The output signal status is toggled with each edge 0->1 on the input (MFE_xy or MFE_xy_inv). – A reset can be performed via the reset input. (1) Switch MFE_xy outputs the input state.
Evaluation	Selection for evaluation of output signals: (0) (0) Digital input (1) (1) Thresholds, 4 stages, fixed (2) (2) Alarm button (3) (3) Reset button (4) (4) Paddle, 560 ohms (5) (5) Voltage measurement (6) (6) Thresholds, configurable
Threshold stage 1	Switch threshold for "MFE_xy_stage_1" relative to supply voltage as % and parametrised evaluation.
Threshold stage 2	Switch threshold for "MFE_xy_stage_2" relative to supply voltage as % and parametrised evaluation.
Threshold stage 3	Switch threshold for "MFE_xy_stage_3" relative to supply voltage as % and parametrised evaluation.
Threshold stage 4	Switch threshold for "MFE_xy_stage_4" relative to supply voltage as % and parametrised evaluation.

Outputs

	Width	Digital input	Threshold values	Alarm button	Reset button	Paddle	Voltage measurement	Thresholds, configurable	Description
MFE_xy (digital)	1 bit	30%	25%	20%	20%	18%	n/a	Value, stage 1	Input voltage > activation threshold
MFE_xy_analog	16 bit	x	x	x	x	x	x	x	Input voltage in mV
MFE_xy_inv	1 bit	x	x	x	x	x	n/a	x	Logical inversion from "MFE_xy"
MFE_xy_rel	8 bit	x	x	x	x	x	x	x	Input voltage relative to operating voltage as %
MFE_xy_Stufe_1	1 bit	0%	0%	0%	0%	0%	n/a	K	Input voltage > threshold stage 1 → (1)
MFE_xy_Stufe_2	1 bit	30%	25%	20%	20%	18%	n/a	K	Input voltage > threshold stage 2 → (1)
MFE_xy_Stufe_3	1 bit	n/a	50%	27%	27%	22%	n/a	K	Input voltage > threshold stage 3 → (1)
MFE_xy_Stufe_4	1 bit	n/a	75%	41%	41%	36%	n/a	K	Input voltage > threshold stage 4 → (1)
Hysteresis (digital/stages)		-1%	-1%	-1%	-1%	-1%	-1%	-1%	Hysteresis for threshold evaluation

Key

- Not available

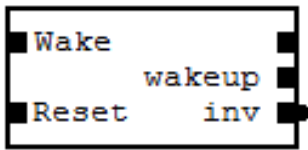
x Is active or available

C Is configurable

n/a Stage with 0xFFFF = Not used



2.5.1.2 Function block type MFE_DIGITAL



There are a total of 16 digital inputs (MFE_Digital). These are designated MFE_09 – MFE_24 in the configuration tool.

The MFEs consist of:

- 8 earth active (low-side) digital inputs
- 8 positive active (high-side) digital inputs

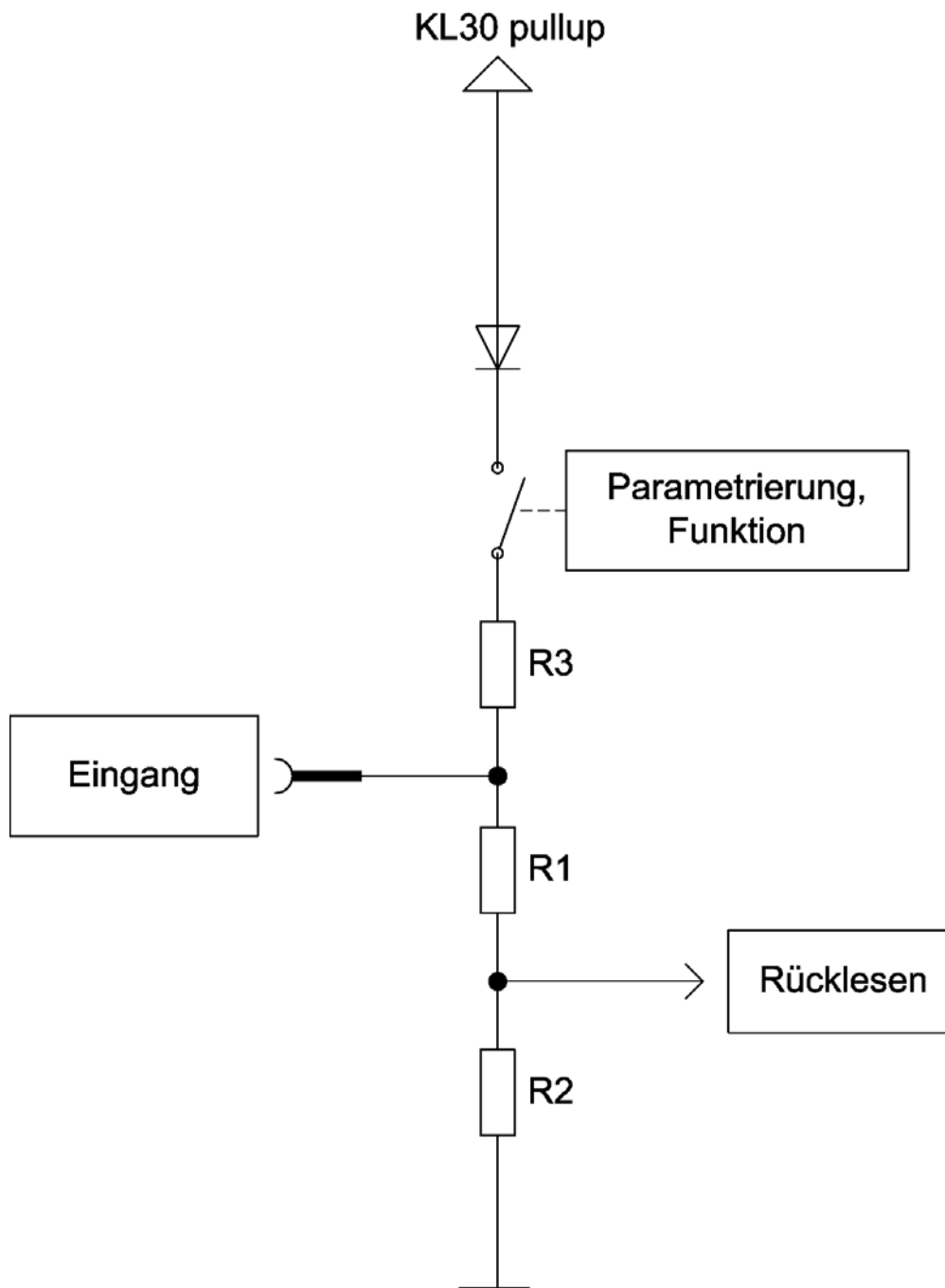
MFE_Digital

Circuit diagram and special features

Low active

Block	Description
MFE_09	Low active, pull-up resistor
MFE_10	Low active, pull-up resistor
MFE_11	Low active, pull-up resistor
MFE_12	Low active, pull-up resistor
MFE_13	Low active, pull-up resistor
MFE_14	Low active, pull-up resistor
MFE_15	Low active, pull-up resistor
MFE_16	Low active, pull-up resistor

Circuit diagram (low active)



Resistance values:

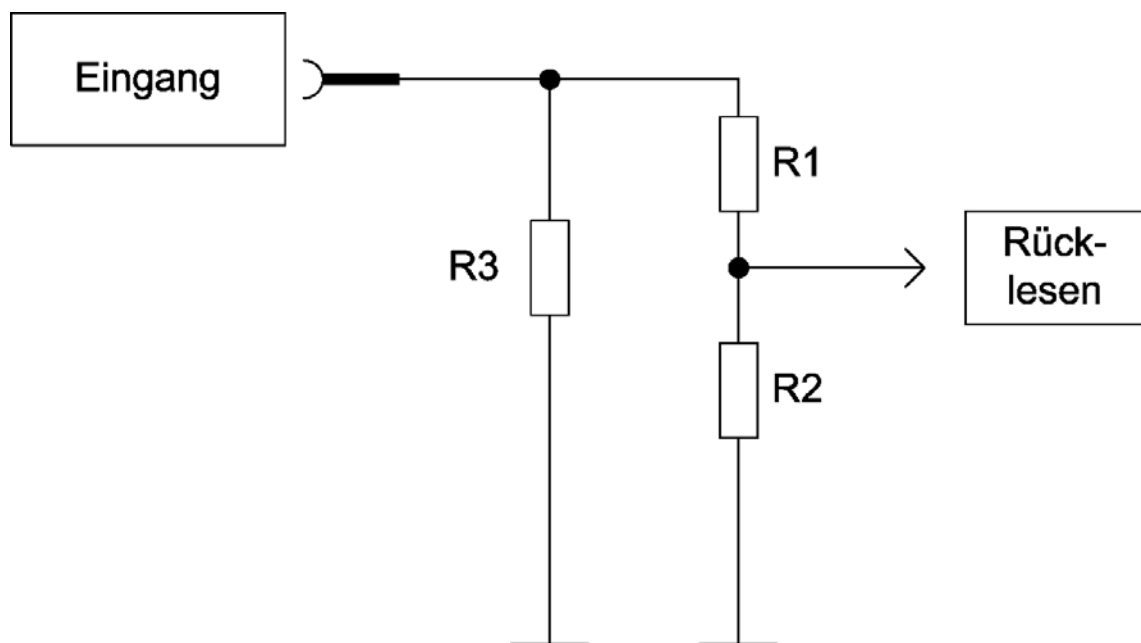
- R1 = 100 kOhm
- R2 = 60.4 kOhm
- R3 = 1.82 kOhm

The pull-up resistor (R3) is activated in the sleep mode of the control unit, if the input is configured as wakeup-capable. If the input is not configured as wakeup-capable, the pull-up resistor is deactivated in the sleep mode.

High active

Block	Description
MFE_17	High active, pull-down resistor
MFE_18	High active, pull-down resistor
MFE_19	High active, pull-down resistor
MFE_20	High active, pull-down resistor
MFE_21	High active, pull-down resistor
MFE_22	High active, pull-down resistor
MFE_23	High active, pull-down resistor
MFE_24	High active, pull-down resistor

Circuit diagram (high active)



Resistance values:

- R1 = 100 kOhm
- R2 = 60.4 kOhm
- R3 = 1.82 kOhm

The pull-down resistor (R3) is deactivated in the sleep mode of the control unit, irrespective of how the inputs are configured.

Electrical properties

Current flow convention: positive currents flow into the device, negative currents flow out. Unless otherwise defined, voltages are specified against local earth (term. 31) of the device.

Parameters	Conditions	Symbol	min	Type	max	Unit
Maximum input voltage	$U_{Bat} = 9 - 16V$, $T_{amb} = -40 - 85^{\circ}C$	U_{in}	0		U_{bat}	V
Input current	$U_{Bat} = U_{in} = 14V$, 25°C, Pull-up off	I_{in_14V}	70		150	μA
Input current	$U_{Bat} = 14V$, $U_{in} = 0V$, 25°C, Pull-up on	I_{in_pull}	-8.0		-6.0	mA
Maximum low-level	$U_{Bat} = 9 - 16V$, $T_{amb} = -40 - 85^{\circ}C$	U_L			2.8	V
Maximum high-level	$U_{Bat} = 9 - 16V$, $T_{amb} = -40 - 85^{\circ}C$	U_H	6.1			V
Hysteresis	$U_{Bat} = 9 - 16V$, $T_{amb} = -40 - 85^{\circ}C$	U_{hyst}	0.25			V

Inputs

Block	Width	Filter	Description
Wake	1 bit	no	Capable of waking up control unit (0) OFF (1) ON
Reset	1 bit	no	If the input is parametrised to the button (see parameter "Type"), an edge change (0) → (1) sets the output signal "MFE_xy" to (0). Static operation is not possible. With parameters set to "switch", the reset input has no function.

Parameters

Name	Description
Type	Selection: (0) Push-button The output signal status is toggled with each edge 0->1 on the input (MFE_xy or MFE_xy_inv). – A reset can be performed via the reset input. (1) Switch – MFE_xy outputs the input state.

Outputs

Name	Width	Description
MFE_xy (digital)	1 bit	Input status: (0) Not active (1) Active
MFA_xy_wakeup	1 bit	Wakeup status of control unit: (0) MFE is not reason for ECU being weak (1) MFE has woken up ECU
MFE_xy_inv	1 bit	Inverted input status

2.5.2 Multifunction outputs (MFA)

The KFG has a total of 24 discreet outputs. These MFAs can be parametrised individually using a configuration tool. It should be noted that not all MFAs have the same capability. The available outputs are listed as follows

- 4 half bridges with load current 5A, PWM-capable
- 2 high-side with load current 10A
- 6 high-side with load current 5A
- 2 low-side with load current 1A, PWM-capable
- 4 high-side with load current 0.5A, PWM-capable
- 4 low-side with load current 0.5A, PWM-capable
- 2 relay outputs with 1A each

All outputs can issue a soft PWM and they also feature a flash function.

Essentially, the outputs can be assigned to four categories. These categories are also the same in the configuration tool.

- MFA_POWER
- MFA_HALFBRIDGE
- MFA_PWM
- MFA_RELAIS

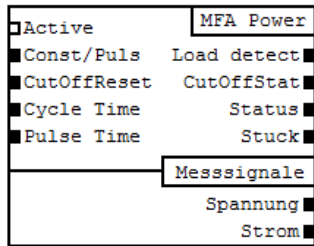


All outputs can hold their respective nominal load indefinitely. Even simultaneous use of all nominal currents at all outputs is permissible.



One of the eight "0.5A outputs" (4 high-side (MFA_09 – MFA_12) and 4 low-sides (MFA_15 – MFA_18)) may be loaded up to 0.6A provided the others have a maximum of 0.5A and the sum of the currents from these 8 outputs does not exceed 4A.

2.5.2.1 Function block type MFA_POWER



There are a total of 8 power outputs (MFA_POWER). These are designated MFA_01 – MFA_08 in the configuration tool.

The MFAs consist of:

- 2 high-side outputs with a load current of 10A
- 6 high-side outputs with a load current of 5A

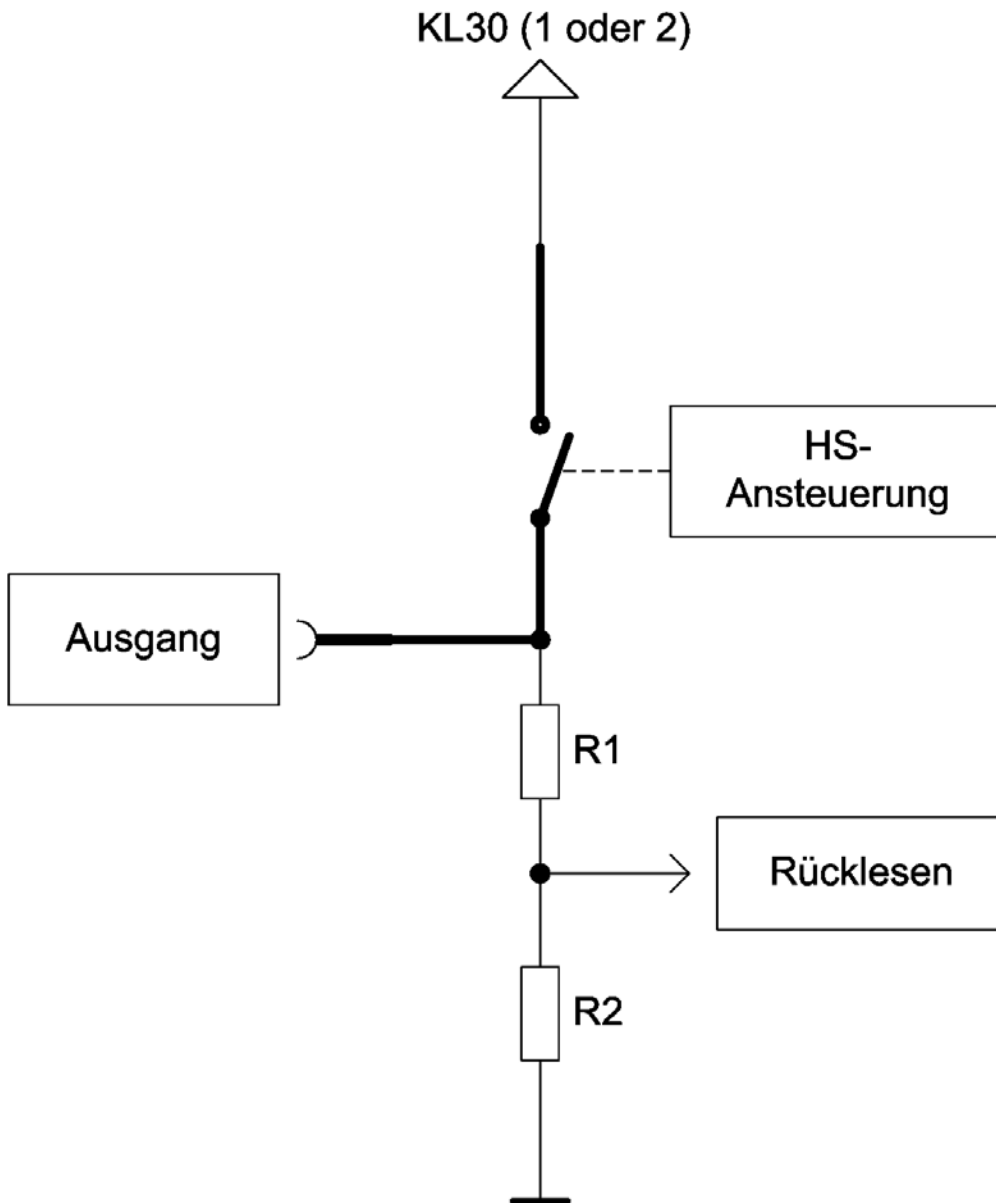
MFA_POWER

Circuit diagram and special features

Block	Nominal current	Description
MFA_01	5A	High-side switch with overload protection
MFA_02	10A	High-side switch with overload protection
MFA_03	5A	High-side switch with overload protection
MFA_04	10A	High-side switch with overload protection
MFA_05	5A	High-side switch with overload protection
MFA_06	5A	High-side switch with overload protection
MFA_07	5A	High-side switch with overload protection
MFA_08	5A	High-side switch with overload protection



Circuit diagram



Resistance values:

- R1 = 100 kOhm
- R2 = 18.7 kOhm

Properties

Properties	Description						
Soft PWM	Flash function up to 10Hz. Configurable via inputs "Const/Puls", "Cycle Time" and "Pulse Time"						
Current measurement	Measured current is stored in signal pool Measuring range of outputs: typ. 49 A						
Voltage measurement	Measured voltage is stored in signal pool						
Load detection	Parametrisable load detection. The limit can be configured via the parameter "Open load threshold". If the output is activated and a load current exceeds this threshold, a load is detected.						
Short circuit detection and overload detection	Short-circuit currents are limited to 65 to 140 A by a SMART FET. The shutdown upon short-circuits or overloads is controlled by evaluating the current and voltage in the hardware abstraction layer (HAL). Shutdown times <table border="0"> <tr> <td>Current</td> <td>Shutdown time</td> </tr> <tr> <td>>= 25 A</td> <td>< 40 ms</td> </tr> <tr> <td>>= 6.2 A</td> <td>< 20 ms</td> </tr> </table> <p>If the current exceeds the rated current (I_{rated}) (rms) by a factor of more than 1.2, the shutdown time upon overloads decreases as the current increases (HAL functionality). A characteristic curve (e.g. provided by fuses) cannot be guaranteed. If the SMART FET is thermally overloaded or overloaded by excessively high current, the output current can be restricted or the output cycles until final deactivation by the software. Note: For the current measurement, the local measured values of the KFG with the applicable tolerance for current measurements apply (see "Requirements on current measurement"). If PWM and/or the flash function are activated, the shutdown time and shutdown current may differ.</p>	Current	Shutdown time	>= 25 A	< 40 ms	>= 6.2 A	< 20 ms
Current	Shutdown time						
>= 25 A	< 40 ms						
>= 6.2 A	< 20 ms						
Safety	Parameters for "safety target" and "safe condition" can be set individually for each MFA						

Inputs

Block	Width	Filter	Description
Active	1 bit	yes	When "active", MFA is (0) Switched off (1) Switched on
Const./pulse	1 bit	no	MFA either (0) constant (1) Activate flashing (soft PWM)
CutOffReset	1 bit	no	Re-enabling of output after overload deactivation. Edge change (0) → (1) triggers release.
Cycle time	16 bit	No	Cycle time in ms (soft PWM)
Pulse time	16 bit	No	Pulse time in ms (soft PWM)

Parameters

Name	Description
Open load threshold	Threshold for open load detection in mA
Reactivation attempts	Parametrises how many reactivation attempts are carried out with overload or short circuit before MFA is switched off.
Safety target active	Which status of MFA is adopted if "safety target active" is activated and safe status is to be adopted: (0) MFA off (1) MFA on
Safety status	
Check "stuck at" error	Short-circuit monitoring in switching direction: (0) Not active (1) Active

Outputs

Name	Width	Description
MFA_xy_Load_detect	1 bit	Load detection: (0) No load detected (1) Load detected
MFA_xy_Overl_Cutoff_Stat	1 bit	Short circuit/overload status: (0) Not switched off (1) Switched off due to short circuit or overload
MFA_xy_Stat	1 bit	Start status: (0) Not active (1) Active
MFA_xy_Stuck_high	1 bit	Short-circuit monitoring in switching direction: (0) No short circuit (1) Short circuit detected in switch direction
U_MFA_xy	16 bit	Voltage at output, in 1mV/bit
I_MFA_xy	16 bit	Current at output, in 1mA/bit

2.5.2.2 Function block type MFA_HALFBRIDGE

MFA Halfbridge	
<input type="checkbox"/> Active	<input type="checkbox"/> Load detect
<input type="checkbox"/> Const/Puls	<input type="checkbox"/> CutOffStat
<input type="checkbox"/> CutOffReset	<input type="checkbox"/> Status
<input type="checkbox"/> Cycle Time	<input type="checkbox"/> Stuck High
<input type="checkbox"/> Pulse Time	<input type="checkbox"/> Stuck Low
<input type="checkbox"/> PWM Active	<input type="checkbox"/> Polarity
<input type="checkbox"/> PWM Period	
<input type="checkbox"/> PWM Duty	
<input type="checkbox"/> Polarity	
Messsignale	
	<input type="checkbox"/> Spannung
	<input type="checkbox"/> Strom High-Side
	<input type="checkbox"/> Strom Low-Side

There are a total of 4 half bridge power outputs (MFA_HALFBRIDGE). These are designated MFA_19 – MFA_22 in the configuration tool.

The MFAs consist of:

- 4 high-/low-side outputs with a load current of 5A
- 2 of the half bridges respectively can be switched together to make full bridges.

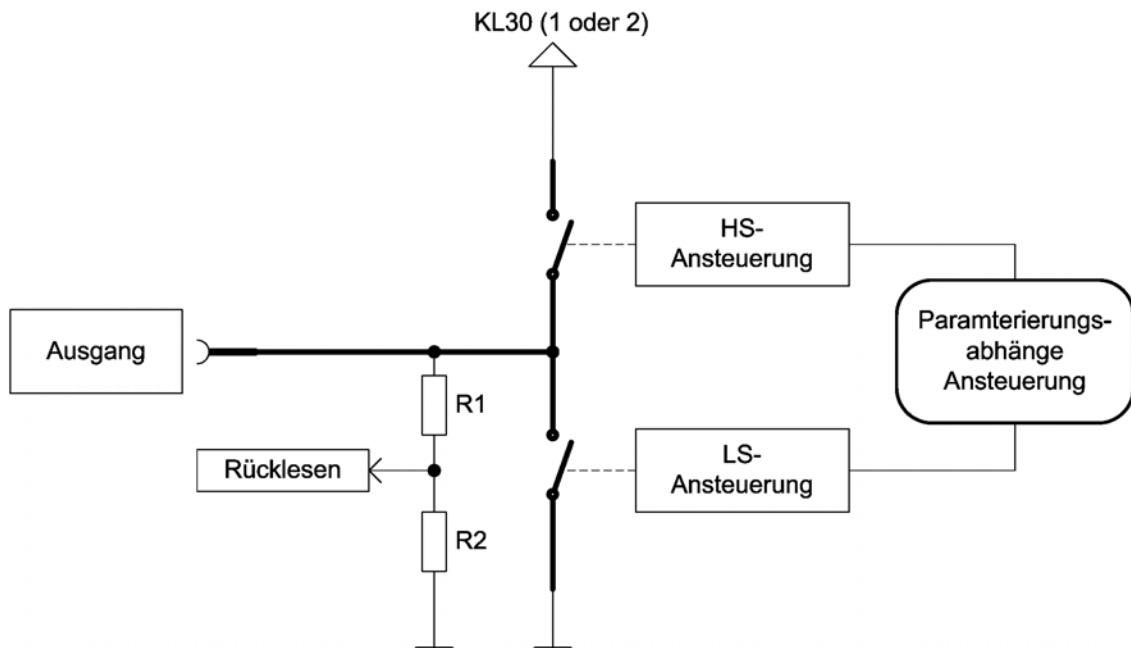
MFA_HALFBRIDGE

Circuit diagram and special features

Block	Nominal current	Description
MFA_19	5A	Configurable high-/low-side switch with overload protection Can be used individually as half bridge or together with MFA_20 as full bridge.
MFA_20	5A	Configurable high-/low-side switch with overload protection Can be used individually as half bridge or together with MFA_19 as full bridge.
MFA_21	5A	Configurable high-/low-side switch with overload protection Can be used individually as half bridge or together with MFA_22 as full bridge.
MFA_22	5A	Configurable high-/low-side switch with overload protection Can be used individually as half bridge or together with MFA_21 as full bridge.



Circuit diagram



Resistance values at output, high-side:

- R1 = 200 kOhm
- R2 = 47.5 kOhm

Resistance values at output, low-side, when activated:

Additional 127 kOhm.

Properties

Property	Description												
Soft PWM	Flash function up to 10Hz. Configurable via inputs "Const/Puls", "Cycle Time" and "Pulse Time"												
Configurable PWM	to 4 - 300Hz Frequency for all outputs can be adjusted together, pulse duty can be selected individually												
Current measurement	Measured current is stored in signal pool Measuring range: typ. 28 A												
Voltage measurement	Measured voltage is stored in signal pool												
Load detection	Parametrisable open load detection. The limit can be configured via the parameter "Open load threshold". If the output is activated and a load current exceeds this threshold, a load is detected.												
Short circuit detection and overload detection	A short-circuit and overload detection is provided for the outputs. Short-circuit currents are limited to 30 to 55 A by the SMART FET. The shutdown upon short circuits or overloads is controlled by evaluating the current and voltage in the hardware abstraction layer (HAL). Shutdown times, high-side, active: <table border="0"> <tr> <td>Current</td> <td>Shutdown time</td> </tr> <tr> <td>> 25 A</td> <td>< 1.4 s</td> </tr> <tr> <td>> 10 A</td> <td>< 360 s</td> </tr> </table> Shutdown times, low-side, active: <table border="0"> <tr> <td>Current</td> <td>Shutdown time</td> </tr> <tr> <td>> 25 A</td> <td>< 120 ms</td> </tr> <tr> <td>> 6.2 A</td> <td>< 60 s</td> </tr> </table> If the current exceeds the rated current (I_{rated}) (rms) by a factor of more than 2 or 1.2, the shutdown time upon overloads decreases as the current increases (HAL functionality). A characteristic curve (e.g. provided by fuses) cannot be guaranteed. If the SMART FET is thermally overloaded or overloaded by excessively high current, the output current can be restricted or the output cycles until final deactivation by the software.	Current	Shutdown time	> 25 A	< 1.4 s	> 10 A	< 360 s	Current	Shutdown time	> 25 A	< 120 ms	> 6.2 A	< 60 s
Current	Shutdown time												
> 25 A	< 1.4 s												
> 10 A	< 360 s												
Current	Shutdown time												
> 25 A	< 120 ms												
> 6.2 A	< 60 s												
Safety	Parameters for "safety target" and "safe condition" can be set individually for each MFA												



Inputs

Block	Width	Filter	Description
Active	1 bit	yes	When "active", MFA is (0) Switched off (1) Switched on
Const./pulse	1 bit	no	MFA either (0) Constant (1) flashing Actuate (soft PWM)
CutOffReset	1 bit	no	Re-enabling of output after overload deactivation. Edge change (0) → (1) triggers reset.
Cycle time	16 bit	No	Cycle time in ms (soft PWM)
Pulse time	16 bit	No	Pulse time in ms (soft PWM)
PWM active	1 bit	No	Configurable PWM (0) Deactivated (1) Activated
PWM period	16 bit	No	If PWM type = frequency → frequency in 0.1 Hz If PWM type = time → cycle time in μs
PWM duty	16 bit	No	If PWM type = frequency → duty cycle in 0.1% If PWM type = time → cycle time in μs
Polarity	1 bit	No	Switches the MFA as: (0) Low-side (1) High-side

Parameters

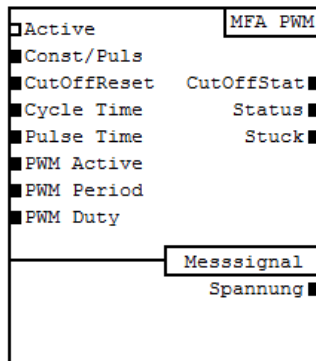
Name	Description
PWM type	Selects the PWM type: (0) Frequency (1) Time Selection influences inputs "PWM period" and "PWM duty"
Open load threshold	Threshold for open load detection in mA
Reactivation attempts	Parametrises how many reactivation attempts are carried out with overload or short circuit before MFA is switched off.
Safety target active	Which status of MFA is adopted if "safety target active" is activated and safe status is to be adopted: (0) MFA off (1) MFA on
Safety status	
Check "stuck at" error	Short-circuit monitoring in switching direction: (0) Not active (1) Active
Full bridge	(0) No (1) Yes with associated half bridge output (19 + 20 OR 21 + 22)
Braking time	In 10ms

Outputs

Name	Width	Description
MFA_xy_Load_detect	1 bit	Load detection: (0) No load detected (1) Load detected
MFA_xy_Overl_Cutoff_Stat	1 bit	Short circuit/overload status: (0) Not switched off (1) Switched off due to short circuit or overload
MFA_xy_Stat	1 bit	Start status: (0) Not active (1) Active
MFA_xy_Stuck_High	1 bit	Short-circuit monitoring in switching direction, if switching direction is towards high-side: (0) No short circuit (1) Short circuit detected in switch direction
MFA_xy_Stuck_Low	1 bit	Short-circuit monitoring in switching direction, if switching direction is towards low-side: (0) No short circuit (1) Short circuit detected in switch direction
MFA_xy_Polarity	1 bit	MFA is: (0) Low-side (1) High-side
U_MFA_xy	16 bit	Voltage at output, in 1mV/bit
I_MFA_xy_HS	16 bit	Current at output (high-side), in 1mA/bit
I_MFA_xy_LS	16 bit	Current at output (low-side), in 1mA/bit



2.5.2.3 Function block type MFA_PWM



MFA_PWM

Circuit diagram and special features

High active

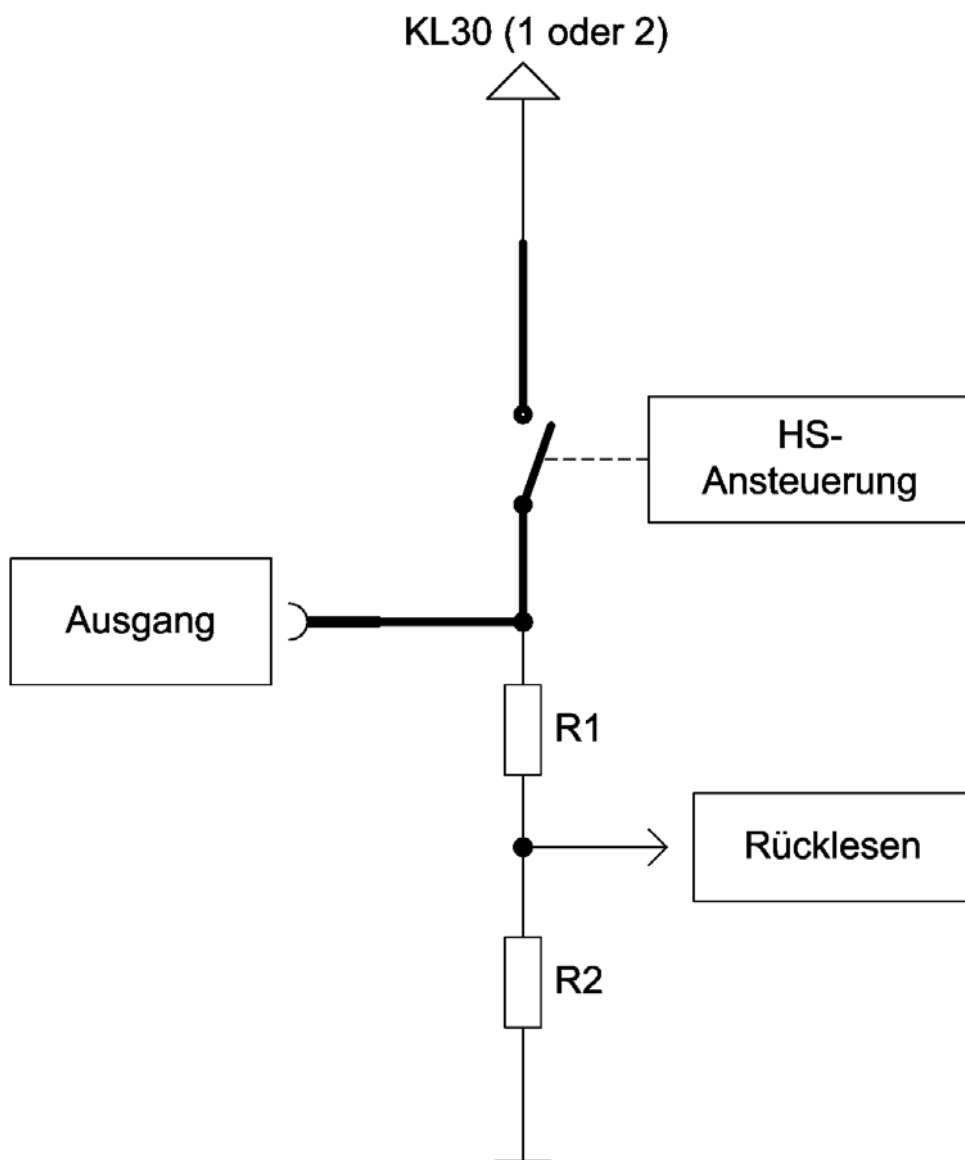
There are a total of 10 signal outputs (MFA_PWM). These are designated MFA_09 – MFA_10 in the configuration tool.

The MFAs consist of:

- 4 high-side outputs with a load current of 0.5A
- 2 low-side outputs with a load current of 1A
- 4 low-side outputs with a load current of 0.5A

Block	Nominal current	Description
MFA_09	0.5A	High-side switch
MFA_10	0.5A	High-side switch
MFA_11	0.5A	High-side switch
MFA_12	0.5A	High-side switch

Circuit diagram (high active)



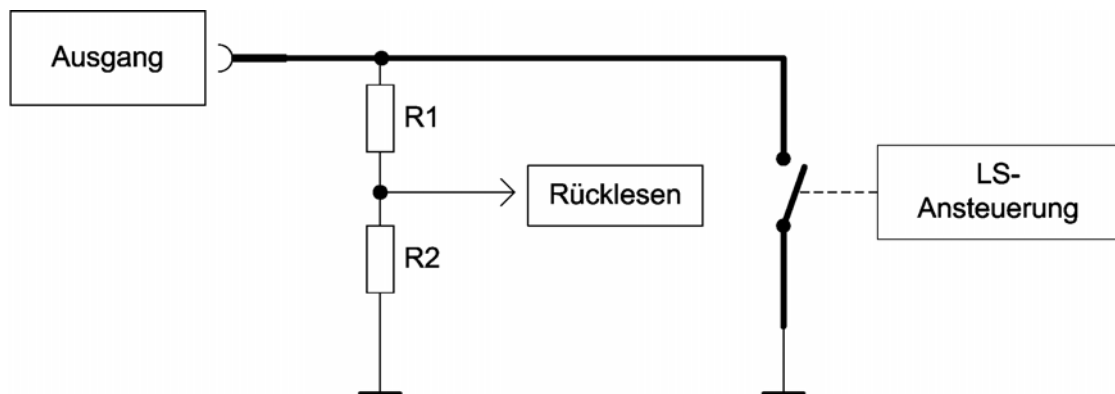
Resistance values:

- R1 = 3.65 kOhm
- R2 = 681 Ohm

Low active

Block	Nominal current	Description
MFA_13	1A	Low-side switch
MFA_14	1A	Low-side switch
MFA_15	0.5A	Low-side switch
MFA_16	0.5A	Low-side switch
MFA_17	0.5A	Low-side switch
MFA_18	0.5A	Low-side switch

Circuit diagram (low active)



Resistance values:

- R1 = 200 kOhm
- R2 = 47.5 kOhm

Properties

Property	Description
Soft PWM	Flash function up to 10Hz. Configurable via inputs "Const/Puls", "Cycle Time" and "Pulse Time"
Configurable PWM	For MFA_13 and MFA_14 (low-side 1A): to 4 - 5kHz Frequency for all outputs can be adjusted together, pulse duty can be selected individually Otherwise: to 4 - 300Hz Frequency for all outputs can be adjusted together, pulse duty can be selected individually
Voltage measurement	Measured voltage is stored in signal pool
Short circuit detection and overload detection	The shutdown upon short circuits or overloads is controlled by evaluating the output in the hardware abstraction layer (HAL). The overload and short-circuit detection is not active until the SMART FET has limited the current or initiated a thermal shutdown. If the SMART FET is thermally overloaded or overloaded by excessively high current, the output current can be restricted or the output cycles until final deactivation by the software.
Safety	Parameters for "safety target" and "safe condition" can be set individually for each MFA

Inputs

Block	Width	Filter	Description
Active	1 bit	yes	When "active", MFA is (0) switched off (1) switched on
Const./pulse	1 bit	no	MFA either (0) constant (1) flashing Actuate (soft PWM)
CutOffReset	1 bit	no	Re-enabling of output after overload deactivation. Edge change (0) → (1) triggers reset.
Cycle time	16 bit	No	Cycle time in ms (soft PWM)
Pulse time	16 bit	No	Pulse time in ms (soft PWM)
PWM active	1 bit	No	Activate or deactivate PWM
PWM period	16 bit	No	If PWM type = frequency → frequency in 0.1 Hz If PWM type = time → cycle time in μs
PWM duty	16 bit	No	If PWM type = frequency → duty cycle in 0.1% If PWM type = time → cycle time in μs



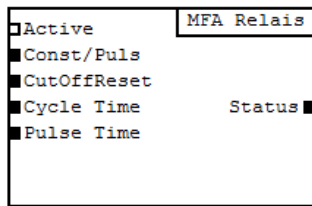
Parameters

Name	Description
PWM type	Selects the PWM type: (0) Frequency (1) Time Selection influences inputs "PWM period" and "PWM duty"
Reactivation attempts	Parametrises how many reactivation attempts are carried out with overload or short circuit before MFA is switched off.
Safety target active	Which status of MFA is adopted if "safety target active" is activated and safe status is to be adopted: (0) MFA off (1) MFA on
Safety status	
Check "stuck at" error	Short-circuit monitoring in switching direction: (0) Not active (1) Active

Outputs

Name	Width	Description
MFA_xy_Overl_Cutoff_Stat	1 bit	Short circuit/overload status: (0) Not switched off (1) Switched off due to short circuit or overload
MFA_xy_Stat	1 bit	Start status: (0) Not active (1) Active
MFA_xy_Stuck_High/Low	1 bit	Short circuit monitoring if high- or low-side: (0) No short circuit (1) Short circuit detected in switch direction
U_MFA_xy	16 bit	Voltage at output, in 1mV/bit

2.5.2.4 Function block type MFA_RELAIS



MFA_RELAIS

There are a total of 2 relay outputs (MFA_RELAIS). These are designated MFA_23 and MFA_24 in the configuration tool.

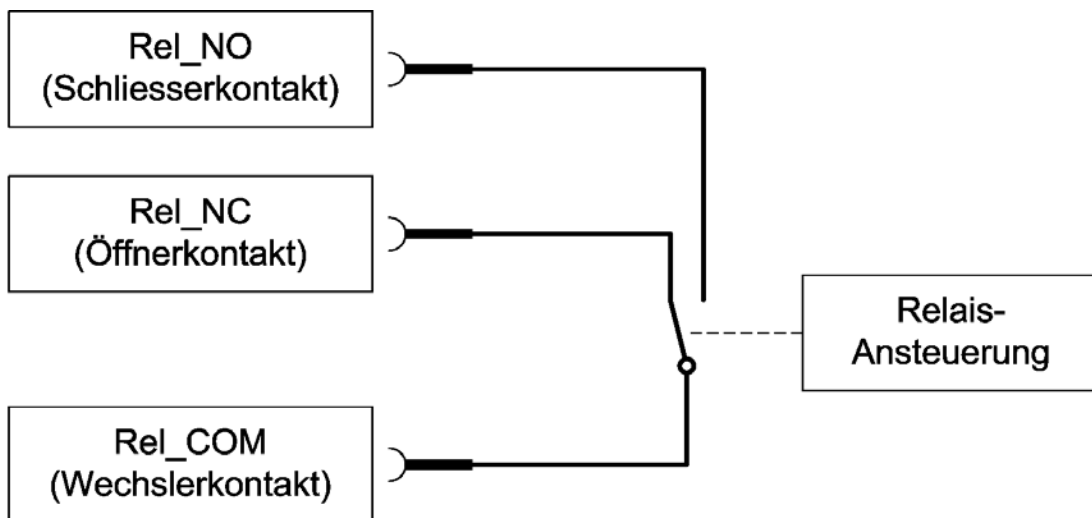
The MFAs consist of:

- 2 outputs with a load current of 1A

Circuit diagram and special features

Block	Nominal current	Description
MFA_23	1A	1 changer, potential free changeover possible Contacts at ST3 (REL_NO1, REL_COM1, REL_NC1)
MFA_24	1A	1 changer, potential free changeover possible Contacts at ST3 (REL_NO2, REL_COM2, REL_NC2)

Circuit diagram



Properties

Properties	Description
Soft PWM	Flash function up to 10Hz. Configurable via inputs "Const/Puls", "Cycle Time" and "Pulse Time"

Maximum switching current:

The maximum load for each path is 1 A.

Minimum switching current:

In order to comply with the service life specification the typical switching current must be at least 10 mA.

No diagnosis, no overload/short-circuit protection:

These outputs do NOT provide any diagnosis functions, voltage feedback measurement or current measurement. In addition, they do not provide any overload protection. Any means of protection must be implemented in the external circuit which is closed via the relay contacts. The specifications must not be exceeded.

If there are capacitive and/or inductive loads, respective means of protection must be provided.

Dynamic parameters

Parameters	Condition	Symbol	min	Type	max	Unit
Maximum switching frequency	$U_{Bat} = 9...16\text{ V}$, $T_{amb} = -40...85^{\circ}\text{C}$	f_{max}			2	Hz
Minimum switch-on time	$U_{Bat} = 9...16\text{ V}$, $T_{amb} = -40...85^{\circ}\text{C}$	T_{on_min}	0.25			s
Minimum shutdown time	$U_{Bat} = 9...16\text{ V}$, $T_{amb} = -40...85^{\circ}\text{C}$	T_{off_min}	0.25			s
Electrical service life	$U_{Bat} = 9...16\text{ V}$, $T_{amb} = -40...85^{\circ}\text{C}$ $I_{max} = 1\text{ A}$		400,000			Gear changes

Inputs

Block	Width	Filter	Description
Active	1 bit	Yes	When "active", MFA is (0) Switched off (1) Switched on
Const./pulse	1 bit	No	MFA either (0) constant (1) flashing Activated (soft PWM)
CutOffReset	1 bit	No	Input is available, has no function at MFA_23 and MFA_24
Cycle time	16 bit	No	Cycle time in ms (soft PWM)
Pulse time	16 bit	No	Pulse time in ms (soft PWM)



Outputs

Name	Width	Description
MFA_xy_Stat	1 bit	Start status: Not active Active

2.6 Application functions

2.6.1 Configurable logic module

The configurable logic modules are sub-divided into 12 groups. Each group has the same layout and comprises the individual logic blocks. By distributing the logic blocks into groups, it is possible through clever circuitry to achieve minimum cycle times across several groups even with complex sequential circuits. This is made possible through series processing of the groups while the results of the previous group are available to the following groups again without delay.



The cycle time of the PLC is 20 ms.

A group consists of the following logic blocks:

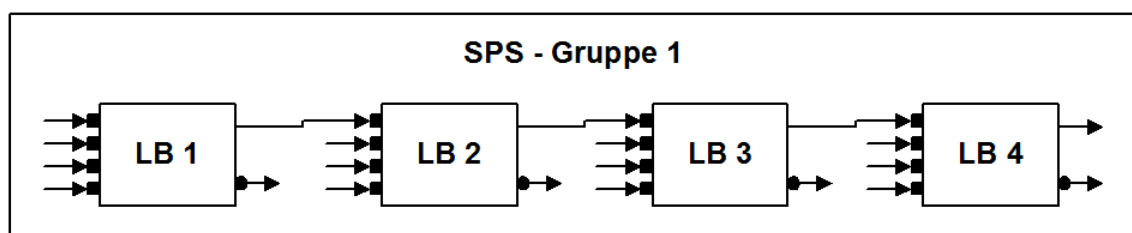
- 8 logic blocks (AND/OR/EXOR etc.)
- 2 flip-flops (D / RS)
- 1 hysteresis block (Schmitt trigger)
- 2 threshold switches
- 1 timer (monoflop)
- 1 counter (up, down)
- 1 value table with 4 entries
- 5 constants

Processing of the logic blocks is in groups: first group 1, then group 2, etc. up to group 12. Within the group is the following sequence:

- 1 Threshold value switch
- 2 Hysteresis blocks
- 3 Flip-flops
- 4 Timer
- 5 Counter
- 6 Logic blocks
- 7 Value tables

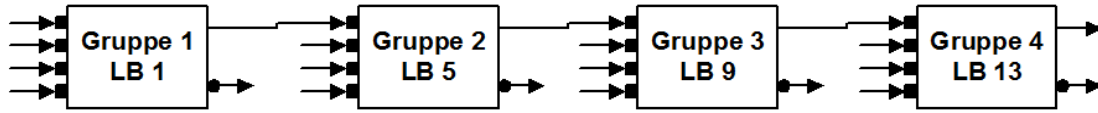
The results of the individual blocks are written back to the signal pool straightaway and are directly available for further processing in the following blocks. All other input signals such as e.g. CAN signals, status of inputs/outputs, ... are not changed while the SPS is being processed to prevent possible problems with consistency.

If a function created with the PLC blocks consists e.g. of 4 successively switched logic blocks, where all logic blocks belong to the same group, the cycle time of the input signal is up to 4 PLC cycles, due to this function, until a current signal is available at the output of the last logic block. The figure below shows unfavourable circuitry for the cycle time.

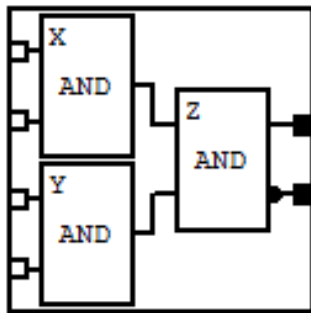


If the same function is switched together from a logic block of 4 groups, the cycle time can be reduced to one SPS cycle. It is therefore important for the order in which the SPS function blocks are worked

through and the signal paths and their dependencies in the sequential circuit to be as closely aligned as possible. In this way, SPS functions can be realised with the shortest cycle times. Circuit connection of the blocks is then realised as follows:



2.6.1.1 Function block type SPS_LOGIC_BLOCK



Integrated in one logic block are three 2-fold logic elements as per the diagram. These 2-fold logic elements can perform various logic functions independently of each other.

This makes it possible to realise logic operations effectively, which for the most part consist of 2-fold logic elements, without having to use a complete logic block in each case. At the same time, however, there is still the option of making 3-fold and 4-fold logic elements possible with just one logic block.

This approach saves resources and shortens the cycle time of the SPS system as the number of successive blocks can be reduced.

SPS_LOGIC_BLOCK

Inputs

Block	Width	Filter	Description
Input 1	1 bit	yes	Digital input with upstream input filter
Input 2	1 bit	yes	Digital input with upstream input filter
Input 3	1 bit	yes	Digital input with upstream input filter
Input 4	1 bit	yes	Digital input with upstream input filter



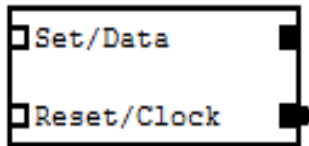
Parameters

Name	Description
Sub-block X	Type of operation from logic element x (0) And (1) Or (2) Exor (3) Nand (4) Nor (5) Exnor
Sub-block Y	Type of operation from logic element y (0) And (1) Or (2) Exor (3) Nand (4) Nor (5) Exnor
Sub-block Z	Type of operation from logic element z (0) And (1) Or (2) Exor

Outputs

Name	Width	Description
SPS_Gx_LBy	1 bit	The result of the logic operation
SPS_Gx_LBy_inv	1 bit	The inverted result of the logic operation

2.6.1.2 Function block type SPS_FLIPFLOP



SPS_FLIPFLOP

The flip-flop can be configured either as D-Flip-Flop with data input and cycle input or as RS-Flip-Flop with set input and reset input.

If configured as D-Flip-Flop, it is possible for 16-bit signals to be stored at the data input and to be issued at "SPS_Gx_FFy".

Inputs

Block	Width	Filter	Description
Set/data	16 bit	yes	Analogue input with upstream input filter
Reset/clock	1 bit	yes	Digital input with upstream input filter

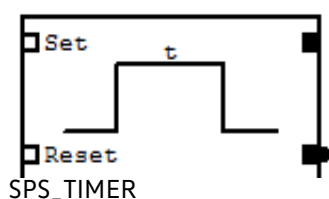
Parameters

Name	Description
Type	Configures FlipFlop type (0) RS-FF (1) D-FF
Evaluation	Configures the evaluation (0) If type is configured as RS-FF: (1) Status-controlled (2) Edge-controlled (3) If type is configured as D-FF: (4) Positive edge evaluation (5) Negative edge evaluation

Outputs

Name	Width	Description
SPS_Gx_FFy	16 bit	If type is configured as D-FF: <ul style="list-style-type: none"> ■ Output signal equates to set/data
SPS_Gx_FFy_inv	1 bit	Inversion from "SPS_Gx_FFy"

2.6.1.3 Function block type SPS_TIMER



The timer block is active for a set period when triggered. The setting can be used to determine whether the timer block is retriggerable and whether the input signal is evaluated with edge control or status control.

Inputs

Block	Width	Filter	Description
Set	1 bit	yes	Triggers the timer
Reset	1 bit	yes	Is always status controlled and resets the timer

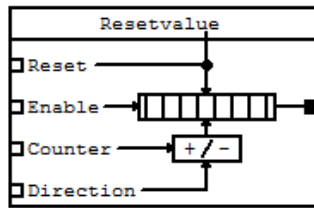
Parameters

Name	Description
Evaluation	Configures the evaluation (0) Status triggered (1) Edge triggered
Edge evaluation	Configuration of edge evaluation (0) Positive edge (1) Negative edge Only applicable if evaluation is edge controlled
Retriggerable	Time reset after triggering repeats (0) No (1) Yes
Time	The time that the timer should run down from (observe time basis)
Time basis	Configures the time basis (0) 100ms (1) 1s (2) 10s (3) 1min (4) 10min

Outputs

Name	Width	Description
SPS_Gx_TBy	1 bit	After triggering, the timer is active for the set period
SPS_Gx_TBy_inv	1 bit	Inversion from "SPS_Gx_TBy"

2.6.1.4 Function block type SPS_COUNTER



SPS_COUNTER

The counter block is an element used to count edge changes. The internal counter register is 16 bit wide. If the counter status rises to 0xFFFF, an overflow to 0x0000 takes place with the next edge at the "counter" input.

Only positive edges are evaluated.

Inputs

Block	Width	Filter	Description
Reset	1 bit	yes	Edge change (0) → (1) sets the counter to a parametrised reset value
Enable	1 bit	yes	Enables or releases evaluation of the "counter" input
Counter	1 bit	yes	Positive edge of input increases/reduces counter status
Direction	1 bit	yes	Determines the counter direction (0) Up (1) Down

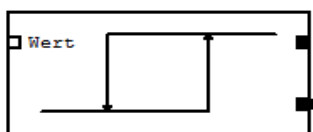
Parameters

Name	Description
Reset value	After a reset, the counter has the value defined here

Outputs

Name	Width	Description
SPS_Gx_CNT	16 bit	Includes the current counter status

2.6.1.5 Function block type SPS_HYSTERESE



The hysteresis block can be used, for example, to convert an analogue input signal with a Schmitt trigger to a digital signal.

SPS_HYSTERESE

Inputs

Block	Width	Filter	Description
Value	16 bit	yes	Analogue input with upstream input filter

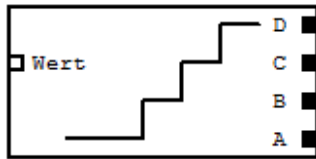
Parameters

Name	Description
Top	Top switching threshold
Bottom	Bottom switching threshold

Outputs

Name	Width	Description
SPS_Gx_HB	1 bit	Set or not set depending on the switching threshold
SPS_Gx_HB_inv	1 bit	Inversion from "SPS_Gx_HB"

2.6.1.6 Function block type SPS_THRESHOLD



SPS_THRESHOLD

The threshold switch compares the input value with up to four thresholds and sets the associated output.

For the block to function correctly, ensure that the parametrised values for the thresholds meet the following condition:

threshold A < threshold B < threshold C < threshold D

If this regulation for parametrisation of the thresholds is not observed, the response of the threshold switch will not be defined.

Inputs

Block	Width	Filter	Description
Value	16 bit	yes	Analogue input with upstream input filter

Parameters

Name	Description
SS type (operating mode)	Configures the operating mode (0) Bar graph (1) Running point
Threshold A	Threshold for output A
Threshold B	Threshold for output B
Threshold C	Threshold for output C
Threshold D	Threshold for output D

Outputs

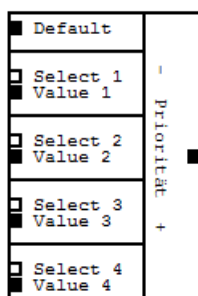
Operating mode bar graph

	A	B	C	D
Input <= threshold A	0	0	0	0
Threshold A < input <= threshold B	1	0	0	0
Threshold B < input < threshold C	1	1	0	0
Threshold C < input <= threshold D	1	1	1	0
Threshold D < input	1	1	1	1

Operating mode running point

	A	B	C	D
Input <= threshold A	0	0	0	0
Threshold A < input <= threshold B	1	0	0	0
Threshold B < input < threshold C	0	1	0	0
Threshold C < input <= threshold D	0	0	1	0
Threshold D < input	0	0	0	1

2.6.1.7 Function block type SPS_VALUETABLE



SPS_VALUETABLE

If an input condition is "true", the value of the allocated signal ID is applied to the output. The value of the input is not placed at the output. If several input values are valid at the same time, the value benefits in the value table with the highest position number. (Priority from position 1 to 4 upwards).

Inputs

Block	Width	Filter	Description
Default	16 bit	No	If no "select" is not chosen → Output = default
Select 1	1 bit	Yes	Selects signal ID at "value 1"
Value 1	16 bit	No	If selected through "select 1" → Output = value 1
Select 2	1 bit	Yes	Selects signal ID at "value 2"
Value 2	16 bit	No	If selected through "select 2" → Output = value 2
Select 3	1 bit	Yes	Selects signal ID at "value 3"
Value 3	16 bit	No	If selected through "select 3" → Output = value 3
Select 4	1 bit	Yes	Selects signal ID at "value 4"
Value 4	16 bit	No	If selected through "select 4" → Output = value 4

Outputs

Name	Width	Description
SPS_Gx_WTy	16 bit	The signal ID of the selected input

2.6.1.8 Function block type SPS_CONST



SPS_KONSTANTE

The constant can be used via the signal ID like a signal pool signal.

Inputs

Name	Description
Constant 1	Unsigned 16 bit value
Constant 2	Unsigned 16 bit value
Constant 3	Unsigned 16 bit value
Constant 4	Unsigned 16 bit value
Constant 5	Unsigned 16 bit value

Outputs

Name	Width	Description
SPS_Gx_K1	16 bit	Signal has parametrised constant 1
SPS_Gx_K2	16 bit	Signal has parametrised constant 2
SPS_Gx_K3	16 bit	Signal has parametrised constant 3
SPS_Gx_K4	16 bit	Signal has parametrised constant 4
SPS_Gx_K5	16 bit	Signal has parametrised constant 5

2.6.2 Configurable arithmetic modules (AU)

The module makes general arithmetic functions available. The cycle time is 20ms. All AU blocks are processed in one cycle.

The arithmetic unit comprises the following blocks:

- 20 calculation blocks
- 8 filter/ramp function blocks
- 8 differentiators/integrators
- 8 comparators
- 4 characteristic curves
- 16 NVM accumulators
- 20 constants

Working through the individual AU blocks is sub-divided into 4 calculation stages. Each AU block can be assigned to one of these calculation stages. The results of the block are written to the signal pool straightaway and are available to the following blocks for further processing. In this way, minimum cycle times can be achieved even where operations are more complex. All other input signals such as e.g. CAN signals, status of inputs/outputs, ... are not changed while the SPS is being processed to prevent possible problems with consistency.

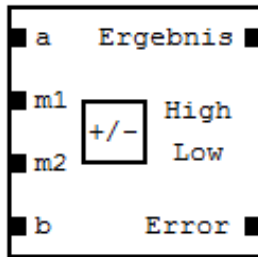
Essentially, the following sequence is adopted for processing:

- Calculation blocks
- Filter blocks
- Differentiators/integrators
- Comparators
- Characteristic curve
- Accumulators

As previously mentioned, processing of the individual blocks can be distributed over the 4 individual calculation stages of the AU in order to maintain the maximum calculation speed during an AU cycle. The internal calculation resolution is 32 bit signed. All results of the AU blocks also have this resolution.

Parametrisation of the blocks is analogous to the SPS blocks via the configuration tool.

2.6.2.1 Function block type AU_CALC



AU_CALC

The calculation totals four operands.

Function:

$$e b n i s = \frac{(a \times m_1)}{m_2} \pm b$$

The result is then checked for violation of (fallen below / gone over) the upper and lower thresholds.

Inputs

Block	Width	Filter	Descriptions
a	16 bit	No	Operand a (see function)
m1	16 bit	No	Operand m1 (see function)
m2	16 bit	No	Operand m2 (see function)
b	16 bit	No	Operand b (see function)

Parameters

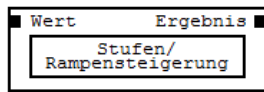
Name	Description
Calculation step allocation	Allocates processing to one of the 4 calculation steps
Type	Configures the operator Addition Subtraction
Upper threshold	If "result" > upper threshold: → "result" = upper threshold:
Lower threshold	If "result" < lower threshold → "result" = lower threshold

Outputs

Name	Width	Description
AU_Calc_xy	32 bit	Includes the "result" of the function
AU_Calc_xy_Err	1 bit	Is set if division by 0 "Result" is then invalid and is set to 0



2.6.2.2 Function block type AU_FILTER



AU_FILTER

If "mean value" is configured:
The "result" is the gradual mean value across "n-stage" support points.
Function:

$$\frac{(n - 1) \times MW_{(t-1)} + Wert_{(t)}}{n}$$

Current mean value "result" at time t

Previous mean value "result" at time t-1

Input value at time t

Number of supporting points

Time distance between support points equates to PLC
cycle time (20 ms)

If "ramp" is configured:

The result is a ramp function with the set ramp increase. The time basis is
the SPS cycle time (20ms)

If

$$\text{Ergebnis}_{(t-1)} + \text{Rampensteigung}$$

Then:

$$\text{Ergebnis}_{(t-1)} + \text{Rampensteigung}$$

If

$$\text{Ergebnis}_{(t-1)} - \text{Rampensteigung}$$

Then:

$$\text{Ergebnis}_{(t-1)} - \text{Rampensteigung}$$

Otherwise the following applies:

$$gebnis_{(t)} = Wert_{(t)}$$

Inputs

Block	Width	Filter	Description
Value	16 bit	No	Is occupied with the value to be processed

Parameters

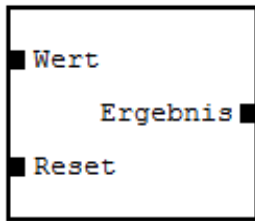
Name	Description
Calculation step allocation	Allocates processing to one of the 4 calculation steps
Type	Configures the type (0) Mean value (1) Ramp
Filter stages/ramp increase	If type "mean value" → Support points of mean value formation If type "ramp" → Ramp increase

Outputs

Name	Width	Description
AU_Filter_x	32 bit	Includes the "result" of the function



2.6.2.3 Function block type AU_DIFF_INT



AU_DIFFERENZIERER
INTEGRIERER

Scaling of the input value takes place first:

$$\frac{(Wert \times SkalierungsfaktorF1)}{SkalierungsfaktorF2}$$

Function:
As differentiator:

$$Wert_{skaliert(t)} - Wert_{skaliert(t-1)}$$

As integrator:

$$Wert_{skaliert(t)} + Wert_{skaliert(t-1)}$$

The following applies here:

$$(t - 1) = Zeitbasis$$

Time basis = AU_DiffInt_1_Timebase * PLC cycles (20 ms)

At the end, the result is checked for violation of (fallen below/exceeding) the upper and lower thresholds.

Inputs

Block	Width	Filter	Description
Value	16 bit	No	Is the value to be integrated or differentiated
Reset	16 bit	No	If "reset" ≥ 1 , result is 0

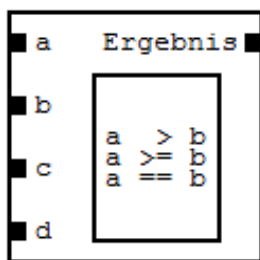
Parameters

Name	Description
Calculation step allocation	Allocates processing to one of the 4 calculation steps
Type	Configures the type (0) Integrator (1) Differentiator
Scaling factor F1	Scaling factor for scaling that takes place before the operation. (see function)
Scaling factor F2	Scaling factor for scaling that takes place before the operation. (see function)
Upper threshold	If "result" > upper threshold: → "result" = upper threshold
Lower threshold	If "result" < lower threshold → "result" = lower threshold

Outputs

Name	Width	Description
AU_Diff_Int_x	32 bit	Includes the "result" of the function

2.6.2.4 Function block type AU_COMPARE



Comparator compares two inputs and gives input "c" or input "d" as "result".

AU_KOMPARATOR

Inputs

Block	Width	Filter	Description
a	16 bit	No	First value for comparison
b	16 bit	No	Second value with which value is compared
c	16 bit	No	Signal for successful comparison
d	16 bit	No	Signal for unsuccessful comparison

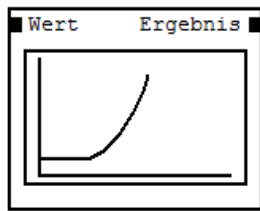
Parameters

Name	Description
Calculation step allocation	Allocates processing to one of the 4 calculation steps
Type	Configures the type (0) $a > b$ (1) $a \geq b$ (2) $a == b$

Outputs

Name	Width	Description
AU_Compare_x	32 bit	Includes value of signal from → input "c" if comparison "true" → input "d" if comparison "false"

2.6.2.5 Function block type AU_INTERPOLATION



AU_KENNLINIE

A characteristic curve is defined with 8 x/y pairs and is used to convert the input value.

Function:

$$\text{Ergebnis} = f(\text{Wert})$$

If "value" \leq x1 \rightarrow "result" = y1

If "value" \leq x8 \rightarrow "result" = y8

Otherwise, "result" is the linear interpolation between the two next value pairs

Inputs

Block	Width	Filter	Description
Value	16 bit	No	Is the value to be converted

Parameters

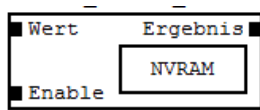
Name	Description
Calculation step allocation	Allocates processing to one of the 4 calculation steps
Value x1	x-coordinate of value pair 1
Value x2	x-coordinate of value pair 2
Value x3	x-coordinate of value pair 3
Value x4	x-coordinate of value pair 4
Value x5	x-coordinate of value pair 5
Value x6	x-coordinate of value pair 6
Value x7	x-coordinate of value pair 7
Value x8	x-coordinate of value pair 8
Value y1	y-coordinate of value pair 1
Value y2	y-coordinate of value pair 2
Value y3	y-coordinate of value pair 3
Value y4	y-coordinate of value pair 4
Value y5	y-coordinate of value pair 5
Value y6	y-coordinate of value pair 6
Value y7	y-coordinate of value pair 7
Value y8	y-coordinate of value pair 8

Outputs



Name	Width	Description
AU_Interpolation_x	32 bit	Includes the "result" of the function

2.6.2.6 Function block type AU_MEMORY



AU_MEMORY

Enables storage of a value even after ECU power has shutdown. If "enable" = 0 the last "result" is stored in the NVRAM. Initialises the stored value at restart.

Inputs

Block	Width	Filter	Description
Value	16 bit	No	Is occupied with the value to be processed.
Enable	16 bit	No	If "enable" > 0 → Transfer to output "result"

Parameters

Name	Description
Activation delay	If "enable" > 0 → result is value of signal at input "value"

Outputs

Name	Width	Description
AU_Memory_xy	32 bit	Is the value of the signal at the input "value" once the activation delay has elapsed

2.6.2.7 Function block type AU_CONST



AU_CONST

20 constants are available in order to supply the individual blocks with relevant conversion constants.

Parameters

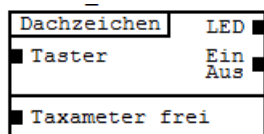
Name	Description
Value	Signed 31 bit

Outputs

Name	Width	Description
AU_Const_x	32 bit	Signed 31 bit

2.6.3 Permanently coded functions

2.6.3.1 Function block type taxi roof sign



Control for taxi roof sign.

AU_TAXI_ROOF_SIGN

Inputs

Block	Width	Filter	Description
Roof sign button	1 bit	No	Request taxi roof sign
Taxameter free	1 bit	No	Taxameter free activates run-on time of 30min for roof sign

Parameters

Name	Description
apk_dachzeichen_nachlaufzeit	Duration roof sign is active after withdrawal of ignition key
apk_dachzeichen_taster_einzeit	Function display (LED) lights up (flash frequency)
apk_dachzeichen_taster_auszeit	Function display (LED) does not light up (flash frequency)

Outputs

Name	Width	Description
LED roof sign	1 bit	Function display (LED) signal following flash frequency
Roof sign on/off	1 bit	Set if roof sign is active

2.6.3.2 Function block type taxi alarm

General	Codierung
Aus Taster	Status
Passiv	Taster Stat.
Ein	Funkhilferuf
Taster	LED
	Status
	Blinktakt
Aktiv	Taster Stat.
Ein	Alarm Status
Taster	Fernli. Stat.
	Innenli. Stat.
	SigHorn. Stat.
	Status
	Blinker Takt
	Fernlicht Takt
Funk Ein	Funkalarm

Function block serves as a means of parametrising cycle signals and enables activation of a passive or active alarm.

TAXI ALARM

Inputs

Block	Width	Filter	Description
Button Alarm off	1 bit	No	Request for alarm off
Button Alarm passive	1 bit	No	Request for alarm passive
Button Alarm active	1 bit	No	Request for alarm active
Button Alarm radio on	1 bit	No	Request radio call for help on

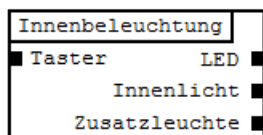
Parameters

Name	Description
Led_Ein_Intervall	Duration of function displays LED on (frequency)
Led_Aus_Intervall	Duration of function displays LED off (frequency)
Signalhorn_Ein_Intervall	Duration signal horn on (frequency with active alarm)
Signalhorn_Aus_Intervall	Duration signal horn off (frequency with active alarm)
Blinkertakt_Ein_Intervall	Duration turn signal on (frequency with active alarm)
Blinkertakt_Aus_Intervall	Duration turn signal off (frequency with active alarm)
Intermettierendes_Licht_Ein_Intervall	Duration main beam on (frequency with active alarm)
Intermettierendes_Licht_Aus_Intervall	Duration main beam off (frequency with active alarm)

Outputs

Name	Width	Description
mf_taxialarm_codiert	1 bit	Set if taxi and alarm type are coded
subcan_taster_alarm_aus	1 bit	Alarm reset button information
subcan_taster_alarm_passiv	1 bit	Alarm button information
Radio call for help	1 bit	Set if radio call for help active
led_fkt_dachzeichen	1 bit	Set if function display (LED) active
subcan_passiver_alarm	1 bit	Alarm status
leds_dachzeichen	1 bit	Signal actuates LEDs in roof sign
subcan_taster_alarm_aktiv	1 bit	Alarm button information
mf_alarm_blinken	1 bit	Set if taxi alarm active
mf_interm._fernlicht_alarm	1 bit	Set if taxi alarm active
mf_innenraumbel._alarm	1 bit	Set if taxi alarm active
mf_signalhorns	1 bit	Set if taxi alarm active
subcan_aktiver_alarm	1 bit	Set if taxi alarm active
mf_alarm_blinkertakt	1 bit	Determined frequency of turn signal cycle
mf_interm_takt	1 bit	Determined frequency of main beam flash cycle
subcan_passiver_alarm	1 bit	Alarm status

2.6.3.3 Function block type control interior light



INTERIOR LIGHT CONTROL

The function allows for switch the interior light on and off.

Inputs

Block	Width	Filter	Description
Button Interior light	1 bit	No	Activates or deactivates the interior light

Parameters

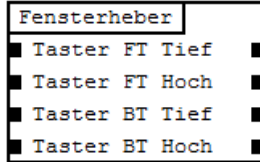
Name	Description
apk_innenlicht_nachlaufzeit	After activation, system is deactivated automatically after parametrised time

Outputs

Name	Width	Description
Interior lighting LED	1 bit	Set if function display (LED) on

Name	Width	Description
Interior light	1 bit	Transmitter signal for interior lighting
Auxiliary light	1 bit	Set if auxiliary light on

2.6.3.4 Function block type "Window regulator control"



The function makes it possible to actuate the window regulators.

WINDOW_REGULATOR_CON
TROL

Inputs

Block	Width	Filter	Description
Window regulator button, driver door, down (Taster Fensterheber FT Tief)	1 bit	No	Request driver door window down
Window regulator button, driver door, up (Taster Fensterheber FT Hoch)	1 bit	No	Request driver door window up
Window regulator button BT Tief	1 bit	No	Request passenger door window down
Window regulator button, front passenger door, up (Taster Fensterheber BT Hoch)	1 bit	No	Request passenger door window up

Outputs

Name	Width	Description
Fensterheber FT Tief	1 bit	Set if driver door window down
Fensterheber FT Hoch	1 bit	Set if driver door window up
Fensterheber BT Tief	1 bit	Set if passenger door window down
Fensterheber BT Hoch	1 bit	Set if passenger door window up



2.6.3.5 Function block type official vehicle

■ SOSI	Status
■ Funkhauptschalter	
■ Taster Sendesignal	
	Status
■ Funkgerät 1	Status
■ Kurzschluss Versorg.	
■ Funkgerät 2	Status
■ Kurzschluss Versorg.	
■ Tagfahrl. Abschaltung	
■ Taster	Aus
	LED
■ 1 Funkfrei-	Status
■ 2 Sprechen	Intern
■ Stadt/Land Umsch.	
■ Innenbeleuchtung	
■ Unterdrück	Status
■ Taster	Intern
	LED
■ Anlassersperre	
■ Waffenkasten	Summer

The function block "Official vehicle" integrates individual functions, such as special signal system, radio, weapons box etc.

OFFICIAL_VEHICLE

Inputs

Block	Width	Filter	Description
SOSI	1 bit	No	Status of special signal system
Radio master switch button	1 bit	No	Radio master switch activates radio equipment
Short circuit radio 1	1 bit	No	Connection of error status signal
Short circuit radio 2	1 bit	No	Connection of error status signal
Daytime running light shutoff button	1 bit	No	Suppresses daytime running light
Radio hands-free 1	1 bit	No	Activates radio hands-free
Radio hands-free 2	1 bit	No	Activates radio hands-free 2
Urban/rural switch	1 bit	No	Changeover of acoustics of SOSI
Button suppresses interior lighting	1 bit	No	Suppresses interior lighting
Starter interlock	1 bit	No	Prevents engine from starting
Weapons box	1 bit	No	Enables opening of the weapons box for the duration "Waffenkasten_Pulsdauer".

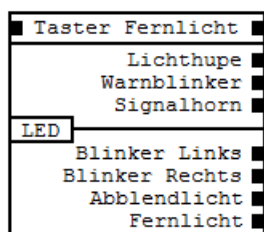
Parameters

Name	Description
Waffenkasten_Pulsdauer	Period for opening of weapons box once enabled
Funkhauptschalter_Led_Ein_Intervall	Function display radio equipment on (frequency)
Funkhauptschalter_Led_Aus_Intervall	Function display radio equipment off (frequency)
Funkhauptschalter_Led_Ein_Intervall_Fehler	Function display radio equipment error on (frequency)
Funkhauptschalter_Led_Aus_Intervall_Fehler	Function display radio equipment error off (frequency)
Abschaltverzögerung_Digitalfunk	Delay until radio master switch is deactivated
Funkhauptschalter_Spannung_Normal	Normal voltage for radio master switch
Funkhauptschalter_Spannung_Blinken	Voltage from which "Funkhauptschalter_Led_Ein Aus_Intervall_Fehler" occurs.
Funk_Abschaltung_Klemme_S_Nachlauf	Run-on until radio master switch is deactivated
Summer_Einschaltdauer	Duration that buzzer is active if radio master switch is active, key is not in ignition lock and door is opened.

Outputs

Name	Width	Description
SOSI status	1 bit	Set if SOSI is active
Radio status	1 bit	Set if radio master switch is active
Digital radio status	1 bit	Set if radio supply is active (transmission signal)
Digital radio	1 bit	Set if radio supply is active
Radio 1 status	1 bit	Set if radio 1 is active (transmission signal)
Radio 1 supply	1 bit	Set if radio 1 supply is active
Radio 2 status	1 bit	Set if radio 2 is active (transmission signal)
Radio 2 supply	1 bit	Set if radio 2 supply is active
Daytime running lights shutoff	1 bit	Set if daytime running lights off
Daytime running lights shutoff LED	1 bit	Set if function display (LED) active
Daytime running lights shutoff status	1 bit	Set if daytime running lights off (transmission signal)
Daytime running lights shutoff internal	1 bit	Set if daytime running lights off
Urban/rural switch	1 bit	Set if changeover requested
Interior lighting suppressed	1 bit	Transmission signal with status interior light for K-CAN
Interior lighting status	1 bit	Transmission signal with status interior light for CiA
Interior lighting internal	1 bit	Signal with status request interior light off
Interior lighting LED	1 bit	Function display (LED) on if interior light off
Buzzer	1 bit	Signal with status request buzzer on

2.6.3.6 Function block type light control driving aid and driving school



The function block "Light control driving aid and driving school" allows for main beam control for driving aids or driving school.

LIGHT_CONTROL_DRIVING_A
ID/DRIVING_SCHOOL

Inputs

Block	Width	Filter	Description
Button for main beam	1 bit	No	Request main beam

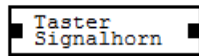
Parameters

Name	Description
Tastendruck_Fernlicht_Lichthupe	Maximum period of actuation of button for headlight flasher
Intermittierendes_Licht_Ein_Intervall	Duration for "light on" (frequency)
Intermittierendes_Licht_Aus_Intervall	Duration for "light off" (frequency)

Outputs

Name	Width	Description
Main beam request	1 bit	Set if main beam requested
Headlight flasher	1 bit	Set if main beam requested and actuation period is shorter than "Tastendruck_Fernlicht_Lichthupe"
Hazard warning lights	1 bit	Set if hazard warning lights requested by KFG
Horn	1 bit	Set if signal horn requested by KFG
Left turn signal	1 bit	Set if function indicator (LED) for left turn signal is on
Right turn signal	1 bit	Set if function indicator (LED) for right turn signal is on
Dipped beam	1 bit	Set if function indicator (LED) for dipped beam is on
Main beam	1 bit	Set if function indicator (LED) for main beam is on

2.6.3.7 Function block type signal horn control driving aid and driving school



The function block "Signal horn control driving aid and driving school" allows for activating the signal horn.

SIGNAL_HORN_CONTROL_DR
IVING_AID_AND_DRIVING_SC
HOOL

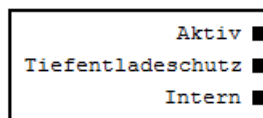
Inputs

Block	Width	Filter	Description
Signal horn button	1 bit	No	Request signal horn

Outputs

Name	Width	Description
Horn	1 bit	Set while signal horn is requested

2.6.3.8 Function block type exhaustive discharge protection



The function block "Exhaustive discharge protection" allows for configuring the exhaustive discharge protection.

EXHAUSTIVE_DISCHARGE_PR
OTECTION

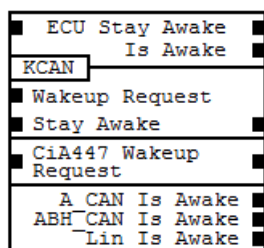
Parameters

Name	Description
Apk_Tes_U_Schwelle_Vorwarnung	Lower threshold for prewarning of undervoltage
Apk_Tes_O_Schwelle_Vorwarnung	Upper threshold for prewarning of undervoltage
Apk_Tes_Schwelle_Unterspannung	Threshold for undervoltage
Apk_Tes_Schwelle_Normalspannung	Threshold for normal voltage
Apk_Tes_Verzögerung_Vorwarnung	Delay until prewarning is detected
Apk_Tes_Verzögerung	Delay when undervoltage is detected
Apk_Versorgung_Nachlauf	Follow-up time

Outputs

Name	Width	Description
Active	1 bit	Set as soon as shutoff is active
Exhaustive discharge protection	1 bit	Set as soon as exhaustive discharge is active
Internal	1 bit	Internal status

2.6.3.9 Function block type "Control of ECU operation state handler"



The function block "Control of ECU operation state handler" allows for controlling the operation states. (sleep, wake, stay awake etc.)

CONTROL_OF_ECU_OPERATI
ON_STATE_HANDLER

Inputs

Block	Width	Filter	Description
ECU stay awake	1 bit	No	Signalises that the ECU is to stay awake
Wakeup request	1 bit	No	Request for convenience CAN bus to wake-up
Stay awake	1 bit	No	Request for convenience CAN bus to stay awake
CiA447 wake-up request	1 bit	No	Request for CiA447 to wake-up

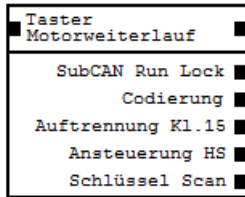
Parameters

Name	Description
p_cia447_tcanopen_wakeup_min_aktiv	Minimum time for which CANopen must be active after wakeup
p_cia447_tnachlauf_canopen	Stopping time for CANopen

Outputs

Name	Width	Description
ECU_StayAwake	1 bit	Set as long as control unit is kept awake
ECU_IsAwake	1 bit	Set if control unit is awake
K_CAN_Is_Awake	1 bit	Set if convenience CAN bus is awake
CiA447_CAN_Is_Awake	1 bit	Set if CiA447 is awake
A_CAN_Is_Awake	1 bit	Set if powertrain CAN bus is awake
ABH_CAN_Is_Awake	1 bit	Set if ABH CAN bus is awake
LIN_Is_Awake	1 bit	Set if energy LIN is awake

2.6.3.10 Function block type continued engine running switching (MWS)



The function block "Continued engine running switching (MWS)" allows for continued engine running switching.

CONTINUED_ENGINE_RUNNI
NG_SWITCHING

Inputs

Block	Width	Filter	Description
Continued engine running button	1 bit	No	Request continued engine running

Parameters

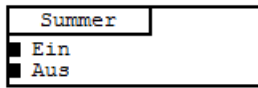
Name	Description
Motordrehzahl_Überwachungsunterbrechung	Start/stop interruption speed monitoring at start
Motordrehzahl_Verzögerung	Monitoring delay at engine start
Verzögerung_Elv_Abschaltung	Delay with MWS - Electronic steering column lock request term. 15 off
Verzögerung_Absch._Weiterl._Zulassen	Delay with MWS run-on authorisation
Verzögerung_MWS_Auftrennung_KI15	Delay with MWS separation BSG term. 15
Verzögerung_Klemmen_Freischalten	Delay with MWS reset of separation term. 15 after unauthorised drive attempt
Motorweiterlauf_Intervall	Interval for immobiliser query
Motorweiterlauf_Intervall_Elv	Interval for immobiliser query with ELV
Wegfahrsperranfragen_Anzahl	Period for immobiliser queries
Wartezeit_Wegfahrsperranfrage	Waiting time until immobiliser query

Outputs

Name	Width	Description
Continued engine running	1 bit	Set if continued engine running requested
SubCAN Run Lock	1 bit	
Coding	1 bit	Set if coding condition met
Separation term. 15	1 bit	Actuation signal for opener relay
Actuation HS	1 bit	Actuation signal for high-side supply of term. 15 bridge



2.6.3.11 Function block type vehicle buzzer



The function block "Vehicle buzzer" allows for configuring the vehicle buzzer.

VEHICLE_BUZZER

Inputs

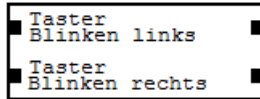
Block	Width	Filter	Description
ON	1 bit	No	Request buzzer on for duration of "Summer_Einschaltdauer"
OFF	1 bit	No	Request buzzer off

Parameters

Name	Description
Summer_Einschaltdauer	Duration for buzzer on

2.6.4 Permanently coded functions (safety)

2.6.4.1 Function block type "Turn signal control driving school"



The function block "Turn signal control driving school" allows for controlling the turn signals.

TURN_SIGNAL_CONTROL_DRIVING_SCHOOL

Inputs

Block	Width	Filter	Description
Left turn signal button	1 bit	No	Request left turn signal
Right turn signal button	1 bit	No	Request right turn signal

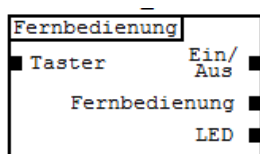
Parameters

Name	Description
apk_lenkradwinkel_rückstellung	Configures number of degrees for reset

Outputs

Name	Width	Description
Indicate left	1 bit	Set if left turn signal requested by KFG
Indicate right	1 bit	Set if right turn signal requested by KFG

2.6.4.2 Function block type enable remote control for driving aid and driving school



The function block "Enable remote control for driving aid and driving school" allows for activating and deactivating the remote control.

ENABLE_REMOTE_CONTROL
_FOR_DRIVING_AID/
DRIVING_SCHOOL

Inputs

Block	Width	Filter	Description
Remote control button	1 bit	No	Activates remote control if button is pressed shorter than "dauer_tastendruck_fernbedienung_off"

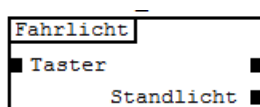
Parameters

Name	Description
nachlauf_fernbedienung	Run-on time for remote control on after ignition off
dauer_tastendruck_fernbedienung_off	Amount of time button has to be pressed to switch remote control off

Outputs

Name	Width	Description
Remote control on/off	1 bit	Set if remote control is activated
Remote control	1 bit	Set if remote control is activated (transmission signal)
Remote control LED	1 bit	Set if function lamp (LED) on

2.6.4.3 Function block type "Dipped beam control driving aid and driving school"



The function block "Dipped beam control driving aid and driving school" enables the KFG to request the dipped beam.

DIPPED_BEAM_CONTROL_DR
IVING_AID/DRIVING_SCHOOL

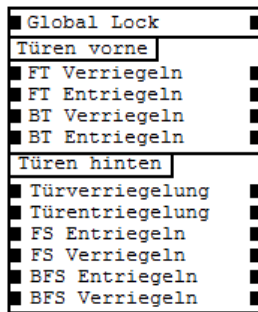
Inputs

Block	Width	Filter	Description
Button for dipped beam	1 bit	No	Request for dipped beam by KFG

Outputs

Name	Width	Description
Dipped beam request	1 bit	Set if dipped beam requested by KFG
Side lights request	1 bit	Set if side lights requested by KFG

2.6.4.4 Function block type central locking



The function block "Central locking" is used for the central locking function.

CENTRAL_LOCKING

Inputs

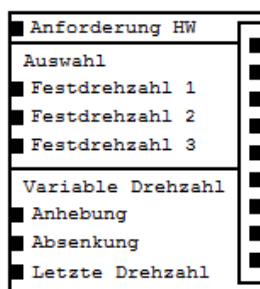
Block	Width	Filter	Description
Global lock	1 bit	No	Locks all doors on request
FT Verriegeln vorne (driver door lock front)	1 bit	No	Driver door lock request
FT Entriegeln vorne (driver door unlock front)	1 bit	No	Driver door unlock request
BT Verriegeln vorne (passenger door lock front)	1 bit	No	Front passenger door lock request
BT Entriegeln vorne (passenger door unlock front)	1 bit	No	Front passenger door unlock request
Türverriegelung hinten (door lock rear)	1 bit	No	Rear door lock request
Türentriegelung hinten (door unlock rear)	1 bit	No	Rear door unlock request
FS Entriegeln hinten (driver side unlock rear)	1 bit	No	Rear door on driver side unlock request
FS Verriegeln hinten (driver side lock rear)	1 bit	No	Rear door on driver side lock request
BFS Entriegeln hinten (front passenger side unlock rear)	1 bit	No	Rear door on front passenger side unlock request
BFS Verriegeln hinten (front passenger side lock rear)	1 bit	No	Rear door on front passenger side lock request

Outputs

Name	Width	Description
Global lock	1 bit	Set if all doors locked
FT Verriegeln vorne (driver door lock front)	1 bit	Set if driver door locked
FT Entriegeln vorne (driver door unlock front)	1 bit	Set if driver door unlocked
BT Verriegeln vorne (passenger door lock front)	1 bit	Set if front passenger door locked
BT Entriegeln vorne (passenger door unlock front)	1 bit	Set if front passenger door unlocked
Türverriegelung hinten (door lock rear)	1 bit	Set if rear door locked
Türentriegelung hinten (door unlock rear)	1 bit	Set if rear door unlocked
FS Entriegeln hinten (driver side unlock rear)	1 bit	Set if rear door on driver side locked
FS Verriegeln hinten (driver side lock rear)	1 bit	Set if rear door on driver side unlocked
BFS Entriegeln hinten (front passenger side unlock rear)	1 bit	Set if rear door on front passenger side locked
BFS Verriegeln hinten (front passenger side lock rear)	1 bit	Set if rear door on front passenger side unlocked

2.6.5 Enhanced functions

2.6.5.1 Function block type working speed governor (ADR)



The function block "Working speed governor (ACC)" allows for sending a speed request to the engine control unit in order to operate power take-off units or for elevated performance from the vehicle's onboard supply.

WORKING_SPEED_GOV
R

Inputs

Block	Width	Filter	Description
Anforderung HW (request HW)	1 bit	No	Working speeding governor request
Selection fixed speed 1	1 bit	No	Selects the fixed speed "p_adr_festdrehzahl_1"
Selection fixed speed 2	1 bit	No	Selects the fixed speed "p_adr_festdrehzahl_2"
Selection fixed speed 3	1 bit	No	Selects the fixed speed "p_adr_festdrehzahl_3"
VarDrehzahl Anhebung (variable speed increase)	1 bit	No	Increases the variable speed by "VarDrehzahl_Schritt"

Block	Width	Filter	Description
VarDrehzahl Absenkung (variable speed reduction)	1 bit	No	Decreases the variable speed by "VarDrehzahl_ Schritt"
VarDrehzahl Letzte Drehzahl (variable speed last speed)	1 bit	No	Sets the last variable speed to "VarDrehzahl"

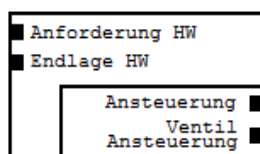
Parameters

Name	Description
p_adr_p_s_ADR_min	Minimum time for which a button must be pressed to activate the function
p_adr_p_s_ADR_max	Maximum time for which a button may be pressed to activate the function
p_adr_festdrehzahl_1	Speed, can be placed on "Festdrehzahl" (fixed speed)
p_adr_festdrehzahl_2	Speed, can be placed on "Festdrehzahl" (fixed speed)
p_adr_festdrehzahl_3	Speed, can be placed on "Festdrehzahl" (fixed speed)
p_adr_VarDrehzahl_Start	Start speed for variable speed
p_adr_VarDrehzahl_Max	Maximum adjustable speed
p_adr_VarDrehzahl_Schritt	Speed increments for increase and decrease
p_adr_pos_rampe	Serves to determine the ramp for increase
p_adr_neg_rampe	Serves to determine the ramp for decrease
p_adr_sw_rampenzeit	Serves together with the pos/neg ramp to determine the ramps
p_adr_LED_Einschaltzeit_Abb	Period that LED is on in case of cancellation
p_adr_LED_Ausschaltzeit_Abb	Period that LED is on in case of cancellation
p_adr_LED_Einschaltzeit_Vorbedingung_nio	Period that LED is on if precondition is not met
p_adr_LED_Ausschaltzeit_Vorbedingung_nio	Period that LED is on if precondition is not met

Outputs

Name	Width	Description
Request	1 bit	Set if request is active
Accelerator interlock	1 bit	Set if accelerator pedal is blocked
Negative ramp	16 bit	Signal includes setting of negative ramp
Positive ramp	16 bit	Signal includes setting of positive ramp
Specified speeds	16 bit	Signal includes engine speed to be set
Fixed speed	16 bit	Signal includes selected fixed speed
Fixed speed active	1 bit	Set if fixed speed is active
VarDrehzahl (variable speed)	16 bit	Signal includes set variable speed
LED	1 bit	Function display (LED) cycled on/off

2.6.5.2 Function block type "Power take-off"



POWER_TAKE-OFF

The function block "Power take-off" serves to switch on power take-off on gearbox side. To do this, the KFG must actuate a solenoid valve in the gearbox in order to engage the power take-off.

Inputs

Block	Width	Filter	Description
Anforderung HW (request HW)	1 bit	No	Power take-off request
Endlage HW (end position HW)	1 bit	No	End position reached

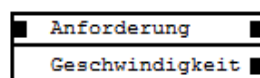
Parameters

Name	Description
p_na_t_to_endlage	Time until end position reached
p_na_t_ventil	Time until valve is actuated
p_na_Drehzahl_max	Maximum permissible speed for engaged power take-off

Outputs

Name	Width	Description
Actuation	1 bit	Set if request is active
Actuation valve	1 bit	Set if valve is actuated

2.6.5.3 Function block type speed limiter (VBEG)



SPEED_LIMITER

The function block "Speed limiter (VBEG)" allows for limiting the vehicle speed. The speed limiter cannot be overridden by the driver.

Inputs

Block	Width	Filter	Description
Request	1 bit	No	Request to limit the vehicle speed

Parameters

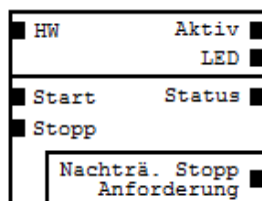
Name	Description
p_vbeg	Indicates the maximum permissible speed

Outputs



Name	Width	Description
Request	1 bit	Set if request is active
Speed	16 bit	Signal includes the maximum permissible speed

2.6.5.4 Function block type remote engine start/stop (MFSS), deactivation engine stop



The function block "Remote engine start/stop (MFSS), deactivation engine stop" allows for starting and stopping the engine remotely.

SPEED_LIMITER

Inputs

Block	Width	Filter	Description
HW	1 bit	No	Request for remote engine start/stop
Start	1 bit	No	Request for remote engine start
Stop	1 bit	No	Request for remote engine stop

Parameters

Name	Description
p_mfss_restart_timeout	Time before engine can be restarted
p_mfss_t_S_min	Minimum time for which a button must be pressed to activate the remote engine start/stop function
p_mfss_t_S_max	Maximum time for which a button may be pressed to activate the remote engine start/stop function
p_mfss_t_Start_max	Timeout for attempt to start engine
p_mfss_t_Stopp_max	Timeout for attempt to stop engine
p_mfss_LED_Einschaltzeit_Abb	Period that LED is on if Abb
p_mfss_LED_Ausschaltzeit_Abb	Period that LED is off if Abb
p_mfss_LED_Einschaltzeit_Vorbedingung_nio	Period that LED is on if precondition is not OK
p_mfss_LED_Ausschaltzeit_Vorbedingung_nio	Period that LED is off if precondition is not OK

Outputs

Name	Width	Description
Active	1 bit	Set if remote engine start is active
LED	1 bit	Function display (LED) cycled on/off
Status	8 bit	Transmission signal with joint result of functions remote engine start/stop and deactivation engine stop.
Retroactive stop request	8 bit	Transmission signal with joint result of functions remote engine start/stop and deactivation engine stop for retroactive stop request.

2.7 Communication interfaces

CAN

- CANopen / ABH CAN interface
 - High speed CAN interface (ISO 11898-2, -5)
 - Wake-capable with CANopen
 - Baud rate:
CANopen 125 kBit/s,
ABH-CAN 125/250/500 kBit/s, configurable
 - Terminal resistor: 66.4 Ohm

LIN

- LIN interface (energy LIN)
 - LIN master with 19200 bit/s

USB

- Max variant
 - USB host with charging function up to 1.5 A
 - If the circuit board temperature exceeds the overtemperature limit, the load current is reduced to 0.5 A as a built-in safety feature

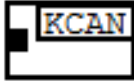
Wireless

- Max variant
 - WLAN access point
 - Bluetooth
- Connection for external aerial



2.7.1 CAN

2.7.1.1 Function block type "C-CAN transmission block"



The function block "C-CAN transmission block" transmits the signal pool value which is linked to the input.

CONVENIENCE_CAN_TRANS
MISSION_BLOCK

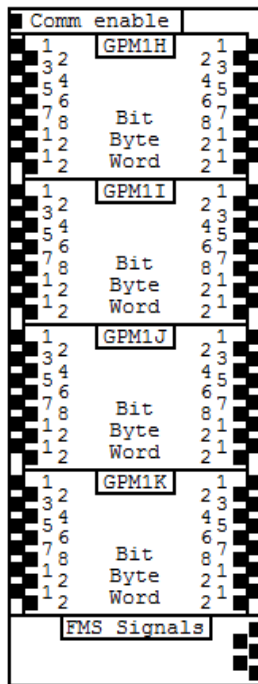
Inputs

Block	Width	Filter	Description
Transmission signal	Dependent on information	no	Is the signal to be transmitted

Parameters

Name	Description
Use safety target	Configure safety target No Yes
Replacement value of signal	If the safety target is activated, this value is transmitted in safe state

2.7.1.2 Function block type "Vehicle body builder CAN (ABH) transmission/reception block"



The function block "Vehicle body builder CAN (ABH) transmission/reception block" allows for transmitting and receiving signal pool data via the ABH-CAN.

There are four 1-bit, four 8-bit and four 16-bit function inputs available for transmitting and receiving.

ABH_CAN

Inputs

Block	Width	Filter	Description
Comm enable	1 bit	No	Activates communication via ABH-CAN
ABH_input_gpm1h_bit_x	1 bit	no	Transmission signals 1 bit
ABH_input_gpm1h_byte_x	8 bit	no	Transmission signals 8 bit
ABH_input_gpm1h_word_x	16 bit	no	Transmission signals 16 bit
ABH_input_gpm1i_bit_x	1 bit	no	Transmission signals 1 bit
ABH_input_gpm1i_byte_x	8 bit	no	Transmission signals 8 bit
ABH_input_gpm1i_word_x	16 bit	no	Transmission signals 16 bit
ABH_input_gpm1j_bit_x	1 bit	no	Transmission signals 1 bit
ABH_input_gpm1j_byte_x	8 bit	no	Transmission signals 8 bit
ABH_input_gpm1j_word_x	16 bit	no	Transmission signals 16 bit
ABH_input_gpm1k_bit_x	1 bit	no	Transmission signals 1 bit
ABH_input_gpm1k_byte_x	8 bit	no	Transmission signals 8 bit
ABH_input_gpm1k_word_x	16 bit	no	Transmission signals 16 bit

Parameters

Name	Description
FMS messages enable	FMS messages active (0) No (1) Yes
ISO messages enable	ISO messages active (0) No (1) Yes
ABH CAN 125kBaud	ABH CAN work with a Baud rate of 125k (0) No (1) Yes
ABH CAN 250kBaud	ABH CAN work with a Baud rate of 250k (0) No (1) Yes
ABH CAN 500kBaud	ABH CAN work with a Baud rate of 500k (0) No (1) Yes
GPM1H active	GPM1H active (0) No (1) Yes
GPM1I active	GPM1I active (0) No (1) Yes
GPM1J active	GPM1J active (0) No (1) Yes
GPM1K active	GPM1K active (0) No (1) Yes
Post run delay	Follow-up time

Outputs

Name	Width	Description
ABH_CAN_BusOff	1 bit	Set if one BusOff state is detected
ABH_gpm2h_in_bit_x	1 bit	1 bit reception signals
ABH_gpm2h_in_byte_x	8 bit	8 bit reception signals
ABH_gpm2h_in_word_x	16 bit	16 bit reception signals
ABH_gpm2i_in_bit_x	1 bit	1 bit reception signals
ABH_gpm2i_in_byte_x	8 bit	8 bit reception signals
ABH_gpm2i_in_word_x	16 bit	16 bit reception signals
ABH_gpm2j_in_bit_x	1 bit	1 bit reception signals
ABH_gpm2j_in_byte_x	8 bit	8 bit reception signals
ABH_gpm2j_in_word_x	16 bit	16 bit reception signals
ABH_gpm2k_in_bit_x	1 bit	1 bit reception signals
ABH_gpm2k_in_byte_x	8 bit	8 bit reception signals

Name	Width	Description
ABH_gpm2k_in_word_x	16 bit	16 bit reception signals
FMS_TotalFuelUsed	8 bit	Sum of total consumption in 0.5l / bit
FMS_HighRes_TotalFuelUsed	8 bit	Sum of total consumption in 1 ml / bit
FMS_TotalEngineHours	8 bit	Sum of total operating hours in 3 min / bit
FMS_Fuel_Rate	8 bit	Current fuel consumption in 0.05 l / bit
FMS_Fuel_Economy	8 bit	Route consumption in 0.001953125 km / bit

2.7.2 Communication module (CM) (max variant)

Communication to connectivity module

Data exchange between the two modules takes place via the internal interface to the connectivity module. The following information is exchanged:

1. General data: (direction: connectivity module → basic module)
 - a. Versions (HW, SW)
 - b. Status information about USB and radio modules
 - c. Type of connected devices (USB category, BT profiles, WLAN devices and number, ...)
2. General signal pool content: (direction: basic module → connectivity module)
 - a. Refresh rate 20ms
 - b. Status of inputs and outputs
 - c. Measured values (U_{term.30}, temperature, ...)
 - d. Selected important CAN signals for the application e.g. in exLap server
 - e. Estimated scope: approx. 250 signals / approx. 200 bytes
3. Signal pool content as required: (direction: basic module → connectivity module)
 - a. Refresh rate 20ms
 - b. Located in the basic module is a block with function inputs that can be occupied with any signals for reuse in the connectivity module.
 - c. 100 function inputs with 32 bit data width are available for the web server.
 - d. 20 function inputs with 32 bit data width are available for the exLap server.
4. Signals of connectivity module: (direction: connectivity module → basic module)
 - a. Refresh rate 20ms
 - b. Output signals of connectivity applications, e.g. for actuation of CAN transmission signals or discreet outputs
 - c. The signal pool has 80 bit signals and 20 signals with 32 bit data for the web server.
 - d. The signal pool has 20 bit signals with 32 bit data width for the exLap server.

2.7.2.1 Web server

The max variant of the control unit provides an HTML server. It allows for the HTML-based access to the signal pool data which was provided via free programming. For this, the function block "Communication module" of the configuration software provides 100 32-bit function inputs, 80 1-bit function outputs and 20 32-bit function outputs.

The signal pool values which are linked to the function inputs "Webserver_{signal_xyz}" can be retrieved using HTML code and can thus be displayed in a browser or client. The values are retrieved by "placeables" in the HTML code which follow the naming pattern listed in the table below.

1st character	2nd character	3rd character	4.-6. Characters	7. Characters
	Direction	Type	Signal ID	
{	o	c	001...999	}
	o = output i = input	c = checkbox n = numerical value b = binary value	Represents the signal number from the signal pool, e.g. Webserver _{signal_001}	

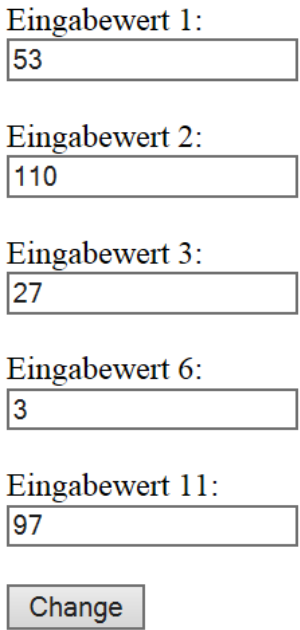


Output values are read-only. Input values can be read and written.

Sample HTML code:		Display in browser:
Values: Value1= {on001} Value2= {on002} Value3= {on003} Value4= {on004} Value5= {on005}	→	Values: Value1= 53 Value2= 110 Value3= 27 Value4= 5 Value5= 91

The HTML server on the control unit automatically replaces the sections in curly brackets "{onXXX}" with the respective values and sends the modified HTML page to the client/browser.

A form must be used for sending values to the signal pool. The sample code in the table below shows HTML code for text input fields. Based on the placeable "{inXXX}" the field shows the respective current values from the signal pool. This value can then be changed. When clicking on the change button, the new values are submitted to the control unit and stored in the signal pool by the web server.

Sample HTML code:	Display in browser:
<pre> <form method="POST">
 Input value 1:
 <input type="text" value="{in001}" name="val[1]"/>
 Input value 2:
 <input type="text" value="{in002}" name="val[2]"/>
 Input value 3:
 <input type="text" value="{in003}" name="val[3]"/>
 Input value 6:
 <input type="text" value="{in006}" name="val[6]"/>
 Input value 11:
 <input type="text" value="{in011}" name="val[11]"/>
 <input name="set_input_values" value="Change" type="submit"> </form> </pre>	

Notes on the input:

1. The transfer method for the form must be set to "POST".
2. Each input value must have a name assigned:
name="val[1].
The name is used for mapping the signal from the signal pool. The fixed mapping to the pool signals must be documented in a separate file, because an index-based access is used here. As a result, the HTML code does not indicate which signal is linked to the code.
3. The index in brackets must match the number in the signal name:
name="val[1]" → {in001}
4. The placeable is used only to output the initial or current status of the values.
The change button at the end of the form is used to initiate the data transfer to the server/control unit.



Sample HTML code:	Display in browser:
<pre><form method="POST"> <table align="left" width="10%"> <tr> <td>Input value 1:</td> <td><input name="val[1]" value="1" type="checkbox" {ic001}></td> </tr> <tr> <td>Input value 2:</td> <td><input name="val[2]" value="1" type="checkbox" {ic002}></td> </tr> <tr> <td>Input value 3:</td> <td><input name="val[3]" value="1" type="checkbox" {ic003}></td> </tr> <tr> <td>Input value 9:</td> <td><input name="val[9]" value="1" type="checkbox" {ic009}></td> </tr> <tr> <td colspan="1" align="center"> <input name="set_input_booleans" value="Change" type="submit"> </td> </tr> </table> </form></pre>	<p>Eingabewert 1: <input type="checkbox"/></p> <p>Eingabewert 2: <input type="checkbox"/></p> <p>Eingabewert 3: <input type="checkbox"/></p> <p>Eingabewert 9: <input type="checkbox"/></p> <p><input type="button" value="Change"/></p>

2.7.2.2 ExLap server

The max variant of the KFG provides an ExLap server which allows for a communication between the control unit and a client via the extensible lightweight asynchronous protocol (ExLap). In this way the client can retrieve or modify signal pool data via certain URLs.



The fleet management services (FMS) can only be retrieved and not modified.

2.7.2.2.1 Service data catalogue

Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum	Units	Remark
DashboardIlluminationLevel		[Rel]	UInt8	dynamic	Provide	0.0 1.0		0.0 = dark 1.0 = bright
DisplayIlluminationLevel		[Rel]	UInt8	dynamic	Provide	0.0 1.0		0.0 = dark 1.0 = bright
FlashingLight		[Act]	Boolean	event	Provide			
FogLight		[Act]	Boolean	event	Provide			
FrontLights	DaytimeRunningLamp	[Act]	Boolean	event	Provide			
FrontLights	HighBeam	[Act]	Boolean	event	Provide			
FrontLights	LowBeam	[Act]	Boolean	event	Provide			
FrontLights	MarkerLamp	[Act]	Boolean	event	Provide			
FrontLights	ParkingLampLeft	[Act]	Boolean	event	Provide			
FrontLights	ParkingLampRight	[Act]	Boolean	event	Provide			
HazardWarningSystem		[Act]	Boolean	event	Provide			
HeadlampFlasher		[Act]	Boolean	event	Provide			
InteriorLights	CabinFront	[Act]	Boolean	event	Provide			BCM1_Innenlicht_V
InteriorLights	CabinRear	[Act]	Boolean	event	Provide			BCM1_Innenlicht_H
RearFogLight		[Act]	Boolean	event	Provide			
TurnSignalLights	FrontLeft	[Act]	Boolean	event	Provide			
TurnSignalLights	FrontRight	[Act]	Boolean	event	Provide			
TurnSignalLights	RearLeft	[Act]	Boolean	event	Provide			
TurnSignalLights	RearRight	[Act]	Boolean	event	Provide			
ChildSafetyCatch	RearLeft	[Act]	Boolean	event	Provide			
ChildSafetyCatch	RearRight	[Act]	Boolean	event	Provide			
DoorLockControl	BootFlap	[Enm]	Boolean	event	Provide			
DoorLockControl	BonnetFlap	[Enm]	Boolean	event	Provide			
DoorLockControl	FrontLeft	[Enm]	UInt2	event	Provide			0 = Unlocked 1 = Locked 2 = Safed



Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum	Units	Remark
DoorLockControl	FrontRight	[Enm]	UInt2	event	Provide			0 = Unlocked 1 = Locked 2 = Safed
DoorLockControl	RearLeft	[Enm]	UInt2	event	Provide			0 = Unlocked 1 = Locked 2 = Safed
DoorLockControl	RearRight	[Enm]	UInt2	event	Provide			0 = Unlocked 1 = Locked 2 = Safed
IgnitionKey		[Enm]	UInt4	event	Provide			0 = NoKeyPresent 1 = KeyRemovable 2 = KeyLock 3 = ElectricityOnMotorOff 4 = MotorOn 5 = StarterActive
MotorRunLock		[Act]	Boolean	event	Provide			
WindowAperture	FrontLeft	[Rel]	UInt8	event	Provide	0.0 1.0		0.0 = opened 1.0 = closed
WindowAperture	FrontRight	[Rel]	UInt8	event	Provide	0.0 1.0		0.0 = opened 1.0 = closed
WindowAperture	RearLeft	[Rel]	UInt8	event	Provide	0.0 1.0		0.0 = opened 1.0 = closed
WindowAperture	RearRight	[Rel]	UInt8	event	Provide	0.0 1.0		0.0 = opened 1.0 = closed
TheftDetection		[Enm]	Boolean	event	Provide			
InteriorTemperature	FrontLeft	[Abs]	SIInt8	dynamic	Provide	-60.0 60.0	C	-60.0 = min 60.0 = max
InteriorTemperature	FrontRight	[Abs]	SIInt8	dynamic	Provide	-60.0 60.0	C	-60.0 = min 60.0 = max
InteriorTemperature	RearLeft	[Abs]	SIInt8	dynamic	Provide	-60.0 60.0	C	-60.0 = min 60.0 = max
InteriorTemperature	RearRight	[Abs]	SIInt8	dynamic	Provide	-60.0 60.0	C	-60.0 = min 60.0 = max
ParkHeating		[Act]	Boolean	event	Provide			
ResidualHeatHeating		[Act]	Boolean	event	Provide			
SeatVentilationLevel	FrontLeft	[Rel]	UInt8	dynamic	Provide	0.0 1.0		0.0 = off 1.0 = full
AutomaticWiper	Front	[Enm]	UInt2	event	Provide			0 = deactivated 1 = idle 2 = operating
AutomaticWiper	Rear	[Enm]	UInt2	event	Provide			0 = deactivated 1 = idle 2 = operating
WiperControl	Front	[Enm]	UInt4	event	Provide			0 = idle 1 = wipeOnce 2 = intermittent 3 = normal 4 = fast 5 = washerActive

Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum	Units	Remark
WiperControl	Rear	[Enm]	UInt2	event	Provide			0 = idle 1 = intermittent 2 = washerActive
WiperSpeed	Front	[Abs]	UInt8	event	Provide	0.0 80.0	1/ min	0.0 = min 80.0 = max
AirbagDeployment	FrontLeft	[Act]	Boolean	event	Provide			
AirbagDeployment	FrontRight	[Act]	Boolean	event	Provide			
AirbagDeployment	RearLeft	[Act]	Boolean	event	Provide			
AirbagDeployment	RearRight	[Act]	Boolean	event	Provide			
CrashDetection		[Enm]	Boolean	event	Provide			
SeatBeltLock	FrontLeft	[Enm]	Boolean	event	Provide			
SeatBeltLock	FrontRight	[Enm]	Boolean	event	Provide			
SeatBeltLock	RearLeft	[Enm]	Boolean	event	Provide			
SeatBeltLock	RearRight	[Enm]	Boolean	event	Provide			
CoolantTemperature		[Abs]	SInt16	dynamic	Provide	-60.0 200.0	C	-60.0 = min 200.0 = max
CurrentGear		[Enm]	UInt4	event	Provide			0 = idle 1 = gear_1_reverse_gear 2 = gear_2 3 = gear_3 4 = gear_4 5 = gear_5 6 = gear_6 7 = gear_7 8 = gear_8
DashboardIndicators	MF_AbsSystem	[Act]	Boolean	event	Provide			
DashboardIndicators	MF_Airbag	[Act]	Boolean	event	Provide			
DashboardIndicators	MF_AlternatorCharge Control	[Act]	Boolean	event	Provide			
DashboardIndicators	MF_Brake	[Act]	Boolean	event	Provide			
DashboardIndicators	MF_SteeringSystem	[Act]	Boolean	event	Provide			
DashboardIndicators	MF_Transmission	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_Beacon	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_ DaytimeRunningLamp	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_Esp	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_FlashingHighBeam	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_FrontFogLight	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_HazardWarning	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_HighBeam	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_LowBeam	[Act]	Boolean	event	Provide			



Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum	Units	Remark
DashboardIndicators	OP_ParkingBrake	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_ParkingLampLeft	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_ParkingLampRight	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_RearFogLight	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_ReversingLamp	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_TurnSignalLightLeft	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_TurnSignalLightRight	[Act]	Boolean	event	Provide			
DashboardIndicators	OP_TurnSignalLights	[Act]	Boolean	event	Provide			
DashboardIndicators	RM_Gearshift	[Act]	Boolean	event	Provide			
DashboardIndicators	RM_Seatbelt	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_AdaptiveLightingSystem	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_BrakeFluidLevel	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_ConvertibleTop	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_DoorClosing	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_EngineCoolantLevel	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_EngineCoolantTemperature	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_EngineHoodOpen	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_EngineOilLevel	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_EngineOilPressure	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_Fuel	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_ParticulateFilter	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_SteeringSystem	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_TankCapLock	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_TirePressure	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_TransmissionFluidTemperature	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_WasherFluidLevel	[Act]	Boolean	event	Provide			
DashboardIndicators	WN_WornBrakeLinings	[Act]	Boolean	event	Provide			
DisplayedEngineSpeed		[Abs]	UInt16	dynamic	Provide	0.0 10000.0	1/ min	0.0 = min 10000.0 = max
DisplayedVehicleSpeed		[Abs]	UInt16	dynamic	Provide	0.0 450.0	km/ h	0.0 = min 450.0 = max
DriveMode	NaturalGas	[Act]	Boolean	event	Provide			
EngineOilLevel		[Abs]	UInt8	dynamic	Provide	0.0 1.0		0.0 = min 1.0 = max

Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum	Units	Remark
EngineSpeed		[Abs]	UInt16	dynamic	Provide	0.0 10000.0	l/ min	0.0 = min 10000.0 = max
ExteriorTemperature		[Abs]	SIInt8	dynamic	Provide	-60.0 60.0	C	-60.0 = min 60.0 = max
FuelConsumption	InstantaneousValue PerMilage	[Abs]	UInt8	dynamic	Provide	0.0 25 May	l/ 100 km	0.0 = min 25.5 = INF
FuelConsumption	TotalValue	[Abs]	UInt32	dynamic	Provide	0.0 100000 0.0	l	0.0 = min 1000000.0 = max
GearSelection		[Enm]	UInt4	event	Provide			0 = Neutral 1 = Park 2 = Reverse 3 = Drive 4 = Sport 5 = Economy 6 = Semiautomatic 7 = OffRoad 8 = Max4 9 = Max3 10 = Max2 11 = Max1
Odometer		[Abs]	UInt32	dynamic	Provide	0.0 210554 06.0	km	0.0 = min 21055406.0 = max
OilTemperature		[Abs]	SIInt16	dynamic	Provide	-60.0 400.0	C	-60.0 = min 400.0 = max
ParkingBrakeApplied		[Act]	Boolean	event	Provide			
RecommendedGear		[Enm]	UInt4	event	Provide			0 = idle 1 = 1 2 = 2 3 = 3 4 = 4 5 = 5 6 = 6 7 = 7 8 = 8 9 = 9 10 = 10 11 = 11 12 = 12 13 = 13 14 = 14



Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum	Units	Remark
SelectedGear		[Enm]	UInt4	event	Provide			0 = idle 1 = 1 2 = 2 3 = 3 4 = 4 5 = 5 6 = 6 7 = 7 8 = 8 9 = 9 10 = 10 11 = 11 12 = 12 13 = 13 14 = 14
ServiceDueMilage		[Abs]	UInt16	dynamic	Provide	0.0 100000 0.0	km	0.0 = min 100000.0 = max
SpeedLimit		[Abs]	UInt8	event	Provide	0.0 250.0	km/h	0.0 = min 250.0 = max
TankLevel	Unleaded	[Rel]	UInt8	dynamic	Provide	0.0 1.0		0.0 = empty 1.0 = full
TankLevel	Premium	[Rel]	UInt8	dynamic	Provide	0.0 1.0		0.0 = empty 1.0 = full
TankLevel	Premium	[Rel]	UInt8	dynamic	Provide	0.0 1.0		0.0 = empty 1.0 = full
TankLevel	Diesel	[Rel]	UInt8	dynamic	Provide	0.0 1.0		0.0 = empty 1.0 = full
TransmissionFluid Temperature		[Abs]	SInt16	dynamic	Provide	-60.0 200.0	C	-60.0 = min 200.0 = max
AntiLockBrakeSystem		[Enm]	UInt2	event	Provide			0 = deactivated 1 = idle 2 = operating
CruiseControlSpeed		[Abs]	UInt16	dynamic	Provide	0.0 450.0	km/h	0.0 = min 450.0 = max
CruiseControlSpeedSet		[Act]	Boolean	event	Provide			
CruiseControlSpeed TipSwitch		[Enm]	UInt4	event	Provide			0 = deactivated 1 = idle 2 = accelerate 3 = decelerate 4 = resume 5 = cancel
CruiseControlSystem State		[Enm]	UInt2	event	Provide			0 = deactivated 1 = idle 2 = operating 3 = suspended
LaunchControl		[Act]	Boolean	event	Provide			
ParkAssistSystemState		[Enm]	UInt2	event	Provide			0 = deactivated 1 = idle 2 = operating 3 = passive
PowerWindowControl	FrontLeft	[Enm]	UInt2	event	Provide			0 = open 1 = close 2 = idle

Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum	Units	Remark
PowerWindowControl	FrontRight	[Enm]	UInt2	event	Provide			0 = open 1 = close 2 = idle
PowerWindowControl	RearLeft	[Enm]	UInt2	event	Provide			0 = open 1 = close 2 = idle
PowerWindowControl	RearRight	[Enm]	UInt2	event	Provide			0 = open 1 = close 2 = idle
SteeringIntervention		[Act]	Boolean	event	Provide			
BatteryTemperature	Main	[Abs]	SInt16	dynamic	Provide	-60.0 200.0	C	-60.0 = min 200.0 = max
BatteryVoltage	Main	[Abs]	UInt16	dynamic	Provide	0.0 400.0	V	0.0 = min 400.0 = max
Altitude		[Abs]	SInt16	dynamic	Provide	-1000.0 18000.0	m	-1000.0 = min 18000.0 = max
DayOfWeek		[Enm]	UInt4	event	Provide			0 = Monday 1 = Tuesday 2 = Wednesday 3 = Thursday 4 = Friday 5 = Saturday 6 = Sunday
GeoPosition	Latitude	[Abs]	UInt32	dynamic	Provide	0.0 90.0	arcDegree	0.0 = min 90.0 = max
GeoPosition	Longitude	[Abs]	UInt32	dynamic	Provide	0.0 180.0	arcDegree	0.0 = min 180.0 = max
GeoPosition	NorthSouth Hemisphere	[Enm]	Boolean	dynamic	Provide			
GeoPosition	EastWestHemisphere	[Enm]	Boolean	dynamic	Provide			
GNSSQuality	FixMode	[Enm]	UInt2	event	Provide			0 = FixNotAvailable 1 = Fix2D 2 = Fix3D
EngineDisplacement		[Abs]	UInt8	static	Provide	0.0 10.0	l	0.0 = min 10.0 = max
EnginePower	Diesel	[Abs]	UInt16	static	Provide	0.0 100000.0	W	0.0 = min 1000000.0 = max
EnginePower	Petrol	[Abs]	UInt16	static	Provide	0.0 100000.0	W	0.0 = min 1000000.0 = max
NumberOfAxles		[Abs]	UInt4	static	Provide	0.0 5.0		0.0 = min 5.0 = max



Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum	Units	Remark
SpecialVehicleType		[Enm]	UInt4	event	Provide			0 = none 1 = police 2 = fireBrigade 3 = ambulance 4 = heavyTransport 5 = publicTransport 6 = taxicab 7 = slow 8 = handicapped 9 = buildingSite 10 = agricultural 11 = accompanying 12 = other
Units	Temperature	[Enm]	Boolean	event	Provide			
Units	Consumption	[Enm]	Boolean	event	Provide			
Units	Distance	[Enm]	Boolean	event	Provide			
Units	Volume	[Enm]	UInt2	event	Provide			0 = litres 1 = Gallons 2 = ImperialGallons
Units	Pressure	[Enm]	UInt2	event	Provide			0 = Kilopascal 1 = Bar 2 = Psi
Units	DateFormat	[Enm]	UInt2	event	Provide			0 = DDMMYYYY 1 = MMDDYYYY 2 = YYYYMMDD
Units	TimeFormat	[Enm]	Boolean	event	Provide			
VehicleIdentificationNumber		[Txt]	ASCII17	static	Provide			
BrakeActuation		[Act]	Boolean	event	Provide			
BrakingPressure		[Abs]	UInt8	dynamic	Provide	0.0 500.0	bar	0.0 = min 500.0 = max
ClutchSwitchActuation		[Act]	Boolean	event	Provide			
DirectionOfDriving		[Enm]	UInt2	event	Provide			0 = standstill 1 = forward 2 = reverse
Horn		[Act]	Boolean	event	Provide			
Kickdown		[Act]	Boolean	event	Provide			
LateralAcceleration		[Abs]	SInt8	dynamic	Provide	-20.0 20.0	m/ s2	-20.0 = min 20.0 = max
LongitudinalAcceleration		[Abs]	SInt8	dynamic	Provide	-20.0 10.0	m/ s2	-20.0 = min 20.0 = max
PanicBraking		[Act]	Boolean	event	Provide			
RainIntensity		[Rel]	UInt8	dynamic	Provide	0.0 1.0		0.0 = noRain 1.0 = shower
RearWindowHeating		[Act]	Boolean	event	Provide			
SteeringWheelAngle		[Abs]	SInt16	dynamic	Provide	-1500.0 1500.0	arcD egre e	-1500.0 = min 1500.0 = max

Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum	Units	Remark
SunIntensity		[Abs]	UInt8	dynamic	Provide	0.0 100000.0	lx	0.0 = min 100000.0 = max
TrailerDetection		[Act]	Boolean	event	Provide			
TurnSignalLever		[Enm]	UInt2	event	Provide			0 = idle 1 = left 2 = right
VehicleSpeed		[Abs]	UInt16	dynamic	Provide	0.0 450.0	km/h	0.0 = min 450.0 = max
WheelSpeed	FrontLeft	[Abs]	UInt16	dynamic	Provide	0.0 450.0	km/h	0.0 = min 450.0 = max
WheelSpeed	FrontRight	[Abs]	UInt16	dynamic	Provide	0.0 450.0	km/h	0.0 = min 450.0 = max
WheelSpeed	RearLeft	[Abs]	UInt16	dynamic	Provide	0.0 450.0	km/h	0.0 = min 450.0 = max
WheelSpeed	RearRight	[Abs]	UInt16	dynamic	Provide	0.0 450.0	km/h	0.0 = min 450.0 = max
WindshieldHeating		[Act]	Boolean	event	Provide			
FMS_Alternator	Status_1	[Enm]	UInt2	dynamic	Provide			0 = not_charging 1 = charging 2 = error 3 = SNV
FMS_Alternator	Status_2	[Enm]	UInt2	dynamic	Provide			0 = not_charging 1 = charging 2 = error, 3 = SNV
FMS_Alternator	Status_3	[Enm]	UInt2	dynamic	Provide			0 = not_charging 1 = charging 2 = error 3 = SNV
FMS_Alternator	Status_4	[Enm]	UInt2	dynamic	Provide			0 = not_charging 1 = charging 2 = error 3 = SNV
FMS_PTO_State		[Enm]	UInt8	dynamic	Provide			0 = Off 5 = Set 31 = SNV
FMS_Driver_Identification		[Txt]	ASCII8	dynamic	Provide			
FMS_ATC_BrkCtrl_Stat		[Act]	Boolean	event	Provide			
FMS_AccelatorPedalLowIdleSw		[Act]	Boolean	event	Provide			
FMS_AccelatorPedalPosition		[Abs]	UInt8	dynamic	Provide	0 100	%	
FMS_EngineLoadAtCurrSpeed		[Abs]	UInt8	dynamic	Provide	0 125	%	
FMS_Fahrzeugtyp		[Enm]	UInt4	static	Provide			0 = Combustion 1 = Hybrid 2 = ElectricVehicle 3 = ElectricVehicleRange Extender



Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum	Units	Remark
FMS_IntakeSystem		[Act]	Boolean	event	Provide			
FMS_FuelType		[Enm]	UInt4	static	Provide			0 = Diesel 1 = Gasoline_E25 2 = Gasoline_E85 3 = CNG 4 = LPG 5 = Hydrogen 6 = Ethanol 7 = SNV
FMS_TotalEngineHours		[Abs]	UInt32	dynamic	Provide	0 210554 000	h	
FMS_SW_VERSION		[Txt]	ASCII4	dynamic	Provide			
FMS_Instantaneous_Fuel_Economy		[Abs]	UInt16	dynamic	Provide	0 125	km/L	
FMS_HResTotal VehicleDistance		[Abs]	UInt32	dynamic	Provide	0 210554 00	m	
FMS_Driver1_WorkingState		[Enm]	UInt4	dynamic	Provide			0 = Rest 1 = DriverAvailable 2 = Work 3 = Driving 6 = Error 7 = SNV
FMS_Driver2_WorkingState		[Enm]	UInt4	dynamic	Provide			0 = Rest 1 = DriverAvailable 2 = Work 3 = Driving 6 = Error 7 = SNV
FMS_Vehicle_motion		[Act]	Boolean	event	Provide			
FMS_Driver1_TimeRelatedStates		[Enm]	UInt4	dynamic	Provide			0 = Normal 1 = 15_min_bef_4h 2 = 4h_reached 3 = 15_min_bef_9h 4 = 9h_reached 5 = 15_min_bef_16h 7 = 16h_reached 14 = Error 15 = SNV
FMS_Driver1_Card		[Abs]	UInt2	dynamic	Provide	0 3		
FMS_Driver2_TimeRelatedStates		[Enm]	UInt4	dynamic	Provide			0 = Normal 1 = 15_min_bef_4h 2 = 4h_reached 3 = 15_min_bef_9h 4 = 9h_reached 5 = 15_min_bef_16h 7 = 16h_reached 14 = Error 15 = SNV
FMS_Driver2_Card		[Abs]	UInt2	dynamic	Provide	0 3		

Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum	Units	Remark
FMS_SystemEvent		[Act]	Boolean	event	Provide			
FMS_HandlingInformation		[Act]	Boolean	event	Provide			
FMS_Tachograph Performance		[Act]	Boolean	event	Provide			
FMS_Tachograph VehicleSpeed		[Abs]	UInt16	dynamic	Provide	0 250,996	km/ h	
FMS_Seconds		[Abs]	UInt8	dynamic	Provide	0 59	s	
FMS_Minutes		[Abs]	UInt8	dynamic	Provide	0 59	min	
FMS_Hours		[Abs]	UInt8	dynamic	Provide	0 24	h	
FMS_Month		[Abs]	UInt8	dynamic	Provide	0 12	mon th	
FMS_Days		[Abs]	UInt8	dynamic	Provide	0 31	days	
FMS_Year		[Abs]	UInt8	dynamic	Provide	1985 2240	year	

2.7.2.2.2 Extended "Service data catalogue" for free programming

In addition to the services from the data catalogue, user defined values can be retrieved from the signal pool or sent to the signal pool using free programming. This is achieved using the function block "Communication module" of the configuration software. The process is similar to handling web server signals.



20 URLs are available both for sending and receiving data.

Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum
VWN_Function_Call_Value_01	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_02	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_03	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_04	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_05	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_06	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_07	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_08	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_09	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_10	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_11	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_12	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_13	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_14	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_15	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_16	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_17	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_18	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_19	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647
VWN_Function_Call_Value_20	Value	[Abs]	SInt 32	dynamic	Receive	-2147483648 2147483647

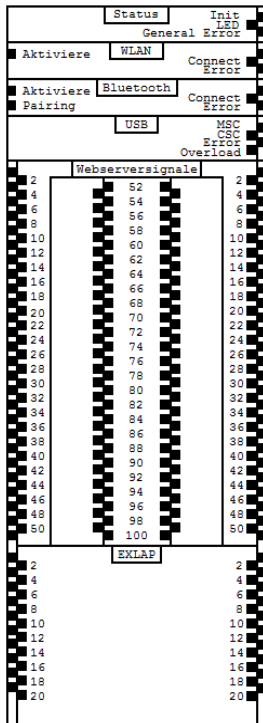
Data which is sent from the ExLap client to the ExLap server via the URL "VWN_Function_Call_Value_xy" are then available in the signal pool and can be retrieved and used via the "EXLAP_Input_Value_xy" output signals of the function block "Communication module" of the configuration tool.

In turn, the results of user-defined functions or other signal pool data can be retrieved by the ExLap server from the ExLap client using an URL "VWN_Function_Input_Signal_xy". In this case, the client sends the signal pool value which was recorded by the "Communication module" function block input "Exlapsignal_xy".

Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristics	Directions	Minimum/maximum
Data object URL	Relative name (if empty, same as URL)	Exlap data type	KFG data type	Characteristic	Direction	Minimum/maximum
VWN_Function_Input_Signal_01		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_02		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_03		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_04		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_05		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_06		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_07		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_08		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_09		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_10		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_11		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_12		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_13		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_14		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_15		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_16		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_17		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_18		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_19		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647
VWN_Function_Input_Signal_20		[Abs]	SInt32	dynamic	Provide	-2147483648 2147483647



2.7.2.3 Function block type "Communication module" (CM) (max variant)



COMMUNICATION_MODULE

The communication module serves as a means of activating/deactivating and configuring the WLAN, Bluetooth and USB functionality.

In addition, the function block allows for the data exchange between the KFG and a client using a web server or ExLap. For the web server 100 inputs and outputs are available. For ExLap 20 inputs and 20 outputs are available.

Inputs

Block	Width	Filter	Description
WLAN activation	1 bit	No	Activates/deactivates the WLAN function
Bluetooth activation	1 bit	No	Activates/deactivates the Bluetooth function
Bluetooth pairing	1 bit	No	Initiates the Bluetooth pairing process
Webserversignal_xyz	1 bit	No	Web server signal
Exlapsignal_xy	1 bit	No	EXLAP signal

Parameters

Name	Description
u_usb_charger_u_min	Voltage limit for deactivation of USB charging process due to undervoltage
t_usb_charger_u_min	Period in undervoltage until deactivation of USB charging process
t_sw_min_wlan	Minimum actuation period of input 'WLAN activation'
t_sw_max_wlan	Maximum actuation period of input 'WLAN activation'
t_sw_min_bt	Minimum actuation period of input 'Bluetooth activation'
t_sw_max_bt	Maximum actuation period of input 'Bluetooth activation'
t_sw_min_bt_pairing	Minimum actuation period of input 'Bluetooth pairing request'
t_sw_max_bt_pairing	Maximum actuation period of input 'Bluetooth pairing request'
t_bt_pairing_time	Maximum period for successful pairing

Name	Description
t_bt_pairing_led_on	Period that function indicator (LED) is on during pairing
t_bt_pairing_led_off	Period that function indicator (LED) is off during pairing

Outputs

Name	Width	Description
Initialisation status	1 bit	Set if applications are initialised and all software modules are running
LED status	1 bit	Set if function display (LED) on
General error	1 bit	Set as soon as any error is detected
WLAN status	1 bit	Set if WLAN is activated
WLAN connected status	1 bit	Set if connection to an access point or at least one client is made
WLAN error	1 bit	Set if an error in the WLAN module is detected
Bluetooth status	1 bit	Set if Bluetooth is activated
Bluetooth connected status	1 bit	Set if at least one Bluetooth device is connected
Bluetooth error	1 bit	Set if an error in the Bluetooth module is detected
Status USB MSC	1 bit	Set if a mass storage device is attached and has been detected
Status USB CDC	1 bit	Set if a communication device class is attached and has been detected
USB error	1 bit	Set if an error in the USB module is detected
USB overload	1 bit	Set if an overload at the USB port is detected
Webserver_Control_Bit_xy	1 bit	Status of control element of web server Not confirmed Confirmed
Webserver_Input_Value_xy	32 bit	Value of numerical control element of web server Maximum value 0xFFFFFFFF
EXLAP_Input_Value_xy	32 bit	Value of EXLAP-Call object "VWN_Function_Call_Value.Value_x" Maximum value 0xFFFFFFFF



2.7.2.3.1 General settings

General settings General settings of the communication module

Name	Description
Web Interface Passwort (web interface password)	The password for the web interface
Externe Antenne vorhanden? (external aerial available?)	External aerial available yes/no
Logging Schnittstelle (logging interface)	Determines from which interface the logs are issued
Logging Maske (logging template)	Level from which log messages are created
Logging Speichermaske (logging memory template)	Level from which log messages are saved
Host Name	Host Name

2.7.2.3.2 WLAN settings

WLAN settings WLAN settings of the communication module

Name	Description
WLAN Modus (WLAN mode)	Client or access point
Standard	The WLAN standard
Sicherheitsverfahren (security method)	Security method for securing WLAN
channel	0 = Auto
Region	Country code
SSID	Name used to address the service set
SSID verstecken? (hide SSID?)	Service set ID visible or invisible.
Wi-Fi secret key	The secret key for the security method
MAC Adresse Filtereintrag 1 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 2 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 3 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 4 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 5 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 6 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 7 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 8 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 9 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 10 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 11 (MAC address filter entry 1)	MAC address for which a connection is authorised



Name	Description
MAC Adresse Filtereintrag 12 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 13 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 14 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 15 (MAC address filter entry 1)	MAC address for which a connection is authorised
MAC Adresse Filtereintrag 6 (MAC address filter entry 1)	MAC address for which a connection is authorised



2.7.2.3.3 Bluetooth settings

Bluetooth settings

Bluetooth settings of the communication module

Name	Description
BT Modus (BT mode)	The Bluetooth mode
Device name	The displayed device name
PIN code	The Bluetooth PIN code
PAN mode	Personal Area Network
Pairing data 1 (MAC)	MAC address for which a connection is authorised
Pairing data 1 (Link key)	Link key for the connected device
Pairing data 2 (MAC)	MAC address for which a connection is authorised
Pairing data 2 (Link key)	Link key for the connected device
Pairing data 3 (MAC)	MAC address for which a connection is authorised
Pairing data 3 (Link key)	Link key for the connected device
Pairing data 4 (MAC)	MAC address for which a connection is authorised
Pairing data 4 (Link key)	Link key for the connected device
Pairing data 5 (MAC)	MAC address for which a connection is authorised
Pairing data 5 (Link key)	Link key for the connected device

2.7.2.3.4 IPv4 settings

IPv4 settings

Settings for the IPv4 protocol of the communication module

Name	Description
IPv4 active?	IP address static or from DHCP
KFG IP Adresse Ethernet (KFG Ethernet IP address)	The Ethernet IP address of KFG with static IP assignment
Subnet-Maske Ethernet (Ethernet subnet mask)	The Ethernet subnet mask with static IP assignment
KFG IP Adresse WLAN (KFG WLAN IP address)	The WLAN IP address of KFG with static IP assignment
Subnet-Maske WLAN (WLAN subnet mask)	The WLAN subnet mask with static IP assignment
KFG IP Adresse BT PAN (KFG BT PAN IP address)	The BT PAN IP address of KFG with static IP assignment
Subnet-Maske BT PAN (BT PAN subnet mask)	The Bluetooth subnet mask with static IP assignment
Gateway Adresse (Gateway address)	Gateway Adresse (Gateway address)
DNS Server 1	The primary DNS server
DNS Server 2	The secondary DNS server

2.7.2.3.5 IPv6 settings

IPv6 settings

Settings for the IPv6 protocol of the communication module

Name	Description
IPv6 active?	IP address static or from DHCP
KFG IP Adresse Ethernet (KFG Ethernet IP address)	The Ethernet IP address of KFG with static IP assignment
Netzwerk Präfix Länge Ethernet (Network Ethernet prefix length)	The Ethernet prefix length for static IP assignment
KFG IP Adresse WLAN (KFG WLAN IP address)	The WLAN IP address of KFG with static IP assignment
Netzwerk Präfix Länge WLAN (Network WLAN prefix length)	The WLAN prefix length for static IP assignment
KFG IP Adresse BT PAN (KFG BT PAN IP address)	The BT PAN IP address of KFG with static IP assignment
Netzwerk Präfix Länge BT PAN (Network BT PAN prefix length)	The BT PAN prefix length for static IP assignment
Gateway Adresse (Gateway address)	Gateway Adresse (Gateway address)
DNS Server 1	The primary DNS server
DNS Server 2	The secondary DNS server

2.7.2.3.6 DHCPv4 settings

DHCPv4 settings

DHCPv4 settings of the communication module

Name	Description
DHCPv4 active?	DHCPv4 activated/deactivate
Address Pool Start Ethernet	Start of the potential Ethernet IP addresses
Address Pool End Ethernet	End of the potential Ethernet IP addresses
Subnet Mask Ethernet	Ethernet subnet mask for DHCP
Default Gateway Ethernet	Ethernet gateway for DHCP
Address Pool Start WLAN	Start of the potential WLAN IP addresses
Address Pool End WLAN	End of the potential WLAN IP addresses
Subnet Mask WLAN	WLAN subnet mask for DHCP
Default Gateway WLAN	WLAN gateway for DHCP
Address Pool Start BT PAN	Start of the potential BT PAN IP addresses
Address Pool End BT PAN	End of the potential BT PAN IP addresses
Subnet Mask BT PAN	BT PAN subnet mask for DHCP
Default Gateway BT PAN	BT PAN gateway for DHCP
Lease time	The validity period for the IP configuration (comprises IP address, lease time, renewal time and rebind time)
Renewal time	The time until the client is permitted to attempt to extend the lease time. Approx. half the lease time period.
Rebind time	The time until the client is permitted to attempt to obtain an IP configuration from another server, if a request for extending the lease time has not been answered. Approx. 7/8 (87.5%) of the lease time period.



Name	Description
Offered wait time	The validity period of the offered IP configurations after a client request.
Default Domain	The default domain for DHCP
DNS Server 1	The primary DNS server
DNS Server 2	The secondary DNS server
Address reservation MAC 1	The reservation of an MAC address
Address reservation IP 1	The reservation of an IP address
Address reservation MAC 2	The reservation of an MAC address
Address reservation IP 2	The reservation of an IP address
Address reservation MAC 3	The reservation of an MAC address
Address reservation IP 3	The reservation of an IP address
Address reservation MAC 4	The reservation of an MAC address
Address reservation IP 4	The reservation of an IP address
Address reservation MAC 5	The reservation of an MAC address
Address reservation IP 5	The reservation of an IP address
Address reservation MAC 6	The reservation of an MAC address
Address reservation IP 6	The reservation of an IP address
Address reservation MAC 7	The reservation of an MAC address
Address reservation IP 7	The reservation of an IP address
Address reservation MAC 8	The reservation of an MAC address
Address reservation IP 8	The reservation of an IP address
Address reservation MAC 9	The reservation of an MAC address
Address reservation IP 9	The reservation of an IP address
Address reservation MAC 10	The reservation of an MAC address
Address reservation IP 10	The reservation of an IP address
Address reservation MAC 11	The reservation of an MAC address
Address reservation IP 11	The reservation of an IP address
Address reservation MAC 12	The reservation of an MAC address
Address reservation IP 12	The reservation of an IP address
Address reservation MAC 13	The reservation of an MAC address
Address reservation IP 13	The reservation of an IP address
Address reservation MAC 14	The reservation of an MAC address
Address reservation IP 14	The reservation of an IP address
Address reservation MAC 15	The reservation of an MAC address
Address reservation IP 15	The reservation of an IP address
Address reservation MAC 16	The reservation of an MAC address
Address reservation IP 16	The reservation of an IP address

2.7.2.3.7 DHCPv6 settings

DHCPv6 settings

DHCPv6 settings of the communication module

Name	Description
DHCPv6 active?	DHCPv6 activated/deactivate
Address Pool Prefix Ethernet	The Ethernet address pool prefix
Address Pool Prefix Länge Ethernet (Ethernet address pool prefix length)	The length of the Ethernet address pool prefix
Address Pool Prefix WLAN	The WLAN address pool prefix
Address Pool Prefix Länge WLAN (WLAN address pool prefix length)	The length of the WLAN address pool prefix
Address Pool Prefix BT PAN	The BT PAN address pool prefix
Address Pool Prefix Länge BT PAN (BT PAN address pool prefix length)	The length of the BT PAN address pool prefix
Preferred life time	The preferred life time of the IP configuration (comprises IP prefix, valid life time and preferred life time)
Valid life time	Validity period for the IP configuration
Renewal time	The time until the client is permitted to attempt to extend the valid life time.
Rebind time	The time until the client is permitted to attempt to extend the valid life time. if a request for extending the valid life time has not been answered. Approx. 7/8 (87.5%) of the valid life time period.
DNS 1	The primary DNS server
DNS 2	The secondary DNS server
Domain search	Domain search
Address reservation DUID LL 1	The reservation of a DUID LL address
Address reservation IP 1	The reservation of an IP address
Address reservation DUID LL 2	The reservation of a DUID LL address
Address reservation IP 2	The reservation of an IP address
Address reservation DUID LL 3	The reservation of a DUID LL address
Address reservation IP 3	The reservation of an IP address
Address reservation DUID LL 4	The reservation of a DUID LL address
Address reservation IP 4	The reservation of an IP address
Address reservation DUID LL 5	The reservation of a DUID LL address
Address reservation IP 5	The reservation of an IP address
Address reservation DUID LL 6	The reservation of a DUID LL address
Address reservation IP 6	The reservation of an IP address
Address reservation DUID LL 7	The reservation of a DUID LL address
Address reservation IP 7	The reservation of an IP address
Address reservation DUID LL 8	The reservation of a DUID LL address
Address reservation IP 8	The reservation of an IP address
Address reservation DUID LL 9	The reservation of a DUID LL address
Address reservation IP 9	The reservation of an IP address
Address reservation DUID LL 10	The reservation of a DUID LL address
Address reservation IP 10	The reservation of an IP address



Name	Description
Address reservation DUID LL 11	The reservation of a DUID LL address
Address reservation IP 11	The reservation of an IP address
Address reservation DUID LL 12	The reservation of a DUID LL address
Address reservation IP 12	The reservation of an IP address
Address reservation DUID LL 13	The reservation of a DUID LL address
Address reservation IP 13	The reservation of an IP address
Address reservation DUID LL 14	The reservation of a DUID LL address
Address reservation IP 14	The reservation of an IP address
Address reservation DUID LL 15	The reservation of a DUID LL address
Address reservation IP 15	The reservation of an IP address
Address reservation DUID LL 16	The reservation of a DUID LL address
Address reservation IP 16	The reservation of an IP address

3 Safety concept

3.1 Initialisation

Initialisation of the control unit after POR or wakeup runs through the following phases:

1. The μ C-internal RC oscillator vibrates
2. The reset generator of the μ C starts
3. and activates the self test for flash and RAM.
4. Upon completion of the
5. self test, the code execution is approved.
6. The application checks the result
7. of the self test. In the event of an error, a new reset is triggered.
8. If the tests were successful, the actual control unit application starts to run

In the event of complete failure of the micro controller unit, the following status is assumed by the control unit:

- No communication via CAN or LIN, also no physical influences of buses
- No reaction to discreet inputs
- All outputs, except for the relay outputs, are set to a condition which has been parametrised beforehand:
- Either static on or off. This functionality is realised via a small second micro controller unit, the safety controller.
- If a connectivity circuit board (max variant) is installed,
- the voltage supply of this module is not activated.

3.2 Error detection

- Sensors (e.g. analogue inputs for alarm buttons, turn signals, dipped beam, etc.) are monitored for plausible values. If the defined limits are violated, this is rated as an error.
- Actuators (e.g. outputs for additional lighting) are monitored via read-back wires for correct activation and external errors such as overload/open circuit/short circuit.
- CAN interface
For error detection with individual CAN messages, a message counter is used: CRC via the data and time-out monitoring as agreed with VW. The communication matrix must reserve the signals for the message counter and the CRC values.
- Application
For the operating period, correct operation (procedural control) is assured by means of an intelligent watchdog and data security with separate data storage (back-up by MPU – Memory Protection Unit) and/or redundant data store. In addition, all memories are backed up with ECC.

3.3 Signal flow

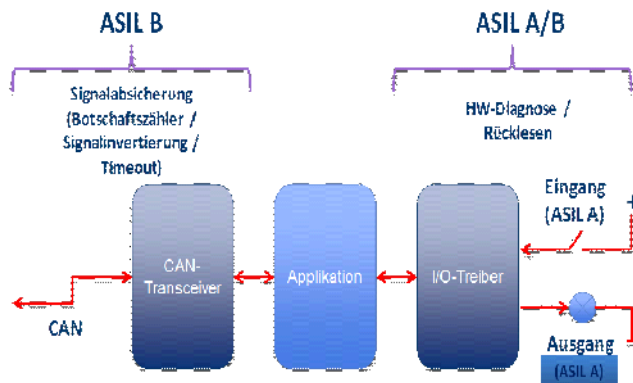


Abb. 3.1 Signal flow

3.4 Application

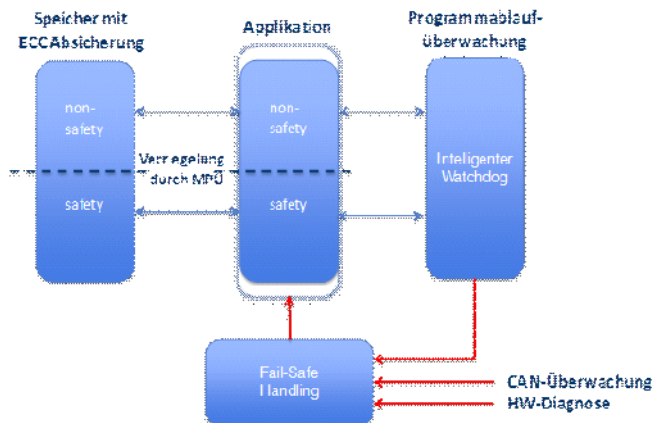


Abb. 3.2 Application

Safety-relevant functions are mapped to special tasks. No mixing of safety-relevant and unsafe functions within one task.

The data storage of the secure application functions is protected by the MPU against unauthorised write accesses from unsafe functions. The MPU is configured during initialisation in such a way that the secure storage areas are protected against any write accesses. If a task is now started with safety-relevant functions, this task opens the MPU first in order to access its own secure data. The MPU is closed again at the end of the task. In this way, assurance is provided that only the secure functions can write to the secure data area.

Opening of the MPU, in particular, to gain access to the secure storage area is only possible within the previously determined tasks with safety relevance. This is checked by the task ID. In this way, assurance is given that an unsafe task may be able to call up this function (faulty) but will have no access to the MPU register with its task ID. The function for opening is exited again directly in a case such as this.

If during execution of the secure function a (generally insecure) interrupt occurs, the interrupt handler will detect this and close the MPU again before the actual ISR is called up. At the end of the ISR, the MPU is opened again if the ISR was executed within a secure function. The status of the MPU is saved in the interrupt handler. It may not be in a safe area but it is protected against manipulation by a double storage with inversion, which means that erroneous opening of the MPU by the interrupt handler can be avoided. In the event of an incorrectly blocked MPU, a trap exception is triggered by the MPU. If several exceptions of this kind occur during operation, a reset is triggered. Likewise, the secure

context is only used within the task. The insecure operating system has no connection or configuration tasks with safety relevance.

3.5 Watchdog

- The System Basis Chip (SBC) is used as the watchdog. Triggering of this watchdog is by means of a serial interface. The watchdog is realised as a window watchdog in the hardware. It cannot be deactivated by interventions in the software.
- All of the SW modules to be monitored report to the watchdog module at task level. This counts the messages and compares them with the task cycle time stored for the SW module
- Once all of the SW modules have reported correctly within the monitoring window, resetting of the watchdog is enabled.
- Actual triggering of the watchdog then takes place in the system tick interrupt.
- Advantage: Execution of the SW modules at task level and also the interrupt function of the system tick are checked.
- With this intelligent watchdog, monitoring is carried out to ensure that all tasks are carried out in the predefined time scale and cycle, which is also a requirement of ISO 26262.

3.6 Signal pool

- Introduction of additional signal types: SafeBit (1 bit), SafeByte (1-8 bits), SafeWord (1-16 bits), (SafeDWord) (1-32 bits), SafeSignedDWord (1-31 bits)
- All safe signals are stored as redundant. In this way, a check can be carried out during interrogation to determine whether the signal value is still valid. The signals each have an error flag that indicates to the signal user whether the signal value is faulty.
- The safe signals are kept in a dedicated storage area, which is protected by the MPU against access from insecure modules.



Abb. 3.3 Signal pool

3.7 Secure discreet input

- Analogue inputs can be evaluated by means of parametrising a switch input with serial or parallel resistance.
- In this mode, the input driver uses the associated secure signal in the signal pool.
- All other signals are used as normal.
- If an error (open circuit, short to GND or hanging switch) is detected, a signal error is flagged up and can be evaluated by higher levels.

3.8 Secure CAN signal

- CAN reception signals
 - If reception signals are equipped with a relevant E/E safety feature and are required by a secure function, they are stored in a secure signal in the signal pool. In this way, the signals are available to the functions through parametrisation.
 - If a CAN signal is received with a faulty checksum/message counter, the error flag of the secure signal is set and a default value is used.
- CAN transmission signals
 - KFG transmission signals with E/E safety feature are assigned to secure signals. Before the signal value is transferred, the error flag of the signal is checked and, in the event of an error, a parametrisable default value is used.
- Special case:
 - QM CAN reception signals that are used for checking plausibility of a function activation within a secure function:
To enable this, so-called QM+ signals are added to the secure part of the signal pool. These CAN signals are evaluated by means of time-out monitoring and debounce in order to achieve the lowest level of error detection and error tolerance. Errors within this monitoring do not lead to an error being flagged up and therefore do not directly result in the secure status of the downstream function.

3.9 Secure function

- Secure functions can be parametrised via function inputs in the same way as (previous) normal functions.
- If a secure signal is placed at a function input, an error flag of the signal can be used to trigger a fail-safe reaction according to the assigned safety target.
- If a normal signal is used, the basic function is still not given but a fail-safe reaction can no longer be triggered.
- The result of the secure function is saved as a secure signal in the signal pool.

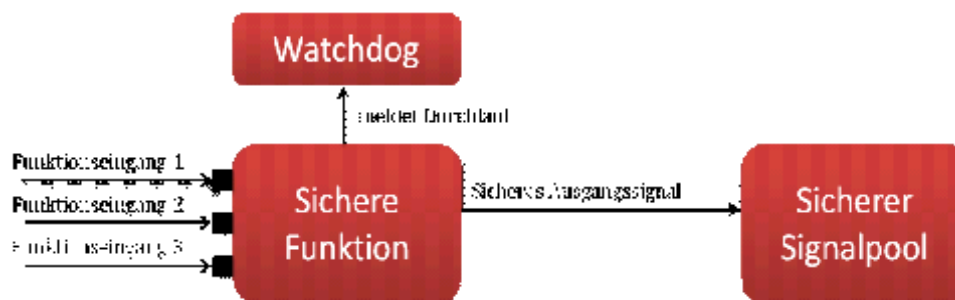


Abb. 3.4 Secure function

3.10 Secure SPS

- To facilitate maximum flexibility even with safety-relevant functions, not only the whole SPS but also the whole AU are realised as a secure function (ASIL B) as well.
- As with the secure functions, the period of operation and correct execution of the individual logic blocks are monitored here as well.
- All output signals of the SPS are stored in the secure signal pool.

3.11 Secure output

- The function inputs of the discreet outputs recognise a secure signal and, depending on the parametrisation, can retract or set actuation of the output if an error flag is set.
- To check whether an output has been switched correctly, the read-back line of the output is used. In the event of an error, retraction of the actuators takes place and an entry is made in the fault memory.

3.12 Secure parametrisation

- All parameter sets are stored with a checksum in the NVM.
- As the NVM parameter sets in the RAM mirrors are interrogated during system initialisation, the checksum is checked and, if an error is found, a substitute data set is selected and an error code is entered
- With safety-relevant parameter sets, this checksum inspection is applied on the RAM mirrors cyclically in order to detect errors in the parameter set during operation as well

3.13 Detection of main controller failure

Communication between the main controller and safety controller is linked via a serial interface. The telegrams are transferred by means of an ISO 14230-based frame with length information and checksum. Details about this transfer are included in the document [SC_DIAG]. In this way, transmission errors are generally detected by bit dumpers or similar and the frame in question is rejected. The content of the frames is laid out on a higher level in accordance with the KWP2000 protocol, which implements a client/server concept. In other words, only applicable queries of the main controller are responded to with a corresponding response from the safety controller.

The main controller sends a request cyclically to reset the time-out of the communication monitoring in the safety controller. The request includes a serial counter (message counter) which can be used to detect failure or incorrect operation of the counter. Every time this request is received with correct counter reading, the safety controller triggers the SW module `safety_control.c`. If this triggering does not take place, the outputs are moved to secure status.

The secure status (output on/off) is predefined by the configuration (NVM block) in the safety controller. This configuration is queried by the main controller each time communication starts and compared with the specifications of the KFG configuration. If there is a discrepancy, the configuration in the safety controller is updated with the new values.

3.14 Detection of safety controller failure

The main controller sends a request cyclically to reset the time-out of the communication monitoring in the safety controller. If this request is not responded to by the safety controller, the main controller triggers a reset of the safety controller via a GPIO cable. If in repeated cases requests are not responded to, an error is set in the main controller.



4 Specifications

4.1 Absolute thresholds

Parameters	Symbol	Min	Type	Max	Unit
Voltage		7.0	13.5	18.0	V
Jump-start				28.0	V
Load dump				35.0	V
Electricity				80	A
Current through a connector pin MCP 2.8 mm (with suitable mating connector and connection line with 2.5mm ² at 25°C)				25	A
Current through a connector pin MQS 0.64 mm (connection line with 0.5mm ² at 25°C)				4	A

4.2 Technical data

Parameters	Symbol	Min	Type	Max	Unit
Temperature range ambient air operation		-40	25	85	°C
Temperature range ambient air storage		-40	25	90	°C
Protection type			IP5K1		
Operating voltage		9.0	13.5	16.0	V
Permanent current (distributed over nominal currents of all outputs)				77	A
No-load current at 13.5V and 25°C					
No-load current basic variant only CAN/LIN wake- compatible			270		µA
No-load current increase basic variant by way of wake-compatible low active digital input			140		µA
No-load current increase basic variant by optional wakeup-capable HS CAN			30		µA
No-load current increase by max variant			20		µA

Developer manual

5 The interface library for the KFG app interface

5.1 Introduction

This document is made up of two parts. The first part is a general description of the KFG app interface (customer-specific function control device) and the planned scope of application. The second part offers a technical description in order to enable app developers to use the interface library for the KFG app interface within their own projects. In order to explain some of the design principles, the documentation refers to an example app project which is provided together with this documentation.

5.2 Planned scope of application

The interface library for the KFG app interface should be used to summarise communication between the KFG and the associated application.

It should also reduce the overall expense of development so that app developers can concentrate on the app application code.

5.3 KFG app interface

The interface library for the KFG app interface uses the asynchronous data transfer protocol EXLAP (Extensible Lightweight Asynchronous Protocol) for communication between the KFG control device and the Android smartphone running the app.

EXLAP is an open XML protocol developed by Volkswagen AG which is available via a creative common licence.

5.3.1 Data types

The KFG app interface uses two different data types.

5.3.1.1 Vehicle data

The "vehicle data" category covers all data transferred during communication from the KFG to the app which does not require specific configuration within the KFG.

But the vehicle data interface is generic. This means that if a data URL is available for a specific vehicle data attribute, then a corresponding data source must be available in the actual vehicle in order to ensure proper data content.

Example:

The "TheftDetection" URL provides information as to whether there is an active theft warning system alarm or not. In order for this URL to be able to deliver useful information, a theft warning system needs to be installed in the vehicle.

It is therefore recommended to check during the app design phase whether all vehicle attributes which should be available within the app tally with the planned vehicle configuration.

Further details on the vehicle attributes supported can be found in Chapter 1.4.

5.3.1.2 Customer-specific data

The “customer-specific data” category covers the freely configurable input and output data for the KFG app interface, the content of which is defined via the KFG configuration file. This input and output data is defined during the KFG configuration and is therefore project-specific.

It is very much recommended that a consistent understanding with respect to the meaning of this data is guaranteed between the KFG configuration designer and app developers.

Safety-related aspects

The customer-specific data also allows access to functions, so that data can be defined in the KFG and specific functions started in the relevant vehicle as a function of the relevant KFG configuration.

In this case, the app is part of a complex system. Therefore, the developer of the whole system is responsible for the implementation of safety mechanisms in order to protect the users and resources from damage which could be caused by unintentional actions within the app.

In this case, the app must be considered a safety-related component.

Especially if the app is used via a wireless interface to the vehicle, it should always be taken into consideration that there may no longer be direct visual contact between the user and the unit controlled. If using wireless interfaces, there is also always the risk of interrupted data connections because of distance or other external factors.

Therefore, appropriate design and architecture mechanisms should be implemented in order to prevent unintentional actions within the system.

5.3.1.3 KFG configuration identifier

All app functions must be coupled with the relevant KFG function defined in the KFG configuration at all times. Therefore, the app must be able to ensure that it is communicating with a KFG with the “correct” configuration.

The following approach is recommended to guarantee this:

- Every KFG configuration provided by VW-N incorporates a unique number which is used as an identifier.
- This number can be requested by the app via the URL “xxx”.
- The app development is based on the knowledge of a specific KFG configuration, so the identifier specified is known within the app.
- Once communication has been established between the app and the KFG, the app requests the identifier and then checks whether this is the unique number expected.
- If the number is different or no number can be determined, the app must notify the user accordingly and block functions which could trigger action within the KFG.

5.4 Architecture of KFG client and KFG server

The KFG server is the central point for the collection and distribution of vehicle data. The interface library for the KFG app interface provides a client (*KFGClient*) in order to enable different methods for enquiries to the KFG server and for receipt of the relevant responses. The KFG services are defined based on a range of data objects and functions.

Objects can be subscribed for a specific interval (“send me data no longer than”), while the response for functions is only transmitted once. In principle, the communication is asynchronous.

The following chapter offers a rough overview of the EXLAP-based client/server architecture of the KFG app interface. More detailed information can be found in the EXLAP-specific documentation.

5.4.1 KFG client communication

The first diagram gives an overview of the communication system.

The general approach in line with the steps shown in the diagram is as follows:

- Produce *KFGClient* instance

- Register *KFGListener* and *ConnectionListener* instance
- On successful connection:
- Subscribe to objects
- Access functions
- Asynchronous receipt and asynchronous processing of data in the relevant *KFGListener* methods
- Where necessary, remove subscription to objects
- Exit *KFGClient* (disconnect)

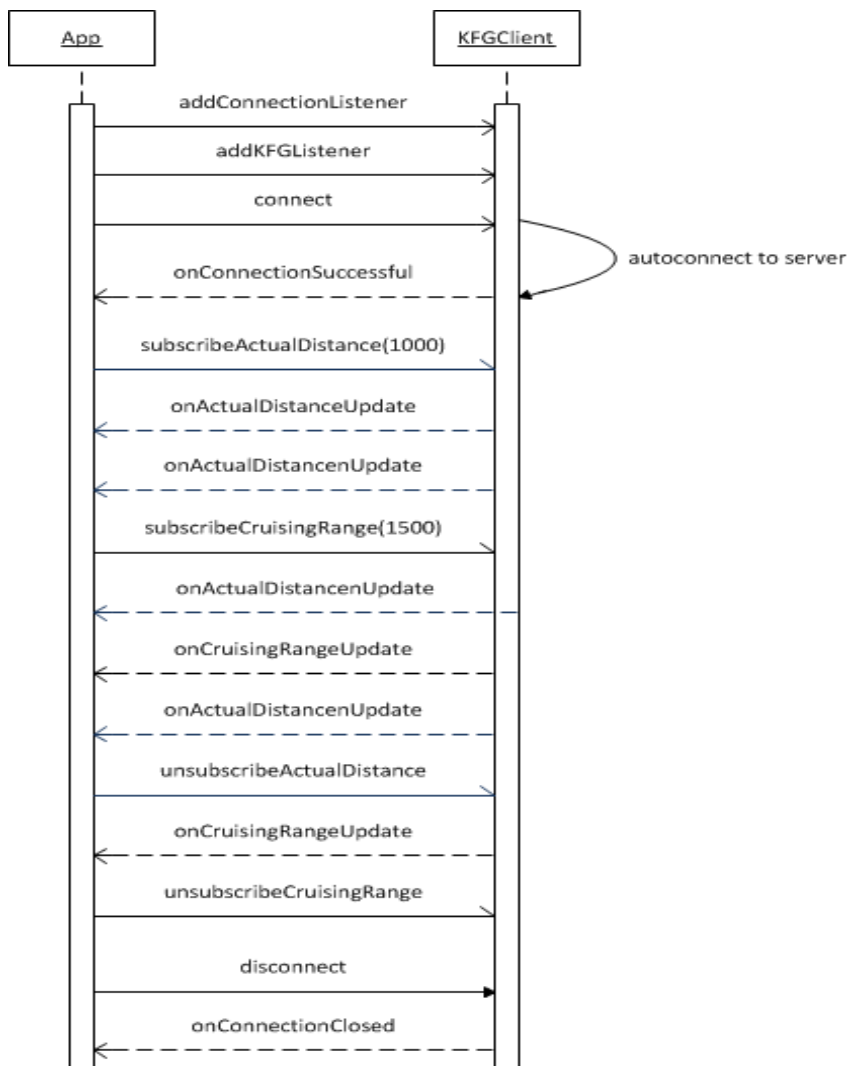


Abb. 4.1 Example program sequence

The *KFGClient* instance is the main class to be used. A *KFGListener* and *ConnectionListener* instance must be added in order to receive data and connection status.

The *KFGListener* instance is notified if data is available for a subscription or a callback for a function call on the server.

The *ConnectionListener* instance is a callback function for any connection problems. Diagram 2 shows callback methods for the *ConnectionListener* instance, each of which has a different connection status.

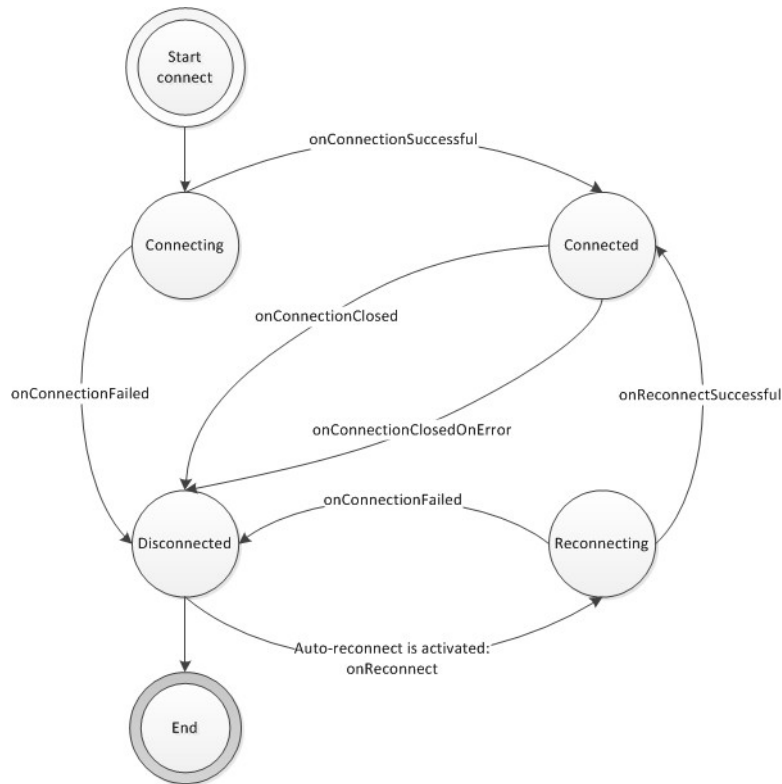


Abb. 4.2 Overview of callback methods for the *ConnectionListener* instance and relevant connection statuses.

To enable proper handling of the connection between the KFG and the app, the developers must implement *ConnectionListener* callback methods in line with the relevant app requirements.

The individual functions of the *ConnectionListener* instance are described in Javadoc (provided as JAR and HTML files in the *jexlap-core* module).

5.4.2 Functions

This section describes various functions which have special features or can be activated via the interface library for the KFG app interface.

5.4.2.1 Automatic connection (autoconnect)

In some devices, as well as the internet authorisation, permission to change the multicast status is required in order to implement the automatic client connection function via the wireless connection.

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" />
```

The multicast block then needs to be set in the code before the actual connection is established.

```
if (multicastLock == null || !multicastLock.isHeld()) {  
    WifiManager wifi = (WifiManager)  
        applicationContext.getSystemService(Context.WIFI_SERVICE);  
    multicastLock = wifi.createMulticastLock("kfg discovery");  
    multicastLock.acquire();  
}  
// Lift multicast block after successful call of onConnectionSuccessful  
multicastLock.release();
```

5.4.2.2 Multiple subscribers (Multisubsciber)

A connection to the KFG is only required for one *KFGClient* instance. It is not possible to create multiple *KFGClient* instances within an app. However, you can subscribe to a signal with different intervals and listeners. This involves adding a listener directly to an existing subscription.

```
client.subscribeActualDistance (2500);  
...  
KFGListener kfgListener = new MyKFGListener();  
client.subscribeActualDistance (5000, kfgListener);
```

5.4.2.3 Restoration of connection

For unstable connections, there is the option of automatically reconnecting the client to the server. This involves activating the relevant connection in the *KFGConfiguration*. If the automatic connection (Autoconnect) function is used (i.e. the client attempts to establish a connection), the repeat connection attempt is executed on a one-off basis for a maximum of 12 seconds. When the connection is restored, the Android app is informed by the *ConnectionListener* by calling up *onReconnect()*. After an attempt to restore the connection, the Android app is either informed either that it has been successful by calling *ConnectionListener.onReconnectSuccessful()* or that it has failed by calling *ConnectionListener.onConnectionFailed()*.

```
client = new KFGClient(new KFGConfiguration(true));  
client.connect();// executes the search and automatic connection
```

5.4.2.4 KeepAlive

In order to use the *keepAlive* function, it must be activated in the *KFGConfiguration*. At the same time, a time limit and a number of repeat attempts for the sending of *keepAlive* commands can also be defined. The default time limit for a *keepAlive* command is 10 seconds. Values of between 0 (excluding) and 10 (inclusive) are allowed. Any other time limit values set trigger the exception *IllegalArgumentException*. The value of the time limit also defines the time between the sending of *keepAlive* commands. You can adjust the number of repeat attempts for sending another *keepAlive* command to the server. By default, only one *keepAlive* command is sent. If a *keepAlive* command does not receive a response from the server x times, the registered *ConnectionListener* is informed using the *onConnectionClosedOnError(Exception e)* method with the exception



ExlapException(ExlapConstants.STATUS_KEEPA_LIVE_TIMEOUT, "Did not receive keep alive response in appropriate time. Connection may be slow or is gone.").

```
boolean useReconnect = true; // activate connection restore
boolean useKeepAlive = true; // activate keepAlive
KFGConfiguration cfg = new KFGConfiguration(useReconnect, useKeepAlive);
cfg.setKeepAliveTimeout(5); // the time for a keepAlive command sent is exceeded after 5
seconds and it is sent again after 5 seconds
cfg.setKeepAliveRetries(4); // attempts to send a keepAlive command 4 times
client = new KFGClient(cfg);
client.connect();// executes search and automatic connection
```

5.4.2.5 "supports" function

"supports" methods can be used to check whether a specific subscription or function call is supported by the server.

```
boolean isActualDistanceSupported = client.supportsActualDistance();
```

5.5 KFG client – general methods

The *KFGClient* represents the main class of the interface library for the KFG app interface. It offers various methods for accessing data and initiating actions on the server. The following section describes some general methods of the *KFGClient* which can be used in app development.

5.5.1 Synchronous data requests

The *getObject* data requests the current value of a data object (or accesses it) set by an object URL. The KFG then immediately delivers the result in a corresponding response. The method request the data for each call, but processes and sends it asynchronously.

```
//url - The URL of the data object
//listener - A reference to AsyncCallback with relevant information on completion of the
task
int getObject (final String url, final AsyncCallback listener)
```

5.5.2 Subscribing to data objects

The *subscribeObject* method requests asynchronous data updates based on content changes to a data object given by an object URL.

```
//url - The URL of the data object to be subscribed
//ival - The minimum interval between the individual data object updates in milliseconds
synchronized void subscribeObject(String url, int ival)
//or
synchronized void subscribeObject(String url)
```

When using object subscriptions, the impact on the overall performance of the app should be taken into consideration.

This applies particularly to URLs updated at short intervals. For example, the URL "EngineSpeed" is updated in the KFG via the relevant CAN signal every 20 milliseconds. So if this URL is subscribed with an *ival* value of "20", the signal is updated 50 times per second as "real engine speed" is constantly changing.

On the other hand, there are URLs whose signal values only change infrequently, which means a low *ival* value has no significant impact on performance. For example, the status for a door is only changed if the door is opened or closed. If the *ival* value is set to "0", the change in signal status is forwarded to the app immediately.

Therefore, during the app design phase, every signal to be subscribed must be evaluated with respect to its dynamic behaviour within the vehicle. The appropriate time interval for the subscription (*ival*) is selected based on this.

At the same time, no URLs should be subscribed for which no asynchronous data updates are required within the application.

For some signals, the update frequency is also dependent on the driving situation of the vehicle. For example, travel speed only changes when the vehicle is in motion. This must be taken into consideration during the system integration test.

5.5.3 Removing a subscription

The *unsubscribeObject* method removes the active subscription for the given URL.

```
//url - The URL of the data object for which the subscription is to be removed  
synchronized void unsubscribeObject (String url)
```

As mentioned in the previous section, object subscriptions which are not required for the normal operation of the app should be removed. The removal of subscriptions is also recommended if there is a communication failure with the KFG.

5.5.4 Accessing functions

The *unsubscribeObject* method starts an asynchronous function call in accordance with the relevant function URL.

```
//url - The URL of the function to be called  
//callParameter - The call parameter as a data object  
int callFunctionAsync(final String url, final DataObject callParameter)
```


6 Example application

This section describes the KFG example application.

6.1 Setting up Android Studio

Android Studio is the official built-in development environment for the development of Android apps. It is based on IntelliJ IDEA. Download the latest version of Android Studio below Android Studio 2.2 and set the program up.



Abb. 5.1 Android Studio: start screen

6.2 Opening an example application project in Android Studio

The following section describes how an example application project is opened in Android Studio.

- 1 In the Welcome to Android Studio window, click on Open an existing Android Studio Project .

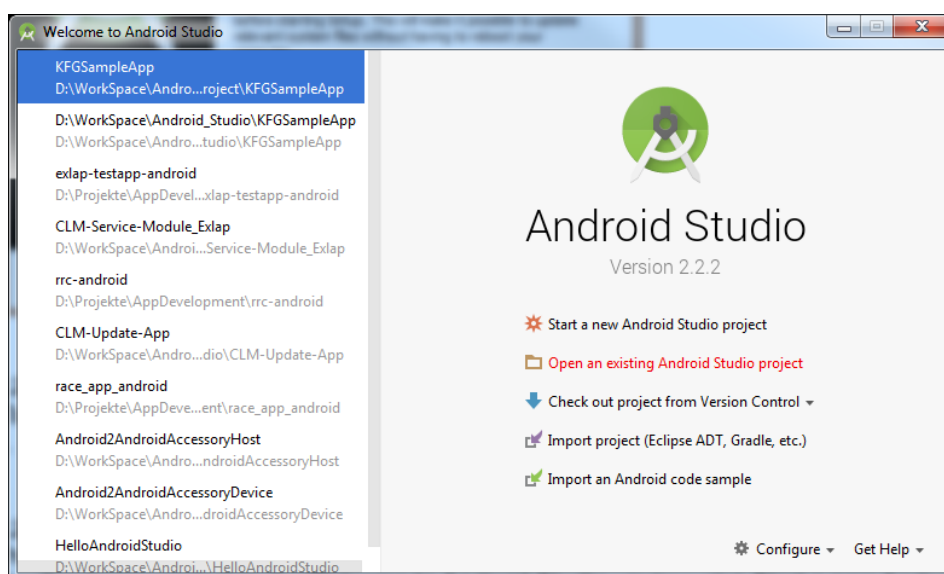


Abb. 5.2 Android Studio: opening a project

- 2 In the Open file or project window, select the folder KFGSampleApp, and click on OK.

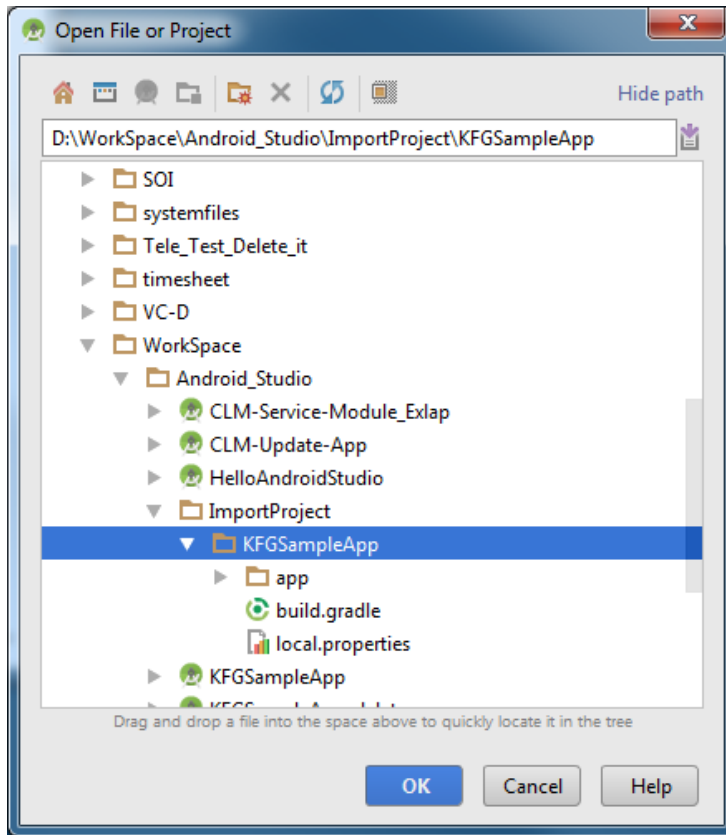


Abb. 5.3 Android Studio: selecting a project

- 3 After a short processing time, a KFGSampleApp is opened in Android Studio.

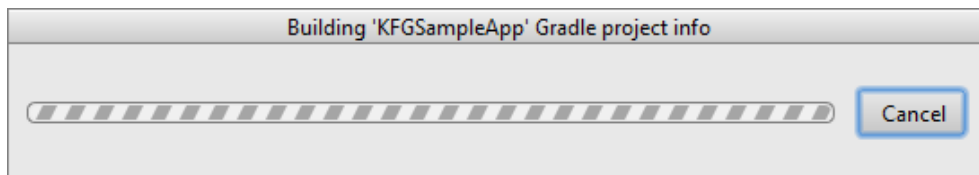


Abb. 5.4 Android Studio: project is being opened

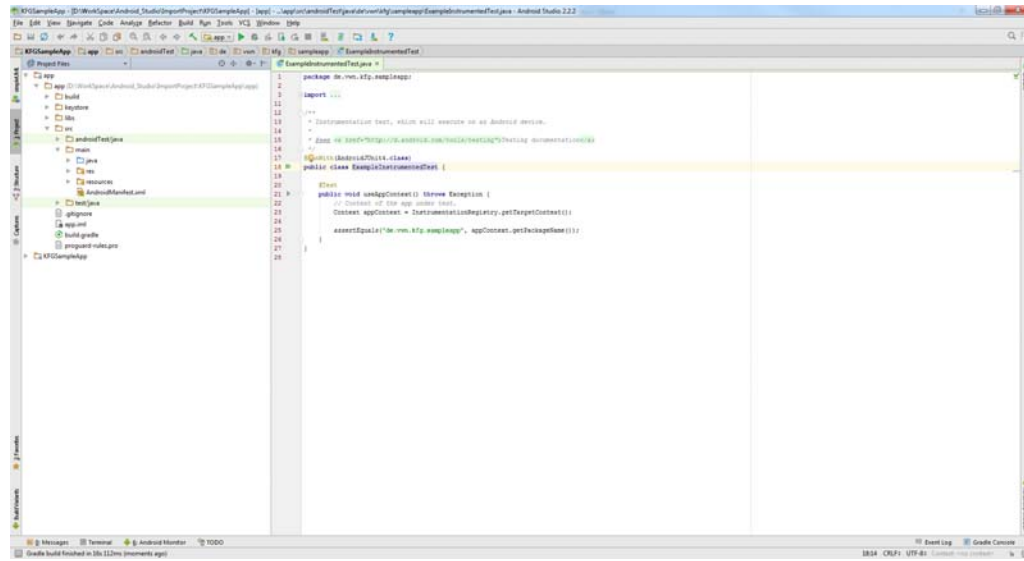


Abb. 5.5 Android Studio: project has been opened

6.3 Creating an Android project

This section describes how a new Android project can be created in Android Studio.

- 1 How to create a new project: In the Welcome to Android Studio window, click on Start a new Android Studio Project .

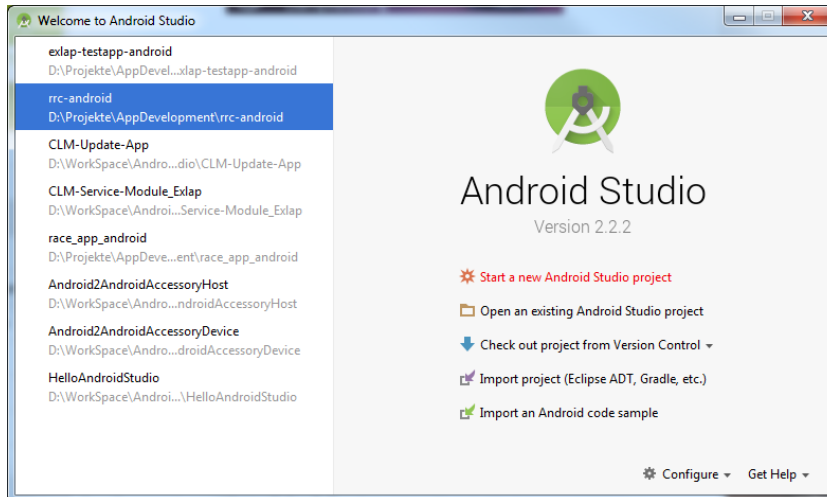


Abb. 5.6 Android Studio: starting a new project

- 2 In the "New Project" dialog, you can enter the following values, for example:
 - Application name: "KFG Sample App"
 - Company domain: "android.kfg.vwn.de"
 - The packet name and the location to which the project is saved are automatically entered by Android Studio but can also be amended by the user. Click on "Next".

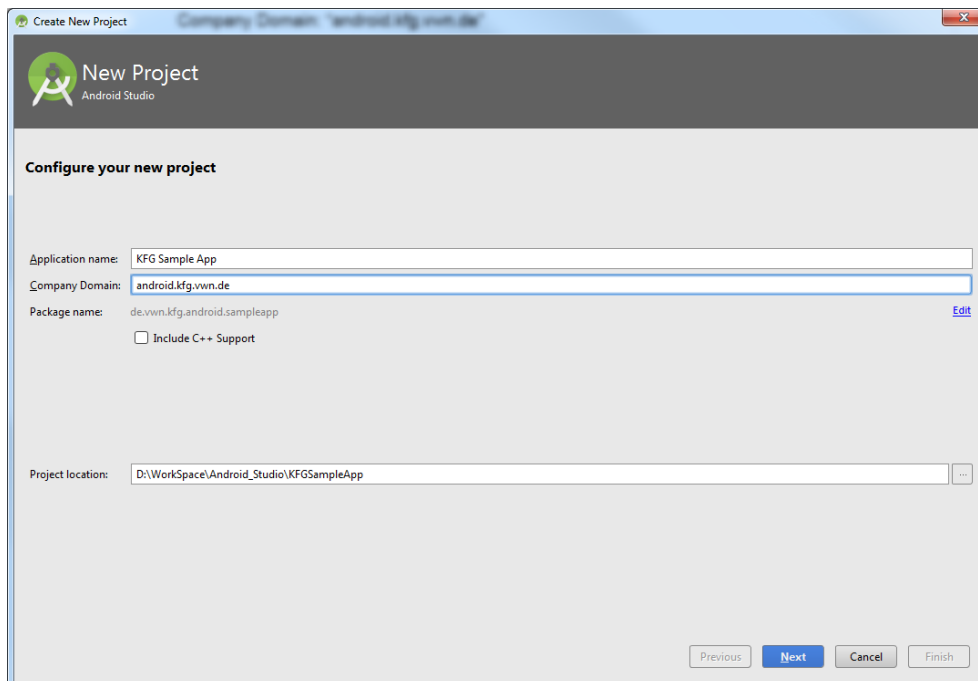


Abb. 5.7 Android Studio: configuring a new project

- 3 In the "Target Android Devices" dialog, select the "Phone" and "Tablet" options. In the "Minimum SDK" list field, enter the API level "15" to support as many devices as possible. Click on "Next".

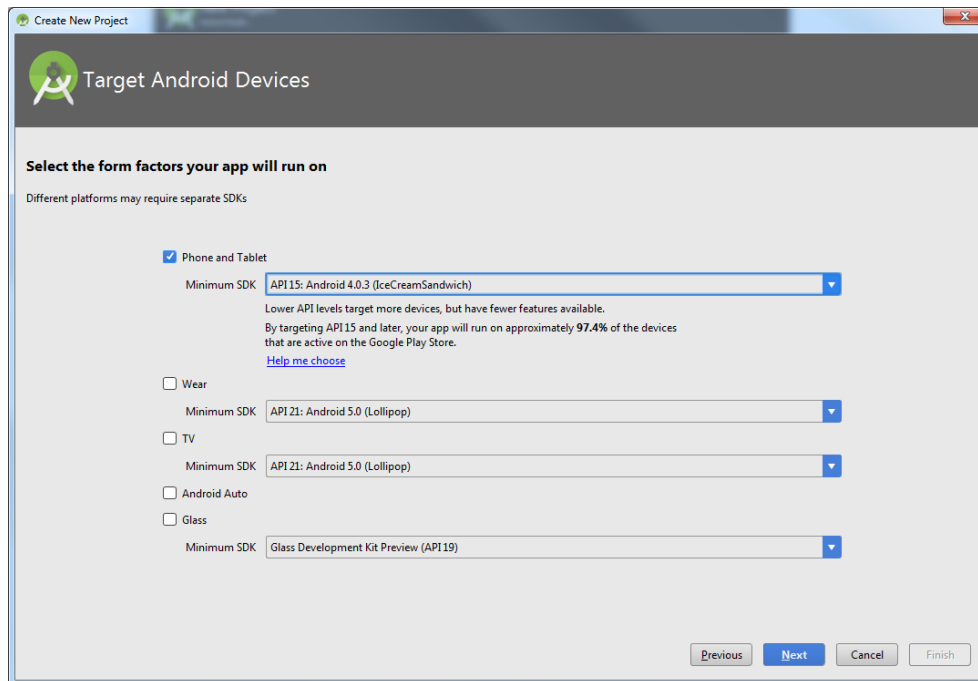


Abb. 5.8 Android Studio: selecting a target device

- 4 In the Add an Activity to Mobile dialog, select Empty Activity and click on Next.

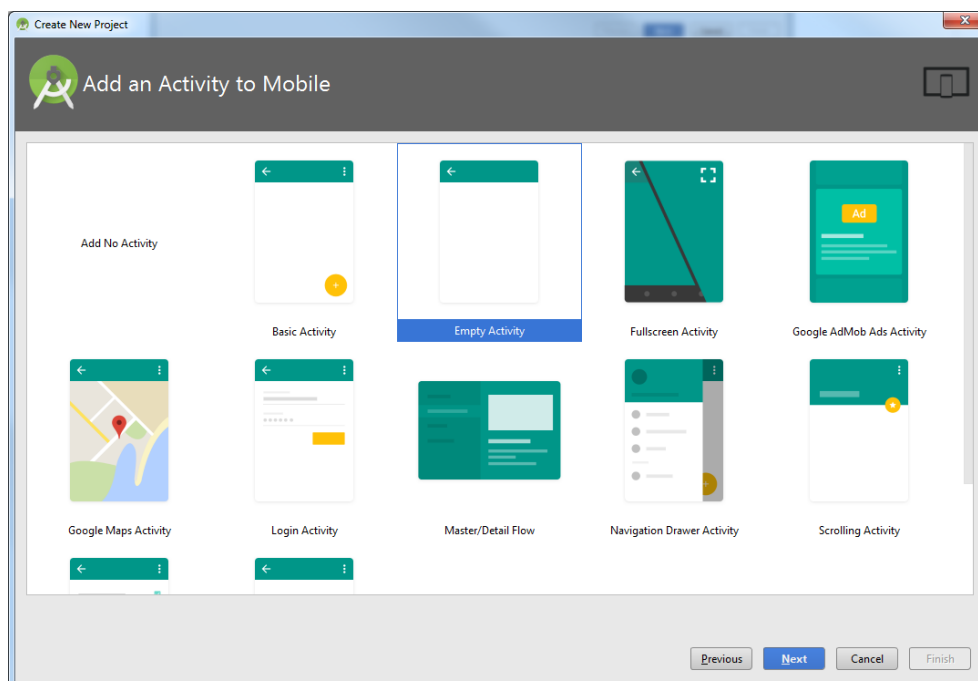


Abb. 5.9 Android Studio: adding an activity

- 5 Leave the default settings in the Customize the Activity unchanged and then click on Finish.

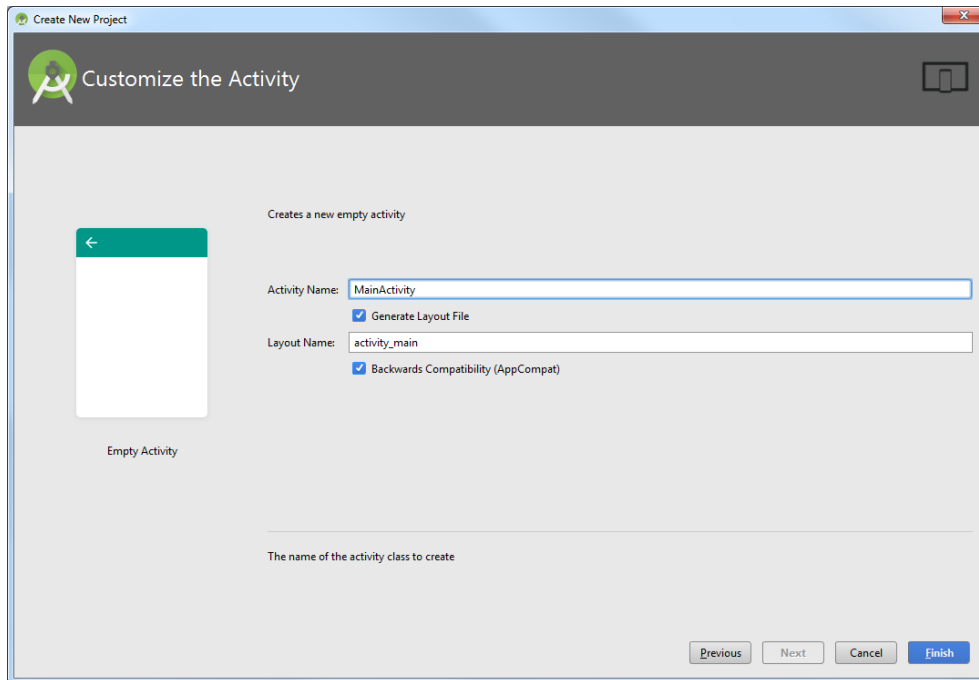


Abb. 5.10 Android Studio: customising an activity

- 6 After a short processing time, the KFG example app is opened in Android Studio with default files. Additional functions can be added to the files.

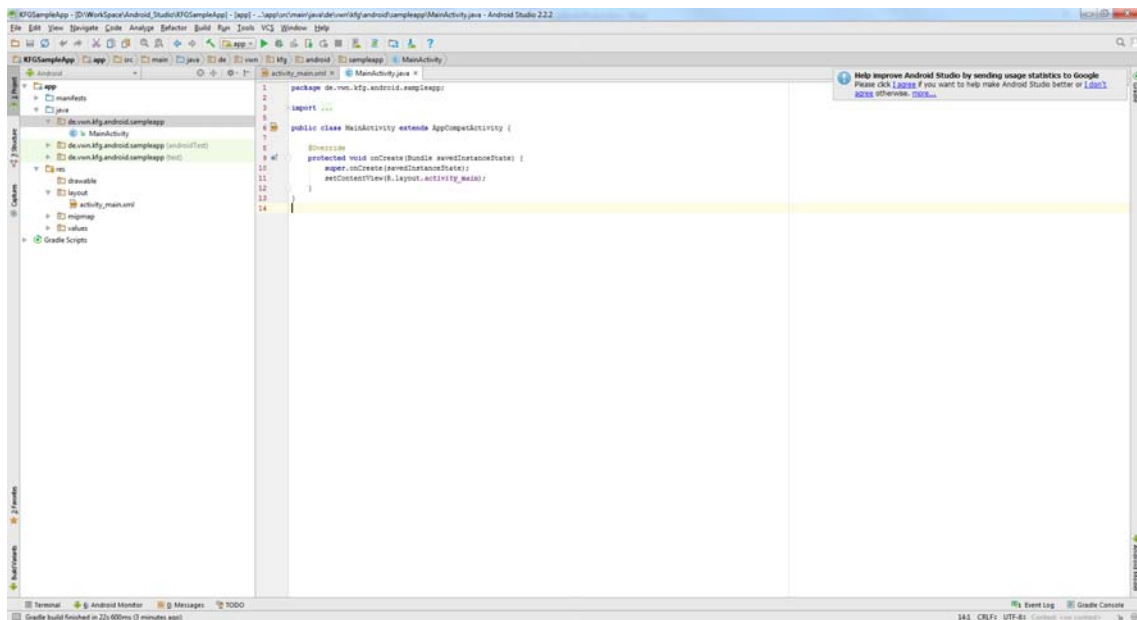


Abb. 5.11 Android Studio: adding functions to files

6.4 Adding files from the interfaces library for the KFG App interface to an Android Project

Create a libs folder in the application folder (if it does not exist already).

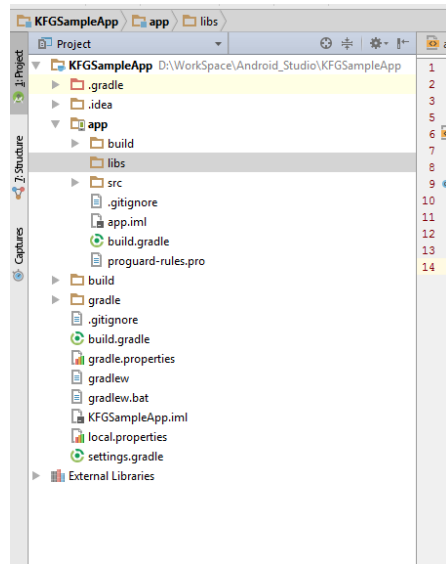


Abb. 5.12 Android Studio: creating libs folder

1 Add the libs folder KFG dependency files (.jar).

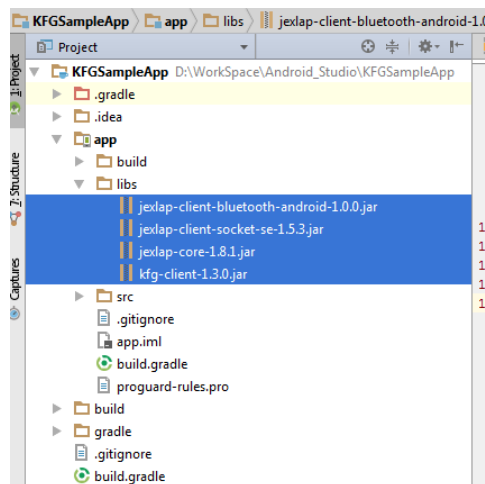


Abb. 5.13 Android Studio: adding KFG dependency files

2 Add (.jar) files for build.gradle dependencies.

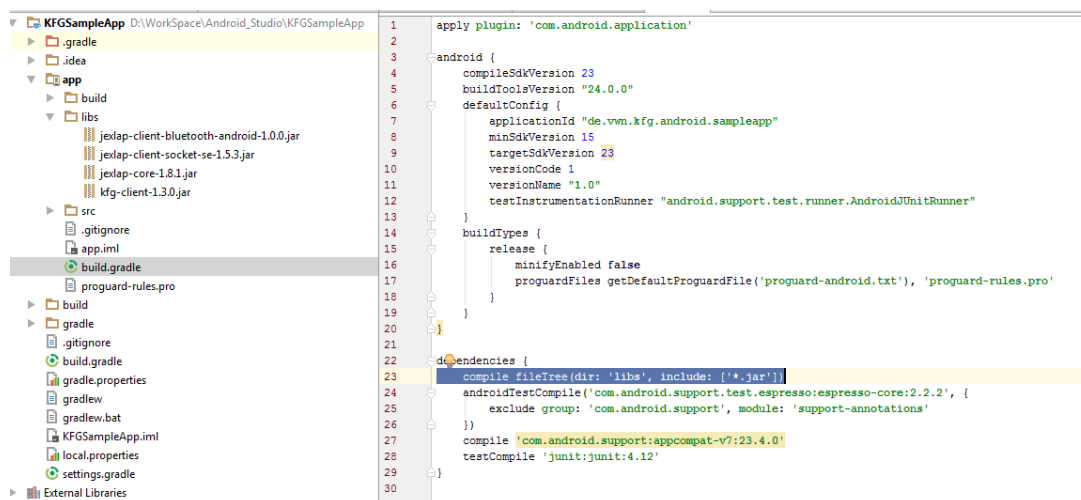


Abb. 5.14 Android Studio: adding KFG build.gradle dependency files

6.5 Setting up the data connection to the KFG

This section describes the different connection options between the KFG and the app. The interface library for the KFG app interface supports wireless and Bluetooth connections. As both wireless and Bluetooth connection models are implemented in the example app, the user must select a connection type before the connection to the KFG is established.

The sections below describe the app development using a single connection type, but both connection types can be used, as in the example app. Although the example code below is different from the example app, it offers app developers an overview of the app development options for different connection types.

6.5.1 App development for WLAN connection

This chapter offers useful information on the use of the interface library for the KFG app interface within the app for a wireless connection.

6.5.1.1 Setting up wireless

The following libraries are required to use the interface library for the KFG app interface.

- kfg-client-xxx.jar
- jexlap-client-socket-se-xxx.jar
- jexlap-core-xxx.jar

6.5.1.2 Setting up and using the KFG client (wireless)

The following describes basic interactions with the KFG client instance based on a complete Java example program from connection through to disconnection.

```
public class KFGSample implements ConnectionListener, KFGListener {
    private KFGClient client;
    public static void main(String[] args) throws IllegalArgumentException,
        ExlapException, IOException, InterruptedException {
        KFGSample kfgSample = new KFGSample ();
        Thread.sleep(20000);
        kfgSample.shutdown();
    }
    public KFGSample() {
        client = new KFGClient();
        client.addKFGListener(this);
        client.addConnectionListener(this);
        client.connect();// executes search and automatic connection
    }
    private void shutdown() {
        // closes all connections; KFG client instance can continue to be
        // used
        client.disconnect();
    }
}
```

After initialisation, the following callbacks are relevant for the example shown:

If the connection is successful (*ConnectionListener*), a subscription is created. Data reception (via the subscription or via a function call) is communicated by means of the relevant methods of the *KFGListener* instance.

```
// ConnectionListener callback
public void onConnectionSuccessful(Capabilities capabilities) {
    System.out.println("Connection successful");
    try {
        client.subscribeActualDistance(1000);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
... // Further ConnectionListenre callbacks follow
```

After the values for the subscription/the return value for the function have been received, the *KFGListener* instance is informed using the relevant methods. In the following example, only one method is called as only one object is subscribed.

```
// KFGListener callback (e.g. for ActualDistance)
@Override
public void onActualDistanceUpdate(final ActualDistance actualDistance) {
    System.out.println("actual distance: " +
        actualDistance.getActualDistance());
}
```

All other callbacks to be implemented would follow. We recommend using a *KFGListenerAdapter*. The *KFGListenerAdapter* includes the *KFGListener* instance so that all the listener callbacks do not need to be implemented. You can select the callbacks required and fill them with individual logic. The *KFGListenerAdapter* is part of the interface library for the KFG app interface.

If it is no longer required, the subscription can be removed:

```
client.unsubscribeActualDistance();
```

6.5.2 App development for Bluetooth connection

This chapter offers useful information on the use of the interface library for the KFG app interface within the app for a Bluetooth connection.

6.5.2.1 Setting up Bluetooth

The following libraries are required to use the interface library for the KFG app interface.

- kfg-client-xxx.jar
- jexlap-client- bluetooth-android-xxx.jar
- jexlap-core-xxx.jar

6.5.2.2 Setting up and using the KFG client (Bluetooth)

In order to use the KFGClient in your app via a Bluetooth connection, the following permissions need to be added to the Android Manifest file:

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />  
<uses-permission android:name="android.permission.BLUETOOTH" />
```

With a Bluetooth connection, the KFG is not automatically detected by and connected to the KFG Client. The app developer must provide the KFG Client with the address of his KFG Bluetooth device. In order to access the address of the KFG Bluetooth devices, you first need to pair your telephone to the KFG Bluetooth device. Proceed as follows to pair your telephone with the KFG Bluetooth device.

- Activate the Bluetooth function in the Android phone settings.
- Start scanning for Bluetooth devices.
- "KFG_Bluetooth" is shown among the devices found.
- Tap "KFG_Bluetooth" to establish a connection. The default PIN for pairing is 1234. If you changed the PIN when configuring the KFG, use the new PIN to pair the device.



The following explains basic interactions with a KFG Client via Bluetooth based on an example Java program:

```
public class KFGSample implements ConnectionListener, KFGListener {
    private KFGClient client;
    public static void main(String[] args) throws IllegalArgumentException,
        ExlapException, IOException, InterruptedException {
        KFGSample kfgSample = new
            KFGSample ();
        Thread.sleep(20000);
        kfgSample.shutdown();
    }
    public KFGSample() {
        String kfgDeviceConnectionURL = getKFGBluetoothDeviceConnectionURL();
        client = new KFGClient(kfgDeviceConnectionURL);
        client.addKFGListener(this);
        client.addConnectionListener(this);
        client.connect();// executes search and automatic connection
    }
    private String getKFGBluetoothDeviceConnectionURL() {
        // call address of KFG Bluetooth device and return connection URL.
        String connectionURL ="";
        BluetoothAdapter mBluetoothDriver = BluetoothAdapter.getDefaultAdapter();
        Set<BluetoothDevice> mPairedDevices = mBluetoothDriver.getBondedDevices();
        for(BluetoothDevice device: mPairedDevices) {
            if( device.getName().equals("KFG_Bluetooth")) {
                // btsp:// is added to the device address to call the connection URL for
                // the device
                connectionURL = "btsp://" + device.getAddress();
                return connectionURL;
            }
        }
    }
    private void shutdown() {
        // closes all connections; KFG client instance can continue to be
        // used
        client.disconnect();
    }
}
```

6.6 Functions

This section describes the most important functions of the example app. These are special functions which can be used to explain the use of the interface library for the KFG app interface. All app functions must be coupled with the relevant KFG function defined in the KFG configuration at all times. In order to test the different functions of the example app, the example KFG configuration needs to be added within the KFG.

The descriptions of the individual functions contain the relevant example code in order to give app developers an overview of how simpler functions could be implemented in the app. The code is not complete. But it does provide an overview of functions and gives information on where similar code could be found in the example app project.

6.6.1 Searching for KFG devices and displaying the devices found in the app

The app uses wireless or Bluetooth to connect in order to search for devices and display devices found in a list. The search for devices is started when the user presses the "Discovery" button. The code below shows how the search for devices can be initiated.

```
String connectionScheme = "Wifi"; // Wireless or Bluetooth
DiscoveryManagerAndroid discoveryManagerAndroid =
    new DiscoveryManagerAndroid (connectionScheme , getActivity());
.....
discoveryManagerAndroid.discoverServices(this, null, null, null, true);
```

As soon as a device is found (*discoveryEvent* for the *DiscoveryListener*), this is shown in the list of devices found and the result *discoveryFinished* is shown.

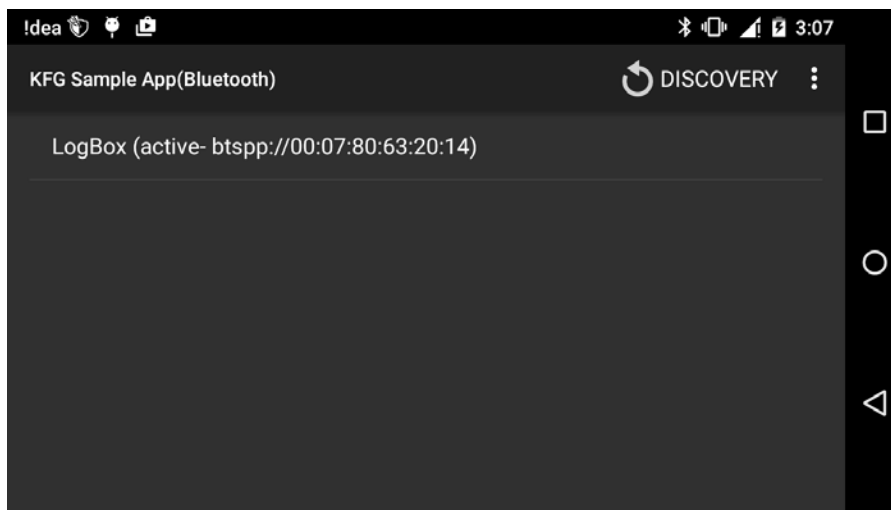


Abb. 5.15 The list of devices found is shown in the app

6.6.2 Displaying the device information and establishing a connection to the device from the app

The app displays information on the selected devices and the connection address.

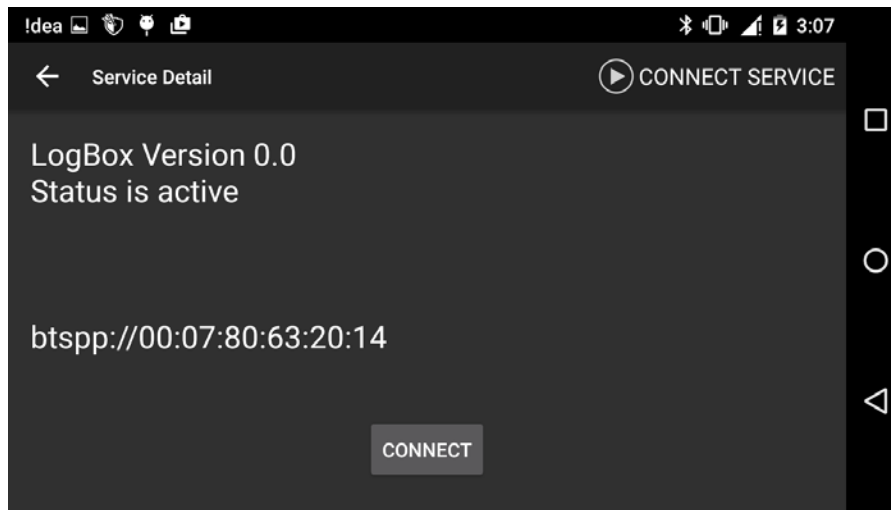


Abb. 5.16 The app displays details of the device

The code below makes it clear how the device address can be used to establish a connection to the device.

```
String connectionAddress = "btspp://00:07:80:63:20:14"; // Device address
// New instance of ConnectionManager to establish a connection to the existing device
// address
ConnectionManager connectionManager = new ConnectionManager(this, connectionAddress);
// Start new thread for calling connection function
new Thread(connectionManager).start();
// Connection function
private void connect() {
    boolean useKeepAlive = true;
    // Create KFGConfiguration
    KFGConfiguration kfgConfiguration = new KFGConfiguration(false, useKeepAlive);
    // Create KPGClient with connection address
    KFGClient kfgClient = new KFGClient(connectionAddress, kfgConfiguration);
    // Add data listener as ConnectionManager
    kfgClient.addDataListener(this);
    // Add connection listener as ConnectionManager
    kfgClient.addConnectionListener(this);
    // Add KFG Listener
    kfgClient.addKFGListener(new KFGListenerAdapter());
    // Call connection
    kfgClient.connect();
}
```

With a successful connection (*onConnectionSuccessful* event for *ConnectionListener*), the *KFGClient* can request data and initiate actions on the server.

6.6.3 Requesting the KFG identifier and testing for the expected unique number by the app

This function demonstrates how static data objects in the app are handled.

For static data objects, the value remains constant while the connection is established.

For example, the values of data objects such as *VehicleIdentificationNumber*, *VehicleColour*, *EnginePower*, *KFGIdentifier*, etc. remain constant while the connection is established.

After a successful connection with the KFG, the *onConnectionSuccessful* callback is called. The first role of the callback is therefore to call the value of the KFG identifier and compare it with the unique number expected. If the number received does not tally with the unique number expected or an error occurs when calling the value, the KFG connection is disconnected. This is made clear in the example code below (*ConnectionManager*).



```
// ConnectionListener callback
public void onConnectionSuccessful(Capabilities capabilities){
// On successful connection, call on KFG identifier
    getKFGIdentifier();
}
private void getKFGIdentifier() {
    KFGClient kfgClient = connectionManager.getKFGClient();
    try {
        //Calls getObject with URL for KFG identifier, ConnectionManager as
        AsyncCallback
        kfgClient.getObject(kfgClient.URL_VWN_FUNCTION_INPUT_SIGNAL_20,connectionManager);
        //If successful, onSuccess callback is called
    } catch (Exception e) {
        //If error message, connection cancelled and message shown
        connectionManager.abortWithInfo("Error in reading KFG Identifier
        message"+e.getMessage());
    }
// AsyncCallback onSuccess method
public void onSuccess(int id, Object object) {
    if (object instanceof DataObject) {
        final DataObject viewDataObject = (DataObject) object;
        // If object URL tallies with KFG identifier URL
        if(viewDataObject.getUrl().equals(kfgClient.URL_VWN_FUNCTION_INPUT_SIGNAL_20)) {
            compareKFGIdentifier(viewDataObject);
        }
    }
}
private void compareKFGIdentifier(DataObject dataObject) {
    // Expected KFG identifier
    double expectedKFGIdentifier = 123456;
    try {
        // Call object data element
        DataElement element = dataObject.getElement(0);
        if (element != null && element.hasValue()) {
            // Call value of data element
            double kfgIdentifier = element.getValueAsDouble();
            // Comparison of values
            if (expectedKFGIdentifier != kfgIdentifier) {
                // If they do not agree, disconnect connection and show a
                report
                connectionManager.abortWithInfo("KFG Identifier did not match");
            }
        }
        else {
            // Disconnection connection and show message
            connectionManager.abortWithInfo ("Error in reading KFG Identifier");
        }
    }
    catch (Exception ex) {
        // Disconnection connection and show message
        connectionManager.abortWithInfo("Error in comparing KFG Identifier " +
        ex.getMessage());
    }
}
```

6.6.4 Displaying the engine speed in the app

This function demonstrates how dynamic data objects in the app are handled.

For a dynamic data object, the value of the data object comes from a source which generates a continuous flow of sensor information (e.g. a sensor which measures the vehicle speed every 10 ms). When new sensor data is measured or generated, data updates are received.

Dynamic data objects include EngineSpeed, VehicleSpeed, GeoPosition, OilTemperature, etc.

The content of dynamic data objects changes very often and the app receives the relevant data frequently. The processing and handling of these values utilises a high level of resources. So it is important to define the minimum interval as well as the object URL when subscribing to the data object.

The code underneath demonstrates how the "DisplayEngineSpeed" is subscribed with an interval (*ExlapServiceActivity*) when a button is tapped.

```
// Button to access the engine speed
Button btnSubscribeDisplayEngineSpeed = (Button) findViewById
(R.id.btn_subscribeDisplayEngineSpeed);
// Adding a click listener to the button
btnSubscribeDisplayEngineSpeed.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Calling the KFGClient object
        KFGClient kfgClient = connectionManager.getKFGClient();
        // Display of a pop-up to add the interval for subscription of the
        // data object URL given
        DisplayObjectSubscribeDialog(kfgClient.URL_DISPLAYEDENGINESPEED);
    }
});
// Display of a pop-up to add the interval for subscription
private void DisplayObjectSubscribeDialog(String objectURL) {
    // Creating a new ObjectDialogfragment instance
    DialogFragment objectDialogFragment = new ObjectDialogFragment();
    // Bundle for setting the object URL
    Bundle args = new Bundle();
    args.putString("objectUrl", objectURL);
    objectDialogFragment.setArguments(args);
    // Display popup
    objectDialogFragment.show(getFragmentManager(), "object");
}
```

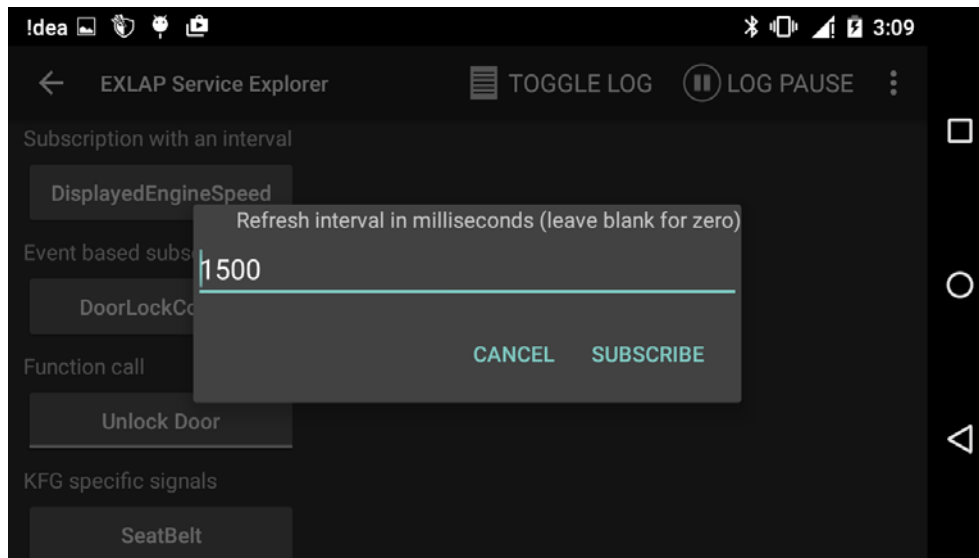



Abb. 5.17 The app shows the pop-up to enter the interval for the subscription
The code below explains how the data object is subscribed with an interval.

```
// Call KFGClient
KFGClient kfgClient = connectionManagerInterface.getConnectionManager().getKFGClient();
// Call URL of object to be subscribed
String objectUrl = kfgClient.URL_DISPLAYEDENGINESPEED;
// Subscription interval
int interval = 500; // Call value entered by user
// Check whether data object is an event or static
if (ExlapIntrospection.isEventOrStatic(objectUrl)) {
    // Subscription without interval
    kfgClient.subscribeObject(objectUrl);
} else {
    // Subscription with interval
    kfgClient.subscribeObject(objectUrl, interval);
}
```

Once data has been received *onData (ConnectionManager)*, the app shows the data object and the value.

```
public void onData(DataObject dataObject) {
    final DataObject viewDataObject = dataObject;
    activity.runOnUiThread(new Runnable() {
        public void run() {
            // Update process to update the object view
            activity.updateObjectsView(viewDataObject, false);
        }
    });
}
```

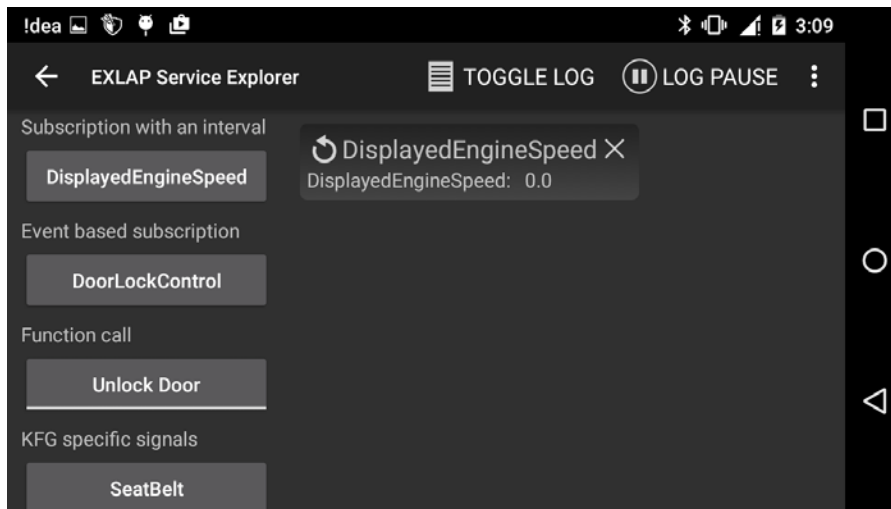


Abb. 5.18 The app shows the data object subscribed and its value

6.6.5 Displaying the door locking status in the app

This function demonstrates how event data objects in the app are handled.

The value of an event data object is updated when a specific event occurs.

This applies to data object such as DoorLockControl, DashboardIndicators, BrakeOverheating, etc.

The code underneath demonstrates how the "DoorLockControl" is subscribed with an interval (*ExlapServiceActivity*) when a button is tapped.

```
// Calling the door lock button
Button btnSubscribeDoorLock = (Button) findViewById(R.id.btn_subscribeDoorLock);
// Adding a click listener to the button
btnSubscribeDoorLock.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Calling KFG client
        KFGClient kfgClient = connectionManager.getKFGClient();
        // Calling the object URL to lock the door
        String doorLockControlURL = kfgClient.URL_DOORLOCKCONTROL;
        try {
            // Subscribing to object with URL
            kfgClient.subscribeObject(doorLockControlURL);
        }
        catch (IOException e) {
            Log.e(TAG, "Could not subscribe to " + doorLockControlURL + ": " +
                e.getMessage());
        }
    }
});
```

Once data has been received *onData* (*ConnectionManager*), the app shows the data object and the value, as described above.

6.6.6 Locking and unlocking the doors using the app

This function is used to demonstrate how a KFG function is called by the app and how values are set for data objects.

For example, the locking and unlocking of the door is activated by the URL of the call function "VWN_Function_Call_Value_02" and the value is set to the input parameter "CM_EXLAP_Input_Value_02".

The KFG function call changes the value of vehicle CAN bus signals directly. If incorrect values are set, this leads to malfunctions in the vehicle. So you need to reset the value of the function call immediately with a delay of 1 second.

If you press the "Lock door" button, the app sets the value of the data object to "0x05" and then back to "0x00" with a delay of 1 second.

The same applies if the "Unlock door" button is pressed, the app sets the value of the data object to "0x09" and then back to "0x00" with a delay of 1 second.

To display the result of the function call, first subscribe to "DoorLockControl" as described in the section above.

The code makes it clear how the value of the function call (*ExlapServiceActivity*) is set using the *FunctionCallHandler*.

```
//Switches for locking and unlocking the door
ToggleButton doorLock = (ToggleButton) findViewById(R.id.btn_doorlock);
// Adding a check for modified listeners to the diverter
doorLock.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        // Calling KFG client
        KFGClient kfgClient = connectionManager.getKFGClient();
        if (isChecked) {
            // To lock the door, call function with value of 5
            setFunctionCallValue(kfgClient.URL_VWN_FUNCTION_CALL_VALUE_02,5);
        }
        else {
            // To lock the door, call function with value of 9
            setFunctionCallValue(kfgClient.URL_VWN_FUNCTION_CALL_VALUE_02,9);
        }
    }
});
// This function creates the FunctionCallHandler and sets the value
private void setFunctionCallValue(String functionURL, double value) {
    // Creating the FunctionCallHandler instance
    FunctionCallHandler functionCallHandler = new FunctionCallHandler(this);
    // Setting the value to the function URL
    functionCallHandler.setFunctionCallValue(functionURL,value);
}
```

The code below demonstrates how the value of the function data object is amended and how it is reset to "0" with a delay of 1 second (*FunctionCallHandler*).



```
// Calling KFG client
KFGClient kfgClient = getConnectionManager().getKFGClient();
// Call function interface of function URL
FunctionInterface functionInterface=
(FunctionInterface)functionInterfaces.get(functionUrl);
// Call Exlap type for function interface
ExlapType exlapType = functionInterface.getIn(0);
// Creating the input parameter object
DataObject callParameter = new DataObject();
// Creating the data element with Exlap type and input value for function
DataElement dataElement = new DataElement(exlapType.getName(),value,exlapType.getType());
// Adding the data element to the input parameter
callParameter.addElement(dataElement);
// Calling the function method with function URL and input parameter
ExlapIntrospection.callFunctionMethod(kfgClient, functionUrl, callParameter);
// Adding a delay of 1 second and resetting the value to "0"
Thread.sleep(1000);
// Creating the input parameter object to reset the value
DataObject resetCallParameter = new DataObject();
// Creating the data element with a value of "0"
DataElement resetDataElement = new DataElement(exlapType.getName(), 0,
exlapType.getType());
//Adding the data element for the input parameter for resetting
resetCallParameter.addElement(resetDataElement);
// Calling the function method with function URL and input parameters for resetting
ExlapIntrospection.callFunctionMethod(kfgClient, functionUrl, resetCallParameter);
```



6.6.7 Displaying the number of fastened seatbelts in the app

This function demonstrates how KFG-specific data objects in the app are handled.

For example, when calling the number of fastened seatbelts using the function URL "VWN_Function_Input_Signal_03". You need to subscribe to the data object using the function URL.

The code below demonstrates how the "Seatbelt" data object is subscribed via the function URL when a button is tapped.

```
// Calling seatbelt status
Button btnSeatBelt = (Button) findViewById(R.id.btn_seatBelt);
// Adding a click listener to the button
btnSeatBelt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Calling KFG client
        KFGClient kfgClient = connectionManager.getKFGClient();
        // Calling the URL for seatbelt
        String url_VWN_FUNCTION_INPUT_SIGNAL_03 = kfgClient.URL_VWN_FUNCTION_INPUT_SIGNAL_03;
        try {
            // Subscribing to object URL
            kfgClient.subscribeObject(url_VWN_FUNCTION_INPUT_SIGNAL_03);
            ...
        }
        catch (IOException e) {
            Log.e(TAG, "Could not subscribe to " + url_VWN_FUNCTION_INPUT_SIGNAL_03 + ": " +
                e.getMessage());
        }
        ...
    }
});
```

Once data has been received *onData* (*ConnectionManager*), the app shows the data object and the value, as described above.

6.7 Overview of classes

The diagram below shows the most important activities and classes in the example app.

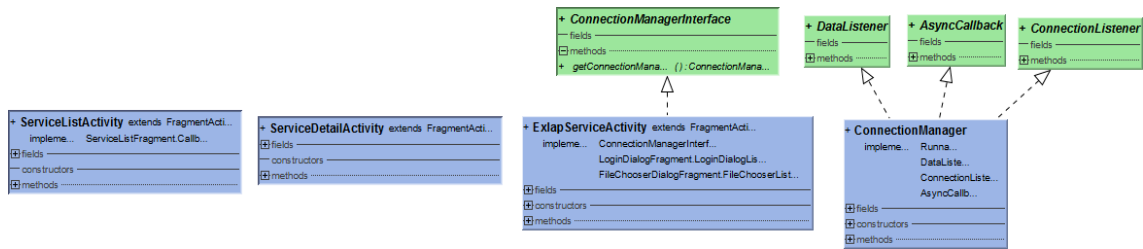


Abb. 5.19 Activities and classes in the example app.