www.ecumaster.com /ecumaster_official /ecumaster

USER MANUAL

PMU-16/PMU-16DL/PMU-16AS/PMU-24DL

Document version 101.1.3
Software version: 101.1 or later
Published on: 17 April 2025

ECU MASTER

Contents

1. Warnings

Attention:

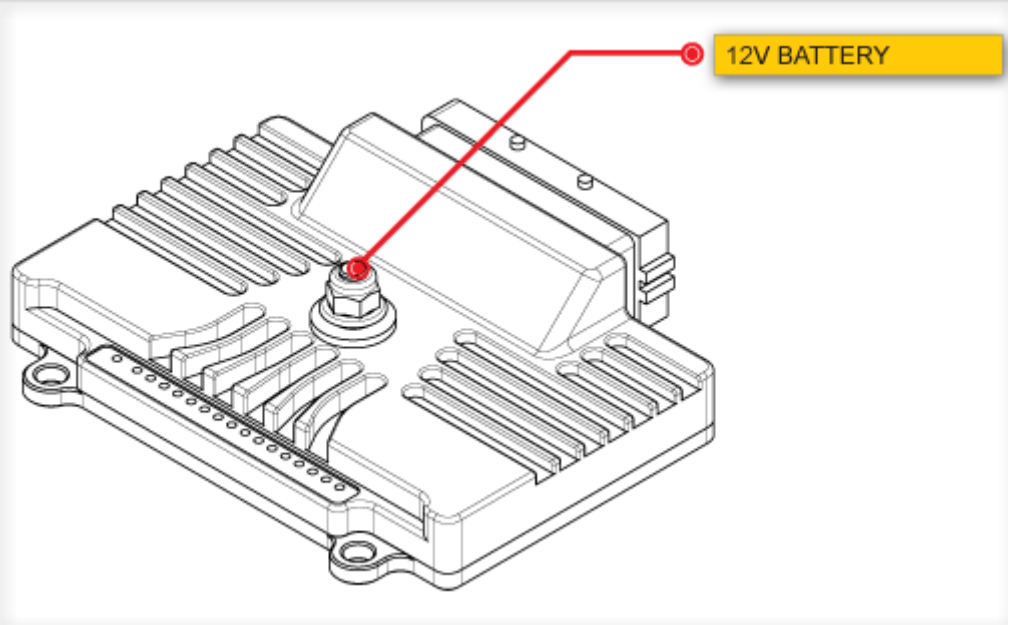
- **The Ecumaster PMU is designed for motorsport applications only and cannot be used on public roads!**
- **The installation of this device should be performed only by trained specialists. Installation by untrained individuals may cause damage to both the device and the vehicle!**
- **Incorrect configuration of the Ecumaster PMU can cause serious damage to vehicle components!**
- **Never modify the device's settings while the vehicle is moving as it may cause an accident!**
- **Ecumaster assumes no responsibility for damage caused by incorrect installation and/or configuration of the device!**
- **To ensure proper use of the Ecumaster PMU and to prevent risk of damage to your vehicle, you must read these instructions and understand them thoroughly before attempting to install this unit.**
- **Never short-circuit the wires of the vehicle's wiring loom or the outputs of the Ecumaster PMU!**
- **All modifications to the vehicle's wiring loom must be performed with the negative terminal of the battery disconnected.**
- **It is critical that all connections in the wiring loom are properly insulated!**
- **The device must be disconnected before performing any welding on the vehicle!**

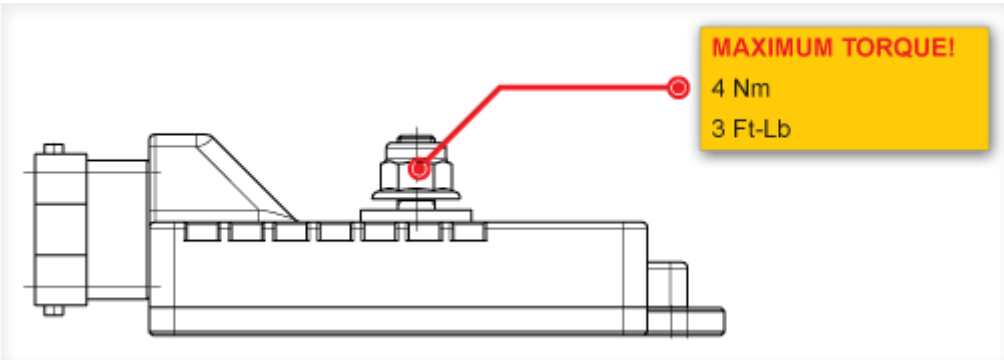
2. Ecumaster PMU

Ecumaster PMU is an intelligent power management unit designed to replace the old, traditional and often unreliable fuses and relays. The PMU is not only an electronic switch, but a device that can perform all kinds of advanced operations and log its parameters (applies to the version of the device with built-in memory). All output status, voltage and current information can be logged and sent over the CAN bus to dashboards to alert users to potential problems.

The advantage of the PMU system is the ability to define the logic of controlling the outputs. This is done using data from analog input channels or CAN bus data, and mathematical operations can be defined for each channel. The PMU allows you to power up external devices such as fans, blinkers, wipers, oil pump etc. and create advanced strategies for those devices using logic with fail-safes, condition checking and many more functions.

It can communicate and work in tandem with other Ecumaster CAN devices. The PMU is equipped with over and under-current protection, surge protection, 3D gyroscope, accelerometer, LED Status lights, Soft Start, Pulse Width Modulation with Duty Cycle control and more.





2.1. Specification

PMU 16, PMU 16DL, PMU 24DL

General	
Operating temperature range	ACEQ100 GRADE1 (-40 – 125 C)
CPU	32 bits automotive, 90 MIPS
Reverse polarity protection	Yes, internal up to 16 V
Operating voltage	6-22 V immunity to transients according to ISO 7637
Enclosure	bespoke CNC machined aluminum
Water and dust resistant	IP 60
Connectors	1 x 39 Automotive connector 1 x M6 stud for battery connection
PC communication	CAN with the interface Ecumaster, Peak PCAN-USB, Kvaser
Size and weight	131x112x32,5 mm, 345 g
Multiple PMUs	Up to 5 PMUs can work in tandem
Outputs	
10 x high-side up to 25 A output	Overcurrent and overheating protection. Outputs may be paired to increase continuous current capability. Current and voltages measured for each output. Inductive load clamp: Vbat – 36 V Inductive energy dissipation Ear: 460 mJ Peak current: 120 A
6 x high-side up to 15 A output	Overcurrent and overheating protection. Outputs may be paired to increase continuous current capability. Current and voltages measured for each output Inductive load clamp with diode to: -0.5 V Peak current: 120 A
8 x high-side up to 7 A output	Outputs sharing terminals with analog inputs (12 bit, 0-12 V)
Total current output	150 A continuous (for PMU 16), 170 A continuous (for PMU 24)
Output current control step	100 mA

PWM	Yes, available for each 25 A output Programmable variable Duty Cycle control for each output Separate frequency setting ranging from 4 Hz to 400 Hz for each output
Soft Start	Yes, available for each 25 A output
Low-side wipers braking output	Dedicated output with wiper braking feature. Pin shared with output O8 Current limit: 6 A
+5V power sources	Monitored 5 V, 500 mA output for powering external sensors
Inputs	
Analog Inputs	**PMU 16 and PMU 16DL**: 16 inputs, 10 bit resolution, 0-5 V (protected), with software selectable 10K ohm pullups / pulldowns **PMU 24DL**: 8 inputs with 10 bit resolution, 0-5 V (protected), 8 inputs with 12 bit resolution, 0-20 V (protected) - sharing terminals with outputs (7 A), with software selectable 10K ohm pullups / pulldowns
CAN keypads	2 x Ecumaster keypads (4, 6, 8, 12 keys), LifeRacing PDU Keypad , 1 x MoTeC/RaceGrade, 1 x Grayhill
CAN BUS	
CAN interface	2 x CAN2.0 A/B
CAN standard	2.0A/B 125, 250, 500 (default for CAN2), 1000 kbps (default for CAN1)
CAN termination	CAN1 - none, CAN2 - software-selectable
Input/Output stream	User defined with bit masking Up to 100 input messages
Other	
Output state indication	16 bicolor LEDs
Accelerometer / Gyroscope	3D accelerometer with 3D gyroscope for logging and crash detection
Real Time Clock	Yes, super capacitor for backup power (up to 3 days)

Temperature sensor	Yes, device temperature monitoring
Logging (PMU 16DL, PMU 24DL only)	
Logging Memory	256 MB
Logging Speed	Variable, defined per channel, up to 500 Hz
PC Logging	
Logging Speed	Variable, defined per channel, up to 500 Hz
Function	
Logical Operations	IsTrue, is False, =, ≠, <, ≤, >, ≥, And, Or, Xor, Flash, Pulse, Toggle, Set/Reset Latch, Hysteresis, Changed
Number of functions	100
Number of operations	250
Update frequency	500 Hz
Special functions	Wipers, Blinkers

PMU 16 Auto Sport

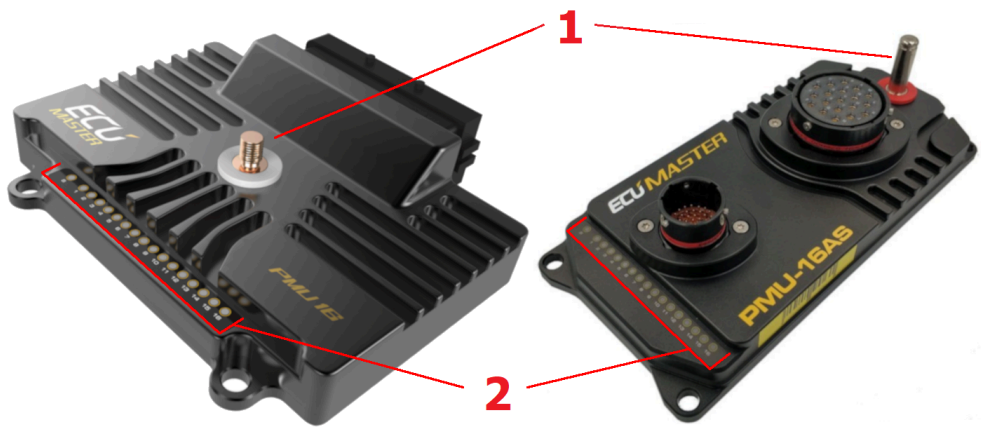
General	
Operating temperature range	ACEQ100 GRADE1 (-40 – 125 C)
CPU	32 bits automotive, 90 MIPS
Reverse polarity protection	Yes, internal up to 16 V
Operating voltage	6-22 V immunity to transients according to ISO 7637
Enclosure	bespoke CNC machined aluminum
Water and dust resistant	IP 65
Connectors	1 x AS shell size 24, 19 x 12AWG 1 x AS shell size 14, 37 x 22AWG 1 x RADLOCK 8mm battery terminal
PC communication	CAN with the interface Ecumaster, Peak PCAN-USB, Kvaser
Size and weight	177x84x46,1 mm, 360 g
Multiple PMUs	Up to 5 PMUs can work in tandem
Outputs	
14 x high-side up to 25 A output	Overcurrent and overheating protection.

	Outputs may be paired to increase continuous current capability. Current and voltages measured for each output. Inductive load clamp: Vbat – 36 V Inductive energy dissipation Ear: 460 mJ Peak current: 120 A
2 x high-side up to 40 A output	Overcurrent and overheating protection. Outputs can be used in parallel to increase current capacity. By pairing both O4 pins (K and U) or both O12 pins (D and R) you get 40 A. Linking the K, U, D and R terminals gives you 80 A. Current and voltages measured for each output.
6 x low-side up to 1 A output	Outputs shorted to ground with thermal overload switch Peak current: 13 A
Total current output	200 A continuous
Output current control step	100 mA
Low-side wipers braking output	Dedicated output with wiper braking feature. Pin shared with output O8 Current limit: 6 A
+5V power sources	Monitored 5 V, 500 mA output for powering external sensors
PWM	Programmable variable Duty Cycle control for each output Separate frequency setting ranging from 4 Hz to 400 Hz for each output
Soft Start	Yes, available for any high-side output
Inputs	
Analog Inputs	16 inputs, 10 bit resolution, 0-5V (protected), with software selectable 10K ohm pullups / pulldowns
CAN keypads	2 x Ecumaster keypads (4, 6, 8, 12 keys), LifeRacing PDU Keypad , 1 x MoTeC/RaceGrade, 1 x Grayhill
CAN BUS	
CAN interface	2 x CAN2.0 A/B
CAN standard	2.0A/B 125, 250, 500 (default for CAN2), 1000 kbps (default for CAN1)
CAN termination	CAN1 - none, CAN2 - software-selectable

Input/Output Stream	User defined with bit masking Up to 100 input messages
Other	
Output state indication	16 bicolor LEDs
Accelerometer / Gyroscope	3D accelerometer with 3D gyroscope for logging and crash detection
Real Time Clock	Yes, super capacitor for backup power (up to 3 days)
Temperature sensor	Yes, device temperature monitoring
Logging	
Logging Memory	256 MB (3 hours of logging at max bandwidth)
Logging Speed	Variable, defined per channel, up to 500 Hz
PC Logging	
Logging Speed	Variable, defined per channel, up to 500 Hz
Function	
Logical Operations	IsTrue, is False, =, ≠, <, ≤, >, ≥, And, Or, Xor, Flash, Pulse, Toggle, Set/Reset Latch, Hysteresis, Changed
Number of functions	100
Number of operations	250
Update frequency	500 Hz
Special functions	Wipers, Blinkers

2.2. Device description

Front view

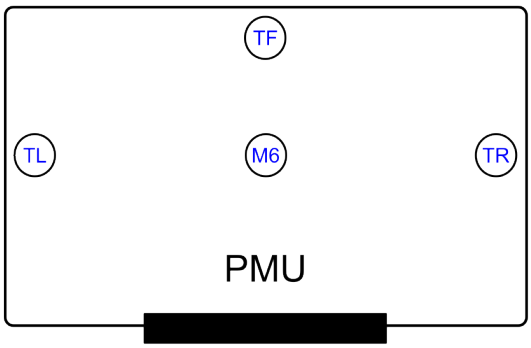


<table>
<tr><th>No</th><th>Function</th><th>Description</th></tr>
<tr><td>1</td><td>Main battery feed +12V</td><td>Main power supply from +12 V battery</td></tr>
<tr><td>2</td><td>Status LED:</td><td>
S – PMU status LED
<table>
<tr><th>Color</th><th>Description</th></tr>
<tr><td>Orange Continuous</td><td>Device is active and connected to PC</td></tr>
<tr><td>Green Continuous</td><td>Device is active (no PC connection)</td></tr>
<tr><td>Green Flashing Slowly</td><td>Device is waiting for Firmware Upgrade</td></tr>
<tr><td>Orange Flashing Slowly</td><td>Device is performing Firmware Upgrade and is connected to the PMU Client</td></tr>
<tr><td>Orange Flashing Fast</td><td>Device is performing Make Permanent operation</td></tr>
<tr><td>Red Continuous</td><td>Device Error - please contact the distributor or manufacturer directly. Be cautious, as the orange and red status colors may be confused easily. To distinguish, unplug the PMU from the PC. If the diode remains red, it indicates a device error.</td></tr>
</table>
#1 - #16 (for PMU 16) – Output status LED
#1 - #24 (for PMU 24)
<table>
<tr><th>Color</th><th>Description</th></tr>
<tr><td>Green</td><td>Output status: ACTIVE</td></tr>
<tr><td>Orange</td><td>Output status: UNDERCURRENT (the output is active, but does not consume current, or consumes less than the declared minimum current)</td></tr>
<tr><td>Red</td><td>Output status: OVERCURRENT (output is tripped-off by software)</td></tr>
</table>
</td></tr>
</table>

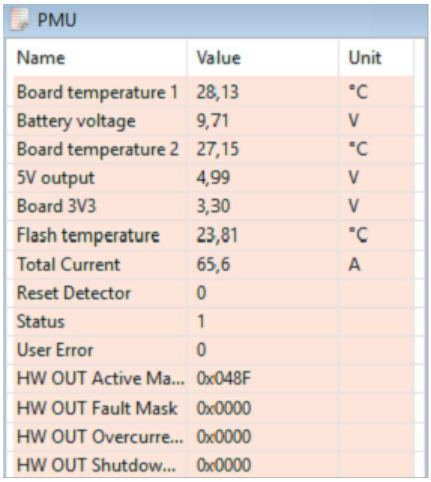
2.3. PMU Temperature

PMU should be located in a place that protects it from weather conditions and other external factors. It is also recommended to place the PMU where heat can be easily dissipated, preferably with good airflow.

The PMU device has three sensors on its board for independent temperature measurement in three separate places.



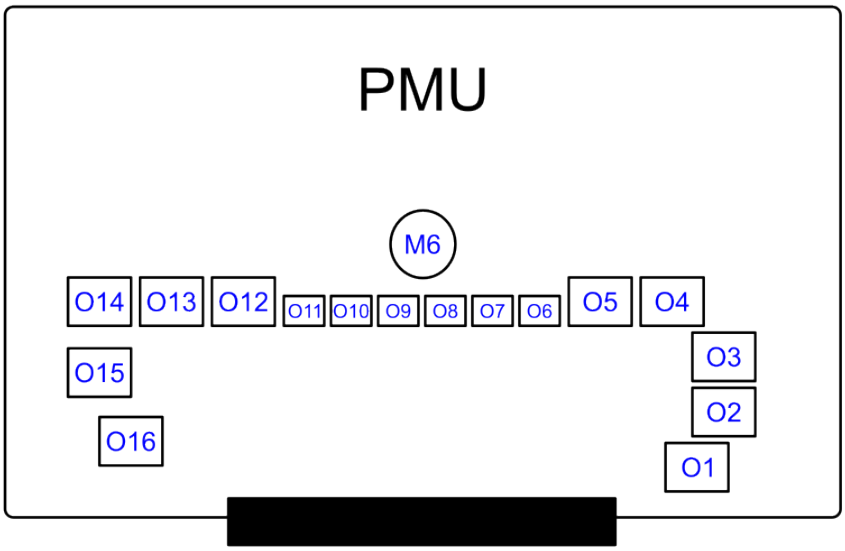
This temperature is displayed on the status bar in PMU Client (TL - Temperature Left, TR - Temperature Right, TF - Temperature Flash) and the PMU panel displaying device parameters logged in real-time.



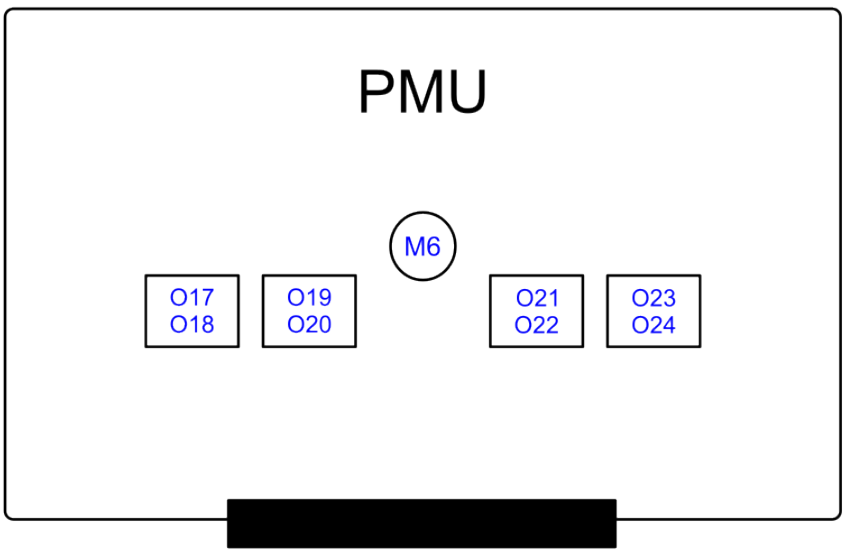
If the PMU experiences excessively high temperatures, it is advised to move it to a colder location or provide better airflow to its current location.

To mitigate high temperatures, connect devices with high current draw to output pins with spaced transistors. This improves heat dissipation, lowering temperatures.

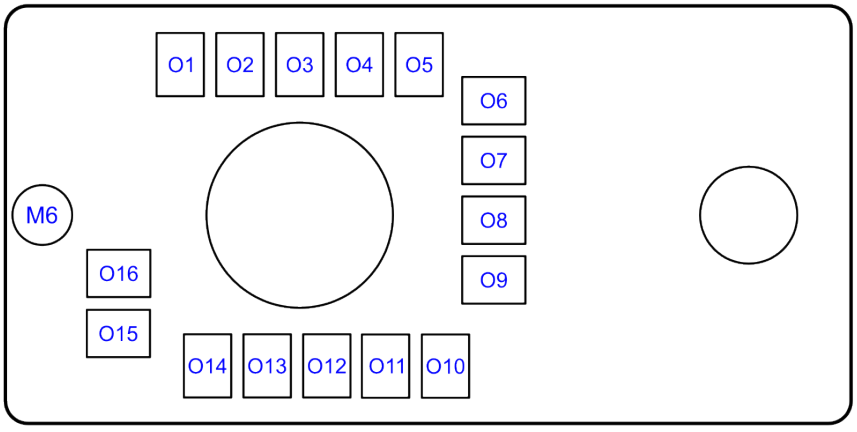
The picture below shows the arrangement of transistors in PMU 16 and PMU 24 for each output pin (top view):



For PMU 24, an additional arrangement of transistors from the bottom:



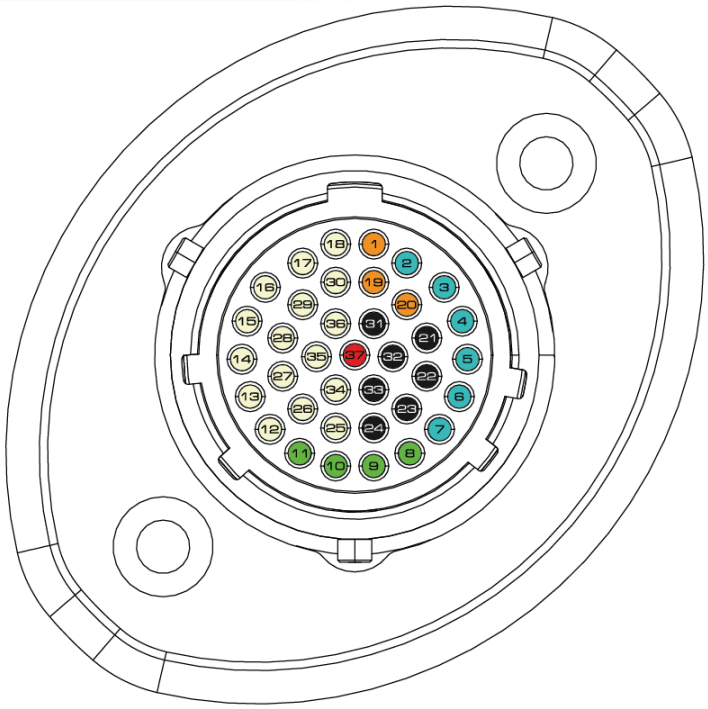
Arrangement of transistors in PMU 16AS:



2.4. Description of the PMU 16AS socket

PINOUT DESCRIPTION:

PMU-16AS



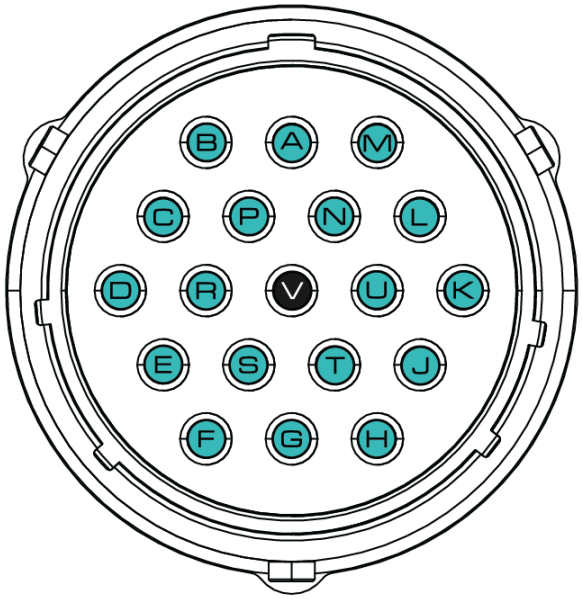
1	+5V OUTPUT
2	LOW-SIDE OUTPUT **L1**
3	LOW-SIDE OUTPUT **L2**
4	LOW-SIDE OUTPUT **L3**
5	LOW-SIDE OUTPUT **L4**
6	LOW-SIDE OUTPUT **L5**
7	LOW-SIDE OUTPUT **L6**
8	CAN2H
9	CAN2L
10	CAN1H
11	CAN1L
12	INPUT **A15**
13	INPUT **A12**
14	INPUT **A9**
15	INPUT **A6**
16	INPUT **A4**
17	INPUT **A2**
18	INPUT **A1**
19	+5V OUTPUT
20	+5V OUTPUT
21	SENSOR GROUND
22	SENSOR GROUND
23	SENSOR GROUND
24	SENSOR GROUND
25	INPUT **A16**
26	INPUT **A14**
27	INPUT **A11**
28	INPUT **A8**
29	INPUT **A5**
30	INPUT **A3**
31	SENSOR GROUND
32	SENSOR GROUND
33	SENSOR GROUND
34	INPUT **A13**
35	INPUT **A10**
36	INPUT **A7**
37	+12V SW

Terminal	Description
1. +5V OUTPUT	+5 V power supply for external sensors. Maximum current consumption 400 mA
2. LOW-SIDE OUTPUT L1	Low-side output 1, maximum current 1 A
3. LOW-SIDE OUTPUT L2	Low-side output 2, maximum current 1 A
4. LOW-SIDE OUTPUT L3	Low-side output 3, maximum current 1 A
5. LOW-SIDE OUTPUT L4	Low-side output 4, maximum current 1 A
6. LOW-SIDE OUTPUT L5	Low-side output 5, maximum current 1 A
7. LOW-SIDE OUTPUT L6	Low-side output 6, maximum current 1 A
8. CAN2.H	CAN H signal for CAN bus 2
9. CAN2.L	CAN L signal for CAN bus 2
10. CAN1.H	CAN H signal for CAN bus 1
11. CAN1.L	CAN L signal for CAN bus 1
12. INPUT A15	Analog input 15
13. INPUT A12	Analog input 12
14. INPUT A9	Analog input 9
15. INPUT A6	Analog input 6
16. INPUT A4	Analog input 4
17. INPUT A2	Analog input 2
18. INPUT A1	Analog input 1
19. +5V OUTPUT	+5 V power supply for external sensors. Maximum current consumption 400 mA
20. +5V OUTPUT	+5 V power supply for external sensors. Maximum current consumption 400 mA
21. SENSOR GROUND	Analog input ground
22. SENSOR GROUND	Analog input ground
23. SENSOR GROUND	Analog input ground
24. SENSOR GROUND	Analog input ground
25. INPUT A16	Analog input 16
26. INPUT A14	Analog input 14
27. INPUT A11	Analog input 11

Terminal	Description
28. INPUT A8	Analog input 8
29. INPUT A5	Analog input 5
30. INPUT A3	Analog input 3
31. SENSOR GROUND	Analog input ground
32. SENSOR GROUND	Analog input ground
33. SENSOR GROUND	Analog input ground
34. INPUT A13	Analog input 13
35. INPUT A10	Analog input 10
36. INPUT A7	Analog input 7
37. +12V SW	+12 V input to switch the PMU on or off

PINOUT DESCRIPTION:

PMU-16AS



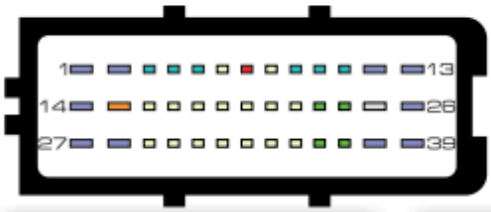
A	OUTPUT **O16**
B	OUTPUT **O15**
C	OUTPUT **O13**
D	OUTPUT **O12** [2)]
E	OUTPUT **O11**
F	OUTPUT **O9**
G	OUTPUT **O8**
H	OUTPUT **O6**
J	OUTPUT **O5**
K	OUTPUT **O4** [1)]

L	OUTPUT **O3**
M	OUTPUT **O2**
N	OUTPUT **O1**
P	OUTPUT **O14**
R	OUTPUT **O12** [2)]
S	OUTPUT **O10**
T	OUTPUT **O7**
U	OUTPUT **O4** [1)]
V	GROUND

[1)] Both O4 terminals must be connected for 40A output. Single terminal is capable of 25A.
[2)] Both O12 terminals must be connected for 40A output. Single terminal is capable of 25A.
[1), 2)] Terminals D, R, K and U must all be connected for 80A output. Double O4 and O12 in PMU Client for 80A.

1. In the case of current consumption above 20 A, **it is necessary** to connect the terminals together:
    - K and U (for O4 output)
    - D and R (for O12 output)
2. Linking the K, U, D, and R terminals gives 80 A.
3. Output O8 is dedicated to wipers.

2.5. Description of the PMU 16 socket

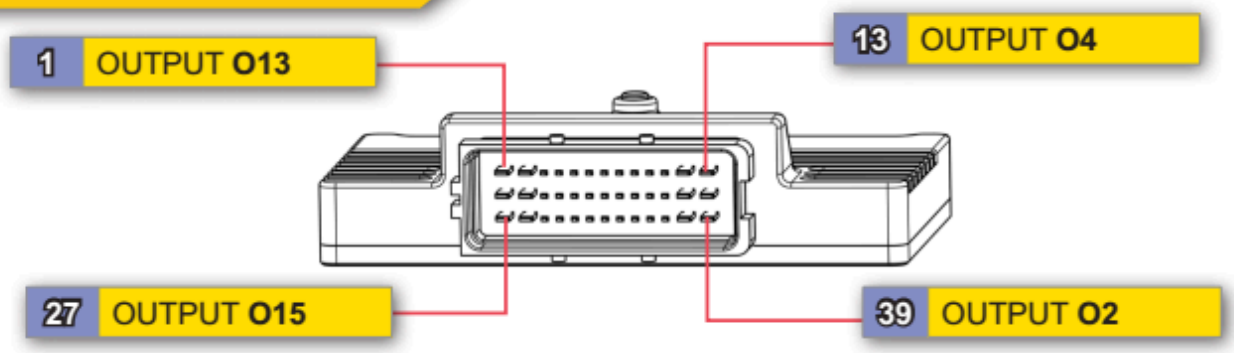


PMU-16
POWER MANAGEMENT UNIT

1	OUTPUT **O13**	14	OUTPUT **O14**	27	OUTPUT **O15**
2	OUTPUT **O12**	15	+5V OUTPUT	28	OUTPUT **O16**
3	OUTPUT **O11**	16	INPUT **A2**	29	INPUT **A1**
4	OUTPUT **O10**	17	INPUT **A4**	30	INPUT **A3**
5	OUTPUT **O9**	18	INPUT **A6**	31	INPUT **A5**
6	INPUT **A9**	19	INPUT **A8**	32	INPUT **A7**
7	+12V SW	20	INPUT **A11**	33	INPUT **A10**
8	INPUT **A14**	21	INPUT **A13**	34	INPUT **A12**
9	OUTPUT **O8**	22	INPUT **A16**	35	INPUT **A15**
10	OUTPUT **O7**	23	CAN1H	36	CAN1L
11	OUTPUT **O6**	24	CAN2H	37	CAN2L
12	OUTPUT **O5**	25	GROUND	38	OUTPUT **O1**
13	OUTPUT **O4**	26	OUTPUT **O3**	39	OUTPUT **O2**

25A	15A	**O8** - Wipers output

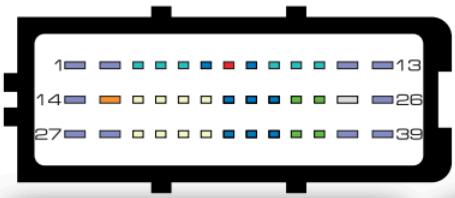
HOW TO READ:



Terminal	Description
1. OUTPUT O13	Output 13, maximum current 25 A
2. OUTPUT O12	Output 12, maximum current 25 A
3. OUTPUT O11	Output 11, maximum current 15 A
4. OUTPUT O10	Output 10, maximum current 15 A
5. OUTPUT O9	Output 9, maximum current 15 A
6. INPUT A9	Analog input 9
7. +12V SW	+12 V input to switch the PMU on or off
8. INPUT A14	Analog input 14
9. OUTPUT O8	Output 8 (dedicated to wipers), maximum current 15 A
10. OUTPUT O7	Output 7, maximum current 15 A
11. OUTPUT O6	Output 6, maximum current 15 A
12. OUTPUT O5	Output 5, maximum current 25 A
13. OUTPUT O4	Output 4, maximum current 25 A
14. OUTPUT O14	Output 14, maximum current 25 A
15. +5V OUTPUT	+5 V power supply for external sensors. Maximum current consumption 400 mA
16. INPUT A2	Analog input 2
17. INPUT A4	Analog input 4
18. INPUT A6	Analog input 6
19. INPUT A8	Analog input 8
20. INPUT A11	Analog input 11
21. INPUT A13	Analog input 13
22. INPUT A16	Analog input 16
23. CAN1.H	CAN H signal for CAN bus 1
24. CAN2.H	CAN H signal for CAN bus 2
25. GROUND	Device ground
26. OUTPUT O3	Output 3, maximum current 25 A
27. OUTPUT O15	Output 15, maximum current 25 A
28. OUTPUT O16	Output 16, maximum current 25 A

Terminal	Description
29. INPUT A1	Analog input 1
30. INPUT A3	Analog input 3
31. INPUT A5	Analog input 5
32. INPUT A7	Analog input 7
33. INPUT A10	Analog input 10
34. INPUT A12	Analog input 12
35. INPUT A15	Analog input 15
36. CAN1.L	CAN L signal for CAN bus 1
37. CAN2.L	CAN L signal for CAN bus 2
38. OUTPUT O1	Output 1, maximum current 25 A
39. OUTPUT O2	Output 2, maximum current 25 A

2.6. Description of the PMU 24 socket

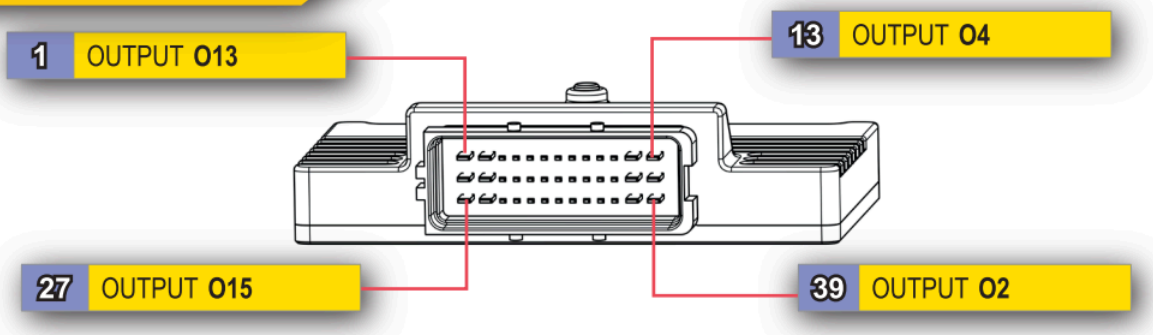


PMU-24 DL
POWER MANAGEMENT UNIT

1	OUTPUT **O13**	14	OUTPUT **O14**	27	OUTPUT **O15**
2	OUTPUT **O12**	15	+5V OUTPUT	28	OUTPUT **O16** (high power flyback)
3	OUTPUT **O11**	16	INPUT **A2**	29	INPUT **A1**
4	OUTPUT **O10**	17	INPUT **A4**	30	INPUT **A3**
5	OUTPUT **O9**	18	INPUT **A6**	31	INPUT **A5**
6	INPUT **A9** / OUTPUT **O17**	19	INPUT **A8**	32	INPUT **A7**
7	+12V SW	20	INPUT **A11** / OUTPUT **O19**	33	INPUT **A10** / OUTPUT **O18**
8	INPUT **A14** / OUTPUT **O22**	21	INPUT **A13** / OUTPUT **O21**	34	INPUT **A12** / OUTPUT **O20**
9	OUTPUT **O8** (wipers)	22	INPUT **A16** / OUTPUT **O24**	35	INPUT **A15** / OUTPUT **O23**
10	OUTPUT **O7**	23	CAN1H	36	CAN1L
11	OUTPUT **O6**	24	CAN2H	37	CAN2L
12	OUTPUT **O5**	25	GROUND	38	OUTPUT **O1** (high power flyback)
13	OUTPUT **O4**	26	OUTPUT **O3**	39	OUTPUT **O2**

	25A		15A		7A

HOW TO READ:



Terminal	Description
1. OUTPUT O13	Output 13, maximum current 25 A
2. OUTPUT O12	Output 12, maximum current 25 A
3. OUTPUT O11	Output 11, maximum current 15 A
4. OUTPUT O10	Output 10, maximum current 15 A
5. OUTPUT O9	Output 9, maximum current 15 A
6. INPUT A9 / OUTPUT O17	Analog input 9 and Output 17 share the same terminal, with a maximum current of 7 A.
7. +12V SW	+12 V input to switch the PMU on or off
8. INPUT A14 / OUTPUT O22	Analog input 14 and Output 22 share the same terminal, with a maximum current of 7 A.
9. OUTPUT O8	Output 8 (dedicated to wipers), maximum current 15 A
10. OUTPUT O7	Output 7, maximum current 15 A
11. OUTPUT O6	Output 6, maximum current 15 A
12. OUTPUT O5	Output 5, maximum current 25 A
13. OUTPUT O4	Output 4, maximum current 25 A
14. OUTPUT O14	Output 14, maximum current 25 A
15. +5V OUTPUT	+5 V power supply for external sensors. Maximum current consumption 400 mA
16. INPUT A2	Analog input 2
17. INPUT A4	Analog input 4
18. INPUT A6	Analog input 6
19. INPUT A8	Analog input 8
20. INPUT A11 / OUTPUT O19	Analog input 11 and Output 19 share the same terminal, with a maximum current of 7 A.
21. INPUT A13 / OUTPUT O21	Analog input 13 and Output 21 share the same terminal, with a maximum current of 7 A.
22. INPUT A16 / OUTPUT O24	Analog input 16 and Output 24 share the same terminal, with a maximum current of 7 A.
23. CAN1.H	CAN H signal for CAN bus 1
24. CAN2.H	CAN H signal for CAN bus 2

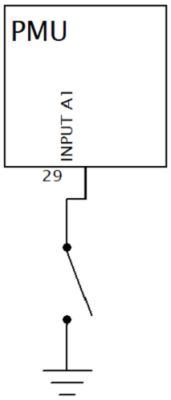
Terminal	Description
25. GROUND	Device ground
26. OUTPUT O3	Output 3, maximum current 25 A
27. OUTPUT O15	Output 15, maximum current 25 A
28. OUTPUT O16	Output 16, maximum current 25 A, high power flyback
29. INPUT A1	Analog input 1
30. INPUT A3	Analog input 3
31. INPUT A5	Analog input 5
32. INPUT A7	Analog input 7
33. INPUT A10 / OUTPUT O18	Analog input 10 and Output 18 share the same terminal, with a maximum current of 7 A.
34. INPUT A12 / OUTPUT O20	Analog input 12 and Output 20 share the same terminal, with a maximum current of 7 A.
35. INPUT A15 / OUTPUT O23	Analog input 15 and Output 23 share the same terminal, with a maximum current of 7 A.
36. CAN1.L	CAN L signal for CAN bus 1
37. CAN2.L	CAN L signal for CAN bus 2
38. OUTPUT O1	Output 1, maximum current 25 A, high power flyback
39. OUTPUT O2	Output 2, maximum current 25 A

2.7. Analog Inputs

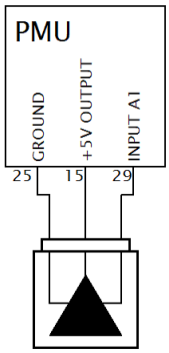
The Analog Inputs of the PMU 16 device are able to process voltages in the range of 0-5 V (voltages above 5 V are read as 5 V) with a frequency of 500 Hz. (The analog input is capable of withstanding a voltage of up to 20 V, but its measurement range is 0-5 V). In the PMU 24 - 8 inputs are the same as the PMU 16, while the next 8 terminals have an increased measuring range of 0-20 V (these inputs are capable of withstanding a voltage of up to 30 V). These terminals can be used as input or output pins.

There are 16 input pins available for both the PMU 16 and PMU 24, as well as a separate +5 V Pin for powering analog inputs such as rotary switches or analog sensors. The basic use of these inputs is to log signals from analog sensors, such as pressure sensors (oil, fuel, water, etc.), or from resistance sensors, such as oil temperature sensor, fuel level sensor. The analog inputs can also be used to connect switches. The voltage of all analog inputs can be transferred to other devices (e.g. EMU PRO) via the CAN bus.

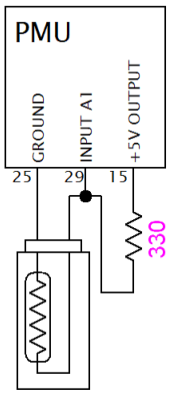
Switch



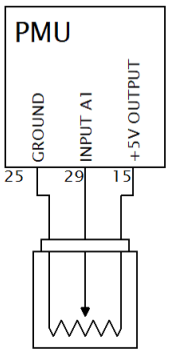
Oil pressure sensor



Oil temperature sensor



Potentiometer

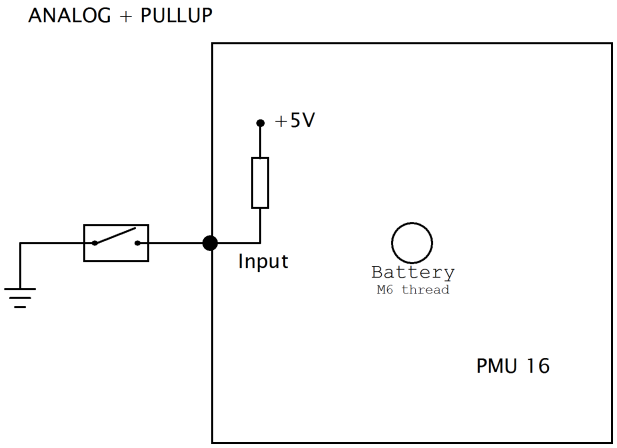


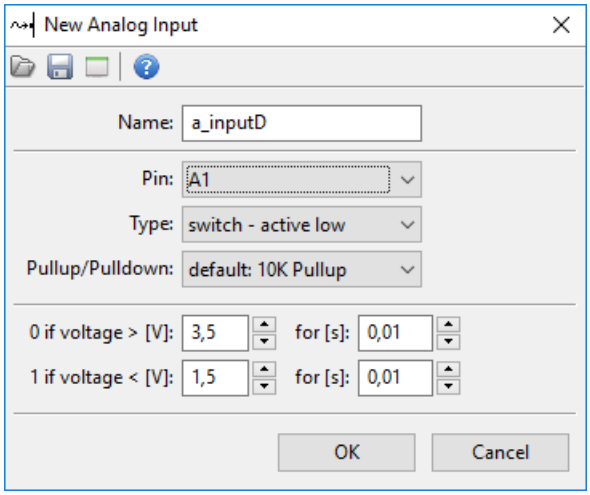
2.8. Internal Pull-up and Pull-down resistors

The PMU device has a built-in pull-up and pull-down [10k] circuit, selectable in the software for each analog input.

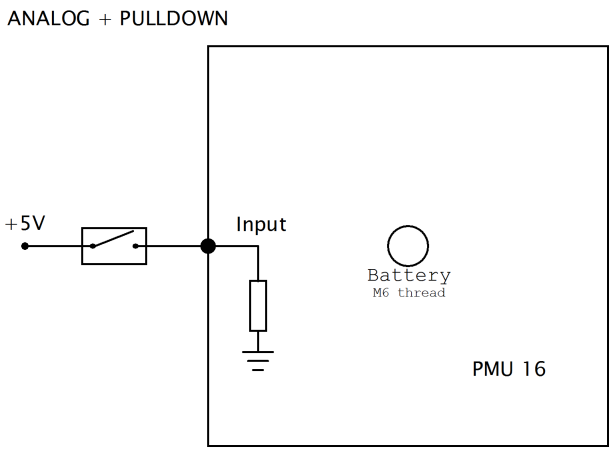
If a low-resistance sensor is connected, e.g. a fuel level sensor, the internal pull-up is too large, which makes it necessary to use an external circuit.

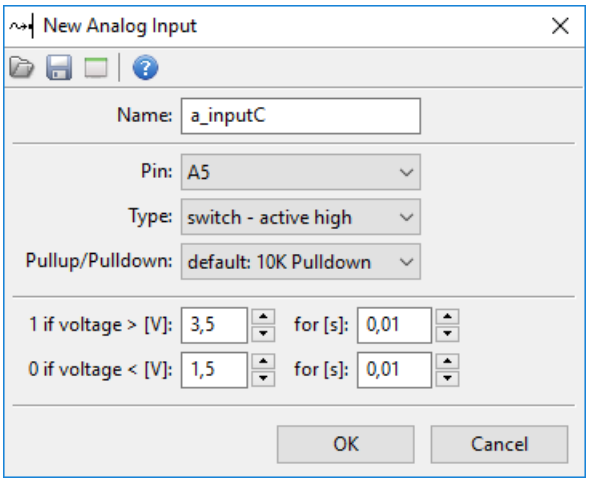
Connection diagram of the switch to ground (switch - active low) with a Pull-up resistor and an example of PMU Client configuration:





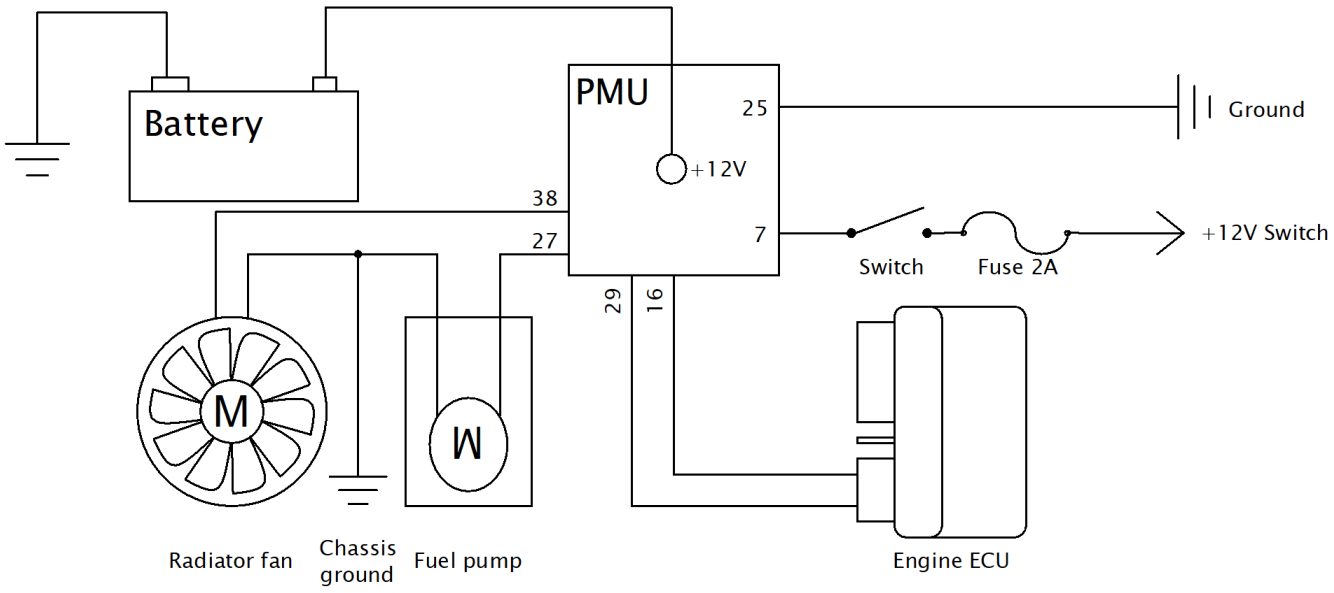
Connection diagram of a switch to +5 V (switch - active high) with a Pull-down resistor and an example of PMU Client configuration:





2.9. Using the analog input to control the fuel pump

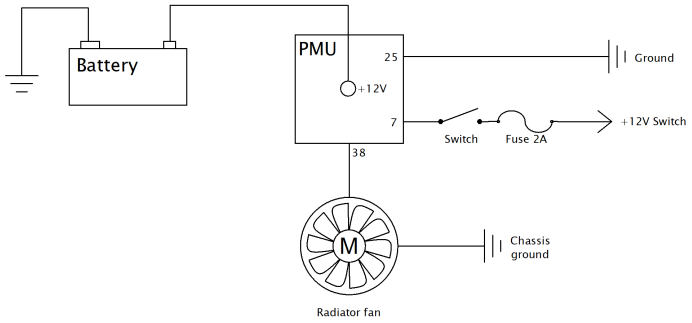
The engine control computer also controls the fuel pump and radiator fan. Traditionally, this is done using a "low-side" signal from the engine control unit output, which activates the fuel pump relay. Similarly, these devices can be controlled using a PMU. In the same way, the computer sends a signal from the output that activates the device (fuel pump or fan). The outputs from the ECU are connected to the analog inputs in the PMU, which operate as switches. They activate the output or outputs (in case of connecting more than one terminal) that power the fuel pump.

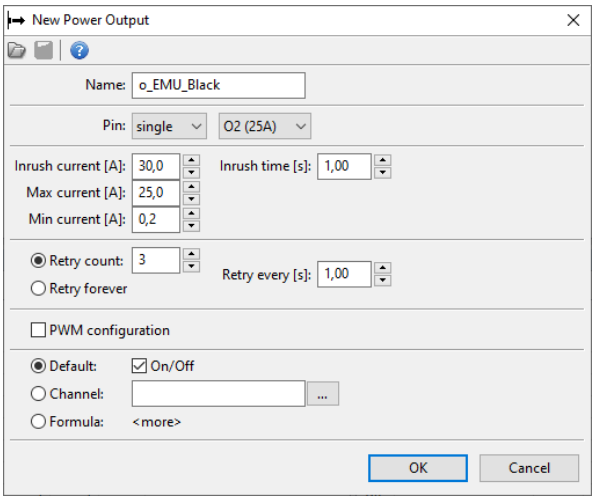


2.10. Outputs

The PMU 16 / 16DL has 16 outputs: 10 pins with a maximum continuous current load of 25 A and 6 pins up to 15 A. The PMU 24DL also has 8 terminals that can be used as input or output pins (with a maximum continuous current load of 7 A). The output pins can also be used in parallel to increase current capacity when the device draws a current of more than 25 A during operation. The PMU in the AS version has 14 outputs up to 25 A and 2 outputs up to of 40 A.

Connection diagram of the output device to the PMU



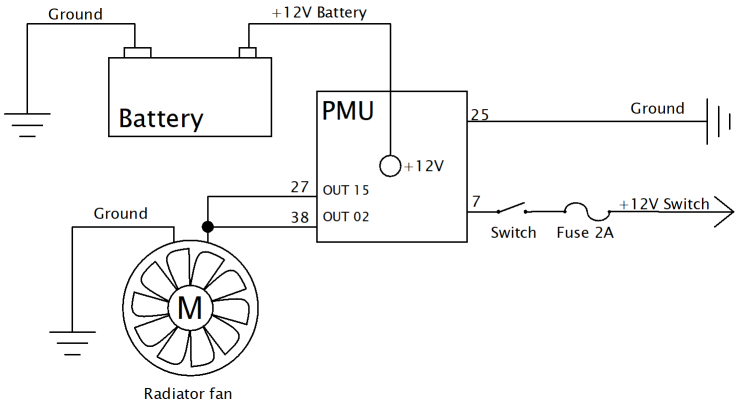


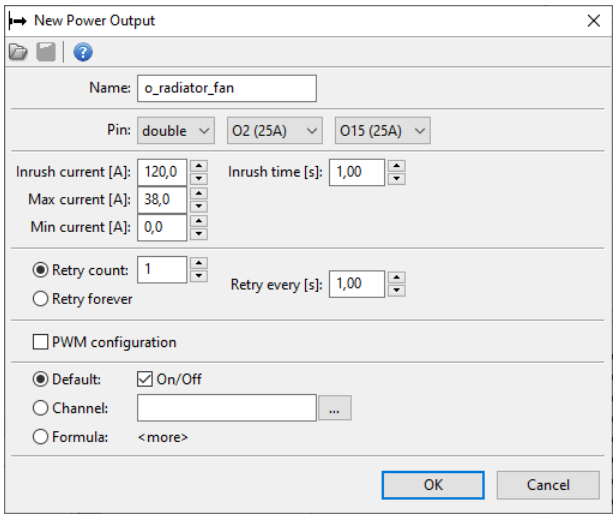
All outputs have overcurrent and undercurrent protection, short-circuit protection, and thermal protection. If any of this protection is activated, the output will be shut down, and an appropriate message will be displayed in the PMU Client and on the PMU device.

For 25 A Output Pins, Soft Start is available and PWM with Duty Cycle control

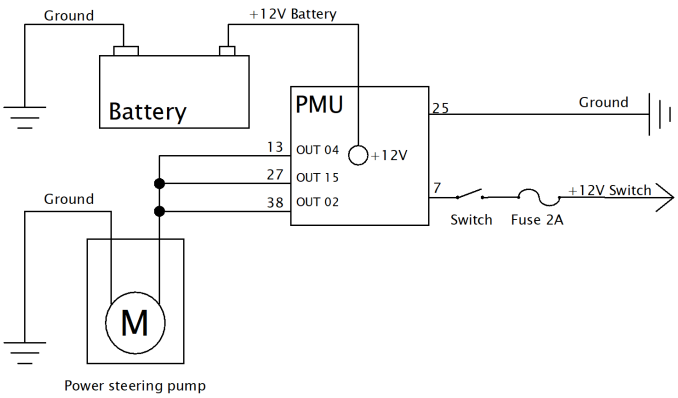
Examples of using parallel outputs:

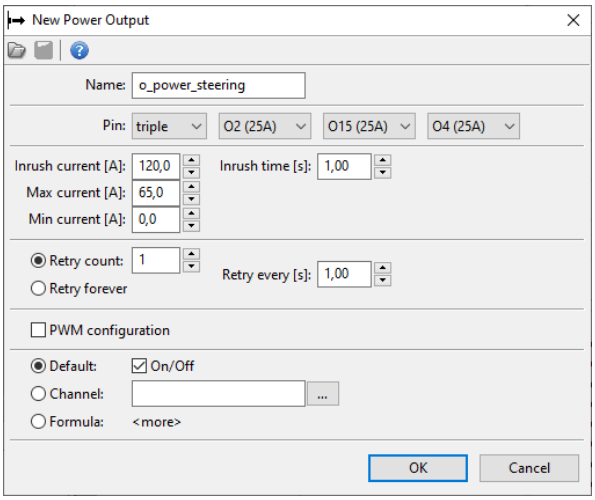
Radiator fan control (at a maximum current consumption of 38 A)





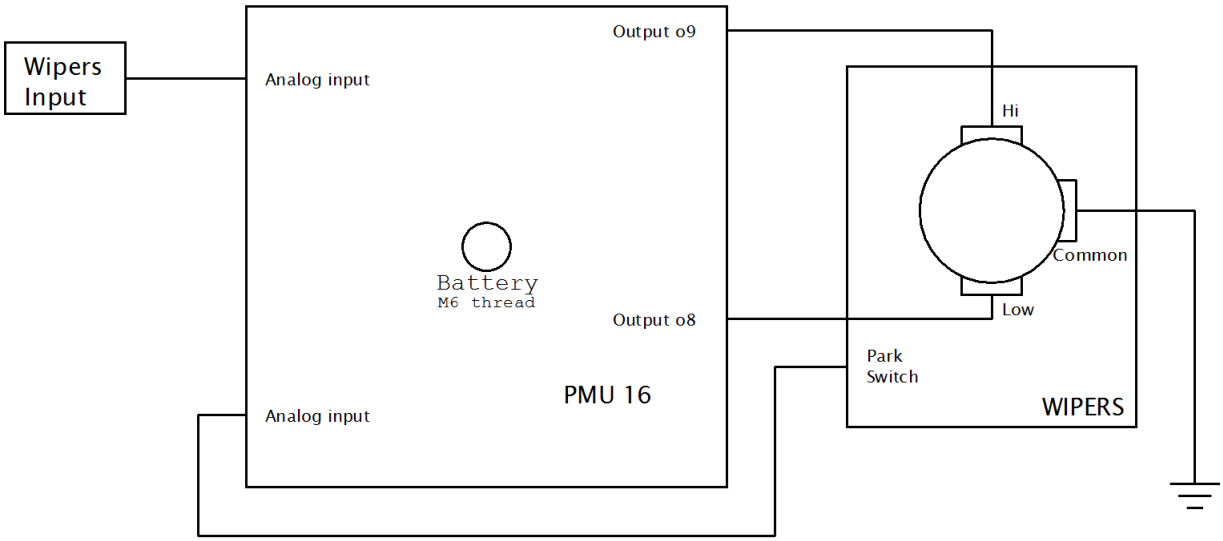
Power steering (maximum current consumption 65 A)





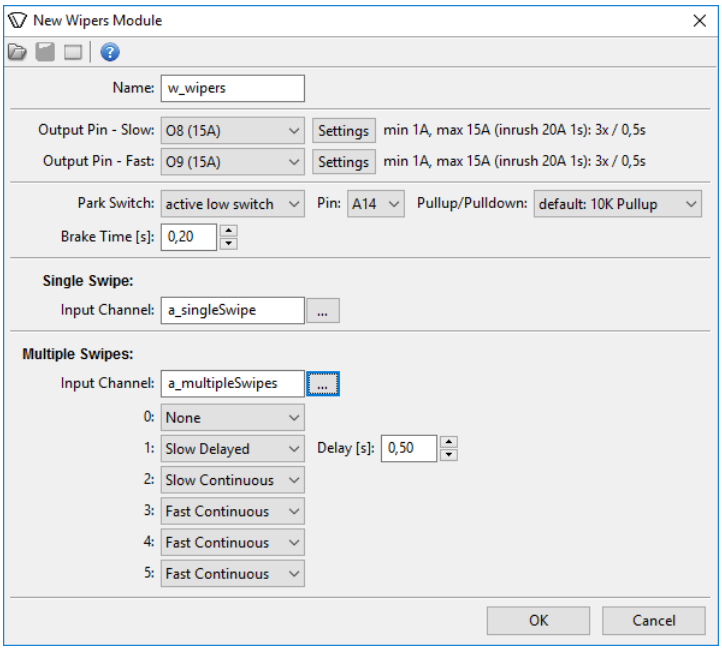
Connecting wipers to the PMU

To connect wipers O8 Output Pin must be used for Slow Wiper output. This is the Pin provided to use and it's the only way to use the park switch ability. For fast wiper output any output pin can be used except for O8. *Park switch* must be connected as an *Analog Input*.

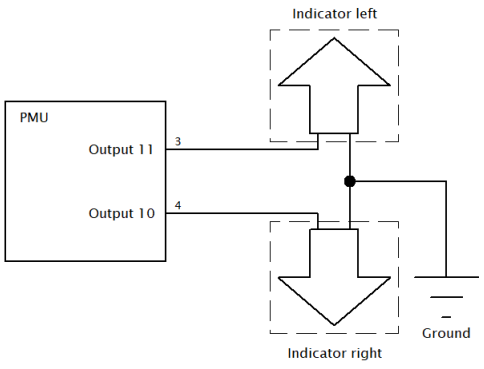


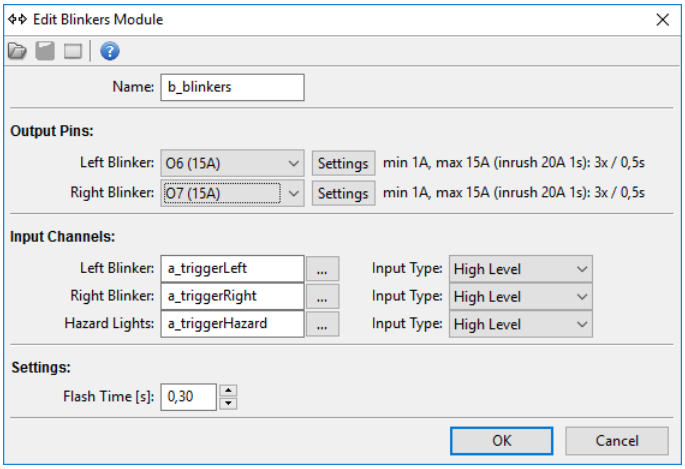
Configuring wipers in PMU Client is pretty easy. There is a module made specifically for wiper configuration that should be used to set up wipers trouble-free (*Project Tree / Wipers Module*).

PMU Client Configuration:



Connecting blinkers to the PMU





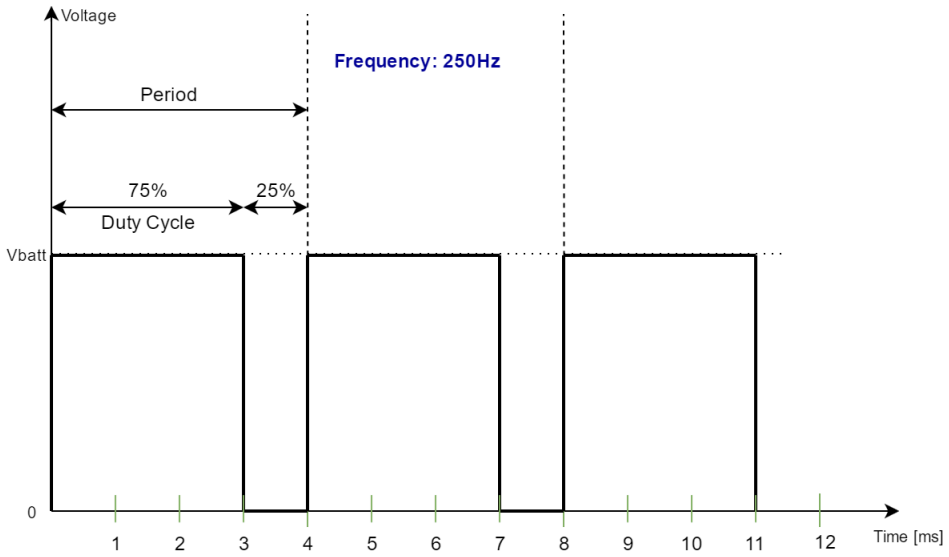
Blinkers should be connected to PMU Output Pins. Three input channels should be used in this configuration: two channels to control blinkers, and one channel to control hazard lights.

PMU Client also provides a special module designed to configure blinkers in an easy way (*Project Tree / Blinkers Module*).

2.11. Pulse Width Modulation – PWM and the use of a flyback diode

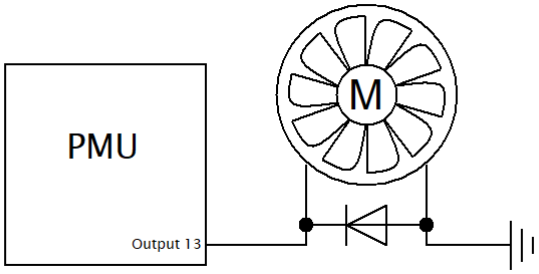
PMU has the ability to apply PWM to 25A Output Pins.

The main principle of Pulse Width Modulation (PWM) is to limit the amount of Power supplied to the Power Output by alternating power Output on and off.



Keep in mind that PWM introduces energy loss to heat due to the transistor switching on and off. Higher frequencies generate more heat, therefore If you are experiencing overheated status on Power Outputs or high heat in general, either lower the Frequency of PWM or use a flyback diode to eliminate flyback and lower the thermal load.

The diagram shows an example of using an external flyback diode.

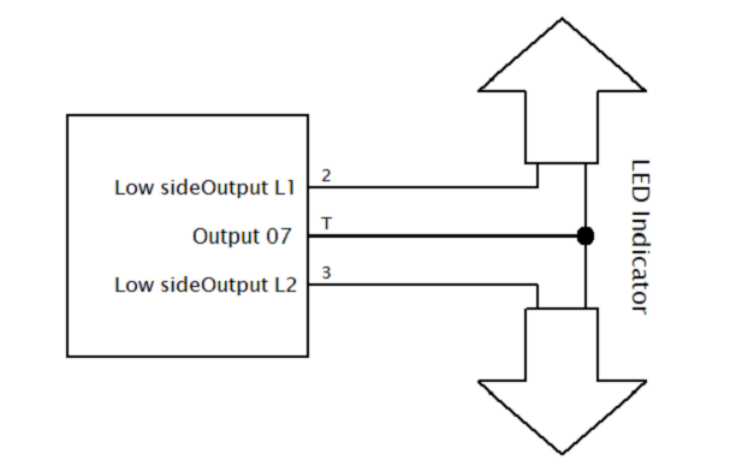


More information about the configuration in the section Soft Start and PWM Duty Cycle *(on page 75)*.

2.12. Low-side outputs – PMU AS

The PMU device in the Auto Sport version has additionally six low-side outputs (shorted to ground).

The maximum continuous current load of each of the outputs is 1 A. They can be used to switch on external receivers with low current consumption, e.g. LEDs or solenoid valves. The configuration of the low-side outputs is similar to the high-side outputs, but the maximum current consumption is fixed. Up to three outputs can be connected in series and generate a PWM signal.



2.13. Connector current capacity

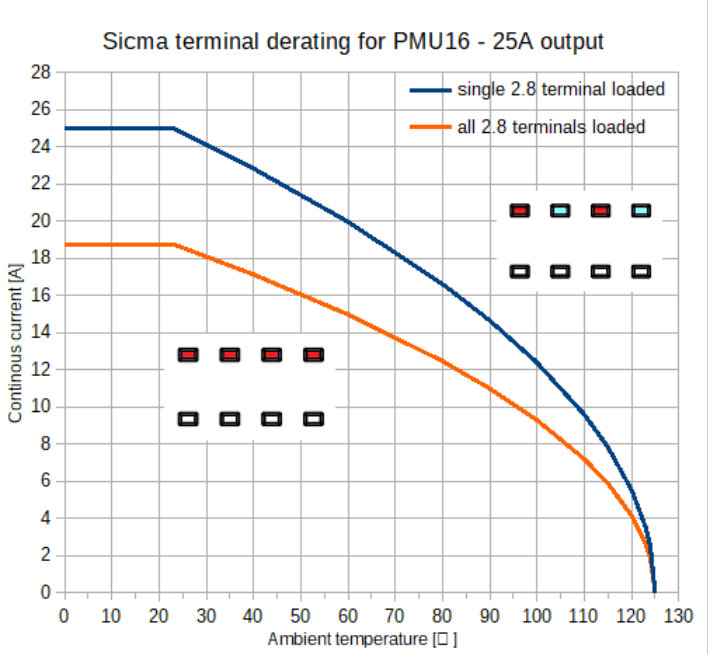
Outputs' current capacity is limited by the connector terminal current capacity.
The Sicma 2.8 terminal current capacity is 25A @ 23°C when only a single terminal is loaded and AWG12 (4mm2) wire is crimped correctly.
The Sicma 1.5 terminal current capacity is 19A @ 23°C when only a single terminal is loaded and AWG14 (2.5mm2) wire is crimped correctly.
Poorly crimped terminal, not completely seated terminal, or smaller than recommended wire size will reduce the terminal current capacity.
Terminals should be derated according to the anticipated temperature of the connector environment and used load balancing pattern.
To safely utilize current capacity, do not place heavily loaded terminals next to each other. During the harness design, the temperature at which the device will operate must be taken into account to prevent overloading the terminal, which may cause overheating and damage to the plug or the device itself.

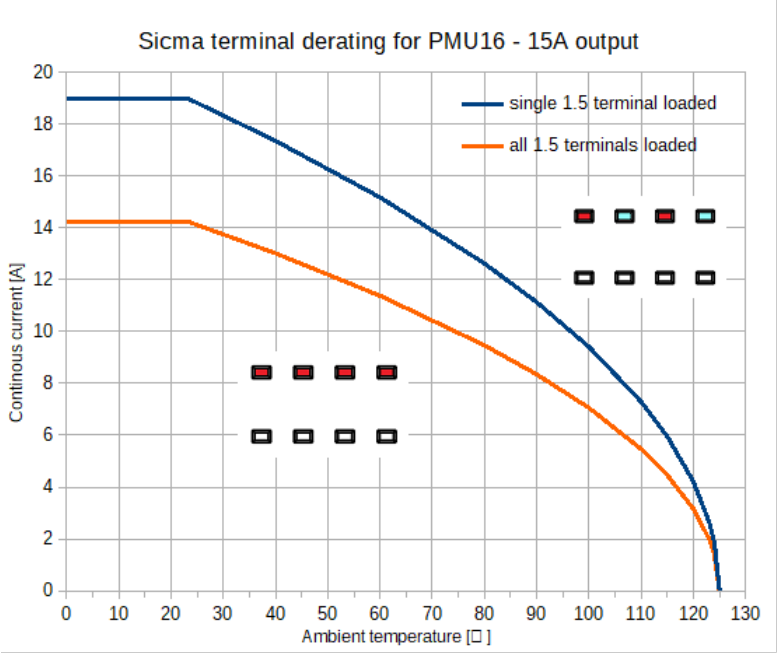
Important:

Two 2.8 terminals placed next to each other can only carry 38A total at 23°C, not 50A!

Important:

A single 2.8 terminal working at 40°C ambient temperature can safely conduct only 23A current.

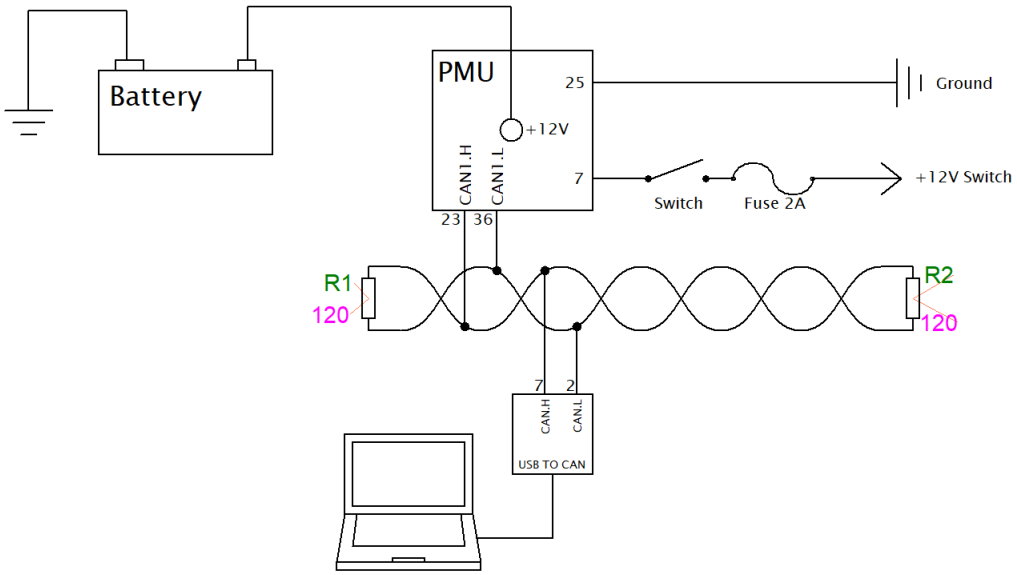




3. Installation

To start the device and to establish communication with a PC, connect the device's power supply and the USB2CAN interface to the CAN1 bus.

This bus has a fixed speed of 1 Mbps and one of its functions is communication with a PC.



The above figure shows the “minimal” configuration enabling communication with the device. The CAN bus is connected via a special USBtoCAN interface.

Interface must be purchased separately!

The software is compatible with the following interfaces:

- Ecumaster USBtoCAN (www.ecumaster.com/products/usb-to-can/)
- Peak Systems PCAN-USB and PCAN-LAN (www.peak-system.com)
- Kvaser USBcan (www.kvaser.com)

All these interfaces feature DB9 connectors, where CANL and CANH signals are on terminals No. 2 and No. 7.

To connect the Ecumaster USBtoCAN interface to a computer, the user must additionally be equipped with a USB A to USB B adapter.

The diagram also includes a 120 ohm terminator which is necessary for the bus to operate correctly (for more information about the terminators go to the CAN bus *(on page 36)* section).

Important:

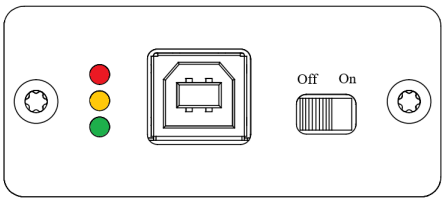
Do not connect the +5 V output from the CAN interface to +5 V PMU!

It is recommended to use interfaces with galvanic isolation. The CAN bus is a differential signal bus and in most cases, there is no need to connect the interface ground to the vehicle/PMU ground. If it is necessary to connect the interface ground, use a digital multimeter to check the potential difference between the interface ground and the vehicle ground. The excessive potential difference may damage the interface.

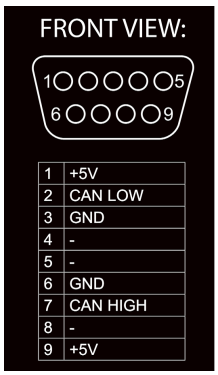
Ecumaster USBtoCAN also has LED signaling ability:

Color	Description
Green Continuous	Device turned on
Green Flashing	Device turned on and connected to PC
Green and Orange Flashing	Data transfer in progress
Orange Continuous	Device turned on, currently in bootloader
Orange Flashing	Device turned on, firmware update in progress
Red Continuous	Temporary CAN communication error
Red Flashing	Permanent CAN communication error

Ecumaster USBtoCAN:



Ecumaster USBtoCAN pinout:

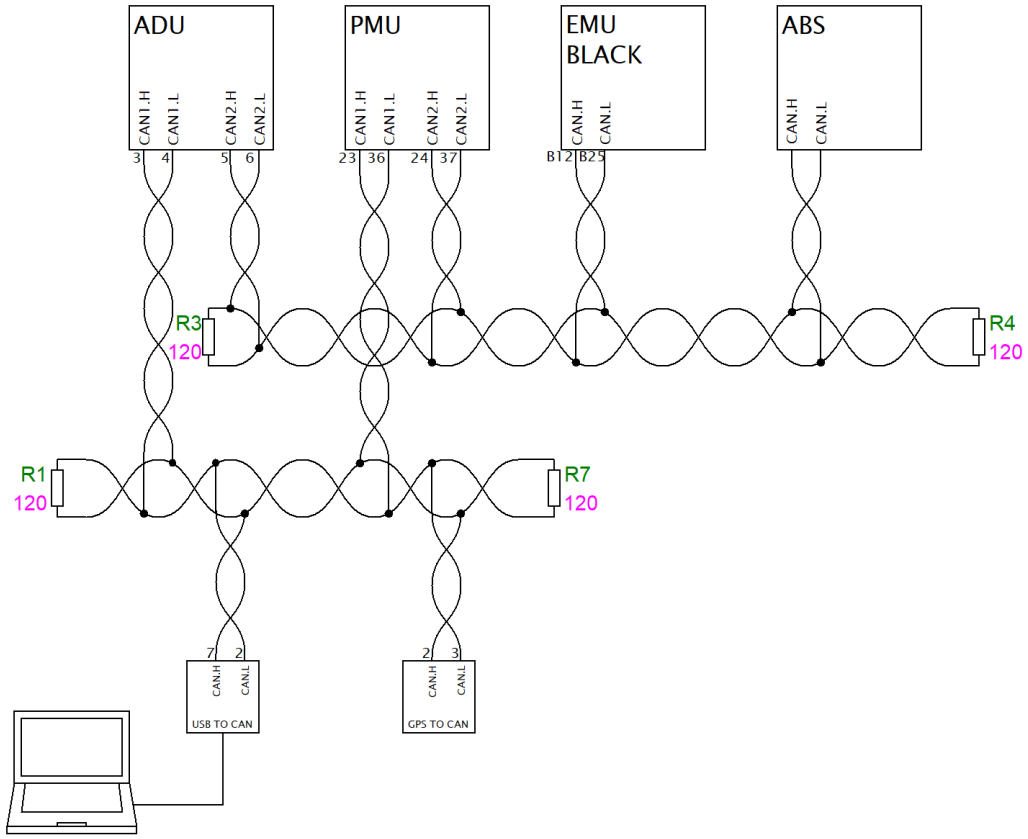


3.1. CAN bus

The CAN (*Control Area Network*) bus was developed for communication in automotive applications. Its construction is very simple (only two wires) and its immunity to interference is very high. In modern vehicles, dozens of different electronic modules may communicate via the CAN bus.

The PMU device is equipped with two CAN buses: CAN1 bus is used for communication with a PC (requires an additional interface), as well as with other CAN devices.

Data frames are sent using a network, the topology of which is shown on the following diagram:



In automotive applications, typical data rates on a CAN bus are 1 Mbps, 500 kbps, and 250 kbps.

The following conditions must be met for each speed:

For 1 Mbps:

- the maximum length of the connection cable between the bus and the node must not exceed 30 cm
- the maximum bus length is 40 m
- maximum number of nods is 30

For 500 kbps:

- the maximum length of the connection cable between the bus and the node must not exceed 30 cm
- the maximum bus length is 100 m
- maximum number of nods is 30

Regardless of the speed, the CAN bus requires termination in the form of 120 ohm resistors at both ends. In the case of CAN1 bus, there is no internal termination resistor, so external termination is required. CAN2 bus is equipped with a software-controlled termination resistor. Additionally, all the connections within the bus must be made using twisted pair wires.
It is important that the data transfer speed on a single bus has to be identical for all devices.

Important:
Failure to follow these rules will lead to the malfunctioning of the CAN bus and communication problems.

The CAN frame comprises an identifier (ID), number of transmitted bytes (DLC) and the data itself. Depending on the bus type, the identifier may be 11 bits (0x0-0x7ff) or 29 bit (0x0-0x1fffffff). The number of data bytes can range from 0 to 8.

ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

Below is a sample CAN frame of the CAN switch board.

ID	DLC	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0x666	8	*Analog#1(mV)*		*Analog#2 (mV)*		*CALPOT 1*	*CALPOT 2*	*Switch mask*	*Heartbeat*

Parameter	Description
Analog#1	Voltage for the analog input #1 0–5000 mV, big endian
Analog#2	Voltage for the analog input #2 0–5000 mV, big endian
Switch mask	Bitmask of pressed buttons (1 means pressed)
CAL POT #1	Discrete value of the position of the rotary switch connected to analog input #1
CAL POT #2	Discrete value of the position of the rotary switch connected to analog input #2
Heartbeat	The counter increments its value by 1 after sending each frame

3.2. Connecting the PMU to the devices

To be able to communicate and log data from devices in the vehicle, connect the PMU to the CAN bus.

When using a CAN bus, you can connect devices to CAN1 (where the device's CAN bus speed is 1 Mbps) or to CAN2 bus where you can define the speed of the CAN bus.

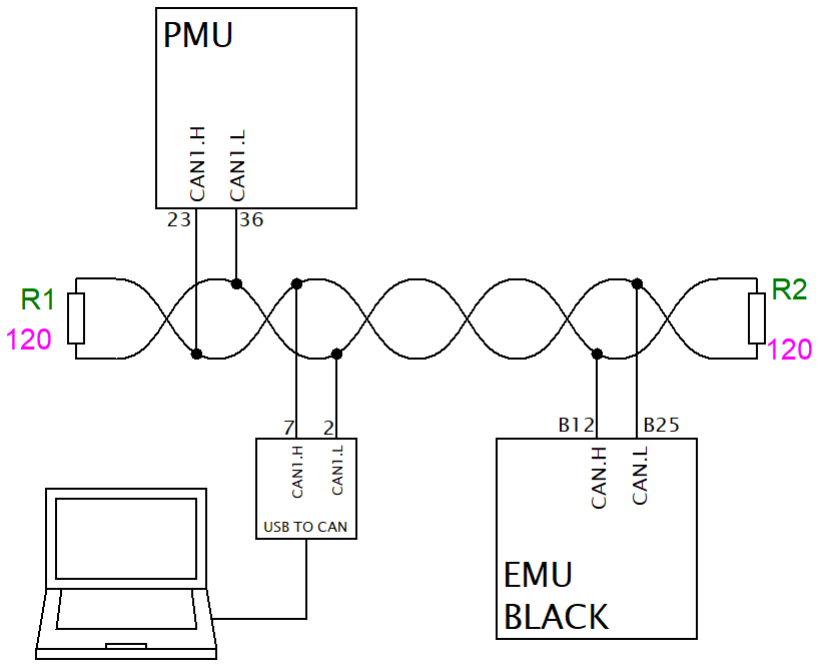
OEM engine management units usually use 500 kbps (ECUs with a 250 kbps bus are rarer), which forces connection to CAN2 bus.

External sensors (e.g. TPS, temperature and oil/fuel pressure sensors, crankshaft position sensor, etc.) may be connected to analog inputs of the device. As a result, the user can monitor and log parameters that are not supported by the original engine controller.

The analog inputs can be used as switch inputs to activate various receivers, e.g. the engine control unit (ECU) sends a low-side signal to start the fuel pump. For this purpose, analog input No. 1 in the PMU is used, set as a *switch*.

3.3. Connecting via a CAN bus

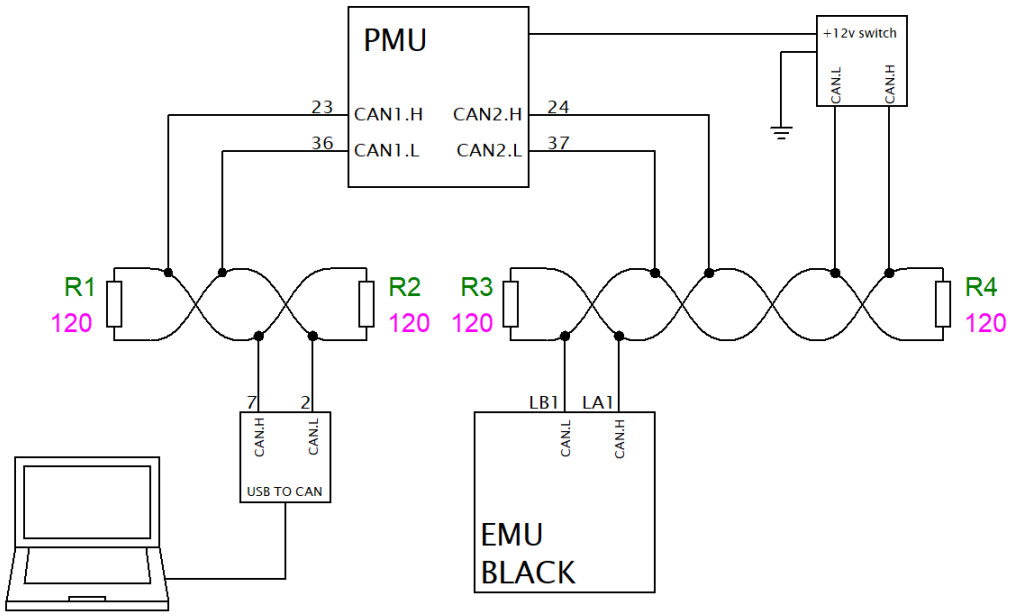
The following diagram shows an example connection of the EMU BLACK to the PMU using CAN1 bus.



Important:

The CAN bus speed of the engine controller must be 1 Mbps, as this is the only speed supported by the CAN1 PMU bus.

If the CAN bus speed is different than 1 Mbps or if you want to isolate the ECU from the PMU CAN1 bus, please use the CAN2 bus. The following diagram shows an example connection of an EMU BLACK.

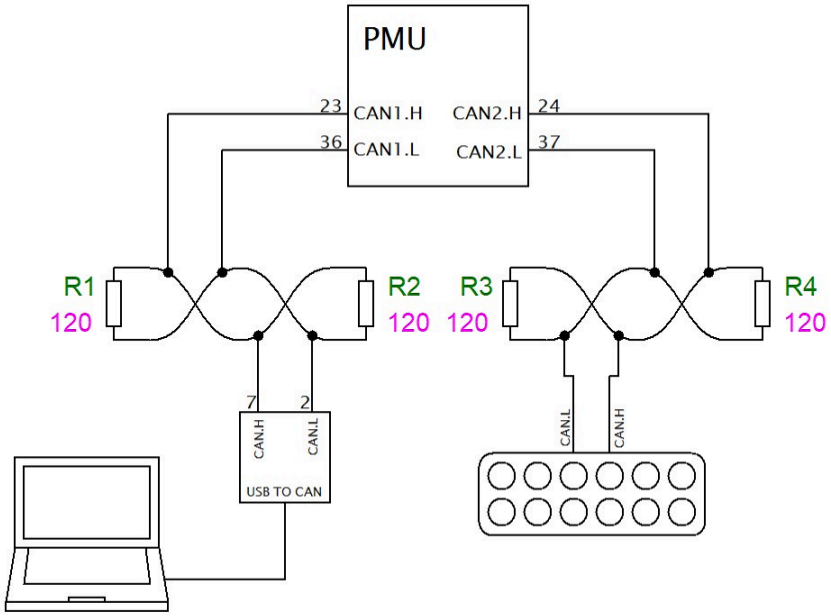


Important:

Care must be taken to ensure correct topology and proper bus termination. For more information see the CAN bus *(on page 36)* section.

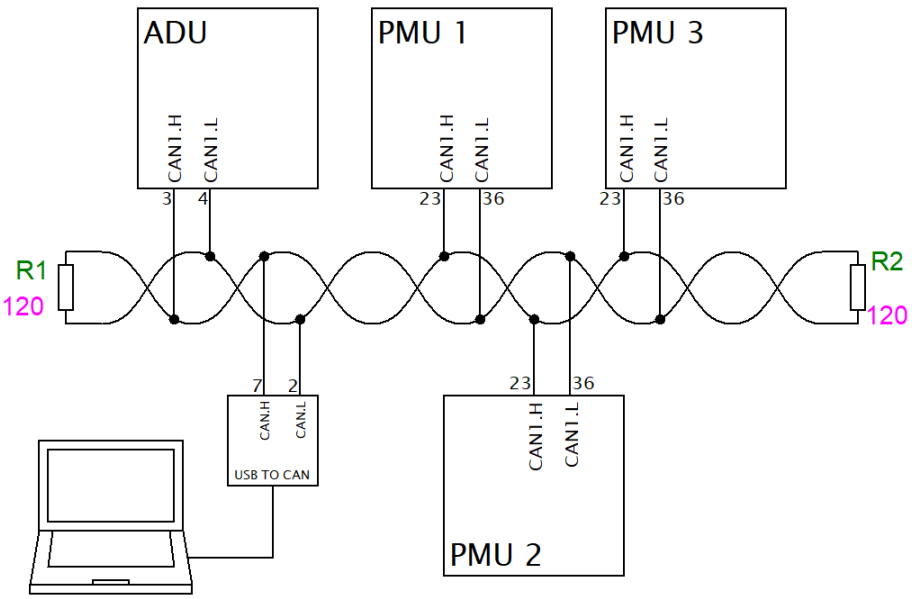
Detailed information on connecting specific ECU brands to PMU can be found in the application notes at www.ecumaster.com/products/pmu/.

Keyboard connection diagram



4. Using multiple PMUs

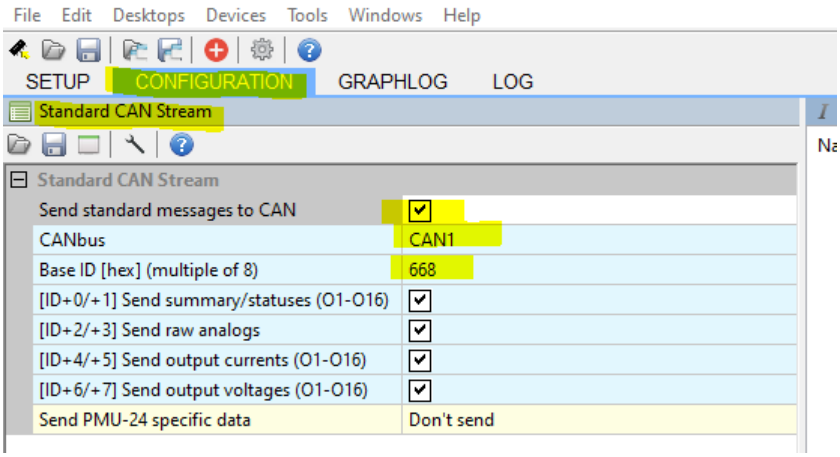
A unique feature of the PMU is working with multiple devices connected to the same CAN bus with one software. Up to five PMUs can be connected on one CAN bus. To use several PMUs, it is recommended to use the CAN1 bus. The devices must be connected to the same bus, terminated with 120 ohm resistors at both ends.



Communication

Setting up proper strategies requires the PMU units used to communicate with each other. For this purpose, it is recommended to use the *CANbus Export* and *CANbus Input* functions to transmit important data between PMUs.

The ***Base ID*** of each PMU must be set correctly when multiple PMUs are installed and connected to the same CAN bus. The screenshot below shows the *Standard CAN Stream* configuration window in the PMU Client.



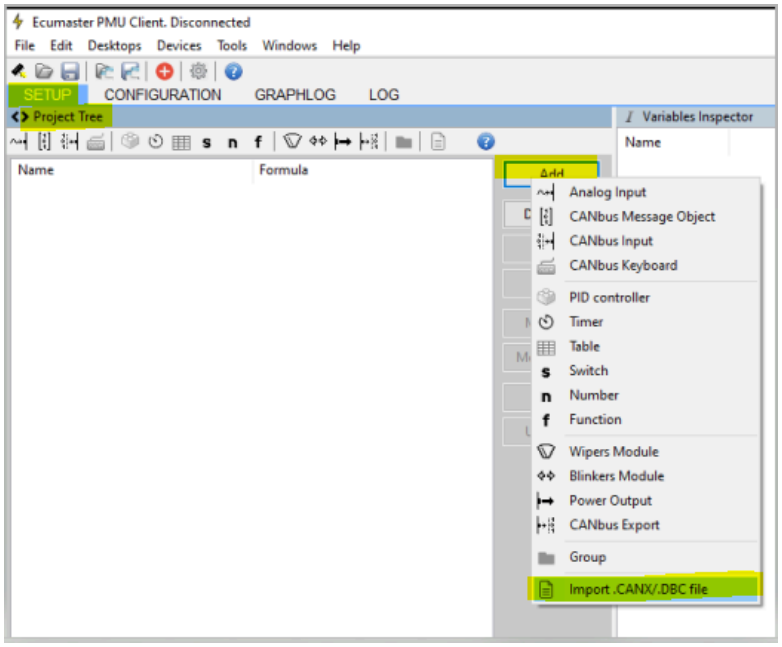
- Enable *Send standard messages to CAN*
- *CANbus*: "CAN1" or "CAN2". Select which CAN bus stream the PMU data is on
- *Base ID*: This field is the dedicated Base CAN ID for each PMU. The following list shows what address to set the *Base ID* on PMU#1, PMU#2, and PMU#3 for the PMU24 and PMU16:

  PMU16:
  - PMU-16_#1 –*Base ID*: 668
  - PMU-16_#2 –*Base ID*: 670
  - PMU-16_#3 –*Base ID*: 678

  PMU24:
  - PMU-24_#1 –*Base ID*: 668
  - PMU-24_#2 –*Base ID*: 678
  - PMU-24_#3 –*Base ID*: 688

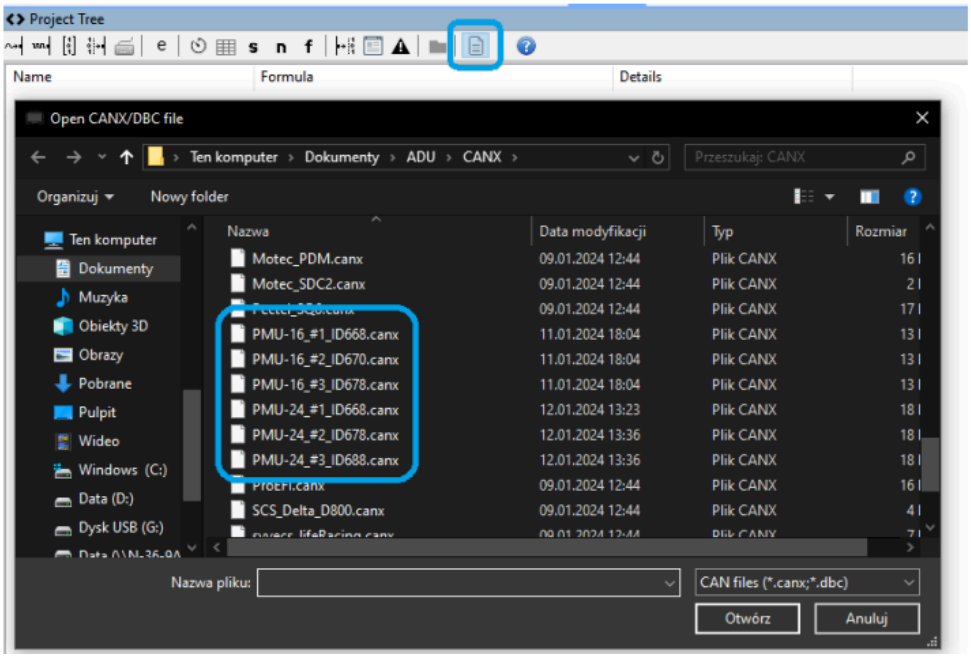
- Leave the *Send* settings enabled

For each of the PMUs import the .CANX files for the other PMUs in the *Project Tree* under the *SETUP* tab.



There are six files to choose from, each containing the PMU device number and default base address in its name:

- PMU-24_#1_ID668.canx
- PMU-24_#2_ID678.canx
- PMU-24_#3_ID688.canx
- PMU-16_#1_ID668.canx
- PMU-16_#2_ID670.canx
- PMU-16_#3_ID678.canx



It's important to note that importing the PMU .CANX file creates only CANbus Message objects in the Project Tree. For PMU-24, these are two CANbus Message Objects:

- PMU1 [1-16]
- PMU1 [17-24]

For PMU-16, there is only one CANbus Message Object:

- PMU1 [1-16]

The CANbus Inputs are not necessary because the channels decode automatically, saving user resources.

The scheme below shows the default CAN stream locations of the PMUs. Please note, that if using PMU 24 and PMU 16 on the same CANbus, the default CAN stream locations may need to be adjusted to avoid conflict.

<table>
<tr><th>ID</th><th>Default CAN stream locations for PMU-16</th><th>Default CAN stream locations for PMU-24</th></tr>
<tr><td>0x668</td><td rowspan="8">PMU-16 #1 at 0x668</td><td rowspan="16">PMU-24 #1 at 0x668</td></tr>
<tr><td>0x669</td></tr>
<tr><td>0x66A</td></tr>
<tr><td>0x66B</td></tr>
<tr><td>0x66C</td></tr>
<tr><td>0x66D</td></tr>
<tr><td>0x66E</td></tr>
<tr><td>0x66F</td></tr>
<tr><td>0x670</td><td rowspan="8">PMU-16 #2 at 0x670</td></tr>
<tr><td>0x671</td></tr>
<tr><td>0x672</td></tr>
<tr><td>0x673</td></tr>
<tr><td>0x674</td></tr>
<tr><td>0x675</td></tr>
<tr><td>0x676</td></tr>
<tr><td>0x677</td></tr>
<tr><td>0x678</td><td rowspan="8">PMU-16 #3 at 0x678</td><td rowspan="16">PMU-24 #2 at 0x678</td></tr>
<tr><td>0x679</td></tr>
<tr><td>0x67A</td></tr>
<tr><td>0x67B</td></tr>
<tr><td>0x67C</td></tr>
<tr><td>0x67D</td></tr>
<tr><td>0x67E</td></tr>
<tr><td>0x67F</td></tr>
<tr><td>0x680</td><td></td></tr>
<tr><td>0x681</td><td></td></tr>
<tr><td>0x682</td><td></td></tr>
<tr><td>0x683</td><td></td></tr>
<tr><td>0x684</td><td></td></tr>
<tr><td>0x685</td><td></td></tr>
<tr><td>0x686</td><td></td></tr>
<tr><td>0x687</td><td></td></tr>
<tr><td>0x688</td><td></td><td rowspan="16">PMU-24 #3 at 0x688</td></tr>
<tr><td>0x689</td><td></td></tr>
<tr><td>0x68A</td><td></td></tr>
<tr><td>0x68B</td><td></td></tr>
<tr><td>0x68C</td><td></td></tr>
<tr><td>0x68D</td><td></td></tr>
<tr><td>0x68E</td><td></td></tr>
<tr><td>0x68F</td><td></td></tr>
<tr><td>0x690</td><td></td></tr>
<tr><td>0x691</td><td></td></tr>
<tr><td>0x692</td><td></td></tr>
<tr><td>0x693</td><td></td></tr>
</table>

ID	Default CAN stream locations for PMU-16	Default CAN stream locations for PMU-24
0x694		
0x695		
0x696		
0x697		

Using PMU Client for multiple PMUs

PMU Client allows up to five PMUs to be connected simultaneously. You can easily switch between them using the menu bar ***Devices/ Set Device#n*** or using the keyboard shortcut ***Ctrl+Shift +1 to 5***.

All currently connected PMUs are also displayed as tabs in the menu bar.

Monitoring PMU Channels

To keep track of PMU channel status information, you can open a separate Log panel for each PMU. Below is a list of channels for each of the three built-in PMU devices:

- pmuX.totalCurrent
- pmuX.battery
- pmuX.boardTemperatureL
- pmuX.boardTemperatureR
- pmuX.boardTemperatureMax
- pmuX.status
- pmuX.userError
- pmuX.oY.status – states for each of the 24 outputs
- pmuX.oY.active – activity flags for each of the 24 outputs
- pmuX.oY.current (*) – current values for each of the 24 outputs
- pmuX.oY.voltage (**) – voltage values for each of the 24 outputs
- pmuX.aY.voltage – voltages in the range of 0-5V for each of the 16 inputs

(*) – For outputs O1-O16, the current value resolution is 0.25A, and for outputs O17-O24, it is 0.1A.

(**) – For outputs O1-O16, the measurement range is 0-16V, and for outputs O17-O24, it is 0-20V.

Extended Voltage Range on Inputs

Inputs A9-A16 can measure voltage in the range of 0-20V, but the information sent on the CAN channels pmuX.aY.voltage is limited to the range of 0-5V. To read the voltage value on these analog inputs across the entire measurement range, you can utilize channels pmuX.oY.voltage. For example, the voltage value on input A9 is also available on pmuX.o17.voltage, and the voltage value on input A16 is also available on pmuX.o24.voltage.

5. PMU Client software for Windows

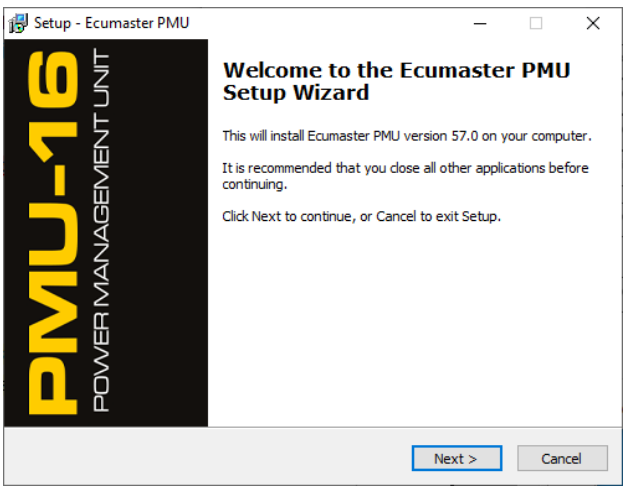
To set up the PMU device, it is necessary to install software for Windows - PMU Client (PMU16, PMU16DL, PMU24DL, and PMU16AS use the same Client software). Official software can be found at www.ecumater.com/products/pmu/.

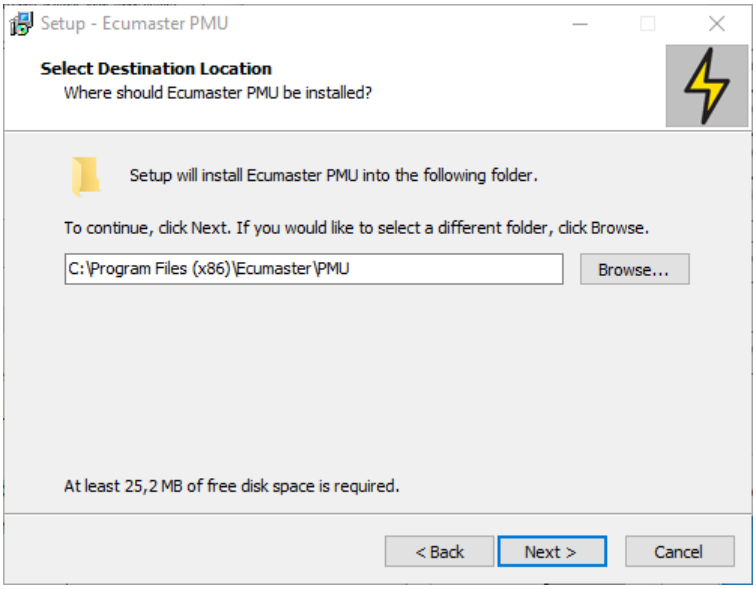
Unofficial, test versions of the software (with additional features) are also available. They can be downloaded from www.ecumaster.com/testVersions.html. Test versions of the software may contain errors! If you have problems with the new software, please contact us at bugs@ecumaster.com. Ecumaster recommends the use of official software.

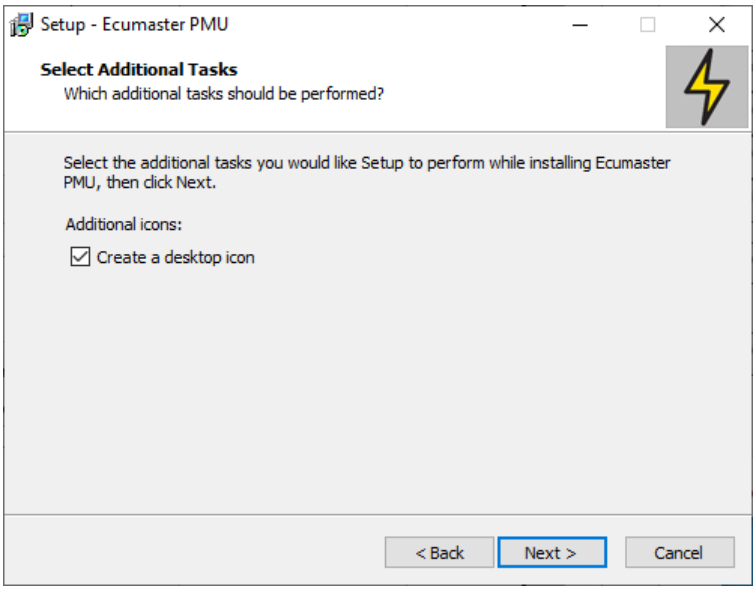
Hardware requirements to run the software:

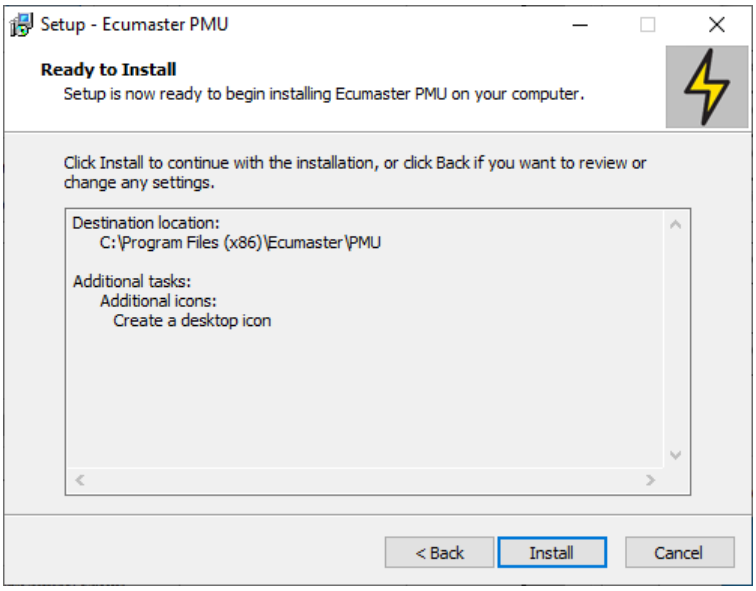
- Windows XP, VISTA, 7, 8, 10, 11 (32 and 64 bits)
- minimum screen resolution of 1366x768
- Open GL compatible graphics card
- 2 GB RAM
- USB port

The following window appears during software installation. Follow the instructions to complete the installation.



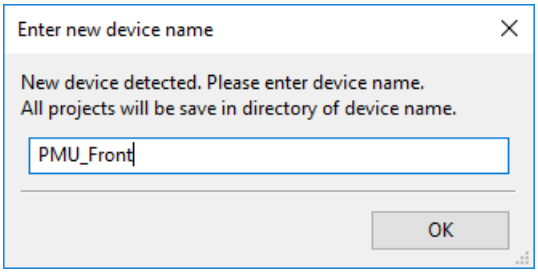

Setup - Ecumaster PMU
Select Destination Location
Where should Ecumaster PMU be installed?
Setup will install Ecumaster PMU into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files (x86)\Ecumaster\PMU
Browse...
At least 25,2 MB of free disk space is required.
< Back
Next >
Cancel


Setup - Ecumaster PMU
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing Ecumaster PMU, then click Next.
Additional icons:
Create a desktop icon
< Back
Next >
Cancel


Setup - Ecumaster PMU
Ready to Install
Setup is now ready to begin installing Ecumaster PMU on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
C:\Program Files (x86)\Ecumaster\PMU
Additional tasks:
Additional icons:
Create a desktop icon
< Back
Install
Cancel

5.1. First connection with the device

Once the software has been installed on a PC, upon the first connection a window will appear asking for the device name.



A subdirectory in users' Documents / PMU / folder will be created under this name, where all settings relating to the device will be stored.

Client and Firmware:

Each PMU is equipped with pre-installed embedded (internal) software that ensures its functioning.

It is referred to as ***Firmware*** .
Client software is used to operate the PMU using a PC.
The Client software installation file contains the latest Firmware versions for the PMU.

Numbering of Client and Firmware versions:

Client version number format:

FirmwareConfiguration.FirmwareMinor.ClientFixNumber (e.g. 51.0.2)

or

FirmwareConfiguration.FirmwareMinor – if ClientFixNumber is zero (e.g. 51.0)

- FirmwareConfiguration, or master version - changes when new settings are made for the Firmware visible in the PMU Client.
- FirmwareMinor, i.e. the minor version - changes when a Firmware fix is published, but WITHOUT making new settings in the PMU Client.
- ClientFixNumber, i.e. Client software fix number - changes each time only the Client is updated.

Firmware version number format:

FirmwareConfiguration.FirmwareMinor (e.g. 51.0)

It should be noted that the Client software version e.g. 83.0:

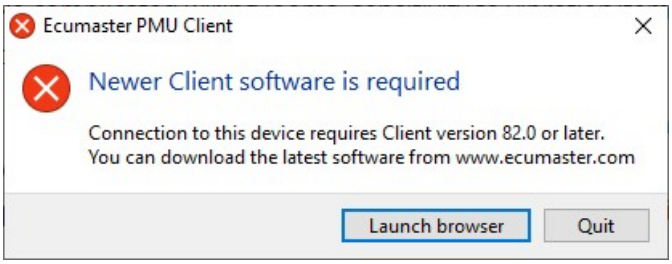
- supports any Firmware from the 83.x family: 83.0, 83.1, 83.2, etc.
- supports all older Firmware versions, e.g. 50.0, 51.1, 82.1, 82.0, 81.0, etc.
- It does NOT support higher FirmwareConfiguration version numbers e.g. 84.0, 84.1

In other words, the latest PMU Client version supports all PMUs that were previously produced. It is possible to update the factory-installed firmware to the latest version or install an older version at any time.

Checking the software version on the first connection:

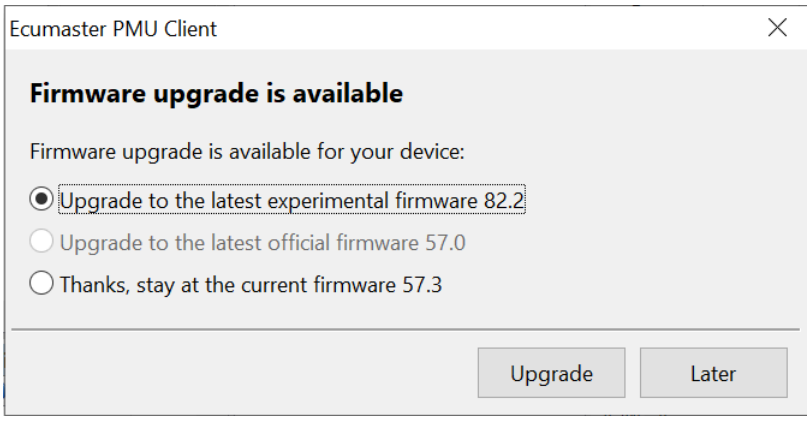
Depending on the Firmware version on the PMU and the Client software version installed on the PC, messages recommending a firmware update may appear the first time you connect to the device:

1. When the Firmware version of the device is higher than the version of the Client software installed on the PC, a message will appear indicating that ***Newer Client software is required***.



   Client software version should be the same or higher than that of the *Firmware* running on the device.

2. When the Firmware version of the device is lower than that of the Client, a message will appear indicating that newer Firmware is available for the PMU (***Firmware upgrade is available***). If an experimental version of the Client software is installed on the PC, three suggestions will appear in a message:

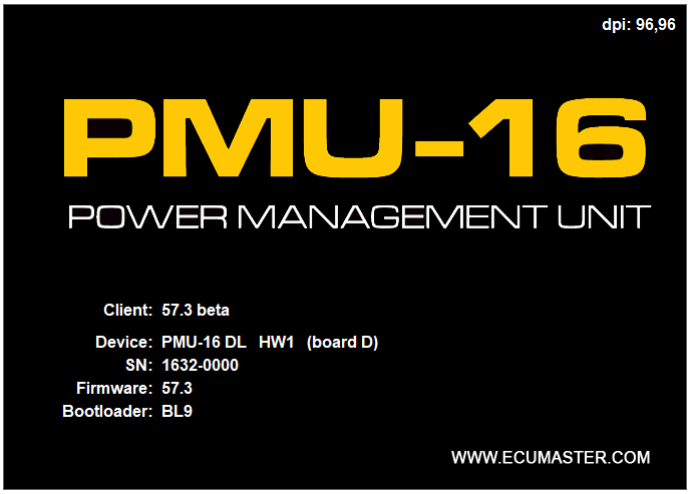


- upgrade to the latest experimental version,
- upgrade to the latest official version (but older than the listed experimental version),
- keep the existing version.

Otherwise, only two options will be available:

- upgrade to the latest official version,
- keep the existing version.

To check the version of the *Firmware* currently installed on the device and the version of the *Client* software, select ***Help / About*** from the main menu. The following message window will appear.



5.2. Firmware update

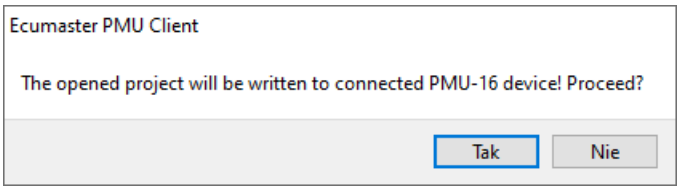
If necessary, the Firmware version can be changed at any time in the PMU. To do this, select ***File / Upgrade firmware...*** from the main menu and then select the relevant firmware version from the list.

5.3. Loading a project

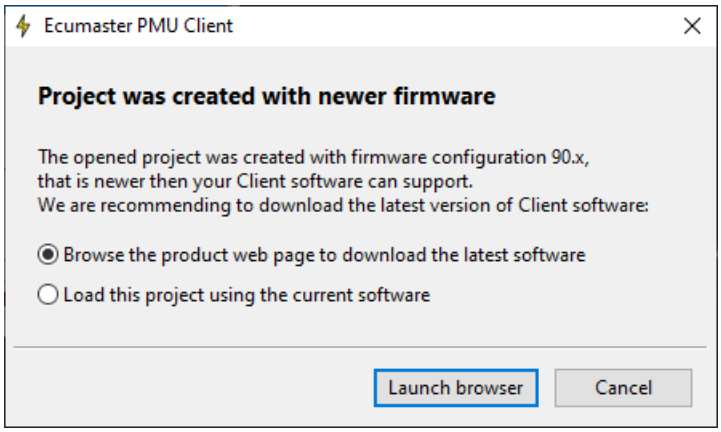
When opening a finished project, different messages may appear depending on:

- the configuration in which the file was created (*firmware configuration*);
- the *Client* software version installed on the PC;
- the *Firmware* version on the PMU;
- the connection status of the PMU or the *"Offline"* operating mode.

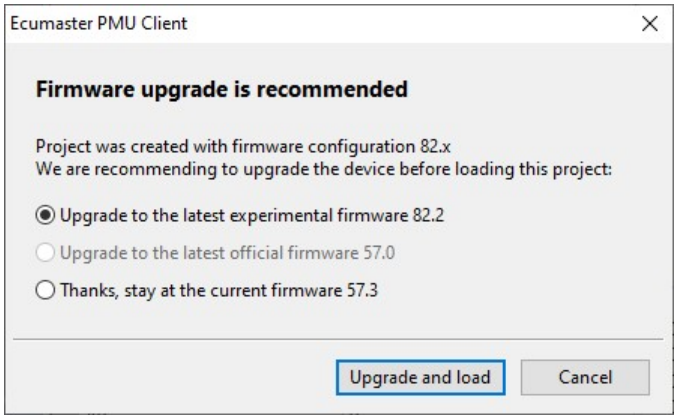
If the device is connected while opening a project, a message may appear indicating that the project being opened will be saved on the connected PMU.



If a project being opened was created using (*firmware configuration*) not supported by the Client software version, a window may appear recommending that a newer version of the Client software be downloaded. This window can appear both when the PMU is connected and when working offline.

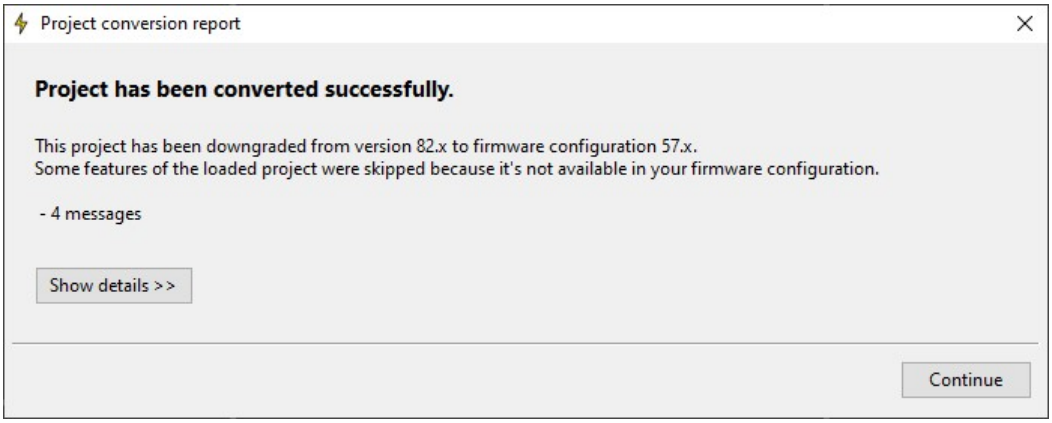


If a project was created using a firmware configuration not supported by the version of Firmware on the PMU, a window may appear recommending a Firmware update before loading the project.

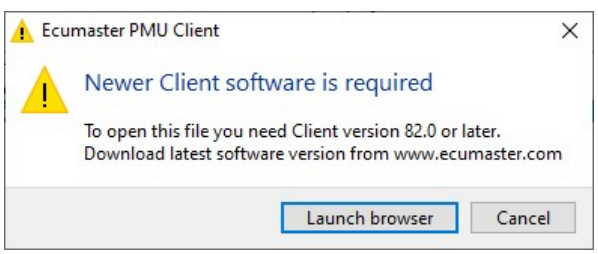


If the software was installed in an experimental (test) version, the message window will offer three choices: update to the latest experimental version, update to the latest official version, or keep the existing version.
If an official version of the software was installed, you will be presented with only two choices: upgrade to the latest official version or keep the existing version.

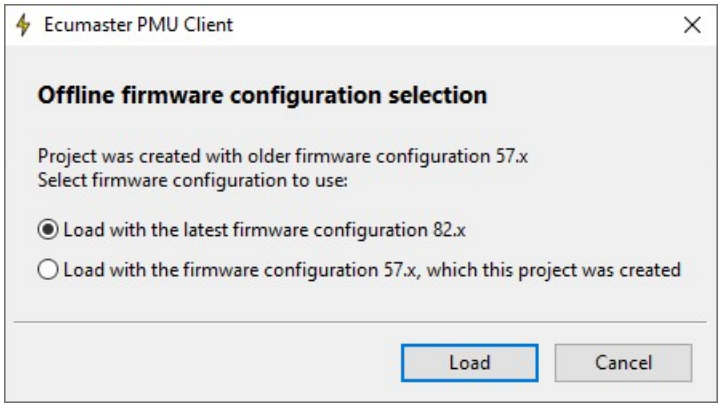
If a project created using a newer configuration is loaded into a device with older configuration software, some functions of the project will not be available because they are not supported by the older configuration.



Opening a project created using a newer configuration and saved in a format other than the format supported by the installed Client software, will cause a message to appear stating that newer software needs to be installed, along with information on the minimum version required to open the file. This window can appear both when the PMU is connected and when working offline.

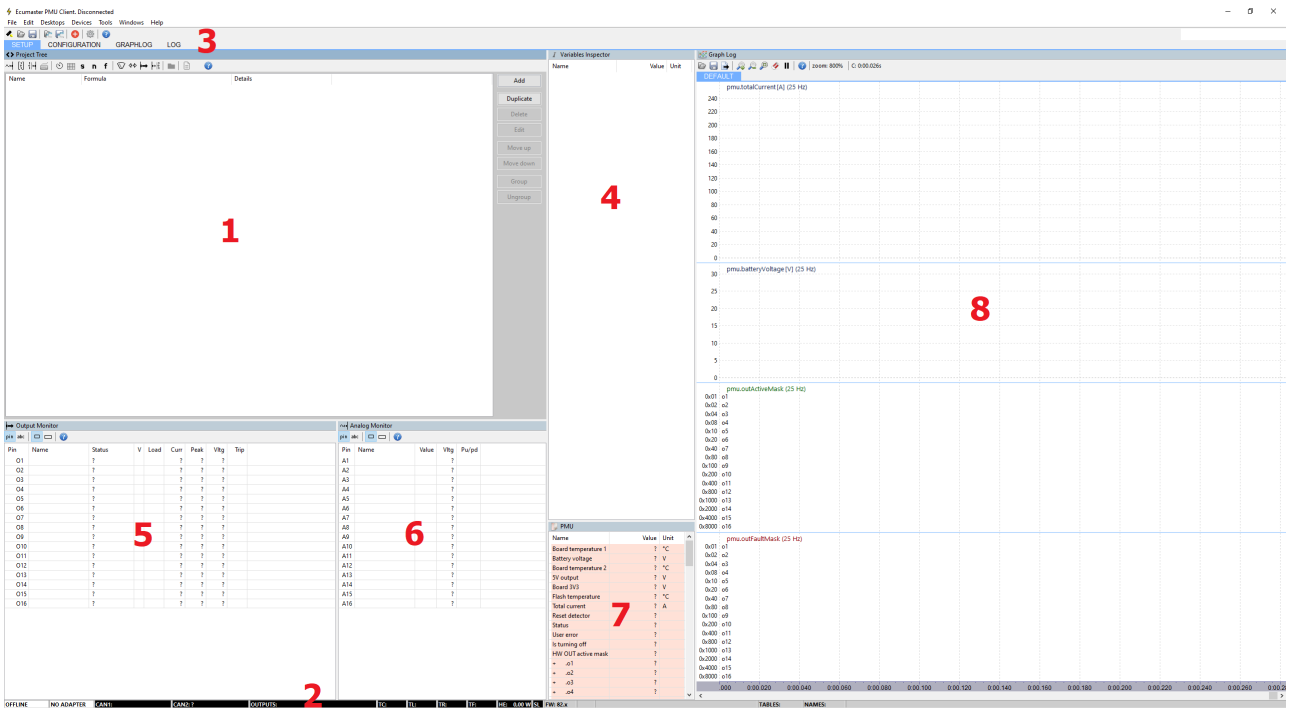


If there is no connection to the PMU and an attempt is made to open a project created using an older configuration than the one supported by the version of the Client software installed on the PC, a window will appear, prompting the user select a software configuration.



5.4. Appearance of the application

Once the application has been installed and started, the screen should look as in the picture below:



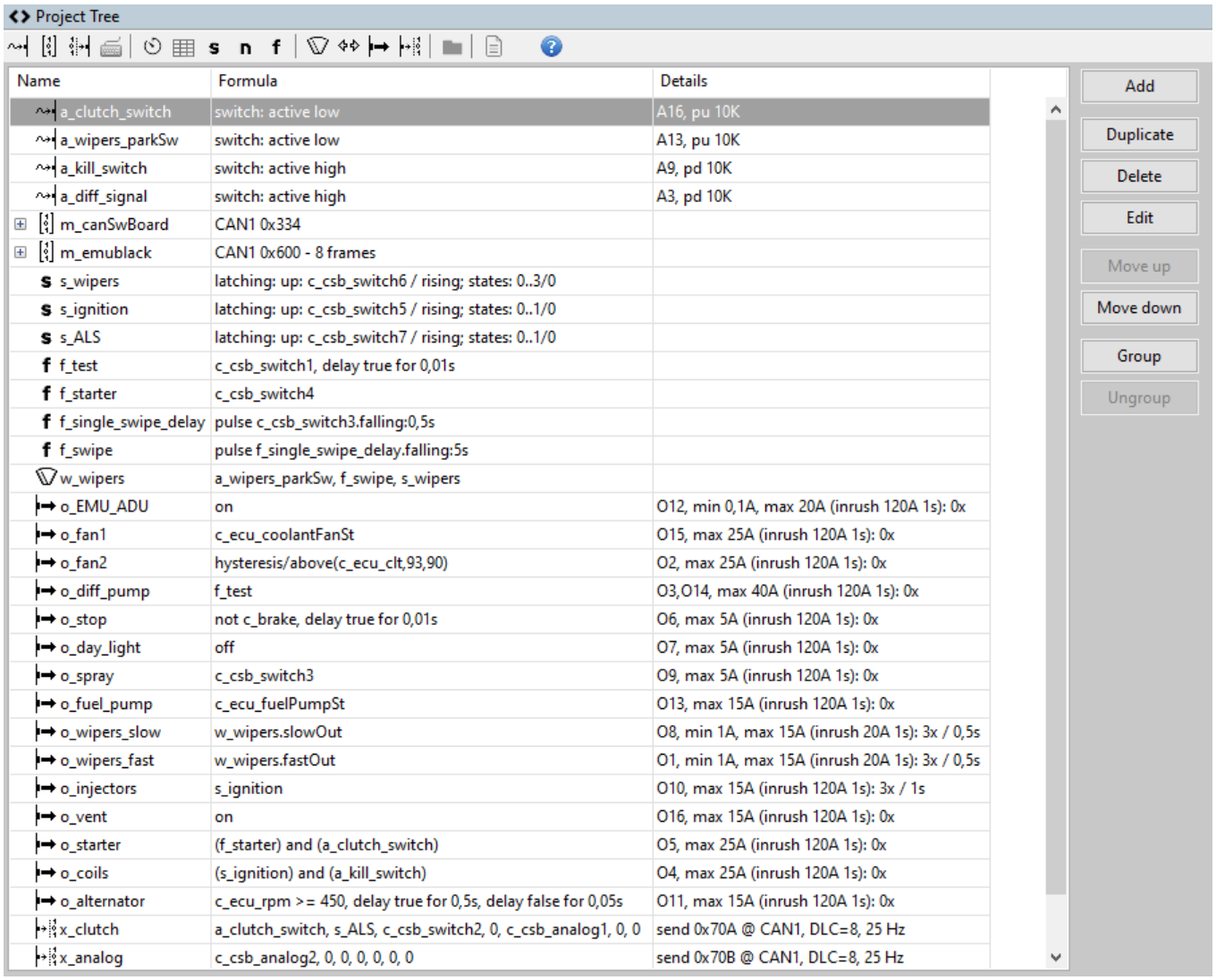
The ***Project Tree*** window is the main window where a project is defined (1). Use this window to define all project elements. Click ***Add*** to add a new element.

The following selection options will appear:

- ***Analog Input*** – an analog input, where all of its Parameters can be defined, e.g. the input number, the name of the variable, the type, pull-up etc.
- ***CANbus Message Object*** – a CAN message element, where an incoming CAN frame can be defined
- ***CANbus Input*** – an element defining variables incoming in CAN frames
- ***CANbus Keyboard*** – an element defining the keyboard
- ***PID Controller*** – an element used for controlling systems with feedback
- ***Timer*** – an element used for counting time
- ***Table*** – an element defining the table that can be used to transform data (e.g. transform an analog input voltage into temperature)
- ***Switch*** – an element defining a switch
- ***Number*** – an element defining a mathematical function in order to convert variable values
- ***Function*** – an element for creating complex logical functions
- ***Wipers Module*** – wiper control module consisting of two Power Outputs - for slow and fast wiper speed, and an Analog Input for the *Park Switch*
- ***Blinkers Module*** – blinkers control element, consisting of two Power Outputs for the left and right blinkers, requiring three inputs to control the Left Blinker, the Right Blinker, and the Hazard Lights.
- ***Power Output*** – an element used to configure power outputs for various devices (such as Fuel Pump, Fan, etc.). They can be turned on by default, controlled by functions, or triggered by another element such as an Analog Input. Every Power Output has over current, under current (both user configurable), and overheat protection.
- ***CANbus Export*** - an element for sending CAN frames with variable and constant values
- ***Group*** – a function for grouping elements; it allows a hierarchy to be introduced into a project in an easy way
- **Import .CANX/.DBC** file – this function is intended for downloading predefined CAN streams for different devices (e.g. EMU BLACK, MoTeC M1 etc.)

When adding different elements to the project, it is recommended to use the ***Group*** element, which allows elements to be grouped into logical sets. You should also make sure to assign correct names to elements. This will facilitate project management in the future. You can also duplicate project elements using the ***Duplicate*** button.

A sample project is shown below



To upload the current project to on-device flash memory, either use the Make Permanent button, or use **F2** keyboard button. The PMU status LED (*S*) will flash with orange color (See Device description *(on page 11)* section).

To save a copy of your current project on hard drive either use **Ctrl+S** keyboard shortcut, or use the Clients toolbar.

The status bar is another important element of the interface (2). It contains useful information on the status of the connected device. For a detailed description see the Status field *(on page 65)* chapter.

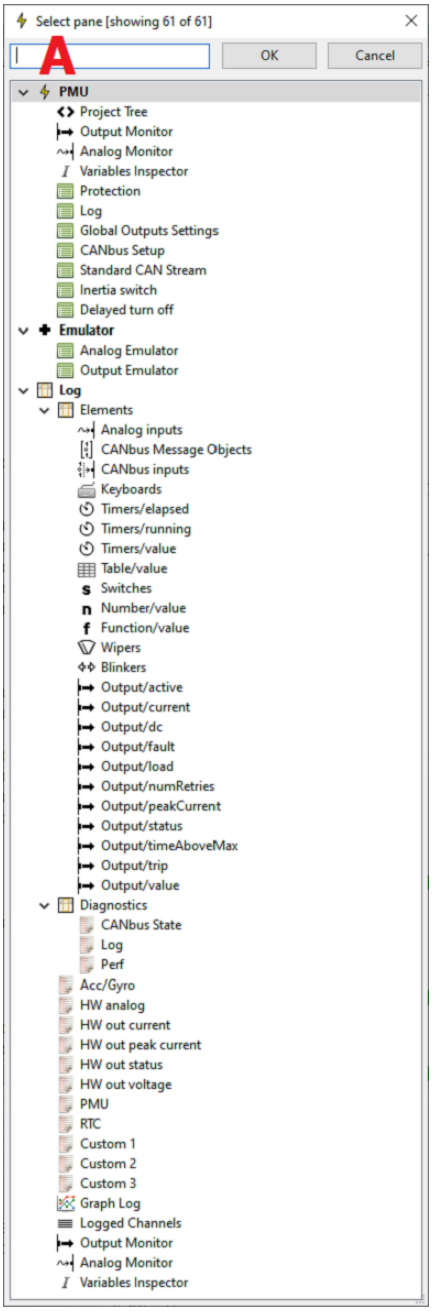
Tabs are a very important element of the application (3). They may be used to create your own sets of windows to facilitate and accelerate handling of the software. Right-clicking a tab brings up the following menu:

New Desktop	Create a new tab. A tab will always be created at the end.
Duplicate	Duplicate a tab. This option creates a new tab and copies the content of a highlighted tab into it.
Delete	Delete a tab
Rename	Rename a tab
Move Left	Move a tab to the left
Move Right	Move a tab to the right

Tabs are saved on the disc after pressing the **F2** (*Make permanent)* or upon leaving the application. They can also be saved to a file and transferred to another PC. To this end, select the *Desktops > Save desktops template* option.

To return to the default settings select *Desktops > Open desktop template* and load the *programDefault.pmulayout2.*

Panels are essential components of the interface for configuring the device. To add a new panel, simply press F9 or click the + icon on the toolbar. This action will open a window displaying all available panels.



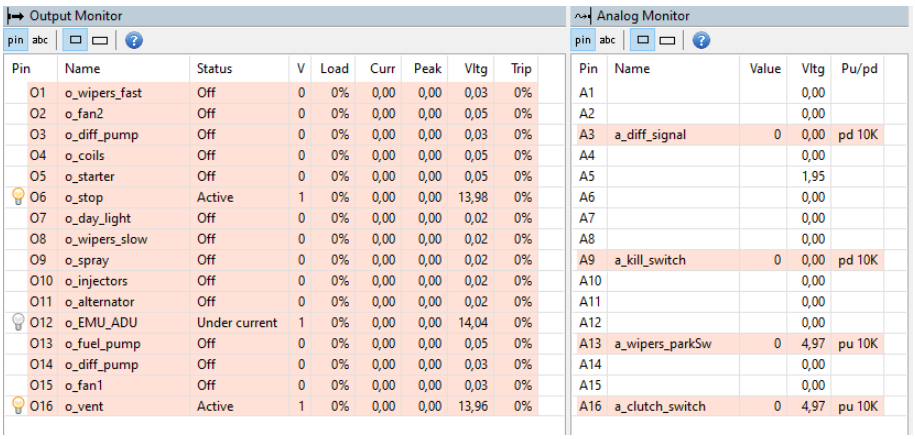
The search can be accelerated by entering the desired option in the filter field (A).

When you click on a selected panel, it will open within the desktop. New panels are automatically positioned on the right side of the desktop, but you can rearrange them by left-clicking on the title bar and dragging them to your desired location. To remove a panel from the desktop, simply right-click on its title bar, and a menu will appear. From there, select the 'Close panel' option to delete it.

There are different panel types. The first type are the configuration panels discussed above. Another type are panels for previewing variables, such as the ***Variables Inspector*** (4), monitoring output parameters ***Output Monitor*** (5), monitoring parameters of analog inputs ***Analog Monitor*** (6),

viewing ***PMU*** device parameters (7), or the ***Graph Log*** (8) showing the trace of logging channels over time.

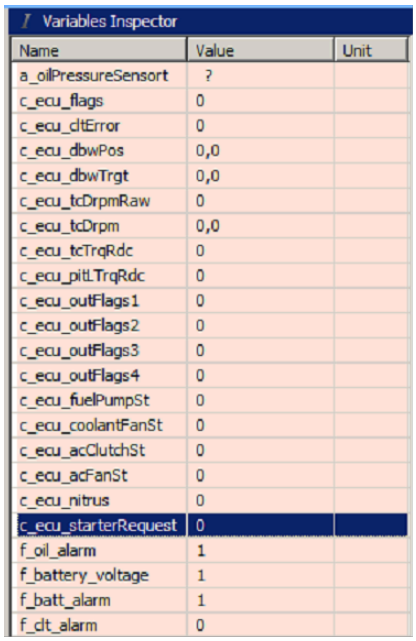
The ***Output Monitor*** and ***Analog Monitor*** panels are updated in real-time and show various parameters reported from individual terminals and the values of user-created elements. They can display all outputs/inputs or only those currently used.



Pin	Name	Status	V	Load	Curr	Peak	Vltg	Trip
O1	o_wipers_fast	Off	0	0%	0,00	0,00	0,03	0%
O2	o_fan2	Off	0	0%	0,00	0,00	0,05	0%
O3	o_diff_pump	Off	0	0%	0,00	0,00	0,03	0%
O4	o_coils	Off	0	0%	0,00	0,00	0,05	0%
O5	o_starter	Off	0	0%	0,00	0,00	0,05	0%
O6	o_stop	Active	1	0%	0,00	0,00	13,98	0%
O7	o_day_light	Off	0	0%	0,00	0,00	0,02	0%
O8	o_wipers_slow	Off	0	0%	0,00	0,00	0,02	0%
O9	o_spray	Off	0	0%	0,00	0,00	0,02	0%
O10	o_injectors	Off	0	0%	0,00	0,00	0,02	0%
O11	o_alternator	Off	0	0%	0,00	0,00	0,02	0%
O12	o_EMU_ADU	Under current	1	0%	0,00	0,00	14,04	0%
O13	o_fuel_pump	Off	0	0%	0,00	0,00	0,05	0%
O14	o_diff_pump	Off	0	0%	0,00	0,00	0,03	0%
O15	o_fan1	Off	0	0%	0,00	0,00	0,03	0%
O16	o_vent	Active	1	0%	0,00	0,00	13,96	0%

Pin	Name	Value	Vltg	Pu/pd
A1			0,00	
A2			0,00	
A3	a_diff_signal	0	0,00	pd 10K
A4			0,00	
A5			1,95	
A6			0,00	
A7			0,00	
A8			0,00	
A9	a_kill_switch	0	0,00	pd 10K
A10			0,00	
A11			0,00	
A12			0,00	
A13	a_wipers_parkSw	0	4,97	pu 10K
A14			0,00	
A15			0,00	
A16	a_clutch_switch	0	4,97	pu 10K

The ***Variable inspector*** panel is used to view values of variables defined in the device. These variables include e.g. *Functions*, *Numbers*, *CANbus inputs* etc.

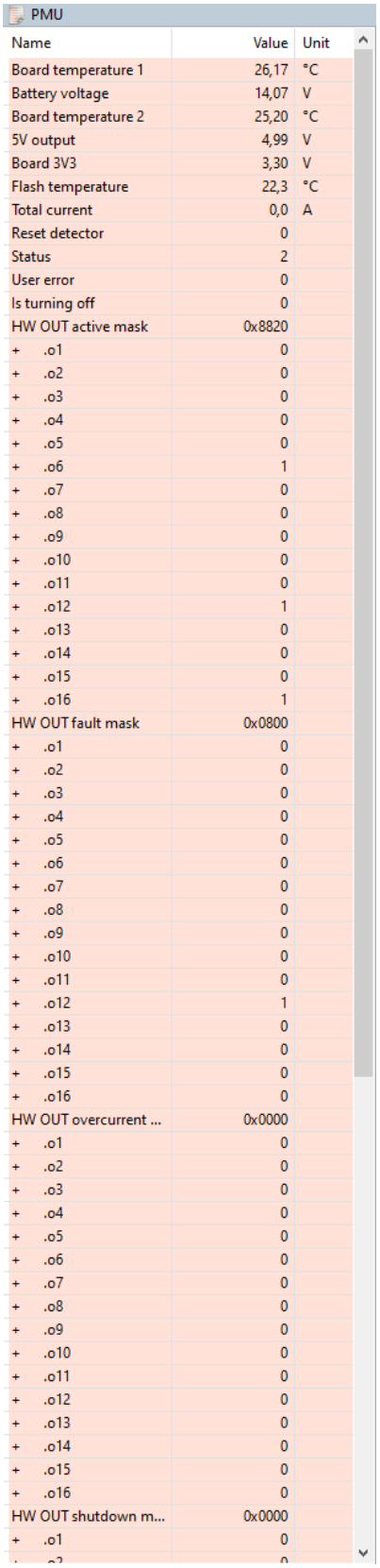


Name	Value	Unit
a_oilPressureSensort	?	
c_ecu_flags	0	
c_ecu_dtError	0	
c_ecu_dbwPos	0,0	
c_ecu_dbwTrgt	0,0	
c_ecu_tcDrpmRaw	0	
c_ecu_tcDrpm	0,0	
c_ecu_tcTrqRdc	0	
c_ecu_pitLTrqRdc	0	
c_ecu_outFlags1	0	
c_ecu_outFlags2	0	
c_ecu_outFlags3	0	
c_ecu_outFlags4	0	
c_ecu_fuelPumpSt	0	
c_ecu_coolantFanSt	0	
c_ecu_acClutchSt	0	
c_ecu_acFanSt	0	
c_ecu_nitrus	0	
c_ecu_starterRequest	0	
f_oil_alarm	1	
f_battery_voltage	1	
f_batt_alarm	1	
f_clt_alarm	0	

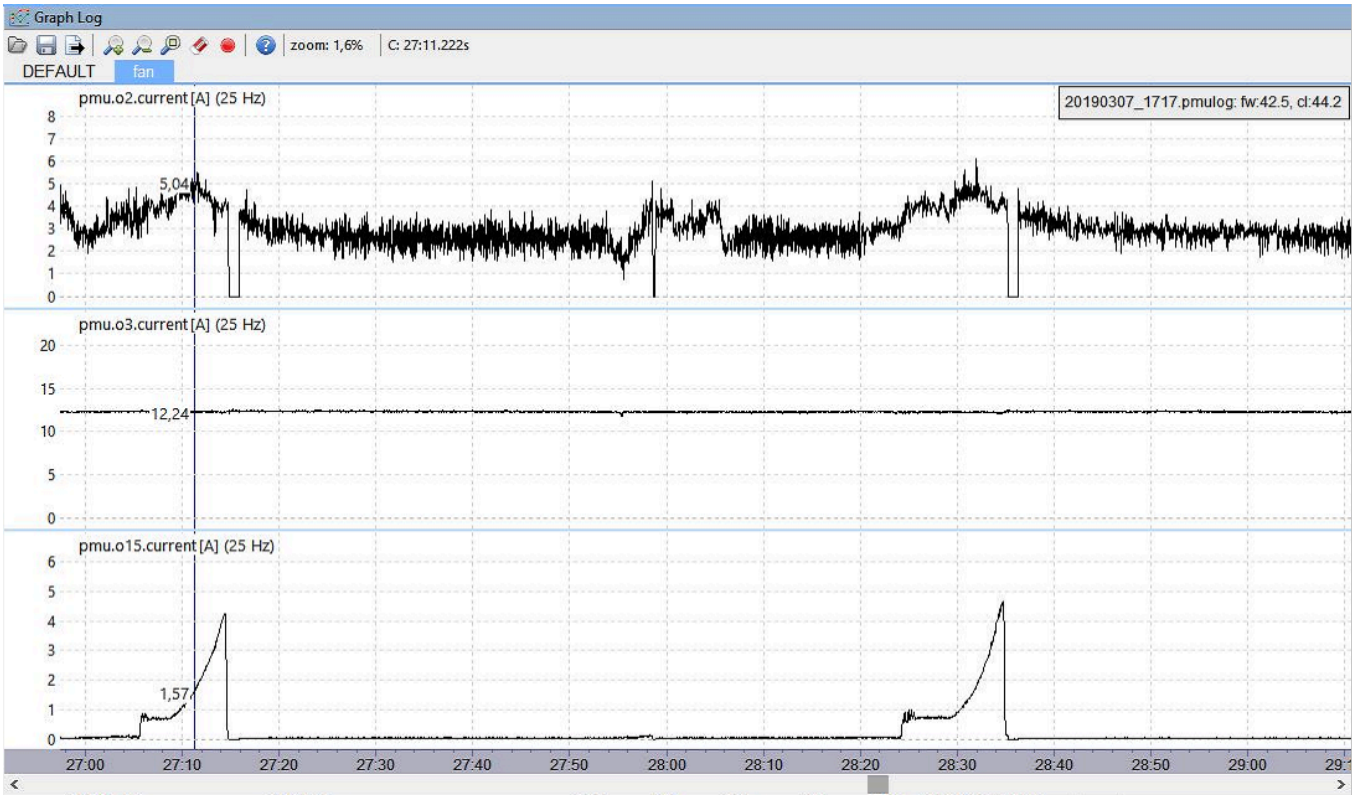
If a value is not a number but the ? symbol, it means that the logging function for this channel is deactivated. To activate logging or change the log frequency for a given channel, click the right mouse button on a given variable and select ***Set log frequency*** and then the desired frequency from the menu that pops up. If logging is suspended on the ***Graph Log*** (***Pause***), logging should be resumed (***Resume log***). It is possible to display values of variables of a given class by opening the following windows: ***Analog inputs*** (displays the analog input values), ***CANbus Message Objects***,

CANbus Inputs (variables from CAN bus), ***Keyboards*** (values of keyboard keys), ***Tables*** (values from the tables), ***Numbers*** (mathematical function values) or ***Functions*** (logical function values).

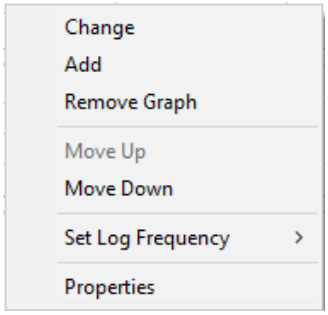
The PMU panel is used to view device parameters.



The logging channel ***Graph Log*** is another panel type. It can be used to display the desired channels.



If you right-click the log area, the following menu will appear.

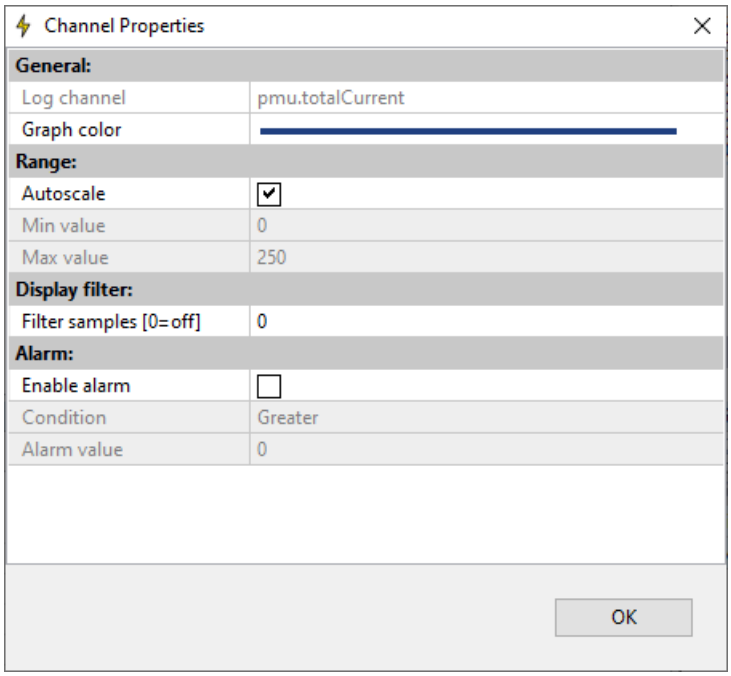


In this menu, new logging channels can be added (***Add***), existing logging channels can be removed (***Remove Graph***), the channel can be changed (***Change***), the position of the graph can be changed (***Move Up***, ***Move Down***), as well as the logging frequency (***Set log frequency***) and the display settings for the channel (***Properties***).

Another way to add a channel to the graph is to select a channel in the ***Output Monitor***, ***Analog Monitor***, ***Variable Inspector***, or any window from the ***Tree View*** → ***Log*** group, and then press the **Insert** key.

Just like the main desktop of the application, the log panel has tabs used to open different logging channel groups. These tabs work exactly the same way as the main desktop tabs.

By selecting ***Properties*** from the menu, you can access the display settings for a logging channel, such as the color of the displayed line (***Graph color***) or the range of values for a given channel (***Min*** and ***Max value***). The ***Autoscale*** option results in an automatic calculation of the values range based on the log data. It is also possible to use the ***Filter samples*** option, It determines the number of samples from which the value at a given point will be determined. The 0 value means no filtering.



The panel toolbar contains icons for:

- reading the log file from the drive (***Open log***);
- saving the log file to drive (***Save log***);
- saving the displayed channels in files (***Export to text format***) with the following formats:

Export to CSV	Export to a **CSV** file with the possibility of changing the settings: ***Export frequency:*** ***Same as channel*** – export frequency including actual logging frequency for individual channels, ***1Hz***, ***5Hz***, ***25Hz***, ***50Hz***, ***125Hz***, ***250Hz***, ***500Hz*** - selection of export frequency, the same for all exported channels. If the export frequency is higher than the logging frequency of the individual channels, the missing data will be completed by the previous value or by interpolation (***Interpolate***).

	Decimal / Column Separator: ***system / ";"*** – the decimal separator is either a comma or a full stop (depending on system settings), the column separator is a semicolon ***"." / ","*** – the decimal separator is a full stop, the column separator is a comma The resulting file may be displayed in a ***Preview*** window ***Table*** – presents data in the form of a table ***Raw output*** – shows the contents of the resulting .csv file
Export to PNG	Export to a **PNG** file

- ***Zoom In***, ***Zoom Out***, ***Zoom extents*** - change of scale;
- ***Clear log***;
- ***Pause / Resume log***

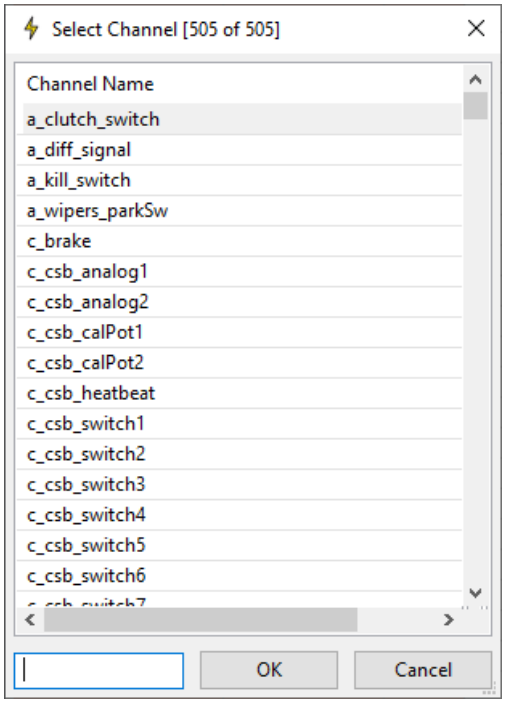
  Please be aware that stopping logging isn't limited to just the Graph Log; it affects all panels displaying logged data. When logging is paused, background logging ceases as well. This means that during the logging pause, data won't be saved to the device's memory and may be lost

The Graph Log panel can also be operated using the keyboard. The following is a list of shortcut keys.

Right / left arrow	Moves a log right or left. To move a log faster, hold the shift key while pressing the arrow keys.
Up arrow or q	Decreases the time base (zoom in)
Down arrow or a	Increases the time base (zoom out)
Z	Adjusts the zoom to a selected log area
Mouse scroll	Time base change
Left mouse button	Area selection
Middle mouse button	Moves the log area

A channel selection window appears when changing or adding a channel to a graph. You can speed up the channel search by entering the channel name in the bottom field of the window.

Available channels will then be displayed. For example, if you enter the word **pmu**, only channels containing the word **pmu** will be displayed.



Some of the functions are available in the application menu. The following is a description of all available Menu functions.

File	
Open project...	Open a previously saved project ***(CTRL + O)***
Save project	Save to last opened / saved file ***(CTRL + S)***
Save project as...	Save to a new file ***(CTRL + SHIFT + S)***
Merge project...	Enables the selection of specific elements from one project for integration into another. More information about merging projects is available in: Appendix A - How-to Merge Projects *(on page 145)*
Import log...	Import a log from a memory stick ***(SHIFT + F4)***
Show full screen	Full screen mode. This increases the screen space available to the application ***(CTRL + F)***.
Change target device...	Changes the device for which the project is created (PMU-16, PMU-24 or PMU AS). Offline mode only.
Upgrade firmware...	Change the internal software of a device
Restore to defaults	Restores a device to the default settings Deletes all settings

Make permanent	Saves changes to the Flash memory of a device Additionally, a file containing the current settings is saved to the *MyDocuments/PMU/DeviceName/QuickSave* ***(F2)***
Exit	Exit the application. The desktop arrangement is saved upon exiting ***(ALT + X)***.
Edit	
Undo	Undo the last operation performed ***(CTRL+Z)***
Redo	Redoing a previously undone operation ***(CTRL+Y)***
Show undo list	Displays a window with all operations performed
Desktops	
Restore desktops	Reads desktop configurations from the following file: *MyDocuments/PMU/Default/desktops.pmulayout2*
Store desktops	Saves desktop configurations to the following file: *MyDocuments/PMU/Default/desktops.pmulayout2*
Open desktops template...	Reads desktop configuration from a selected file. This allows to transfer configurations between PCs.
Save desktops template...	Saves desktop configurations to a selected file. This allows to transfer configurations between PCs.
Add new pane	Adds a new panel to the desktop ***(F9)***
Replace pane	Replaces an existing panel with another ***(SHIFT + F9)***
Switch to desktop	Switch to any selected desktop
Previous desktop	Switches to the previous desktop ***(CTRL+PgUp)***
Next desktop	Toggle to the next desktop ***(CTRL+PgDn)***
Devices	
Device selector	If one or more PMU devices are connected a panel enabling toggling between the devices will pop up. After switching to a device, the data between the PC and the device will be automatically synchronized. The names of particular devices can be found on the right hand side of the application's toolbar. The currently connected device is shown in bold.
Set device #n	Automatic toggling to the connected device no. #n. After switching to a device, the data between the PC and the device will be

	automatically synchronized. The names of particular devices can be found on the right hand side of the application's toolbar. The currently connected device is shown in bold type ***(CTRL+SHIFT+1 to 5)***.
Set device name	Assigns a name to a connected PMU device
Reboot device	Resets a connected device ***(CTRL + SHIFT +R)***
Reconnect	Re-establish communication with the device ***(CTRL + SHIFT + B)***
Receive Log File	Reads logs from a USB storage device connected to the PC ***(SHIFT +F4)***
Set Real Time Clock	Sets the real-time clock of PMU according to the current PC time. This time is used to date the files of a log saved into an external USB storage device.
Restart project in the PMU-16	Restarts the project in the PMU ***(F5)***
Generate pinout	It generates an .html file with device port documentation (it shows the terminals in use and the functions assigned to them). It also generates MOBs for CAN1 and CAN2
Tools	
Customize keys	Change the shortcut keys
Memory report	Displays a window with information on the current usage and the amount of free memory
Logged Channels	Displays a dialogue window with a list of all log channels and their frequency. Current size of the log data is visible at the bottom of the window (number of channels and bites) ***(F8)***
Project Tree	Displays the Project Tree panel ***(SHIFT+F7)***
Output Monitor	Displays the Output Monitor panel ***(SHIFT+F8)***
Analog Monitor	Displays the Analog Monitor panel ***(SHIFT + F10)***
Variables Inspector	Displays the variable monitoring panel ***(SHIFT + F11)***
Configuration Panel	Panel with all the configuration settings in one panel. More information is available in: Configuration *(on page 124)*
Export selected CANbus Message Objects as .CANX	Exports the CANbus messages currently selected in the *Project Tree* as a .canx file for use with other ECUMASTER software

Options	Displays a dialogue window with the application options • ***Save project (Ctrl+S) without dialog*** – save project without dialog • ***2D table size*** – size of 2D maps • ***2D tables color*** – color of map 2D • ***3D tables color scheme –*** color scheme for 3D maps • ***3D table appearance*** – 3D map graphical presentation (table and graph; only graph) • ***Auto save logs –*** automatic saving of logs onto the disc • ***Use mouse wheel to zoom on Graph log –*** log scaling function by means of the mouse wheel

6. Status field

The status field contains important information on the status of a connected device.

Connection status	Specifies, whether a device is connected
CAN adapter	Shows the CAN to USB interface type. The following interface types are supported: • **USBtoCAN** - ECUMASTER interface • **PCAN-USB** - Peak System interface • **Kvaser** - Kvaser interface
CAN 1 status	The status of the CAN1 bus from the USB to CAN interface
CAN 2 status	The status of the CAN2 bus
State of outputs	Output status
Total current	Total device current
Board temperature left	Device temperature on the left
Board temperature right	Device temperature on the right
Flash temperature	Flash memory temperature
Saving log in progress	Log auto-save status
Device firmware version (FW)	Firmware version
Device type	Device type: Standart, DL, AS

Used resources (Functions; Numbers; Operations)	The number of functions used
Free table memory (TABLES)	Memory available for 2D and 3D user maps
Free name memory (NAMES)	Memory available for element names (*CANbus Inputs*, *Functions* etc.)

If the CAN bus (1 or 2) status differs from OK, it means errors along the bus.

Explanation of CAN statuses of the ECUMASTER USBtoCAN adapter

Status	Typical cause of the problem
OK	CAN bus fully functional, no errors
stuff	Not all devices on the CAN bus send frames at the same speed (wrong speed of device along the CAN bus)
form	Not all devices on the CAN bus send frames at the same speed
bitrec	No terminator on the CAN bus
bitdom	CANL and CANH are short-circuited
bit	Two devices send frames with the same ID but with different DLC / DATA fields
ack	Interface is the only device on the CAN bus, no other devices Or: CANL or CANH is disconnected from other devices Or: CAN and CANH are interchanged
Offline	The programme operates in Offline mode - there is no access to the CAN bus

6.1. Output status

The PMU device has the ability to signal the condition of each Power Output. It is displayed on the device itself and in the PMU Client software. When a new Power Output is created, a special variable that represents the status of Power Output, called *output_name.status* is created as well.

The table shows all possible values for each o*.status:

Status value in Client	LED color	Status	Description
0	None	OFF	Output disabled (inactive)
1	Green	ACTIVE (ON)	Active output, working properly
2	Orange	UNDERCURRENT	The output is on, but the device connected to this output consumes less current than the set minimum or does not consume it at all (the output can be active: .active=1, or not active=0)
3	Red	OVERCURRENT	The output is turned off by the software because a software limit has been detected. The current over time is higher than the specified parameters (parameters may need to be tuned).
7	Red	THERMAL SHUTDOWN	The output is turned off by over-temperature (hardware limit exceeded). Thermal shutdown means that no matter what parameters you specify, your setup will not work. You will need to split the output into two pins, install an additional flyback diode, disable the *Soft Start* option, lower the *PWM frequency*, or disable PWM at all.

When the output status is greater or equal to 3, the output behaves as OVERCURRENT (shutdown occurs when software or hardware limits are exceeded).

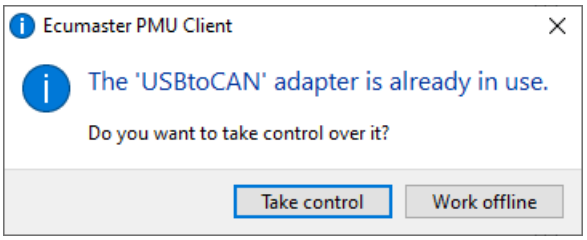
7. Switching the CAN adapter between PMU, ADU, and Light Client programs

ECUMASTER devices connected to the CAN bus communicate with the PC via a CAN adapter: USBtoCAN, Peak or Kvaser.
ECUMASTER software used to configure the devices: PMU Client, ADU Client and Light Client.

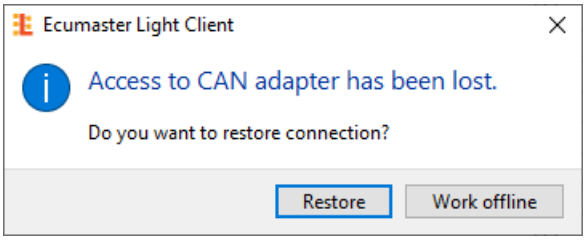
From software version:

- PMU Client - 44.0
- ADU Client - 52.4
- Light Client - 1.7

it is possible to switch easily between programmes without having to close one to open another.



When opening a program (e.g., PMU Client) while another program (e.g., Ecumaster Light Client) is already active, a prompt will automatically appear, asking whether the new program (PMU Client) should take control of the USBtoCAN adapter.



The program that was in use (Ecumaster Light Client) will then lose connection and become inactive.
When the program is called back to the foreground, a message will automatically appear indicating that the previously lost connection can be restored.

8. Configuration of inputs

The PMU device is equipped with 16 analog inputs. In the PMU-24, eight of these inputs can be used as outputs.

8.1. Analog Inputs

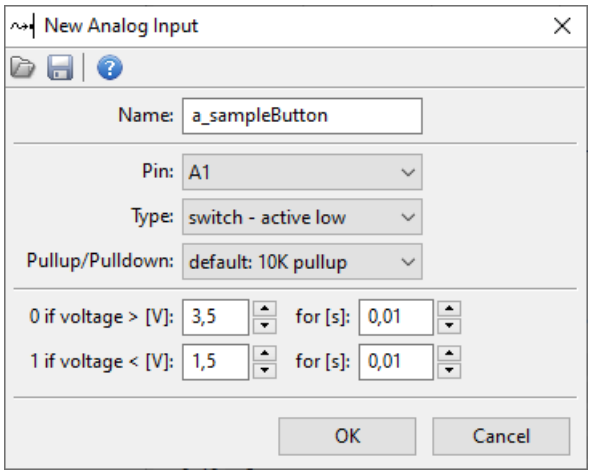
The analog inputs are used for measuring the voltage on the sensors (e.g. oil pressure sensor) or as inputs for the buttons. To add an analog input, add the ***Analog Input*** element in the ***Project Tree***.

The configuration window includes the following options:

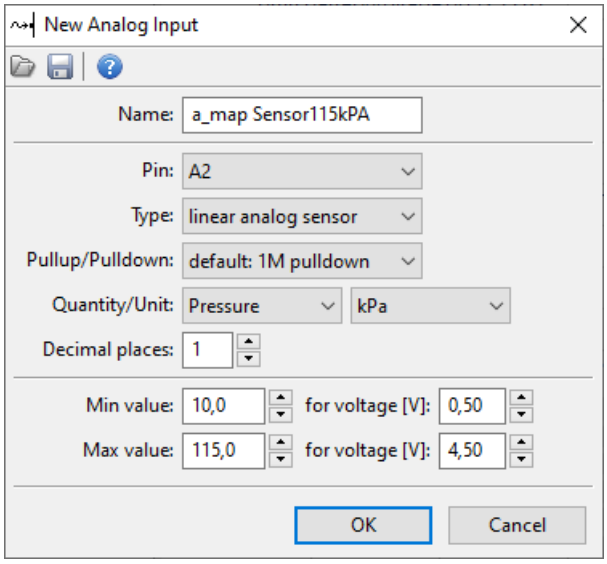
Parameter	Description
Name	Name of the analog input that will be used as the channel name in the project
Pin	Number of the analog input to which the configuration relates
Type	The function that the analog input is to perform: ***Switch – active low*** – the analog input will function as a switch (button) activated by a low state ***Switch – active high*** – the analog input will function as a switch (button) activated by a high state ***Rotary switch*** – the input assumes a value consistent with the position of the *Rotary switch*. The number of positions of the rotary switch is defined as ***Min value / Max value*** ***Linear analog sensor*** – the input is used for measuring voltage (as a [***Unit***] select [***Voltage***]) or any type of linear sensors (e.g. MAP sensor) ***Calibrated analog sensor*** – the input is used for measuring the values from sensors with a non-liner scale (e.g. temperature sensors NTC / PTC). A 2D map is used to define the values.
Pullup / Pulldown	A function for activating the internal 10K resistor connected to ground (Pull-down) or +5 V (Pull-up). These resistors are activated mainly when buttons are connected to analog inputs. For a button activated by a low state, you should activate Pull-up 10K and the button should connect the analog input

Parameter	Description
	to ground. For analog sensors or for measuring voltage, choose the 1M Pulldown option.
Quantity / Unit	For ***Linear*** and ***Calibrated analog sensor*** inputs, the parameter defines the physical value to be measured and its unit
Decimal places	Defines the number of the decimal places for the measured value for the ***Linear*** and ***Calibrated analog sensor*** input type
1 if voltage > [V]	Defines the voltage representing the value 1 for the ***Switch*** input type. For the condition to be satisfied, the voltage must be greater than the indicated value. The voltage value must be maintained for the time defined as the parameter "for [s]"
0 if voltage < [V]	Defines the voltage representing the value 0 for a ***Switch*** input type. In order for this condition to be fulfilled, the voltage needs to be smaller than that defined by time in the [s] field
Min value for voltage	For ***Linear analog sensor*** inputs this value defines the minimum sensor value for the defined Voltage
Max value for voltage	For ***Linear analog sensor*** inputs this value defines the maximum sensor value for the defined Voltage

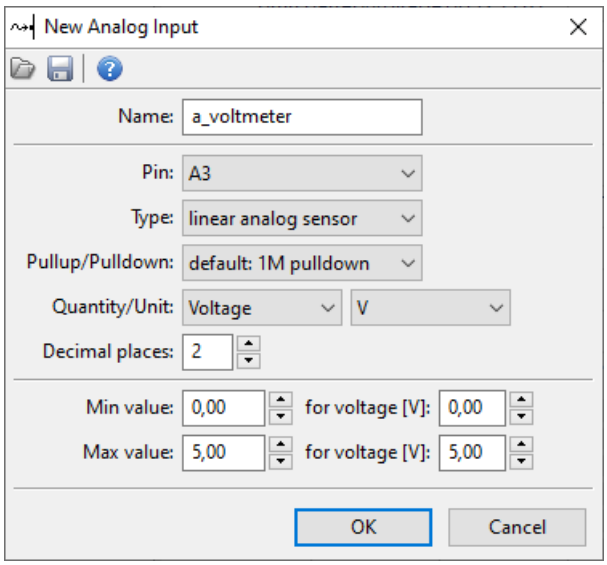
Example configurations:



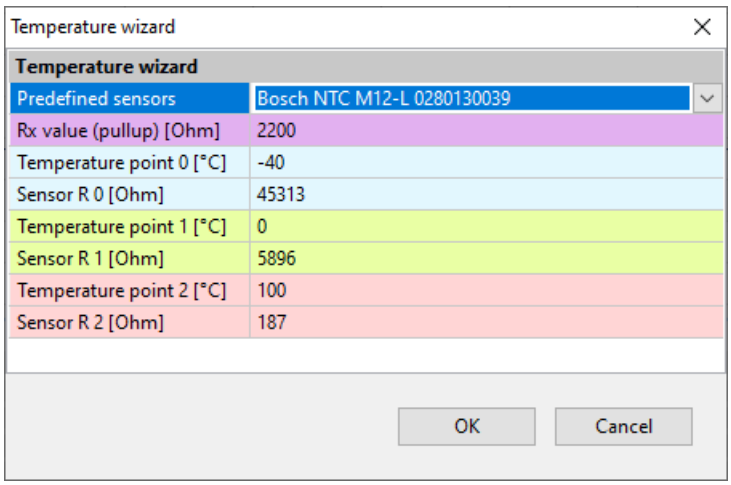
A configuration for a button connected to the ground and to the analog input 1. The *a_sampleButton* value will be 0 when the button is not pressed and 1 when it is.



A configuration for the pressure sensor in the intake manifold (MAP) connected to the analog input 2. The *a_mapSensor115kPa* value will assume values ranging from 10.0 kPa to 115.0 kPa



A configuration for measuring a voltage of 0-5 V for a signal connected to the analog input 3. The a_voltmeter value will assume values ranging from 0.00 V to 5.00 V.



The easiest way to calibrate non-linear temperature sensors is to use the *Wizard*. After pressing the *Wizard* button a window will pop up, allowing to define a sensor using three temperature values and three corresponding sensor resistance values. You can select a defined sensor in the ***Predefined sensor*** field.

The **Rx value** field indicates the value of the pull-up resistor used when connecting the sensor. If the sensor characterization data is correct, a 2D map describing the sensor characteristics will be generated automatically:

If a calibration map is created manually, values can be entered into individual cells. To change the table size, right click on it and select one of the *Modify bins* options.
The values of analog inputs can be viewed in the Analog Monitor panel, where you can find the channel value, its voltage and information about a pull-up resistor connected.

9. Configuration of outputs

The **PMU 16** device is equipped with 16 analog outputs (ten 25 A outputs and six 15 A outputs).
The **PMU 24** device additionally has the ability to use eight input terminals, which can be used as output pins (with a maximum continuous load of up to 7 A).
In the **PMU 16 Auto Sport** version, there are additionally six outputs shorted to ground (low-side) with a load capacity of 1 A each.
The outputs are used to control external devices. The PMU has separate elements with a built-in appropriate strategy for controlling the wipers and blinkers, which makes it very easy for the user to configure these devices.

9.1. Power Output

Power Outputs are elements that control external devices. You can set up the *Minimum Current*, *Maximum Current*, *Inrush Current*, *Inrush Time*, *PWM*, and the mechanisms to switch the Power Output on or off.

To add an analog output, you need to add the ***Power Output*** element in the ***Project Tree***.

The configuration window consists of the following options:

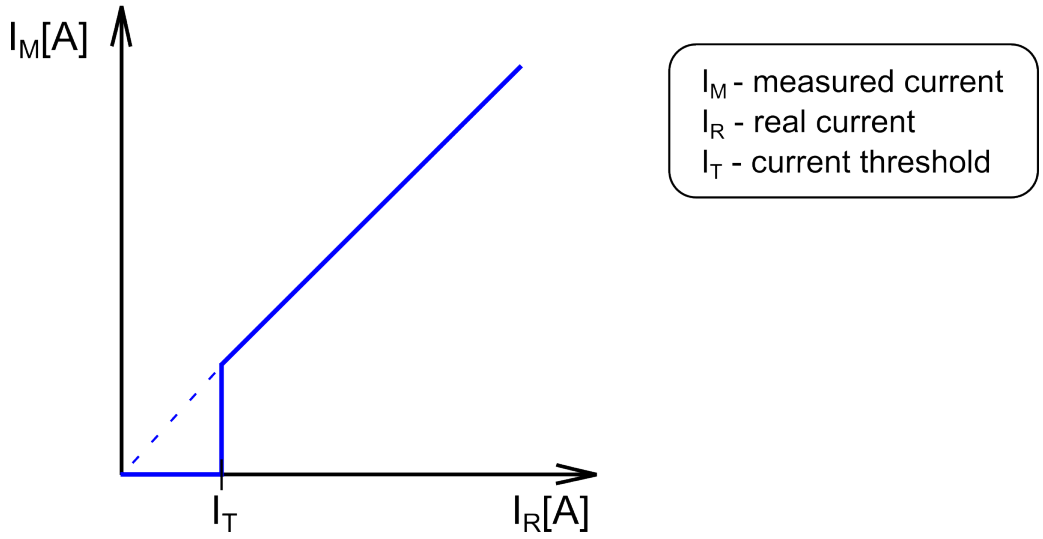
Parameter	Description
Name	Name of the output that will be used as the channel name in the project
Pin	Depending on the current demand, select the number of pins to create the output (single, double, triple) and select the number/s of outputs to which the configuration applies. Up to 3 pins can be used for a maximum output current of 75 A. Pins used in parallel must be of the same type (You cannot use the 25 A and 15 A pins in parallel).
Inrush Current	The maximum current used to start the device during *Inrush Time*. *Inrush current* is higher than *Max Current*. If the device during start-up draws more current than *Inrush Current*, or for a longer period of time than *Inrush Time*, the output status will change to OVERCURRENT and the output will turn off.
Inrush Time	The time in which the current needed to start the device can be greater than *Max Current*, but not greater than *Inrush Current*. (The maximum *Inrush Time* depends on the selected *Inrush Current* and output pin current).
Max Current	Maximum continuous current for the configured output during device operation (after start-up). If this value is exceeded, the output is turned off and the output status changes to OVERCURRENT.
Min Current	Minimum current for the configured output. If the device draws less than *Min Current* or draws no current[1], the output status will change to UNDERCURRENT, but the output will still operate.
Retry count	Number of retries to turn on the output when the current limit is exceeded.
Retry every	The time interval after which the output is switched on again for exceeding the current limit.
Retry forever	Continuous retry attempts to switch on the output at intervals specified by 'Retry every' for exceeding the current limit.

Parameter	Description
PWM configuration	Checking the box allows you to configure pulse width modulation for the 25 A output pins.
Frequency	PWM frequencies
Soft start	*Soft start* on 25 A outputs with *Duration*
Duty cycle	Configure the duty cycle by the percentage of time the output power is on. This cycle can be set to a fixed value or channel controlled (*DC control*).
Default	Default *Power Output* control – **On/Off**. The output is on or off.
Channel	Output control channel
Formula	The formula of the logical function controlling the output.

[1]When measuring very low currents from the outputs, it's important to note that each output has a specific threshold current. Below this threshold, the measured current is clipped to zero, indicated as I_T on the plot.

For the 40A and 25A outputs, this threshold current ranges from 0 to 2A, typically averaging at 0.5A. Meanwhile, for the 15A and 7A outputs, it ranges from 0 to 0.2A.

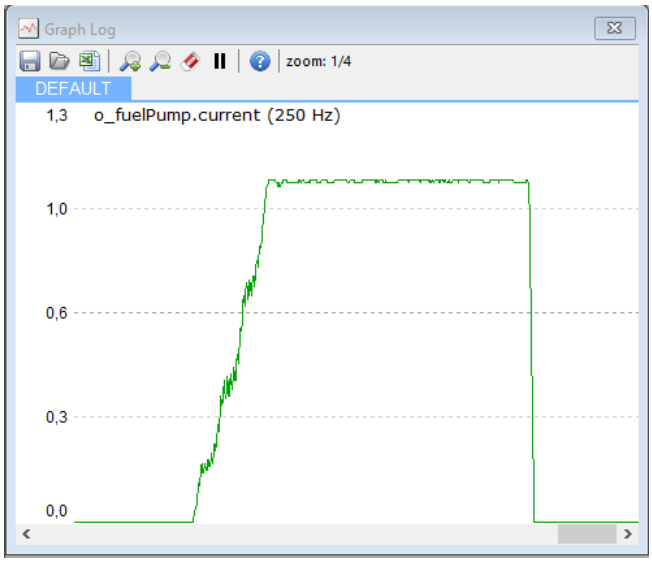
Despite the measured value being clipped to zero, it's essential to understand that the actual current output remains precisely set.



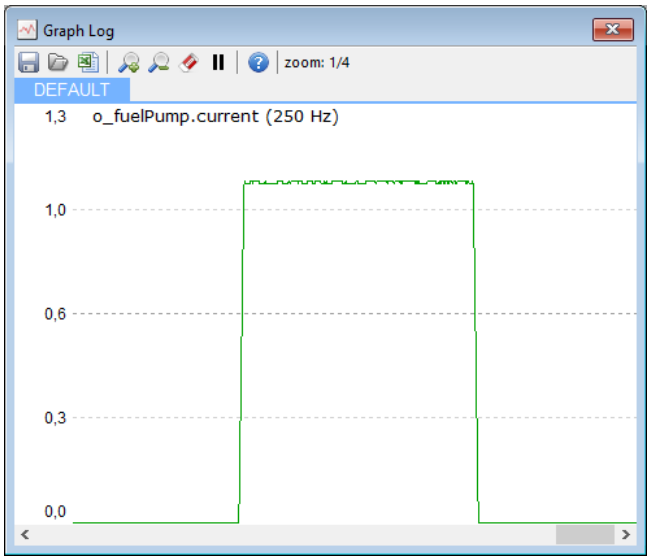
9.2. Soft Start and PWM Duty Cycle

The PMU has the ability to enable a *Soft Start* on the 25 A output pins. *Soft Start* is used in devices that consume a lot of current during start-up, exceeding the maximum *Inrush Current* value for a given output.

The graphs below show the difference between *Soft Start* disabled and enabled.



Soft Start Enabled

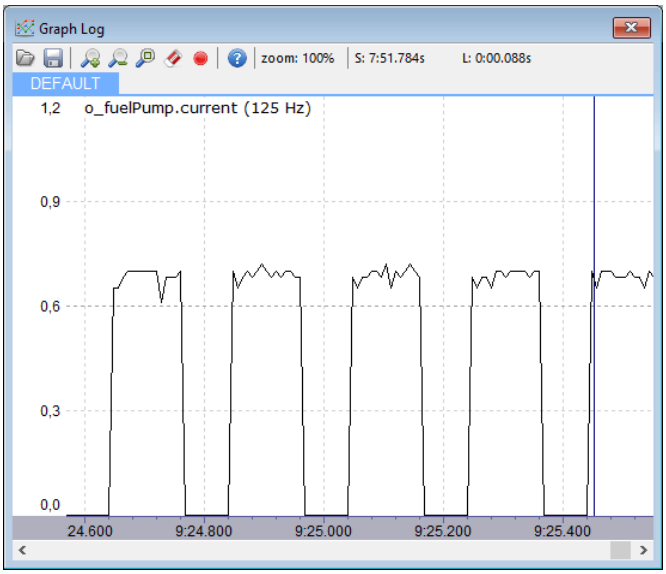


Soft Start Disabled

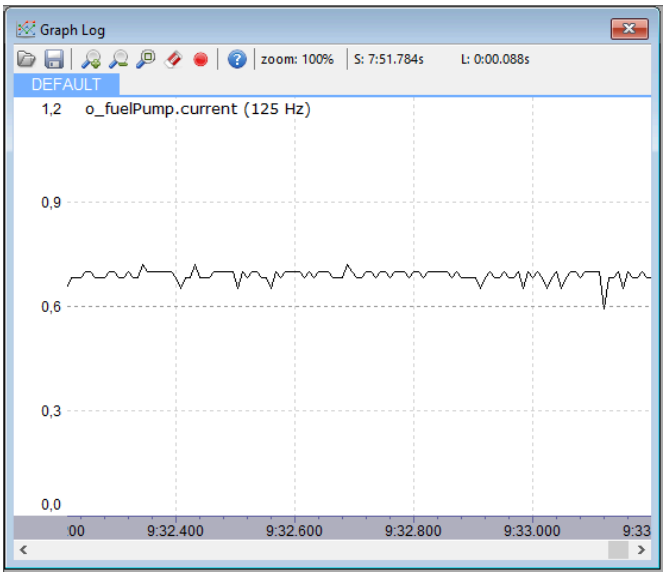
The *Duty Cycle* (*%DC*) represents the percentage of time the *Power Output* is turned on. If the *Duty Cycle* is set to 50% DC, the *Power Output* will be turned on for ½ of the period.

The graphs below show the difference between PWM disabled and PWM with a 70% Duty Cycle.

In this case (PWM control), the fuel pump will operate at a lower speed, as the current is supplied only in 70% of the entire cycle compared to its constant current supply.



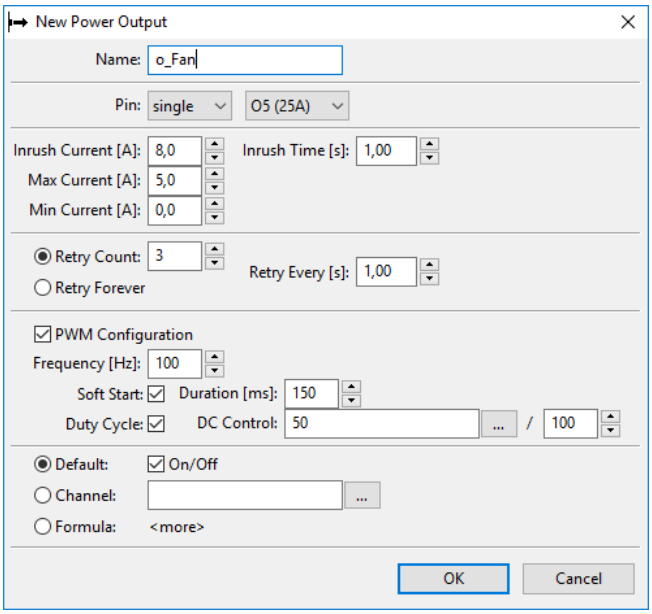
PWM Enabled with Duty Cycle = 70%



PWM Disabled

The duration of the pulse duty cycle can be set to a fixed value that will be the same each time the output is activated, or controlled by logic written with available tools, e.g. by increasing the % DC.

PMU Client Configuration Example with a fixed 100 Hz cycle and 50% DC pulse duty cycle:



9.3. Wipers Module

The PMU has a separate module for configuring wipers with a complex control strategy built in, which is a great convenience for the user. For proper operation of the wipers in the *Wipers Module*, it is enough to configure the parameters described in the table below.

Parameter		Description
Name		Wiper module name
Output pin - Slow		Output pin for low wiper speed. **O8** output is set by default. A dedicated output allows you to use the *Park* function (*Park switch*) by using an additional transistor that reverses the output polarity (low voltage side circuit).
Output pin - Fast		Output pin for high speed wiper. Output pin: *multiple* allows you to select up to 3 pins.
Settings		All settings for the output, such as: *Inrush-*, *max-*, *min-current*, *Inrush time*, *Retry*, and *PWM* configuration described in the "*Power Output*" chapter.
Park switch		Depending on the construction of the Park switch sensor, select the trigger type: • *active low switch* - to a low state • *active high switch* - to a high state • *custom channel* – a channel with the appropriate logic created
Pin		*Park switch* sensor input pin (for the *active low switch* and *active high switch* types).
Pullup/Pulldown		For the *active low* switch type, 10K pullup is set by default. For the *active high* switch, 10K pulldown is set by default.
Channel		The channel that defines the *Park switch* for the *custom channel* type
Brake delay time		Wiper brake delay time starting from the signal appearance
Brake time		Wiper brake duration
Single swipe	***Input channel***	A channel that triggers a single wiper cycle
Multiple swipes	***Input channel***	A channel that triggers the wiper's continuous operation mode
	#0 - #7	Continuous operation modes
	Slow Delay	Continuous operation mode with breaks

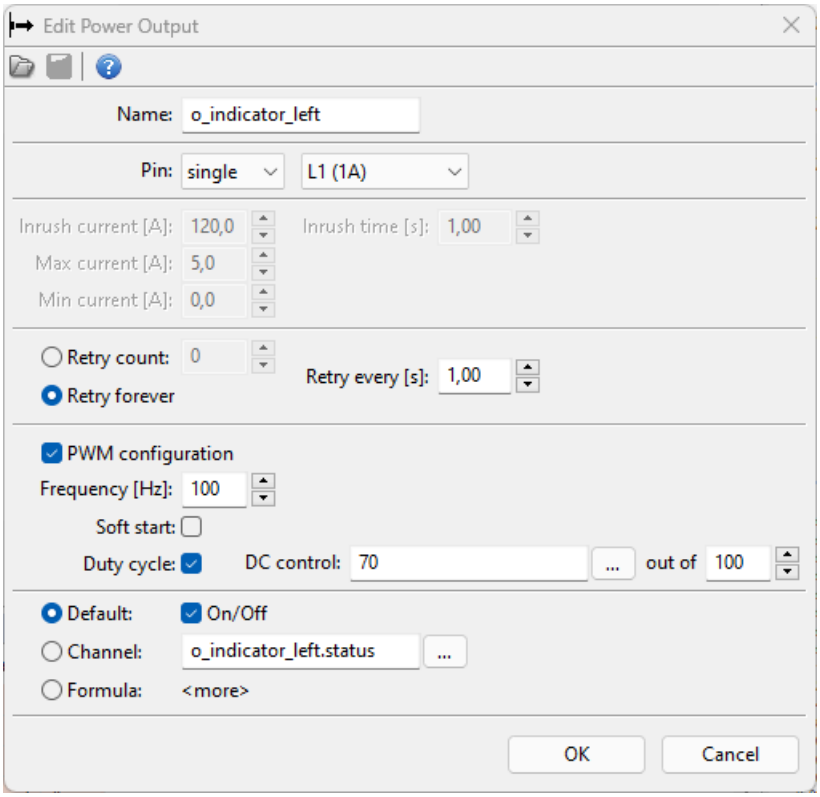
9.4. Blinkers Module

As with the wipers, in the PMU there is also a separate module for controlling the blinkers. The configuration is facilitated by the built-in control logic. You only need to assign the appropriate outputs and input channels.

Parameter		Description
Name		Blinkers module name
Output Pins	***Left Blinker***	The output pin for the left blinker
	Right Blinker	The output pin for the right blinker
Input Channels	***Left Blinker***	Input channel that activates the left blinker
	Right Blinker	Input channel that activates the right blinker
	Hazard Lights	Input channel that activates hazard lights
	Input Type	For each of the input channels, you must specify the input type: ***High level*** – after detecting a value other than 0 (high level), the blinker is turned on, the appearance of a value equal to 0 turns off the blinker ***Rising Edge*** – the status is switched after the appearance of a rising edge, i.e. if the blinkers are off, the appearance of the first rising edge will turn them on, and the next one will turn them off
Settings	***Flash Time***	Specifying in [s] how long the light should be on

9.5. Low-side outputs

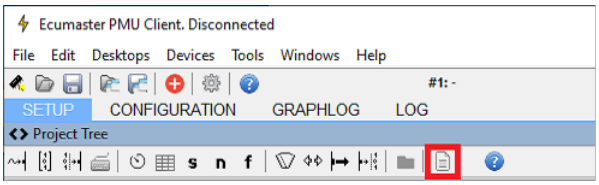
The Low-side outputs in the PMU device in the Auto Sport version are configured in the same way as the standard outputs. The only difference is the inability to set the minimum, maximum, and startup current for these outputs. These options in the ***Power Output*** element for the low-side outputs are inactive.



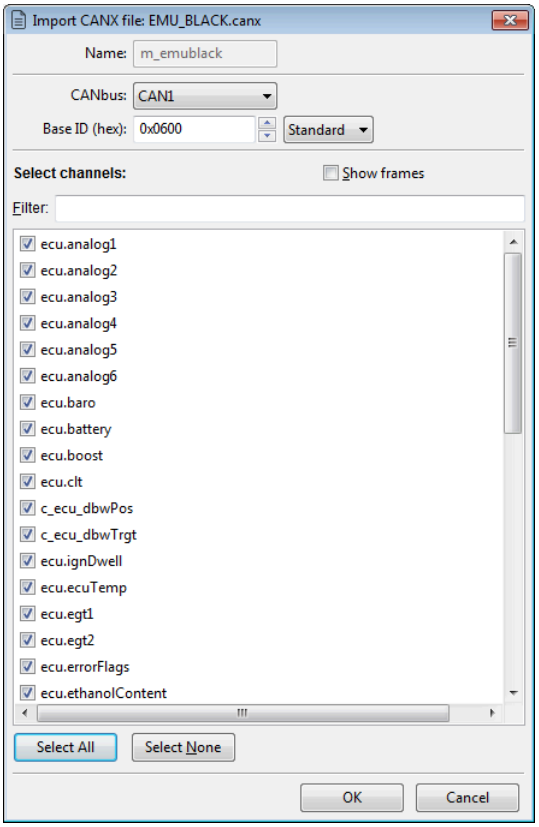
10. Working with CAN buses in PMU

10.1. Using pre-defined streams from *.CANX* and *.DBC* files

The simplest way to work with the CAN bus is to use pre-defined streams from ***.CANX*** and ***.DBC*** files. Streams in files with the *CANX* extension were supplied together with the PMU Client software.



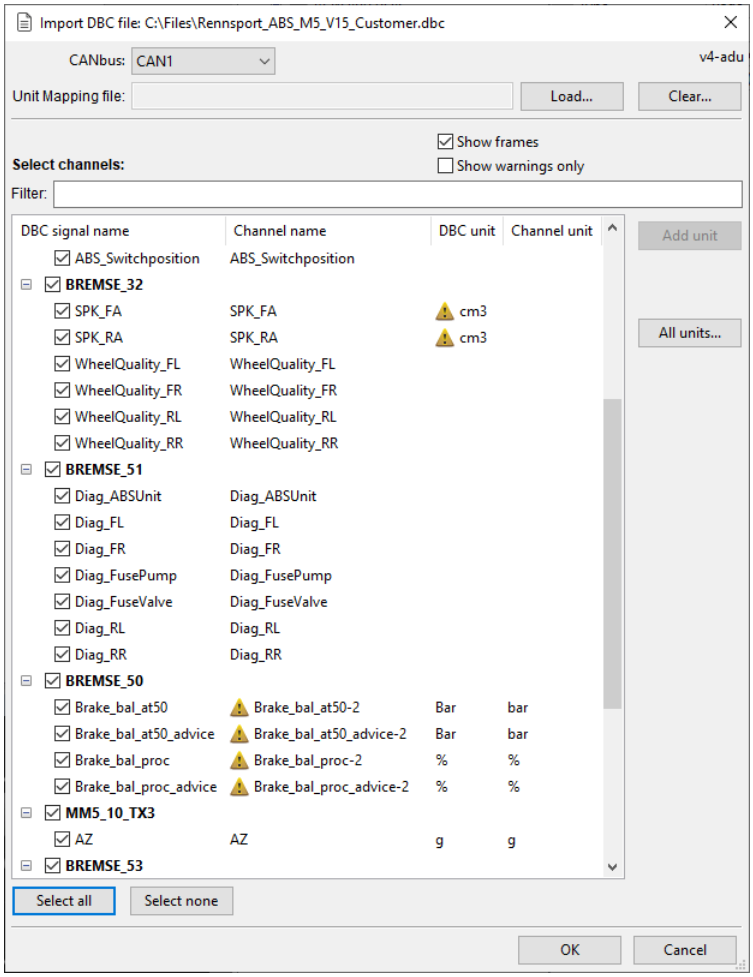
Having selected an icon from the taskbar or choosing the ***Project Tree*** > ***Add*** > ***Import .CANX/.DBC file*** option and pointing to the *.CANX* or *.DBC* file, a window with the import settings will open.



First, choose the CAN bus from which data will be received. The PMU device has two CAN buses: CAN1 and CAN2. Next select the channels to be imported. You can use the filter to select individual channels or select all using the '***Select all***' button. You should remember that the PMU device supports up to 150 CAN channels on both buses.

The following warnings may appear when importing a *.DBC* file:

- about units not defined in the PMU;
- about channels already existing under a particular name.



After confirming with the OK button, the selected channels will be added to the ***Project Tree***. In addition, one or more ***CANbus Massage Object*** responsible for receiving frame groups will be created.

10.2. Own CAN streams – CANbus Message Object

Access to a CAN bus in the PMU device is completely open. You can create your own streams or modify the existing ones supplied together with the programme.

Configuration begins with the creation of a ***CANbus Message Object*** (***MOB***) element in ***Project Tree***. Each MOB receives 1, 2, 4 or 8 CAN frames. After choosing a CAN bus you should select a ***Base ID***, as well as the ***Type*** and the number of received frames – the ***Size*** parameter. If the device in connected, a preview of the stream in real time (***Live Capture***) will be displayed. It facilitates diagnostics and speeds up work.

Important:

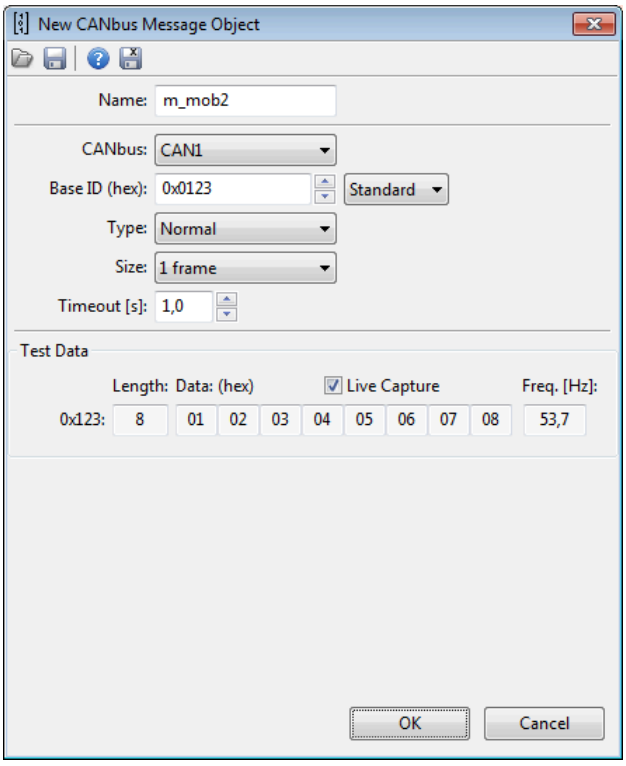
For *Live Capture* preview to work properly, active logging is required - logging cannot remain in pause mode!

Important:

Frame CAN IDs in the PMU Client are always presented in hexadecimal notation (they usually begin with the *0x* prefix, which is a symbol of the hexadecimal notation).

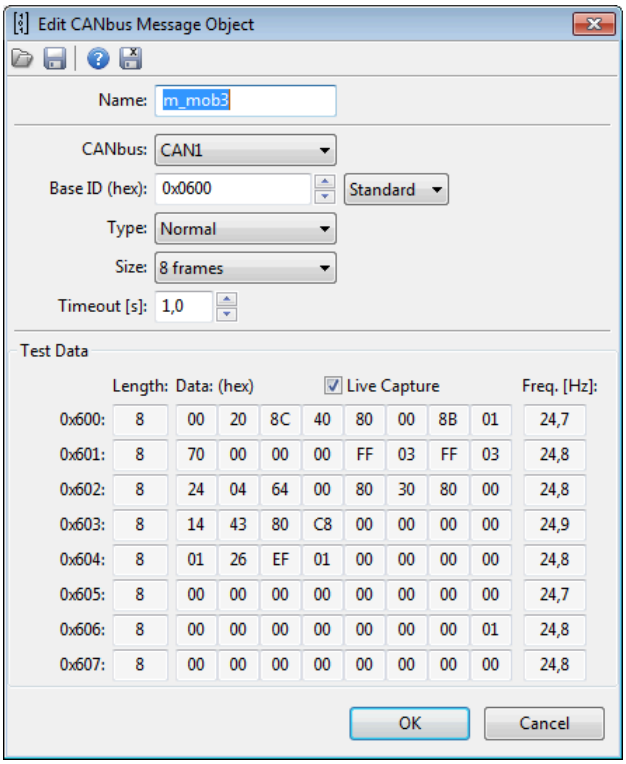
The following are examples of MOB configurations.

Receiving 1 frame ID 0x123 Standard



- **Base ID:** 0x123 Standard
- **Type:** Normal
- **Size:** 1 frame

Receiving 8 frames from the range of ID 0x600–0x607 Standard



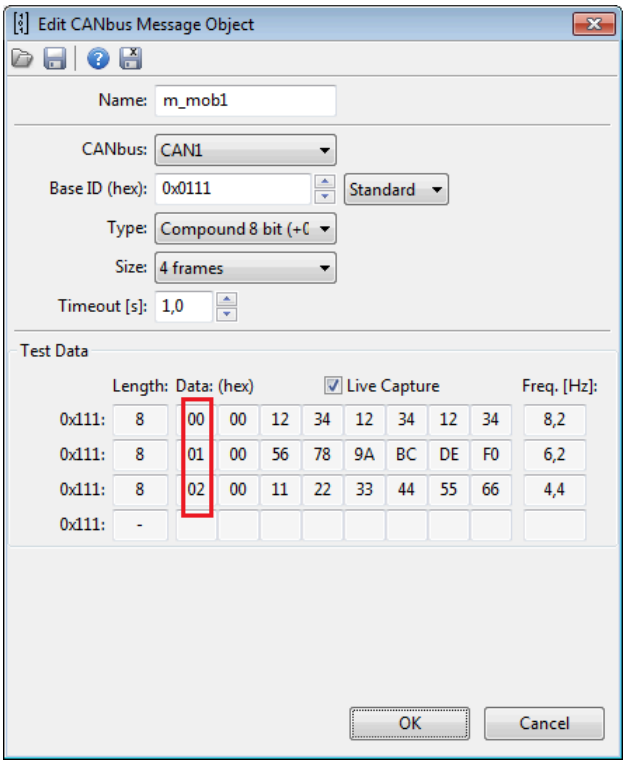
- **Base ID:** 0x600 Standard
- **Type:** Normal
- **Size:** 8 frames

Remember that for regular frames (Type: ***Normal***) Base ID must be divisible by (***Size***) without a remainder. For example, using one MOB, it is possible to receive 8 frames in the range 0x600-607. However, it is not permitted to receive frames in the range 0x601-608 using one MOB. If this is the case, this range should be divided into two MOBs of 0x600 and 0x608.

To check if the Base ID is suitable it is enough to verify the last digit in the hexadecimal notation:

- If the number is 0 or 8, then Base ID is divisible by 8 (Size: 8)
- If the number is 0, 4, 8, C, then Base ID is divisible by 4 (Size: 4)
- If the number is 0, 2, 4, 6, 8, A, C, E, then Base ID is divisible by 2 (Size: 2)
- Each number is divisible by 1 (Size: 1)

Receiving three Compound 8 bit frames from ID 0x111 Standard

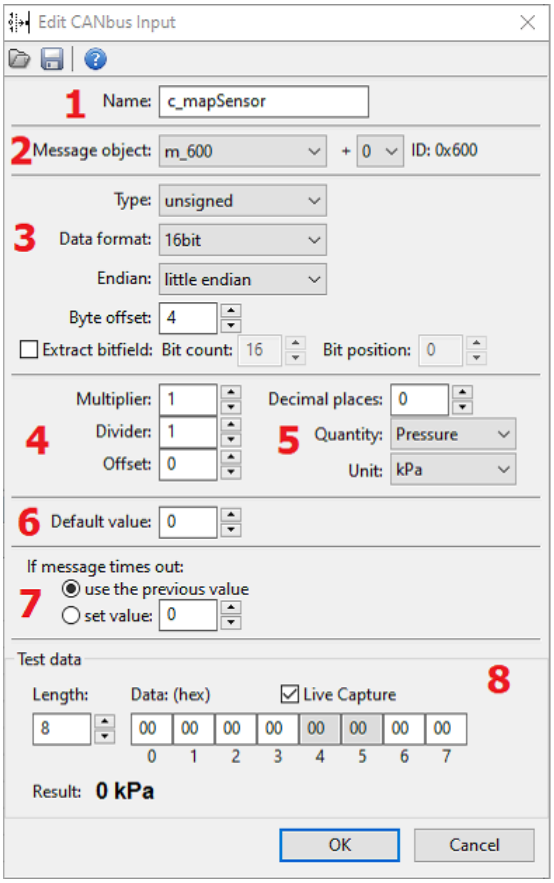


- **Base ID:** 0x111 Standard
- **Type:** Compound 8 bit (+0)
- **Size: 4** frames

For *Compound* frames, the Base ID address doesn't have to be divisible by Size. This results from the fact that the communication takes place using only one CAN ID.
Compound frames (for Multiplexer types '4 bit', '8 bit' or '16 bit') contain an index in the first 4, 8, or 16 bits (according to Compound type respectively). For Multiplexer type 'Custom' index length (in bits), position and endianness is freely defined. In the dialogue window nearby, you can see the first byte in a sequence 00, 01, 02.

10.3. Own CAN streams – CANbus Input

After a ***CAN Message Object*** has been created, you can start defining ***CANbus Input*** channels. First, choose a unique ***Name*** (1) so that it can be identified.



Secondly, select the previously prepared MOB, and select offset in frames from the drop-down list marked "**+**". The scope of this parameter depends on the selected ***Size*** in the parameters of the used ***Message object*** (2).

Next step is setting the ***Byte offset*** parameter (3). It marks the location of the values in question in the CAN frame (0 – 7).

You should select the interpretation of the number:

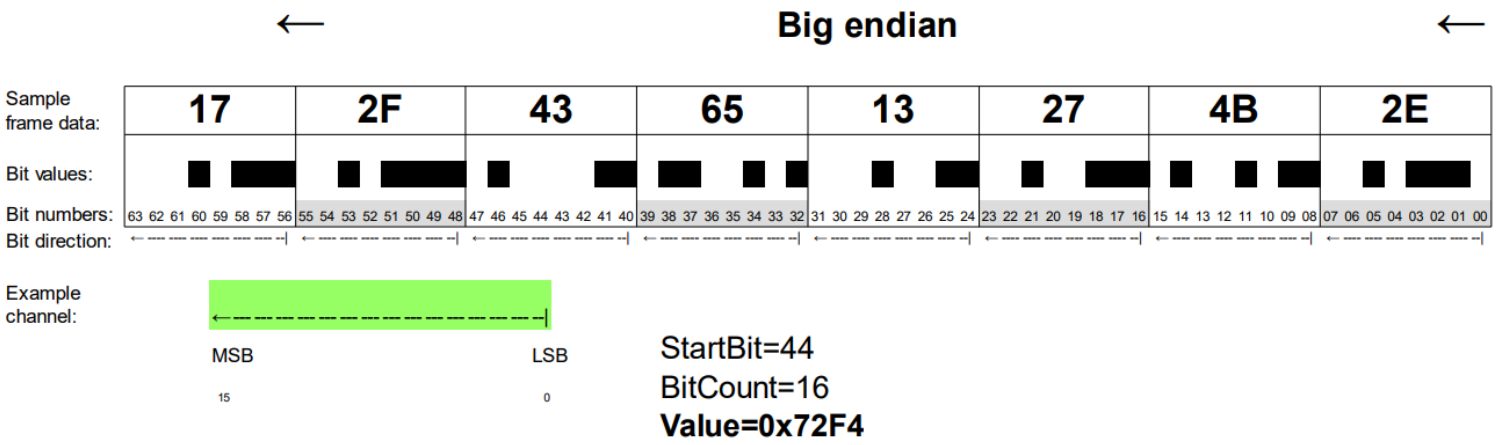
- **signed/unsigned**
  - ***Signed*** – a number with a sign (it can receive positive and negative values, as well as zero). An example of such value is the value from the engine coolant temperature sensor.
  - ***Unsigned*** – positive numbers or zero. For example engine speed (RMP).
- ***8 bit / 16 bit*** – number width in bits; 1 byte or 2 bytes, respectively
  - **signed** 8 bit – range of numbers -128–127
  - **unsigned** 8 bit – range of numbers 0–255
  - **signed** 16 bit – range of numbers -32768–32767
  - **unsigned** 16 bit – range of numbers 0– 65535

- ***big endian / little endian*** – the "sequence" of bytes for 16–bit numbers. It shows how a number stored in two consecutive bytes shall be interpreted. E.g. numbers 0x12, 0x34 can be interpreted as 0x1234 for the ***big endian*** or 0x3412 for the ***little endian***.
- You can also define "***Extract bitfield***", i.e. take only a part of an 8- or 16-bit number. For example, to check the setting of a bit of a 0x80 mask the following settings should be used: Bit count: 1, Bit position: 7.

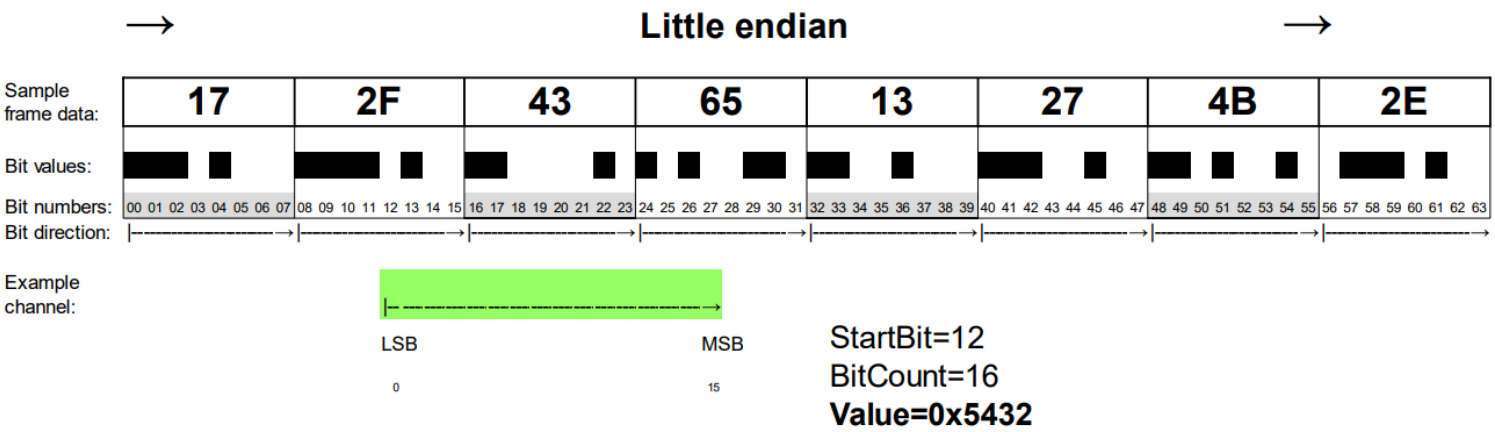
The ***Custom*** data format allows the exact width and position of the information stored in the CAN frame to be determined. The information can occupy a maximum of 16 bits, but these can be taken from 3 bytes. The bit numbering is compatible with *Kvaser Database Editor 2*.

The ***Bit count*** parameter determines how many bits (1-16 bits) of information are present.
The ***Start bit*** parameter specifies the bit number at which the information in the CAN frame starts.

Example for Big Endian, custom Data format:



Example for Little Endian, custom Data format:



The next step is to scale /offset the values by ***Decimal places*** (4).

The “raw” value interpreted in Format (3) field can be scaled.
For example, Lambda in the EMU stream is saved as the value 0..255, where:

- raw value 0 means Lambda = 0.0,
- raw value 255 means Lambda= approx. 2.0,

This value should be scaled. The following settings can be used: ***Multiplier***=1000, ***Divider***=128 and move decimal places using ***Decimal places***=3. This way, you will add the end value of 1.000 to the raw value of 128.

Selection of a physical value and the unit (5). Typical SI units as well as units commonly used in the automotive industry are available. If a requested unit is not on the list, you can also use the ***User*** unit.

Once a unit has been selected, set the selection to the default value (6).
A default value is used from starting the device until receiving the first frame containing the channel.

Important:

Decimal places should be included in this constant. For example, if ***Decimal places = 2*** has been selected, and the default value is to be 1.0, enter the value 100 into the field. This also applies to the following fields: ***Offset*** and ***Timeout value***.

The definition of the behaviour in the event of a fade-out of frame reception on the CAN bus is carried out in the ***If message time out*** field (7).

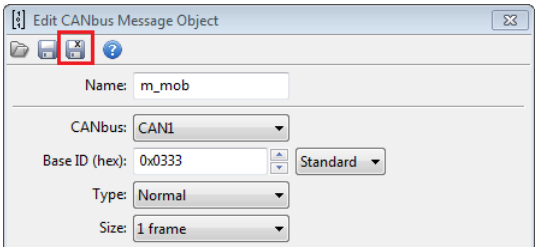
If a given frame cannot be received for a time longer than that defined in ***Message Object*** configuration (***Timeout*** parameter in seconds), two options are available:

- the last value (possibly the default value if a frame was never received) may remain (***Use previous value***);
- a specific value can be set (***Set value***).

The last element of the ***CANbus Input*** defining window are ***Test data*** fields(8). They are used only during editing. You can observe a received frame in real time (***Live capture*** on) or enter test data (***Live capture*** off). In both cases, the calculated final value is displayed, which accelerates configuration.

10.4. Own CAN streams – saving to a .CANX file

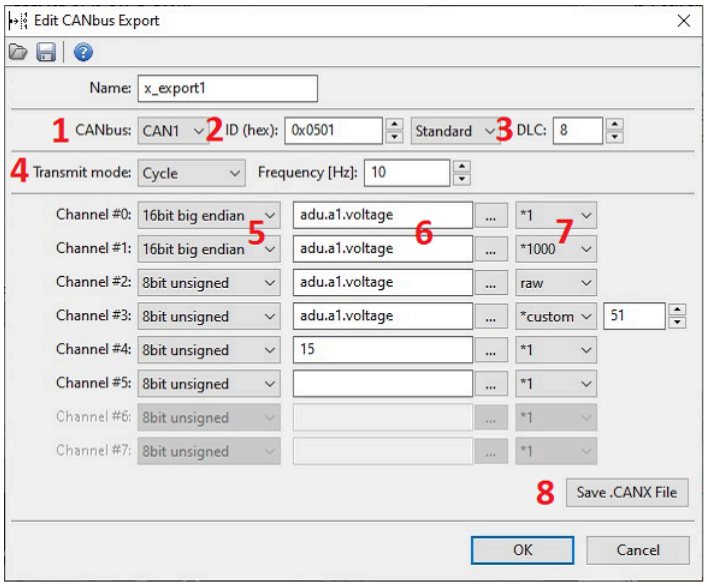
A configured ***Message Object*** together with all ***CANbus Input*** channels can be saved into a .CANX file using a toolbar button.



10.5. Sending frames by means of the CAN bus (CANbus Export)

CANbus Export allows sending any available channel of the device. Frames with any CAN ID can be transmitted on one of two CAN buses. It is a necessary tool to communicate with other PMUs or other ECUMASTER devices.

The ***CANbus Export*** configuration window is comprised of the following sections:



CAN bus selection (1)

Select one of the two available buses: CAN1 or CAN2.

Selecting the CAN frame ID (2)

When selecting the CAN ID frame identifier, it is important to ensure that it does not come into conflict with other communications on the network. Recommended range of identifiers for the

user: 0x500–0x57F. In this range, ECUMASTER devices will never have their default CAN ID in the future.

Important:

In a CAN network, it is not permitted for two devices to transmit frames with the same CAN ID.

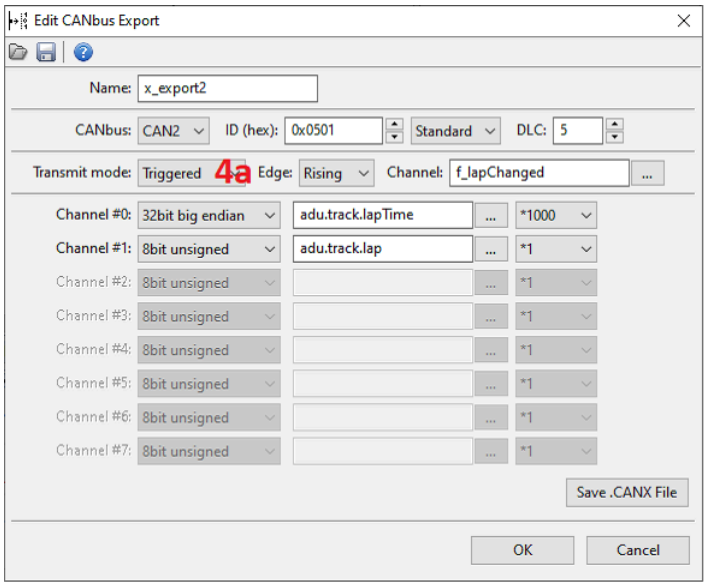
Determination of frame length DLC (3)

The DLC determines the length of a frame: from 0 to 8 bytes.

Selecting transmission type

Continuous transmission Cycle (4)

For continuous transmission, select a sending frequency (Frequency) in the range from 1 to 100 Hz (from 1 to 100 frames sent per second). A maximum of 500 frames in total can be sent per second on a single CAN bus.



Triggered transmission Triggered (4a)

With Triggered transmission, a frame is sent when the appropriate Edge appears on the selected Channel : (Rising or Falling).

Selection of the type of data sent (5)

There are 6 options available:

- ***8bit unsigned*** – the value of the channel is limited to the range of 0..255 and sent as a single byte in a frame.
- ***8bit signed*** – the value of the channel is limited to the range of -128..127 and sent as a single byte in a frame.
- ***16bit big endian*** – the value is transmitted with the most significant byte preceding the least significant byte. For instance, the value 0x1234 will be transmitted as two consecutive bytes: 0x12, 0x34.
- ***16bit little endian*** – the value is transmitted with the least significant byte preceding the most significant byte. For instance, the value 0x1234 will be transmitted as two consecutive bytes: 0x34, 0x12.

Selected channels or constants (6)

You should select a channel from the list or enter a constant. In addition to the decimal notation, a constant can also be saved in hexadecimal notation. To this end, the 0x prefix should be used (e.g. 0xE3 or 0xe3).

Selection of a multiplier or a raw value (7)

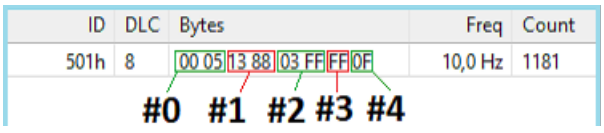
It is possible to multiply the actual value by a constant in the 1-1000 range (the fractional part is discarded) or alternatively to send the ***raw*** value.

Example:

From the drawing of the window *x_ export1*:

- ***Channel #0*** – voltage value at input A1 will be sent as a number from the range: 0, 1, 2, 3, 4, 5 (in volts, but without the fractional part).
- ***Channel #1*** – voltage value at input A1 will be sent as a number from the range 0–5000 (in millivolts).
- ***Channel #2*** – voltage value at input A1 will be sent as a raw value from the ADC converter as a number from the range 0–1023.
- ***Channel #3*** – voltage value at input A1 will be sent as a number from the range 0–255.
- ***Channel #4*** – a constant value will be sent: 15 in the decimal system.

Below is a frame preview as seen in the ECUMASTER Light Client. At the analog input A1, the voltage is exactly 5 V. Accordingly, the channels ***Channel #0*** - ***Channel #4*** present themselves as in the example below:



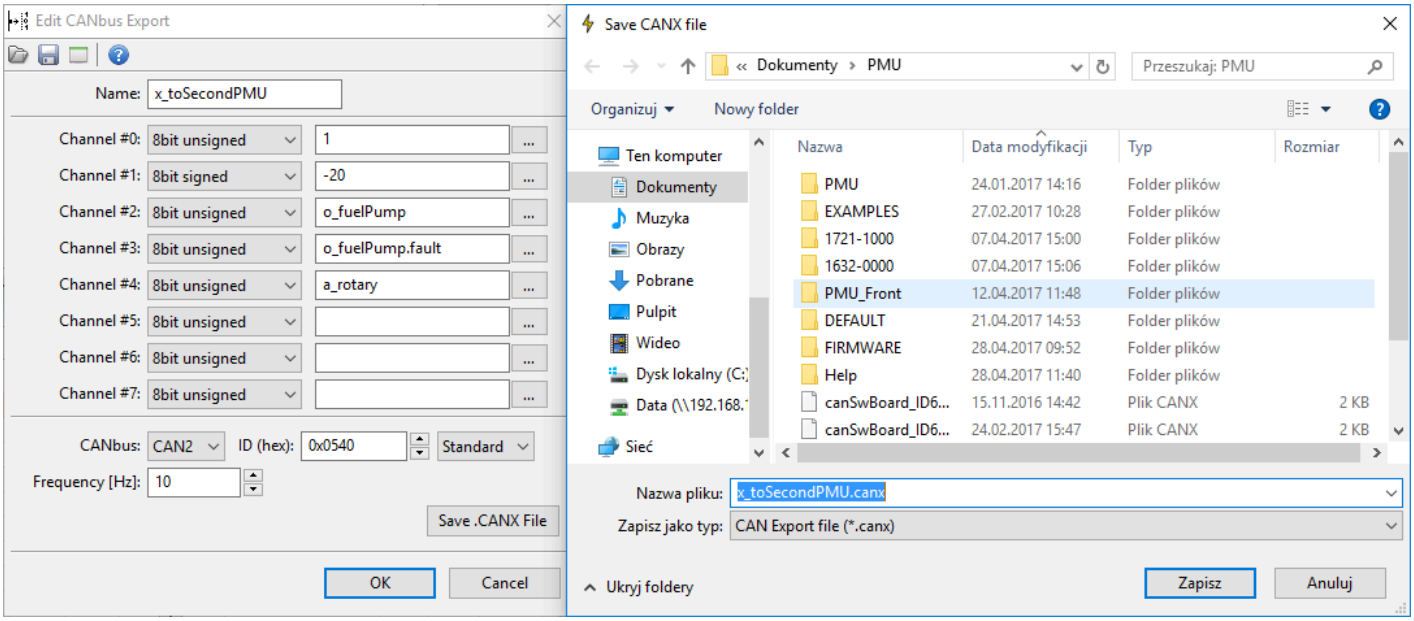
- ***Channel #0*** - value 0x0005, i.e. 5 [V]
- ***Channel #1*** - value 0x1388 , i.e. 5000 [mV]
- ***Channel #2*** - value 0x03FF, i.e. 1023 [adc]
- ***Channel #3*** - value 0xFF, i.e. 255
- ***Channel #4*** - value 0x0F, i.e. 15

Saving to a .CANX file (8)

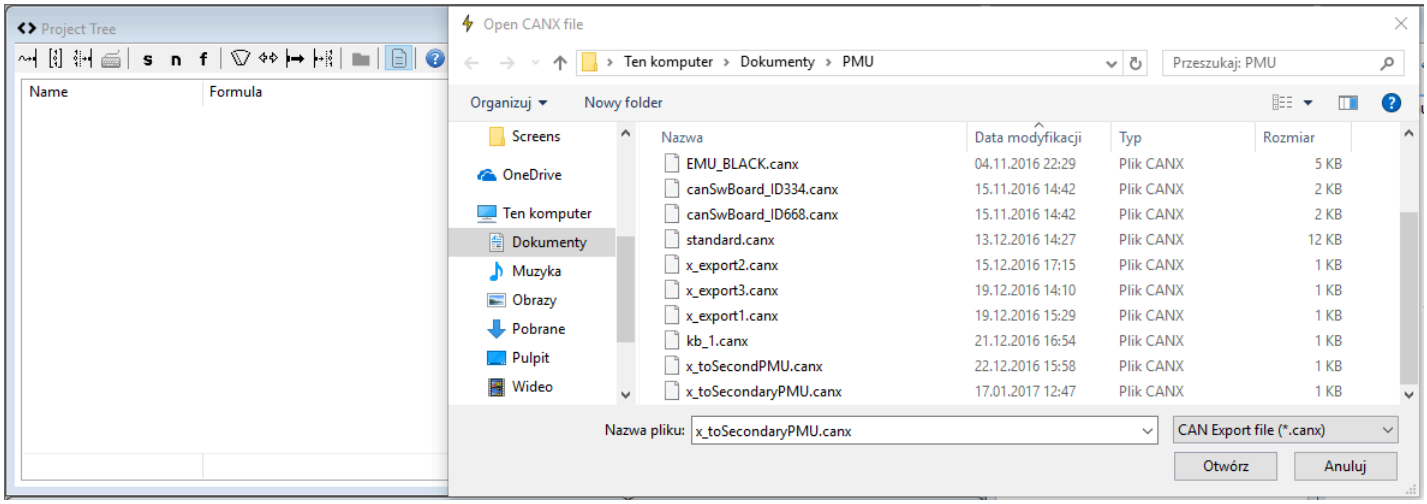
When creating a CANbus Export, you have the option to save this Export as a .CANX file.

This file can be imported by other PMUs to automatically create CANbus MOB with the correct ID, MOB name and CANbus Inputs that correspond to exported Channel.

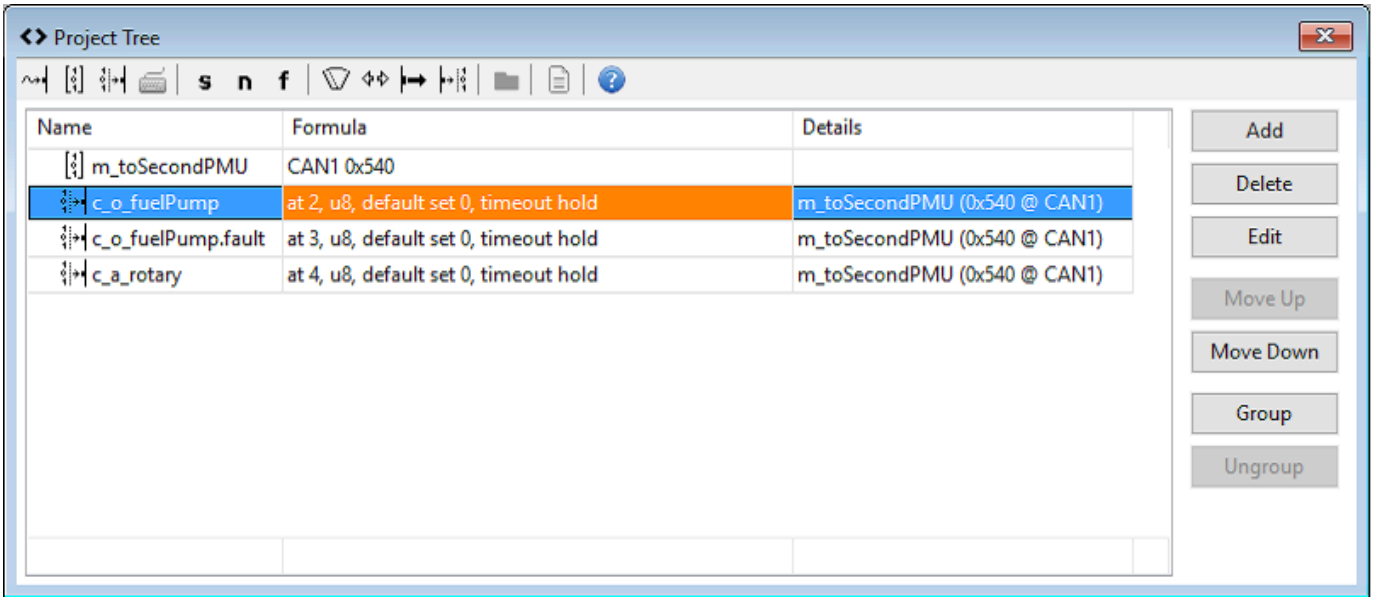
Example:



Create a CANbus Export, choose broadcasted channels and save it as a .CANX file



Use Import .CANX option from *Project Tree* to open the .CANX file



CANbus MOB and CANbus Inputs were automatically created with the correct ID

10.6. Reserved CAN ID

ID range	Default CAN bus	CAN bus configuration possible	Configurable ID	Description
0x012-0x017	CAN1	No, only CAN1	No, ID is fixed	Communication with PMU Client

11. Standard CAN Stream

Standard CAN Stream (Tree View → Standard CAN Stream) gives the user the ability to broadcast key PMU parameters over the CAN bus.
Parameters themselves are predefined (they are fixed and cannot be configured), but the user has the ability to broadcast only some part of them, on chosen CAN bus with chosen ID.

The table shows how the ***CAN Stream*** frames are constructed

ID	**BaseID + 0**			**Frequency: 20 Hz**			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0	(reserved)	4 bits (0xF0)	Unsigned	0-15	-	0	-
	User error	1 bit (0x08)	Unsigned	0-1	-	0	-
	PMU Status [1)]	3 bits (0x07)	Unsigned	0-7	-	0	-
1	Total Current	8 bits	Unsigned	0-255	1A/bit	0	A
2	Battery Voltage	8 bits	Unsigned	0-27.75	0.1088V/bit	0	V
3	Board Temperature Left	8 bits	Unsigned	0-255	1C/bit	0	C
4	Board Temperature Right	8 bits	Unsigned	0-255	1C/bit	0	C
5	Flash Temperature	8 bits	Unsigned	0-255	1C/bit	0	C
6	(reserved)	2 bits (0xC0)	Unsigned	0-3	-	0	-
	I6.active [2)] I5.active I4.active I3.active I2.active I1.active	1 bit (0x20) 1 bit (0x10) 1 bit (0x08) 1 bit (0x04) 1 bit (0x02) 1 bit (0x01)	Unsigned	0-1 0-1 0-1 0-1 0-1 0-1	-	0	-
7	(reserved)	2 bits (0xC0)	Unsigned	0-3	-	0	-
	I6.error [2)] I5.error I4.error I3.error I2.error I1.error	1 bit (0x20) 1 bit (0x10) 1 bit (0x08) 1 bit (0x04) 1 bit (0x02) 1 bit (0x01)	Unsigned	0-1 0-1 0-1 0-1 0-1 0-1	-	0	-

[1] The PMU Status is just the maximum value of all of the 16 outputs. This ***pmu.status*** shows the status of output in the “worst” condition.

[2] The state of the low-side outputs.

ID	**BaseID + 1**			**Frequency: 20 Hz**			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0	o1.status [3] o1.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o2.status [3] o2.active	3 bits (0x0E) 1 bit (0x01)	Unsigned	0-7 0-1	-	0	-
1	o3.status [3] o3.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o4.status [3] o4.active	3 bits (0x0E) 1 bit (0x01)	Unsigned	0-7 0-1	-	0	-
2	o5.status [3] o5.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o6.status [3] o6.active	3 bits (0x0E) 1 bit (0x01)	Unsigned	0-7 0-1	-	0	-
3	o7.status [3] o7.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o8.status [3] o8.active	3 bits (0x0E) 1 bit (0x01)	Unsigned	0-7 0-1	-	0	-
4	o9.status [3] o9.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o10.status [3] o10.active	3 bits (0x0E) 1 bit (0x01)	Unsigned	0-7 0-1	-	0	-
5	o11.status [3] o11.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o12.status [3] o12.active	3 bits (0x0E) 1 bit (0x01)	Unsigned	0-7 0-1	-	0	-
6	o13.status [3] o13.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o14.status [3]	3 bits (0x0E)	Unsigned	0-7	-	0	-

ID	BaseID + 1			Frequency: 20 Hz			
	o14.active	1 bit (0x01)		0-1			
7	o15.status [3)] o15.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o16.status [3)] o16.active	3 bits (0x0E) 1 bit (0x01)	Unsigned	0-7 0-1	-	0	-

[3)] All possible values of each o*.status: 0 – OFF, 1 – ACTIVE (ON), 2 - UNDERCURRENT, 3 – OVERCURRENT, 7 – THERMAL SHUTDOWN. More information is in the Output status *(on page 67)* section.

ID	BaseID + 2			Frequency: 62.5 Hz			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0	a1.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
1	a2.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
2	a3.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
3	a4.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
4	a5.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
5	a6.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
6	a7.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
7	a8.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V

ID	BaseID + 3			Frequency: 62.5Hz			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0	a9.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
1	a10.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
2	a11.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
3	a12.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
4	a13.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
5	a14.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
6	a15.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V
7	a16.voltage	8 bits	Unsigned	0-5	0.0196V/bit	0	V

ID	BaseID + 4			Frequency: 20 Hz			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0	o1.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
1	o2.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
2	o3.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
3	o4.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
4	o5.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
5	o6.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
6	o7.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
7	o8.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A

ID	BaseID + 5			Frequency: 20 Hz			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0	o9.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
1	o10.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
2	o11.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
3	o12.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
4	o13.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
5	o14.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
6	o15.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A
7	o16.current	8 bits	Unsigned	0-63.75	0.25A/bit	0	A

ID	BaseID + 6			Frequency: 20 Hz			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0	o1.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
1	o2.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
2	o3.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
3	o4.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
4	o5.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
5	o6.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
6	o7.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
7	o8.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V

ID	BaseID + 7			Frequency: 20 Hz			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0	o9.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
1	o10.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
2	o11.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
3	o12.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
4	o13.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
5	o14.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
6	o15.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V
7	o16.voltage	8 bits	Unsigned	0-15.84	0.0621V/bit	0	V

Frames only for PMU-24

ID	BaseID + 8			Frequency: 20 Hz			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0	o17.status [3)] o17.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o18.status [3)] o18.active	3 bits (0x0E) 1 bit (0x01)	Unsigned	0-7 0-1	-	0	-
1	o19.status [3)] o19.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o20.status [3)] o20.active	3 bits (0x0E) 1 bit (0x01)	Unsigned	0-7 0-1	-	0	-
2	o21.status [3)] o21.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o22.status [3)] o22.active	3 bits (0x0E) 1 bit (0x01)	Unsigned	0-7 0-1	-	0	-
3	o23.status [3)] o23.active	3 bits (0xE0) 1 bit (0x10)	Unsigned	0-7 0-1	-	0	-
	o24.status [3)] o24.active	3 bits (0x0E) 1 bit (0x01)	Unsigned	0-7 0-1	-	0	-

[3] All possible values of each o*.status: 0 – OFF, 1 – ACTIVE (ON), 2 - UNDERCURRENT, 3 – OVERCURRENT, 7 – THERMAL SHUTDOWN. More information is in the Output status *(on page 67)* section.

ID	**BaseID + 9**			**Frequency: 20 Hz**			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0	o17.current	8 bits	Unsigned	0-25.5	0.1A/bit	0	A
1	o18.current	8 bits	Unsigned	0-25.5	0.1A/bit	0	A
2	o19.current	8 bits	Unsigned	0-25.5	0.1A/bit	0	A
3	o20.current	8 bits	Unsigned	0-25.5	0.1A/bit	0	A
4	o21.current	8 bits	Unsigned	0-25.5	0.1A/bit	0	A
5	o22.current	8 bits	Unsigned	0-25.5	0.1A/bit	0	A
6	o23.current	8 bits	Unsigned	0-25.5	0.1A/bit	0	A
7	o24.current	8 bits	Unsigned	0-25.5	0.1A/bit	0	A

ID	**BaseID + 10**			**Frequency: 62.5 Hz**			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0..1	o17.voltage	16 bits	Unsigned	0-20	0.0155V/bit	0	V
2..3	o18.voltage	16 bits	Unsigned	0-20	0.0155V/bit	0	V
4..5	o19.voltage	16 bits	Unsigned	0-20	0.0155V/bit	0	V
6..7	o20.voltage	16 bits	Unsigned	0-20	0.0155V/bit	0	V

ID	**BaseID + 11**			**Frequency: 62.5 Hz**			
ByteID	Channel	Data Width	Data Type	Range	Resolution	Offset	Unit
0..1	o21.voltage	16 bits	Unsigned	0-20	0.0155V/bit	0	V
2..3	o22.voltage	16 bits	Unsigned	0-20	0.0155V/bit	0	V
4..5	o23.voltage	16 bits	Unsigned	0-20	0.0155V/bit	0	V
6..7	o24.voltage	16 bits	Unsigned	0-20	0.0155V/bit	0	V

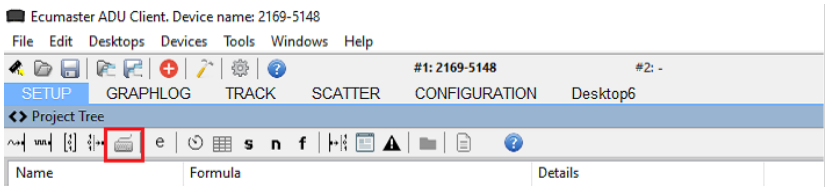
12. CAN bus keypad support

Keyboards from Ecumaster, Grayhill, MoTeC / RaceGrade, Emtron, and Life Racing are supported.

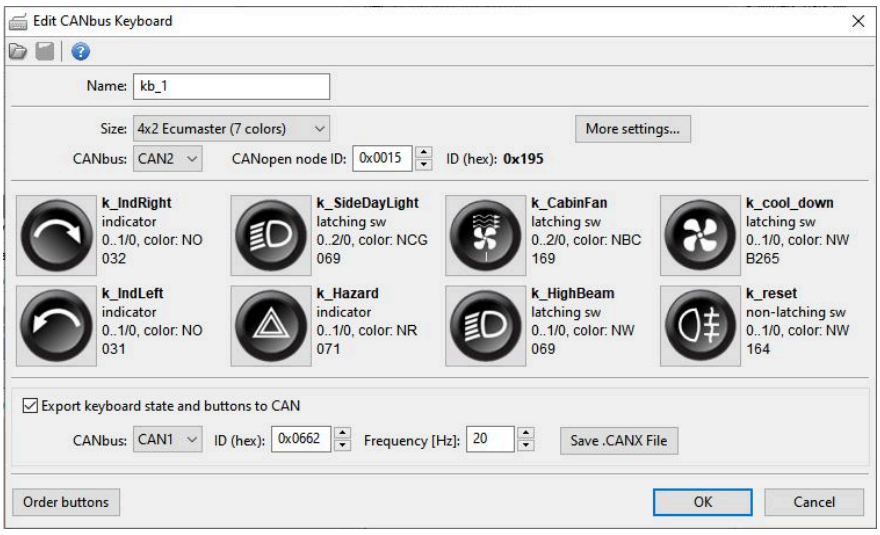
If you're using the 5x3MT keyboard with two encoders, see Appendix B - How-to Configure 5x3MT Keyboard *(on page 149)* for more details.

Up to two keypads can be connected to the PMU device.

To configure a keyboard, click ***Add*** in the ***Project Tree*** and then select ***CANbus Keyboard*** from the list or click the keyboard icon in the toolbar.

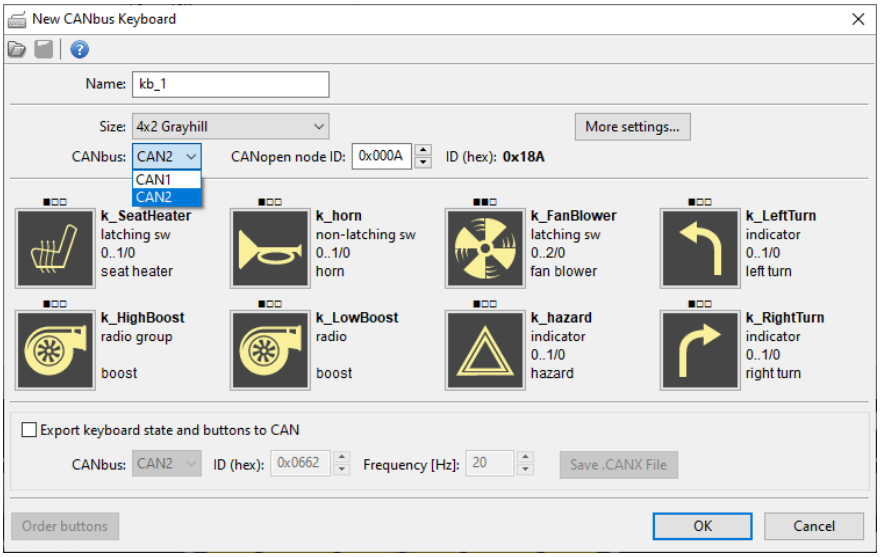


After opening the ***CANbus Keyboard*** window, individual parameters must be set:



- ***Name*** - name of the keyboard
- ***Size*** - size and type of keyboard to be configured
- ***CANbus*** - CAN communication bus to which the keypad is connected

  The default speed of Ecumaster keyboards is 500 kbps, so it is recommended to select the CAN2 communication bus, which is set to the same speed by default. The CAN1 bus can only be selected for keyboards communicating at 1 Mbps



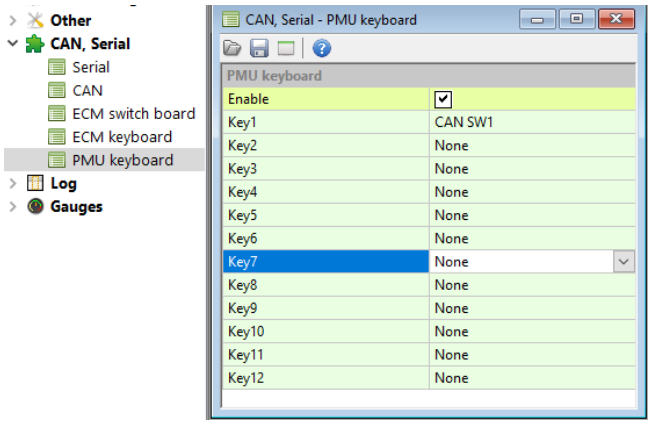
- ***More settings***

Settings	Description
Show open ceremony	Opening ceremony, i.e. key highlight animation when switching on the device (only for Ecumaster keyboards)
Key LED brightness	Backlighting intensity of the keys (Ecumaster keyboards) or LEDs above the keys (Grayhill keyboards)
Back light brightness	Backlighting intensity of the entire keyboard (night backlighting)
Back light color	Full keyboard backlighting color (only for Ecumaster keyboards)
Alternative brightness channel	Channel defining alternative backlighting
Alternative key LED brightness	Intensity of alternative key backlighting (Ecumaster keyboards) or LED above the keys (Grayhill keyboards)
Alternative back light brightness	Alternative backlighting intensity of the entire keyboard (night backlighting)
Alternative back light color	Alternative backlighting color of the entire keyboard (only for Ecumaster keyboards)

- ***CANopen node ID*** – the factory setting for Ecumaster keyboards is 0x0015 (this item needs to be changed if there is a ***CANopen node ID*** conflict with another keyboard connected to the CAN bus)

  A detailed description of CANopen node ID changes can be found at: www.ecumaster.com/files/LightClien/LightClientManual_1_0.pdf in section 11.3.

- ***Export keyboard state and buttons to CAN*** – allows the export of keyboard status from the keyboard to the same or another CAN bus without the need to create and configure the ***CANbus export***. With this feature, the keyboard configured in the **PMU** will also work on other devices, e.g. **ADU, EMU Black** or other **PMU** modules.
  When exporting key status to **EMU Black** the CAN ID address must be 0x662. To receive push button information, select the **PMU Keyboard** on **EMU Black**.

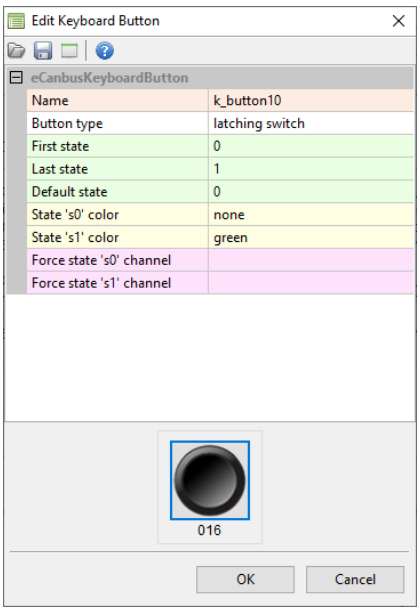


For the **PMU**, simply save the keyboard configuration settings to a ***.CANX*** file and then load the saved file (Import ***.CANX / .DBC*** file), which automatically configures the ***CANbus Message Objects*** and ***CANbus Inputs*** in the software.

- ***Edit Keyboard Button*** - edit individual keys:

Parameter	Description
Name	Button name
Button type	Button mode of operation: ***non-latching switch*** - a non-latching button in which the user-defined backlight color of the button depends on one of the two possible states of the button. ***latching switch*** - a latching button in which two to four stable states can be defined. The color of the key's backlighting depends on the status of the button. ***radiobutton group*** - button to open a group of radio buttons. ***radiobutton*** - further buttons in the radio button group. ***indicator*** - a momentary pushbutton, with illumination to the specified color done by channel-assigned conditions (as opposed to a non-latching switch). Up to three different conditions can be set to change the state of illumination.

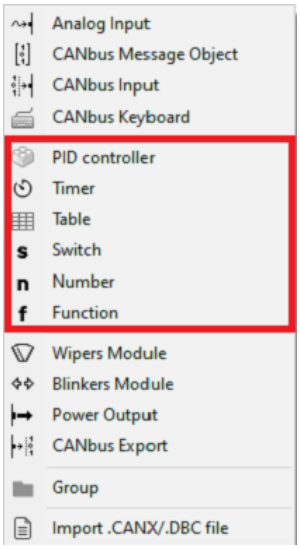
Parameter	Description
	indicator 3 – in Grayhill keyboards, indicator mode 3 is available in addition to indicator mode. The difference between the two is that in indicator 3 mode, the independent illumination of each diode (in each possible configuration) is triggered by the fulfilment of three individual conditions assigned by separate channels. For the indicator mode on Grayhill keyboards however, there is only one channel to assign conditions, the fulfilment of which allows the LEDs to be illuminated in up to three different states - one, two or three LEDs lit.
Default state	Default setting for the status of the button when the keyboard is activated
State's#' color	Setting the color for a button state (only for Ecumaster keyboards)
Force state 's#' channel	Option available in *latching switch* mode. A channel for assigning a condition the fulfilment of which forces a change in the state of the button.
Source channel	Option available for *indicator* mode. Source channel for assigning a condition, the fulfilment of which triggers the illumination assigned to the button state.
Choose Button Image	Choice of button icons



- ***Order buttons*** – prepares copy-ready information on the numbers of plastic key inserts, with selected laser-engraved icons, for order preparation.

13. Processing information in the PMU.

The PMU has six information processing elements:



1. ***PID controller*** – control systems with feedback
2. ***Timers*** - counting time
3. ***Tables*** - 2D or 3D tables
4. ***Switches*** - virtual switches, counters
5. ***Numbers*** - mathematical channels
6. ***Functions*** - logical functions

The above elements are processed at 500 Hz (every 2 ms) just like the other elements in the PMU. The order in which the elements are processed is shown above, i.e. ***Timers*** first, then ***Tables***, etc.

13.1. PID controller – control systems with feedback

The ***PID controller*** is a proportional-integral-derivative controller whose purpose is to maintain the measured value (value of *Process value channel*) at a certain level, called the set point (value of *Set point channel*). It operates in a closed feedback loop. By calculating the difference between the set point and the measured value, it strives to reduce this difference by adjusting the signal applied to the input of the controlled object.

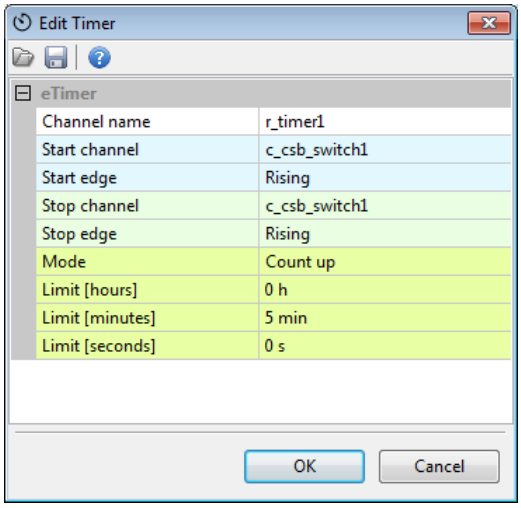
The output value of the controller (the value of the *.output channel*) is the sum of the *Feed forward* value and the values of the components: proportional (the value of the *.proportionalTerm* channel), integral (the value of the *.integralTerm* channel), and derivative (the value of the *.derivativeTerm* channel).

Parameter	Description
Name	PID controller name
Set point channel	Channel defining the set point
Process value channel	The measured value channel
Activation channel	The channel that activates a specific PID controller. When the channel is set to false – the controller's output is equal to the sum of the *Feed forward* and the *Custom term* channel value. Proportional, Integral and Derivative controller components are set to zero.
Feed forward	Constant value added to the controller's output.
Proportional gain	The gain factor of the proportional (P) term of the PID controller. The proportional gain of the PID controller determines how much the control signal responds to the error between the set point and the current state of the controlled system, proportionally increasing control to reduce this error.
Integral gain	The gain factor of the integral (I) term of the PID controller. The integral gain of a PID controller determines how long and how aggressively the controller reacts to long-term differences between the set point and the actual state of the system.
Derivative gain	The gain factor of the derivative (D) term of the PID controller. Increasing this parameter helps to reduce changes in the controller's output when there is a sudden change in the set point or the measured process value.
Integral limit min	The minimum value of the integral term of the PID controller
Integral limit max	The maximum value of the integral term of the PID controller
Custom term	The channel specifying the value added to the controller's output. It is independent of the set point and the measured process value.
Time step	The time step for updating the PID controller
Output min	Minimum duty cycle that is allowed at the output
Output max	Maximum duty cycle that is allowed at the output
Control direction	***Normal*** – Increasing the set point of the process implies increase of the controller output.

Parameter	Description
	Inverted – Increasing the set point of the process implies decrease of the controller output.

13.2. Timers – counting time

Timers are used for counting time.



Two ways of counting are available: the first from zero to the set value (*Count up*) and the second from the set value to zero (*Count down*).

Timers are started by means of a channel defined by ***Start channel*** and an edge defined by ***Start edge***: (Rising or Falling).
Similarly, timer can be stopped using a channel defined by ***Stop channel*** and an edge defined by ***Stop edge*** (Rising or Falling).

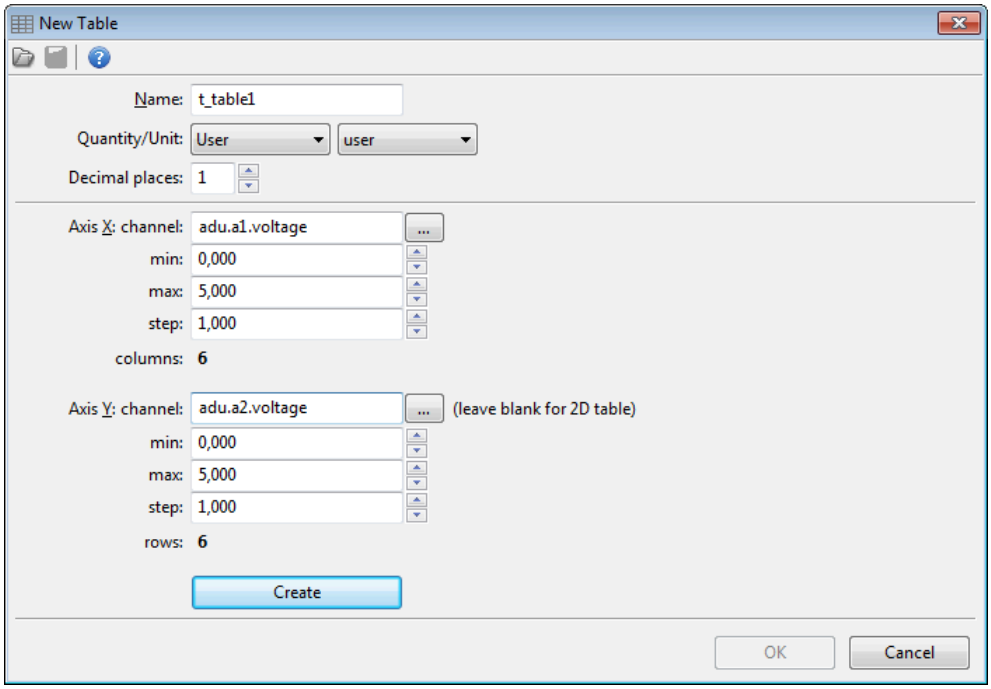
When a timer is stopped, the starting edge appearing in the ***Start channel*** will always result in a change to the initial value. When a timer is already started, the starting edge is ignored. A timer will react to a starting edge only after once it has been stopped or counting has finished. This means starting and stopping is possible using the same channel and edge.

Each created timer has 3 subchannels:

- ***.value*** - value of time in seconds (up to 0.01 s)
- ***.elapsed*** - has a value of “1” when the time has elapsed; it will change to “0” after another start
- ***.running*** - has a value of “1” when a timer is counting time

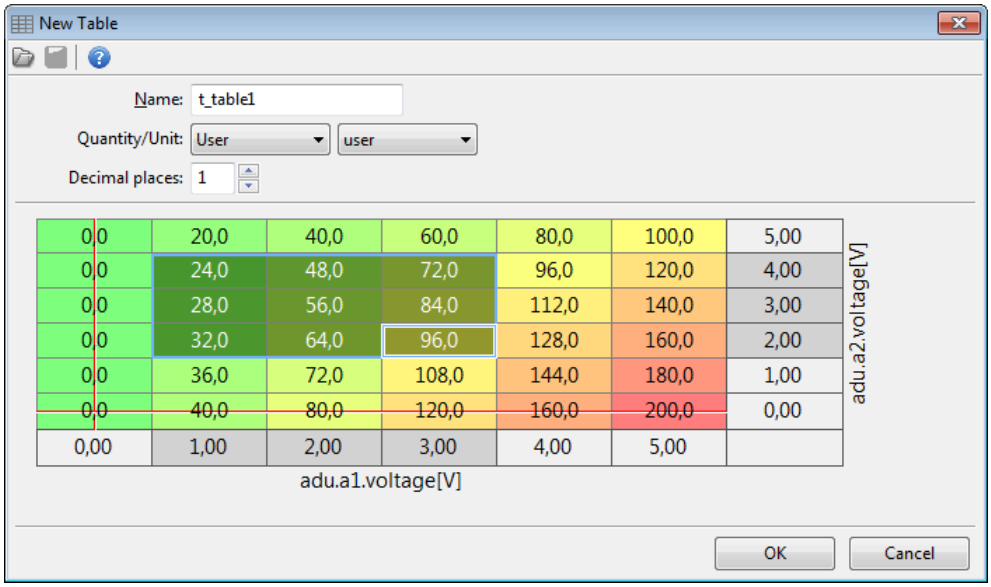
13.3. Tables – 2D / 3D

Configuration of a table starts with defining channels representing axes. Tables can be 2D or 3D. If a table is to be two-dimensional, leave the ***Axis Y: channel*** field empty.

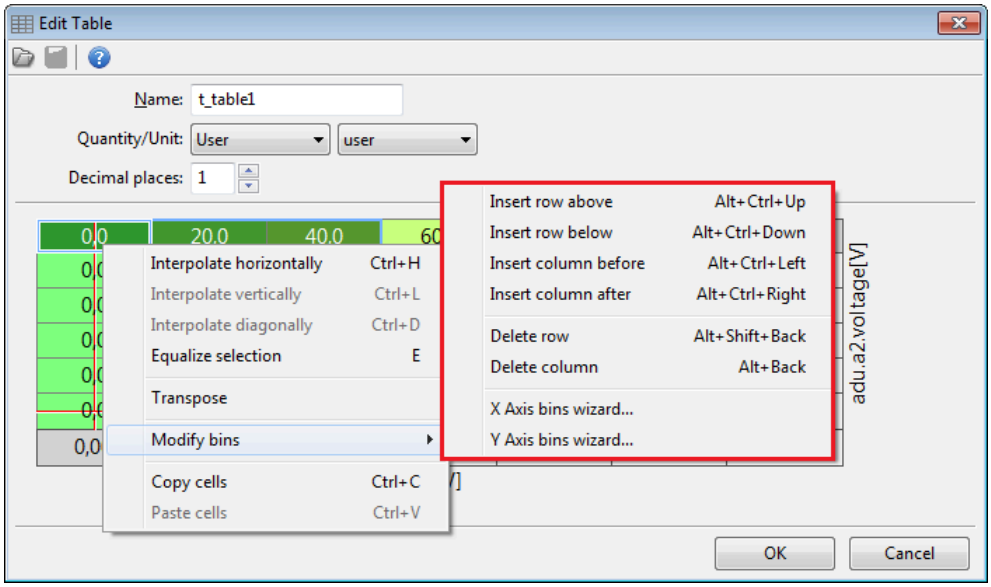


You should also define the axis scope: ***min*** and ***max***. To change the number of elements in a table, change the step parameter which defines a step.
Table size can range from 2x1 (2D) or 2x2 (3D) to 21x1 (2D) or 21x21 (3D).

Next, fill the cells and axes with values. The values defined on axes are independent for each table.



You can select several cells by means of the Shift key. The Ctrl + arrow key copies to adjacent cells. You can also find the horizontal and vertical interpolation commands helpful.



The size of the table (number of columns or rows) can be changed at any time using the context menu available under the right mouse button.

Description of the commands in the context menu:

Command	Key shortcut	Description
Interpolate horizontally	Ctrl+H	Horizontal interpolation: the cell values in the selection area are calculated as a linear interpolation of the cells from the left and right edges of the selection.
Interpolate vertically	Ctrl+L	Vertical interpolation: the cell values in the selection area are calculated as a linear interpolation of the cells from the top and bottom edges of the selection.
Interpolate diagonally	Ctrl+D	Interpolation between apexes. Define the 4 corner points of the selection and the rest of the cells will be counted as bilinear interpolation. The command combines two commands: first the horizontal and then vertical interpolation.
Equalize selection	E	Smooths out selected cells
Insert row above	Alt+Ctrl+Up	Inserts a row above the selected cell
Insert row below	Alt+Ctrl+Down	Inserts a row below the selected cell

Command	Key shortcut	Description
Insert column before	Alt+Ctrl+Left	Inserting a column to the left of the selected cell
Insert column after	Alt+Ctrl+Right	Inserts a column to the right of the selected cell
Delete row...	Alt+Shift+Back	Deletes a row containing the selected cell
Delete column...	Alt+Back	Deletes a column containing the selected cell
X Axis bins wizard		Starts the creator for the X axis allowing to define a new number of columns and to generate the Y axis cells according to the selected interpolation type
Y Axis bins wizard		Launches the Y-axis wizard to define a new number of rows and to generate Y-axis cells according to the selected type of interpolation

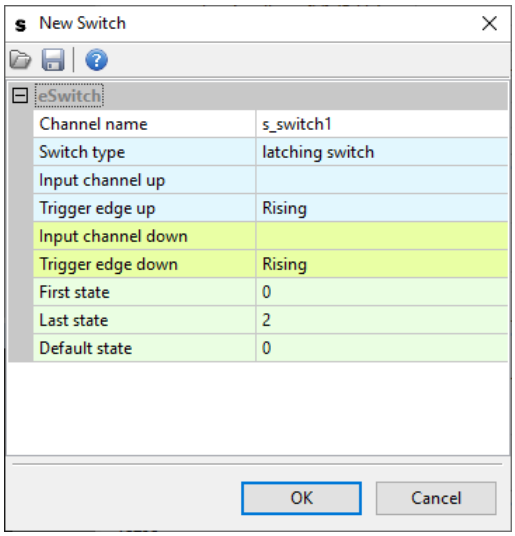
13.4. Switches – virtual switches, counters

The main task of this element is to convert the momentary switch / non-latching switch available as an analog or CAN input into a Latching switch.

You should define the scope using the ***First state*** and ***Last state*** parameters, as well as the default value by means of the ***Default state*** parameter.

This element will operate as a counter. After each appearance of a ***Trigger edge up*** in the ***Input channel up*** element value will increase by 1. If the element already has a value equal to ***Last state*** and a ***Trigger edge up*** appears, the value will “roll back” and assume the value of the ***First state***.

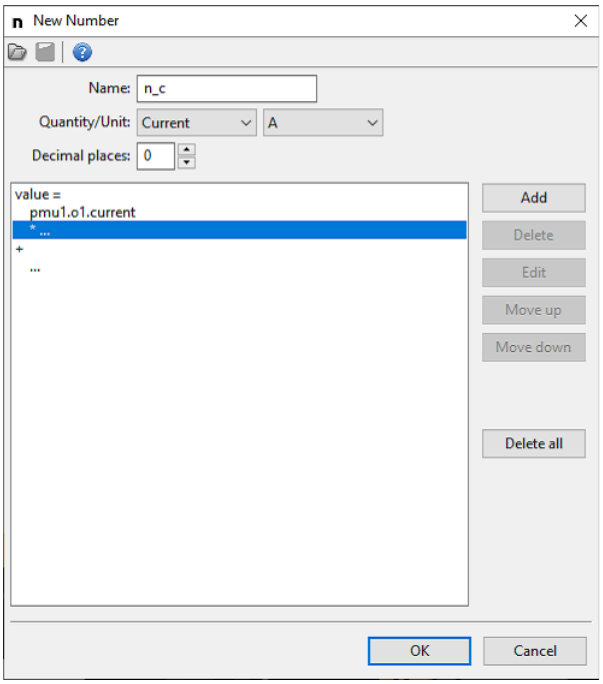
After each appearance of a ***Trigger edge down*** in the ***Input channel down*** the element value will increase by 1. If the element reaches a value equal to ***First state*** and it appears ***Trigger edge down***, it will change to the value of ***Last state***.



13.5. Numbers – mathematical channels

A mathematical channel (***Number***) allows you to create a new channel resulting from the mathematical operations on selected values or channels. When you create a new channel, give it a relevant name (***Name***) and define the physical quantity and unit (***Quantity/Unit***) defining the created channel.

In the simplest form, the *Number* calculates the sum of the products of the selected values or channels.



value =

C1

*** C2**

*** C3**

*** ...**

+

C4

*** C5**

*** ...**

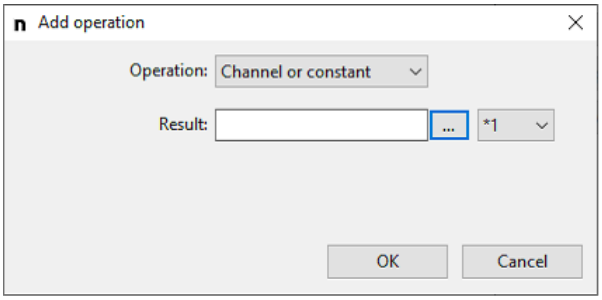
+

...

where C1, C2, C3... - is the selected channel or constant value (*Channel or Constant*)

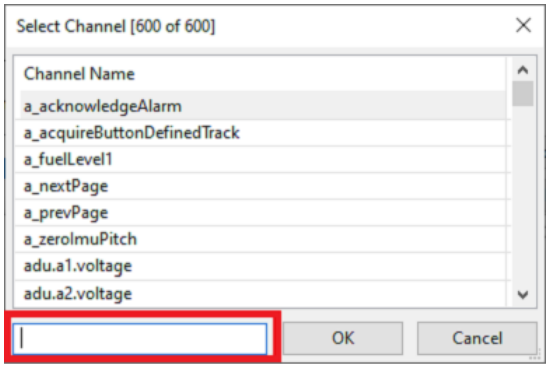
To use the selected channel or constant value for calculations, click on the formula editor and select the field with the ellipsis '**...**'. Then click the ***Add*** button (located on the right).

The ***Add operation*** window will appear, in which you select the relevant operation.



When selecting the ***Operation: Channel or constant***, enter a constant value in the ***Result:*** field or by clicking the button marked '**...**' select the relevant channel from the list.

For a faster search, you can enter the name of the channel you are looking for in the filtering field at the bottom of the ***Select Channel***window.



After confirming with the ***OK*** button, the selected value or channel will appear in the formula field. Remember to select the relevant field (...) when assigning another channel, depending on whether it is multiplied (* ...) or added to the previously selected channel.

After marking channels in the formula editor, you can ***Delete***, ***Edit*** or move one place up (***Move up***) or down (***Move down***) in the formula.

You can also use other mathematical operations, including integer division - the *Divide* operation (“***/***”) or the remainder of the division - *Modulo* operation (***mod***).

e.g. value =

C1

* C2

/ C3

or value =

C1

mod C2

+

C3

* C4

* C5

List of operations available for mathematical channels.

FACTOR is a single multiplier (a constant or a channel) in the C1*C2*C3 notation.

RESULT is the calculated result of previous multiplications or divisions / residues.

Operation	Parameter	Pseudocode
Channel or constant	Result [1]	FACTOR = *Result*
Constant	Result [2]	FACTOR = *Result*
Choose	Condition channel Result if true Result if false	if *Condition_channel* ≠ 0 then FACTOR = *Result_if_true* else FACTOR = *Result_if_false*
Divide	Value	RESULT := RESULT DIV *Value* (DIV - integer division; e.g.: 9 DIV 2 = 4)
Modulo	Value	RESULT := RESULT MOD *Value* (MOD - division reminder; e.g.: 9 MOD 5 = 4)
Addition	Value 1 Value 2	FACTOR = *Value_1* + *Value_2*
Subtraction	Value 1 Value 2	FACTOR = *Value_1* - *Value_2*
Min	Value 1 Value 2	if *Value_1* < *Value_2* then FACTOR = *Value_1* else FACTOR = *Value_2*
Max	Value 1 Value 2	if *Value_1* > *Value_2* then FACTOR = *Value_1* else FACTOR = *Value_2*

Operation	Parameter	Pseudocode
Clamp	Input Min Max	if *Input* < *Min* then FACTOR = *Min* else if *Input* > *Max* then FACTOR = Max else FACTOR = *Input*
Lookup2	Channel First index Value[0], Value[1]	if *Channel* ≤ *First_index* then FACTOR = *Value[0]* else FACTOR = *Value[1]*
Lookup3	Channel First index Value[0], Value[1] Value[2]	if *Channel* ≤ *First_index* then FACTOR = *Value[0]* else if *Channel* = *First_index+1* then FACTOR = *Value[1]* else FACTOR = Value[2]
Lookup4	Channel First index Value[0], Value[1] Value[2], Value[3]	if *Channel* ≤ *First_index* then FACTOR = *Value[0]* else if *Channel* = *First_index+1* then FACTOR = *Value[1]* else if *Channel* = *First_index+2* then FACTOR = *Value[2]* else FACTOR = *Value[3]*
Lookup5	Channel First index Value[0], Value[1] Value[2], Value[3], Value[4]	if *Channel* ≤ *First_index* then FACTOR = *Value[0]* else if *Channel* = *First_index+1* then FACTOR = *Value[1]* else if *Channel* = *First_index+2* then FACTOR = *Value[2]* else if *Channel* = *First_index+3* then FACTOR = *Value[3]* else FACTOR = *Value[4]*

[1] - The constant value for ***Chanel or constant*** operation may be in the [-16384, +16383] range

[2] - The constant value for Constant operation may be in the [-32768, +32767] range

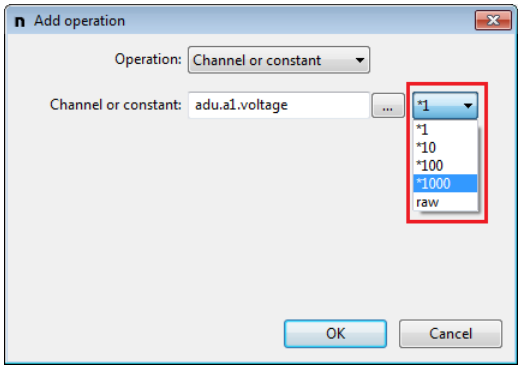
Calculation of the channel value is done in raw values on integers. The fractional part is discarded. In order to obtain the needed accuracy of the created *Number* channel, each constant value or channel used in mathematical operations of the created channel should be multiplied by the appropriate value modifier (multiplier) and then take into account the *Decimal places* by "moving" the Decimal point by the accuracy by which the individual channels / values were multiplied.

Channel value modifiers

The value of each channel available for mathematical operations can be modified. You may multiply by 1, 10, 100 or 1000 (the fractional part is discarded).

You may also choose to use the ***raw*** value with no modification.

Raw applies to the voltage from analogue inputs, for example, when it is a value from an ADC converter with a range of 0-1023.



Decimal places

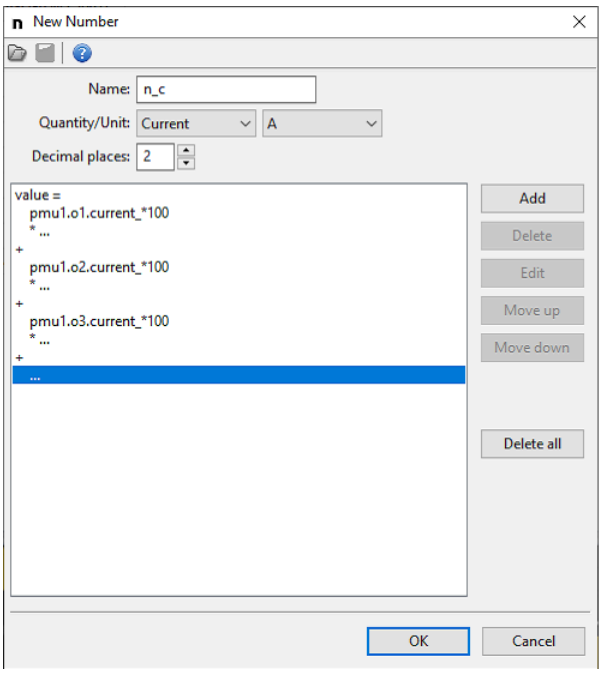
Each mathematical channel can store raw values within the range of [-32768, +32767].

You can additionally define decimal places by, “moving” the point by 0, 1, 2 or 3 places. For example, when Decimal places is set to 1, such channel can store real values in the [-3276.8, +3276.7] range.

Values are calculated in the raw value based on integers and then the point is “moved” by a defined number of decimal places.

Indirect calculations are performed using a broader range of numbers (ca. ±2*109). For example, calculations can be performed for the following values 1000*1000 / 123. At the end, the result is restricted (clamp) to the [-32768, +32767] range.

Calculating the sum of currents of 3 PMU current outputs



Three channels are given:

pmu1_o1.current, *pmu1_o2.current*, *pmu1_o3.current*.

They contain the current sent from a PMU device.

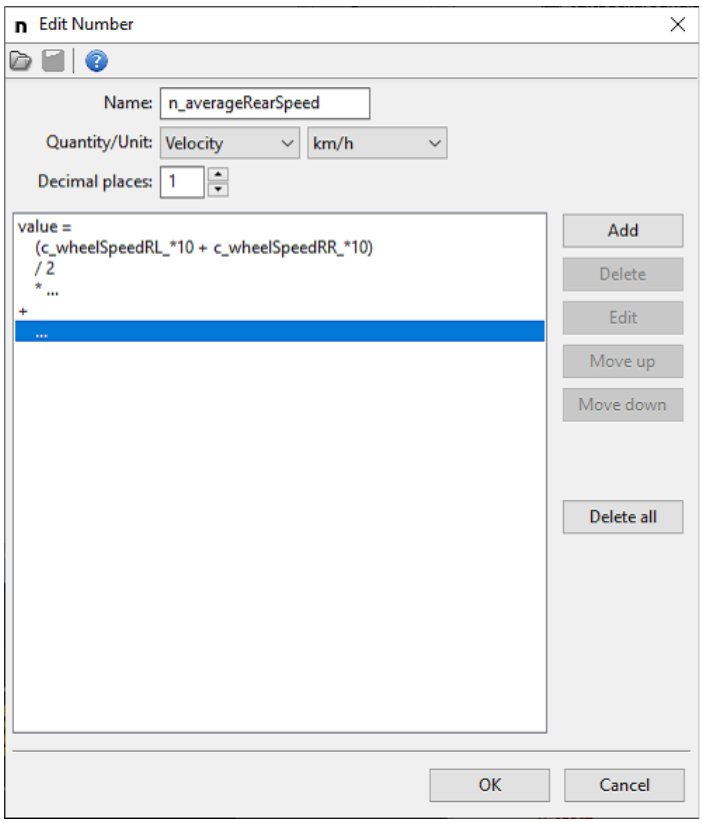
Let's assume that we want to obtain a result with an accuracy of up to 0.01 A.

For each channel, use the value modifier (multiplier) ****100*** and "move" the decimal point to the left by two decimal places (via ***Decimal places: 2***)

The following formula should be input:

n_c = (o1.current*100 + o2.current*100 + o3.current*100)

Calculating the average speed of the rear axle



Two channels are given: *c_speedRL* and *c_speedRR* with speed in km/h.

Let's assume that we want to obtain a result also with an accuracy of up to 0.1 km/h.

The following formula should be input:

n_averageRearSpeed = (c_speedRL*10 + c_speedRR*10) / 2

The decimal point should be moved by 1 place to the left.

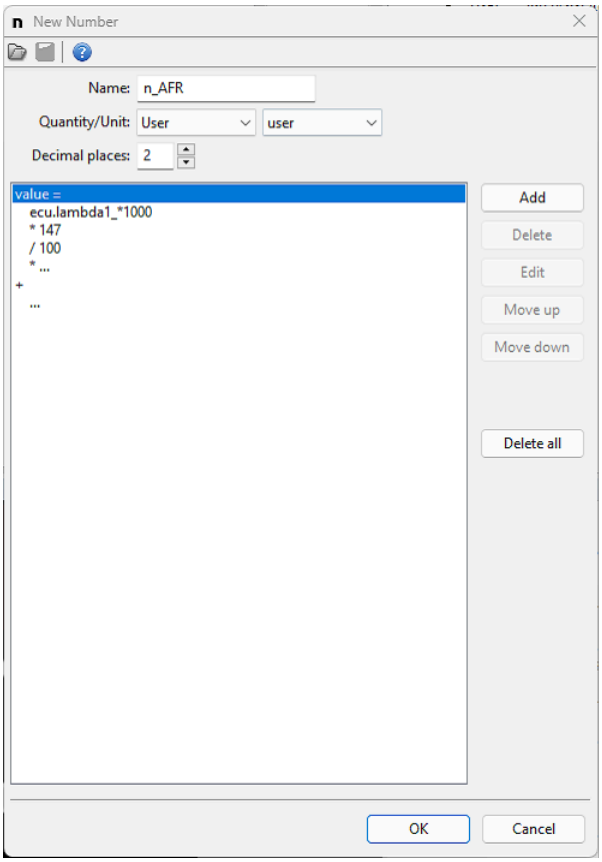
The "/" operation means integer division.

If you need to apply more complex mathematical operations (and it is difficult or impossible to keep the order of operations in one *Number* channel), you should break the operation into several stages by using multiple *Number* channels.

e.g. n_1 = C1+C2+C3
n_2 = C4+C5+C6
n_3 = n_1*n_2

If the final result of such operations should have the appropriate accuracy, remember that this accuracy must be taken into account at each stage of the calculation (in each intermediate channel created).

Changing the unit from lambda to AFR



n_AFR - channel displaying values with an accuracy of two decimal places

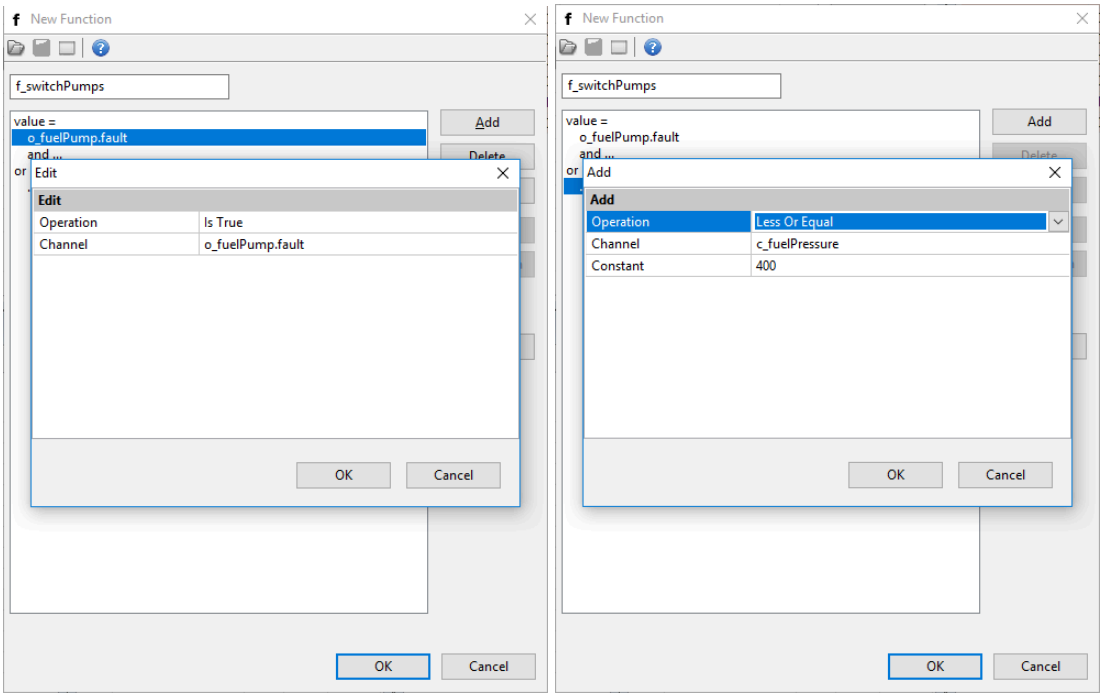
For *ecu.lambda* channel, use the value modifier (multiplier) ***1000*** and "move" the decimal point to the left by two decimal places (via ***Decimal places: 2***)

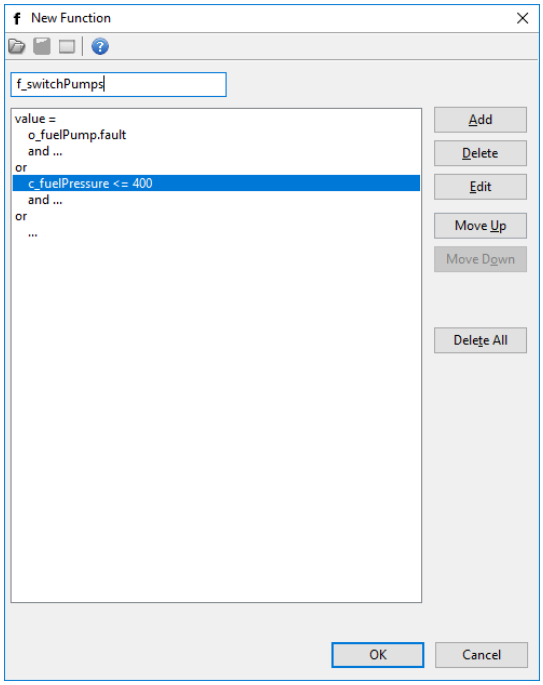
The following formula should be input:
n_AFR = (ecu.lambda*1000) * 147 / 100

13.6. Functions – logical functions

Logical functions are used to create a set of rules and conditions for powering an output device. As a result of these operations, a value of 1 - true or 0 - false can be obtained.

An example would be that the auxiliary fuel pump is turned on if a fault is detected on the main pump or the fuel pressure is less than or equal to 400 kPa.
To achieve this, create a new function (e.g. called f_switch Pumps). Use ***Is True*** for the first condition and ***Less Or Equal*** for the second.





List of operations available for logical functions

Operations for logical functions can be divided into two groups: simple and special.

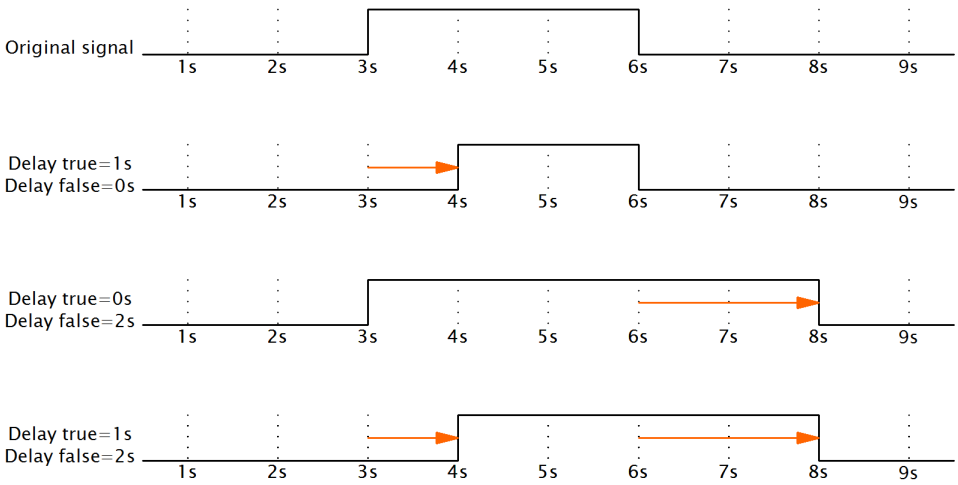
Simple operations are those whose result depends on the input state (alternatively a delay can be used for this result). Simple operations include: testing (*Is False, Is True*), (=, ≠, <, ≤, >, ≥) comparisons and logic operations (*And, Or, Xor*).

Important:

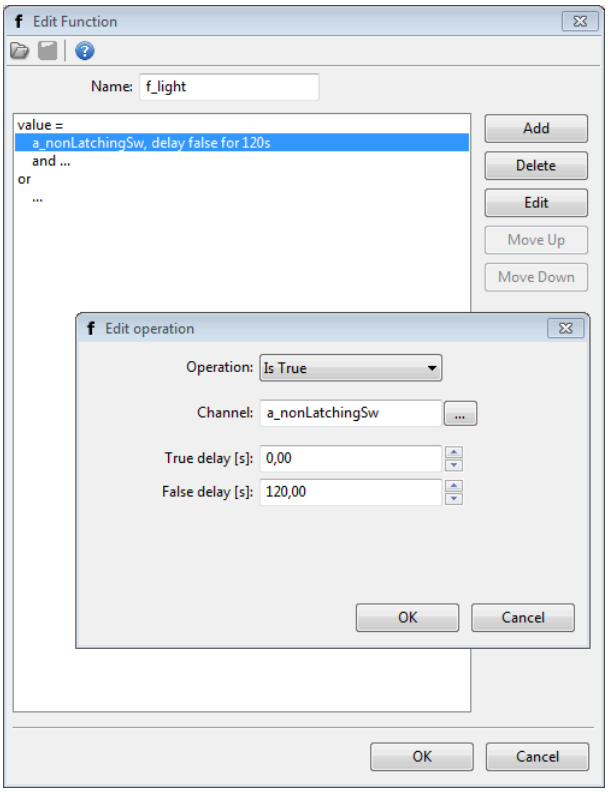
The following description contains **false** and **true** notions. **False** means a value of '**0**' (zero), and **true** means any value **other** than zero (e.g. '**1**').

Testing operations	
Is True	Returns **1** when the Channel value is **true** (non-zero); **0** otherwise.
Is False	Returns **1** when the Channel value is **false** (zero); **0** otherwise. (In electronics a NOT gate is analogous to this operation.)
Comparing operations	
Equal	Returns **1** when the *Channel* value **=** *Constant;* returns **0** otherwise.
Not Equal	Returns **1** when the *Channel* value **≠** *Constant;* returns **0** otherwise.
Less	Returns **1** when the *Channel* value **<** *Constant;* returns **0** otherwise.
Less or Equal	Returns **1** when the *Channel* value **≤** *Constant;* returns **0** otherwise.
Greater	Returns **1** when the *Channel* value **>** *Constant;* returns **0** otherwise.
Greater or Equal	Returns **1** when the *Channel* value **≥** *Constant;* returns **0** otherwise.
Logic operations	
And	Returns **1** when the values of both *Channel #1* and *Channel #2* are **true** (non-zero); returns **0** otherwise.
Or	Returns **1** when at least one of the channels, i.e. *Channel #1* or *Channel #2* is **true** (non-zero), returns **0** otherwise.
Xor	(Exclusive Or) Returns **1** only when exactly one of the channels *Channel #1* or *Channel #2*, has a value of **true** (non-zero), returns **0** otherwise.

All simple operations allow to delay the switching on (***Delay true***) and the switching off (***Delay false***). The figure below shows the original signal and the following figures show how the ***Delay true*** and ***Delay false*** parameters modify with this signal.



The bulb goes on following pressing the button and remains on for another 120 s after releasing it.



This functionality can be achieved by using the *Is True* operation with the parameter *Delay false* = 120 s.

Special operations

<table>
<tr><td colspan="2">Signal generating</td></tr>
<tr><td>Flash</td><td>This operation generates impulses so long as the Channel is true (non-zero).

When the Channel value assumes false (zero), the operation returns the value 0. When high state appears on the Channel channel (non-zero value), the Flash operation starts cycling between the value of 1 (duration defined by Time on) and the value 0 (duration defined by Time off). When the Channel value is false (zero), the operation will immediately start returning 0, thus interrupting the cycle.</td></tr>
<tr><td>Pulse</td><td>This operation generates N impulses following appearance of a trigger edge.

When the selected edge appears (Rising or Falling) on the Channel impulse generation will start. The number of impulses is determined by the Count parameter. Each impulse has an active phase (then the operation returns 1) and a non-active phase (the operation returns 0).

The Retrigger parameter determines if the appearance of a trigger edge during impulse generation will cause the process to restart or if it will be ignored.</td></tr>
<tr><td colspan="2">State-storing operations</td></tr>
<tr><td>Set-Reset Latch</td><td>The operation sets a new or returns the previous one according to the settings of the two input channels: Set Channel and Reset Channel.

<table>
<tr><th>Set channel value</th><th>Reset channel value</th><th>Operation value</th></tr>
<tr><td>true (non-zero)</td><td>false (0)</td><td>1</td></tr>
<tr><td>false (0)</td><td>true (non-zero)</td><td>0</td></tr>
<tr><td>true (non-zero)</td><td>true (non-zero)</td><td>0</td></tr>
<tr><td>false (0)</td><td>false (0)</td><td>previous value</td></tr>
</table>
An analogous operation is performed in the electronic SR latch. SR latch):

https://en.wikipedia.org/wiki/Flip-flop_(electronics)

The initial value of this operation following starting the device can be defined using the Default State parameter.</td></tr>
</table>

Toggle

Toggle changes the state between **0** and **1** each time the selected ***Edge*** (***Rising*** or ***Falling***) appears on the ***Channel***.

Set channel allows setting the value to **1**, and ***Reset channel*** resets the value to **0**. The initial value of this operation following starting the device can be defined using the ***Default State***.

Toggle channel	***Set channel*** value	***Reset channel*** value	Operation value
Rising	false (0)	false (0)	state change
Falling	false (0)	false (0)	previous state
x	true (non-zero)	false (0)	1
x	x	true (non-zero)	0

x - regardless of condition

The table uses the ***Toggle*** channel with an ***Edge: Rising***.

Detecting changes

Changed

When the value of ***Channel*** changes by a predefined ***Threshold***, the operation will initiate an active state (it will return the value **1**) for the amount of seconds defined using the parameter ***Time on***. If, during this time, the channel value changes by the set threshold once again, the active state will be extended again by the number of seconds specified by the parameter ***Time on***. After the end of the active state, the operation will begin returning the value **0**.

Hysteresis

Hysteresis

a) For the ***Polarity=Above*** parameter

If the value of ***Source channel*** is greater than the predefined ***Upper value*** threshold, the value of the operation will be **1**. If it is lower than the ***Lower value*** threshold, the value of the operation will be **0**. If it is within **[*Lower value*, *Upper value*]** range, the value of the operation will be the previous value.

b) For the ***Polarity=Below*** parameter

	If the value of ***Source channel*** is lower than the predefined ***Lower value*** threshold, the value of the operation will be **1**. If it is greater than the ***Upper value*** threshold, the value of the operation will be **0**. If it is within ***[Lower value, Upper value]*** range, the previous value will be the value of the operation.

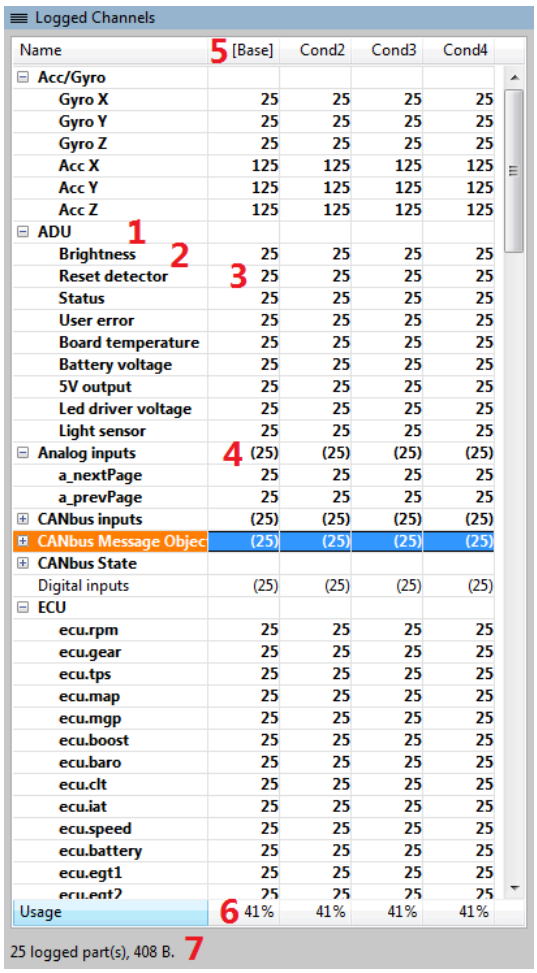
Important:

For ***Pulse***, ***Flash*** and ***Changed*** operations setting the ***Time on*** parameter to 0 s will result in generation of a 2 ms impulse.

14. Logging channels

The ***Logged Channels*** window defines the logging frequencies of channels. These values are expressed in Hz.

It is worth noting that the same frequencies are used for both logging into the memory and for logging directly to the PMU Client program on a PC.



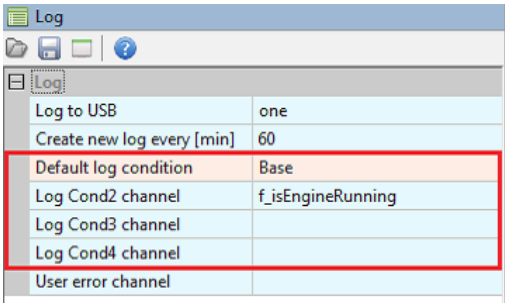
In the configuration window, we can distinguish the following elements:

- Groups (1) containing channels associated with a particular group
- Channels (2) containing data corresponding to their names
- Channel logging frequencies (3)
- Default logging frequencies for new elements (4). When a new element is created (e.g. *Analog Input*), one of these default frequencies will be assigned to its channel.
- ***Log condition*** (5), after which channels from a given column are logged
- The bandwidth usage (6), expressed in [%] for particular Log conditions
- The bandwidth usage expressed in bytes (7).

Parameters in bold are those that are logged. Not all parameters are logged by default! Double-clicking an item will toggle between logging and not logging it. If a new item is created, each subchannel associated with it is also logged.

Configuration can be carried out in the context menu or by using the shortcut keys listed below. If a given command or key is used on an entire group, the frequency will change for all channels within it. However, if they are used in a single channel, they will change the frequency of that channel only. ***Log condition*** values may be changed individually or all at once depending on the column selected.

Key:	**Logging frequency:**
Alt+~	Deactivation of channel / group logging
Alt+1	1 Hz
Alt+2	5 Hz
Alt+3	25 Hz
Alt+4	50 Hz
Alt+5	125 Hz
Alt+6	250 Hz
Alt+7	500 Hz

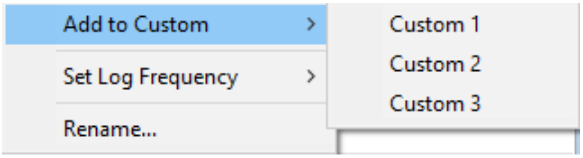


A given log condition is selected 25 times per second. If the values of two condition channels are non-zero (e.g. the one selected in the ***Log Cond2 channel*** field and the one selected in the ***Log Cond3 channel*** field) the first one will be selected.

14.1. Custom Log

PMU Client allows the user to create three separate custom logging groups. Any channels can be added to each of them (***Custom 1***, ***Custom 2***, ***Custom 3***).

To add a selected channel (from any variable preview panel) to a Custom group, right-click on it, then select ***Add to Custom*** and choose the appropriate group (***Custom 1***, ***Custom 2*** or ***Custom 3***).



15. Analog Emulator and Output Emulator

Analog Emulator is used to simulate the value of each input to check whether the assumed strategies or functions work correctly.

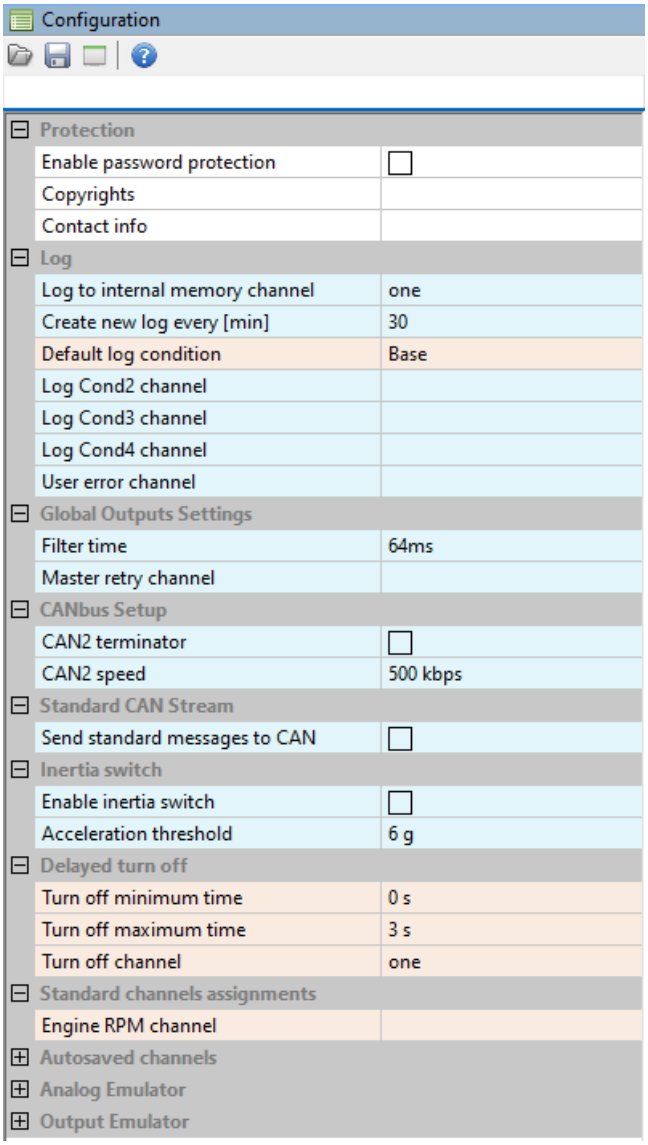
The ***Output Emulator*** is used to simulate the value of each output.
Information flowing from the emulator has the highest priority, as a result of which data flowing on the CAN bus are bypassed.

16. Additional configuration panels

Panels offer additional device configuration options not included in the main menu or the ***Project tree***.

16.1. Configuration

The *Configuration* panel consolidates all settings into a single location. At the top of the panel, just below the toolbar, there is a filter field to help you quickly find the desired setting by typing in its name.



.

16.2. Protection

The *Protection* panel contains options related to the password protection of the device. A password must be provided before any changes can be made to the device if protected. If no password is provided the only way to remove the protection is by restoring the default settings.

Parameter	Description
Enable password protection	Activates password protection of the device
Copyrigths	Information displayed when attempting to connect to the device if protected
Contact info	Information including contact details (e-mail, telephone) displayed when attempting to connect to the device if protected

16.3. Log

The *Log* panel contains the logging configuration.

Parameter	Description
Log to internal memory channel (only for PMU DL and PMU AS)	A channel / variable determining if the device is to log to the internal memory. Only the PMU DL and PMU AS have built-in memory. The value "*one*" means that logging is to be active the whole time. Using this parameter allows to formulate the condition determining when logging is to be active / inactive.
Create new log every [min]	It determines the maximum size of a single file in minutes. When the size of the logged data exceeds the predefined time, a new file will be created.
Default log condition	The PMU device allows to create 4 logging profiles. These profiles are defined in *Menu* > *Logged channels*. They allow to record to a file different channels with different frequencies depending on the current condition. The *Default log condition* parameter defines the default profile that is to be used if none of the conditions for the profiles will be met.
Log Cond2 channel	A channel / function activating logging of the *Cond 2* profile
Log Cond3 channel	A channel / function activating logging of the *Cond 3* profile

Parameter	Description
Log Cond4 channel	A channel / function activating logging of the *Cond 4* profile
User error channel	User-defined error channel

Logging to the built-in internal memory in the PMU DL device is active by default. The log file is saved in the *.PMULOG* format. This is the same format in which the PMU Client logs are saved. Files are created cyclically every defined amount of time (default 1h, maximum 5h). Later analysis is easier when the files are relatively small.

After connecting the device's memory to a PC, in the PMU Client, select the ***Receive logs*** from the ***Devices*** menu (Shift+F4 key) to view the logs stored on this memory.

16.4. Global Outputs Settings

In the *Global Output Settings* panel, you can assign a channel to reset all outputs and set a sample filter for the output current and voltage.

Parameter	Description
Filter time	Averaged recording of output current and output voltage samples over a specified period of time. The default filter value is 64ms. Changing this value may lead to log artefacts or inaccuracies.
Master retry channel	Any channel or component that allows power outputs to be reset. (If the output has signalled a fault due to a not allowed state and is turned off, then triggering this channel will reset all outputs)

16.5. CANbus Setup

The *CANbus setup* panel is used for configuring the CAN bus and for RS232 serial communication.

Parameter	Description
CAN 2 terminator	Activation of the terminator on the CAN2 bus
CAN 2 speed	The speed of the CAN2 bus

16.6. Interia switch

The PMU constantly monitors its gyro values and acceleration in each axis to react quickly in the event of an accident. An emergency stop switch is provided for this purpose, which is activated in the event of too much overload. After exceeding the maximum value of the PMU, it will turn off all outputs.

Parameter	Description
Enable interia switch	The inertia switch instantly shuts down all power outputs to prevent any accidents in the event of failure. It is triggered when a user-defined acceleration threshold is exceeded.
Acceleration treshold	Acceleration limit

16.7. Delayed turn off

By default, the PMU switches off immediately when the ignition is switched off. It is possible to configure an PMU switch-off delay. This requires separate battery (*Battery 12V*) and ignition switch (*Switched 12V*) wiring.

The PMU enters a ***Delayed turning off*** state when the ignition is switched off and remains in this state until it is completely switched off. When the device is operating in ***Delayed turning off*** mode, the value of the ***adu.isTurningOff*** channel is **1**.

To configure the shutdown delay, select ***Desktop / Add new panel*** from the menu or use the **F9** key and select ***Delayed turn off*** from the list.

Define the following parameters in the ***Delayed turn off*** panel:

Parameter	Description
Turn off minimum time	Minimum time after turning off the ignition that must elapse for the ***Turn off channel*** to turn off the PMU
Turn off maximum time	Maximum time after turning off the ignition that the PMU will remain on (provided that the ***Turn off channel*** has not been triggered beforehand)
Turn off channel	PMU shutdown control channel. When the defined channel has a non-zero value between ***minimum time*** and ***maximum time***, the PMU will switch off.

Turning the ignition back on while in ***Delayed turning off*** mode on the unit will restart the PMU 0.5 seconds after the ignition is switched on.

16.8. Standard channels assignment

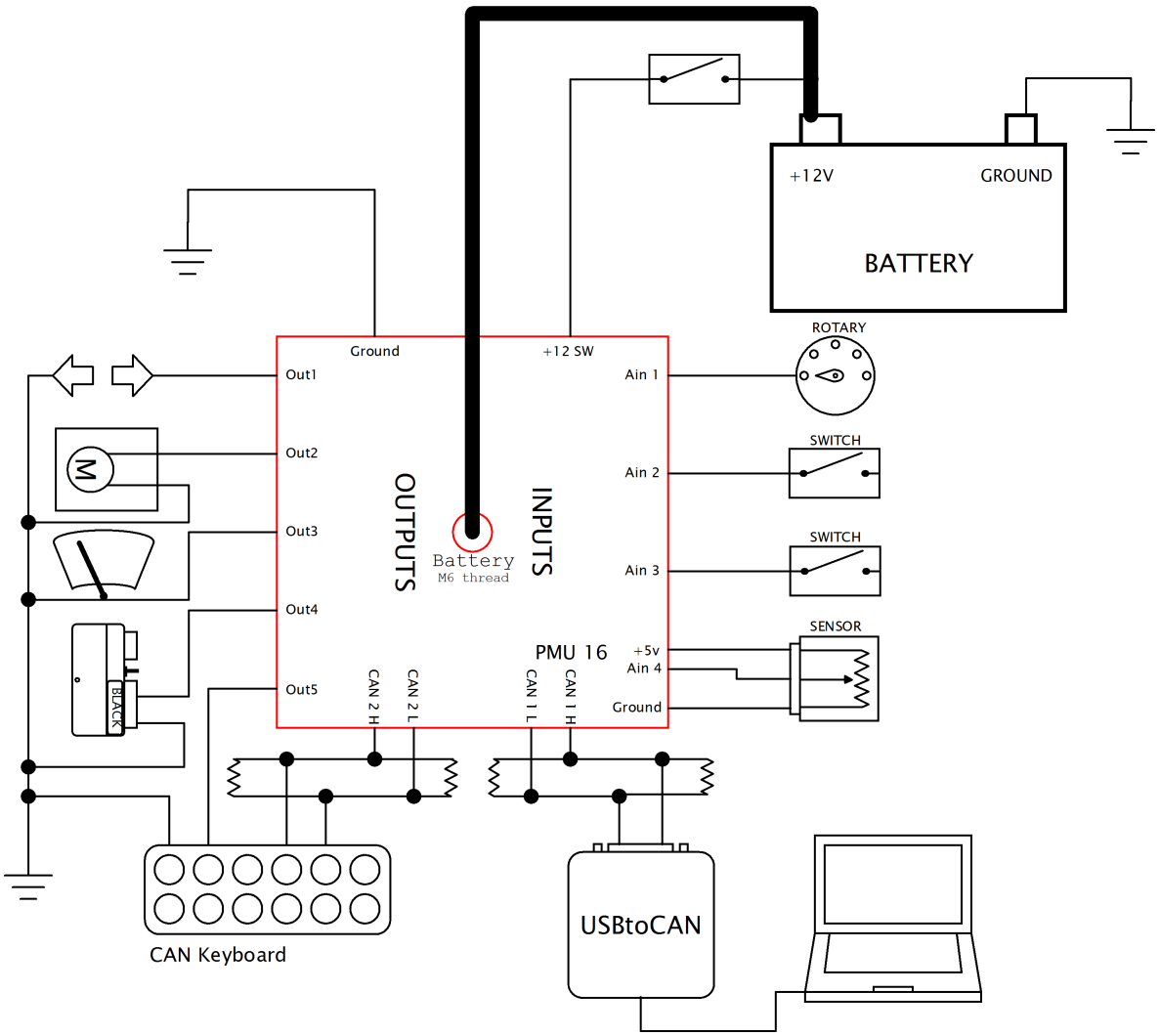
The Standard Channels Assignment panel is used to set the Engine RPM channel, required by the Autosaved Channels panel when the RPM decreases to zero; more details can be found in Appendix C - How-to Configure Autosaved Channels *(on page 156)*.

16.9. Autosaved channels

The Autosave Channels feature allows users to store values for up to 20 channels, which are loaded when the device starts. These values are saved automatically. Full description can be found in: Appendix C - How-to Configure Autosaved Channels *(on page 156)*.

17. Examples of diagrams and settings

Basic diagram of PMU communication and connections.



Key elements:

- PC Communication takes place on CAN1.
- CAN1 has two 120 Ohm terminators on the CAN bus. They are necessary because the PMU does not provide termination on CAN1.
- The CANbus Keyboard is connected to CAN2.
- Power to the PMU is supplied in two ways. First using the ignition which connects to the +12SW Pin. Second, using the positive battery terminal which connects to the M6 Bolt located on the PMU case.

Suggested wire size for continuous current (chassis wiring, Tefzel)

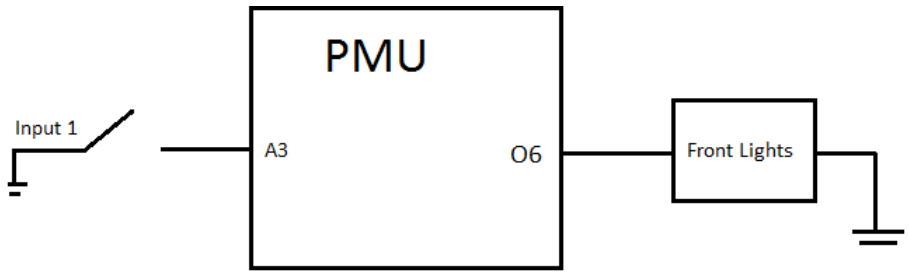
5A	10A	15A	25A
1 mm^2	1,5 mm^2	2,5 mm^2	4 mm^2
AWG 18	AWG 16	AWG 14	AWG 12

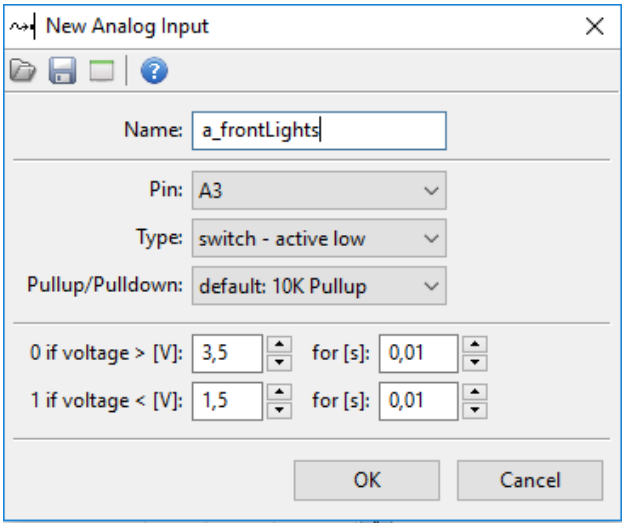
For the power connecter at least 25mm^2 (AWG3) is required!

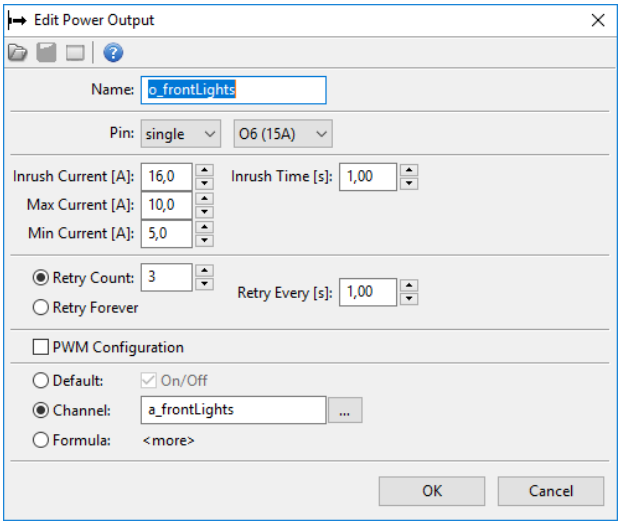
Load examples for popular devices

Device	Inrush Current [A]	Continuous Current [A]
Fuel Pump	15	7
Fan	50	20
Front Lights	16	10
EMU Black	0.12	0.12

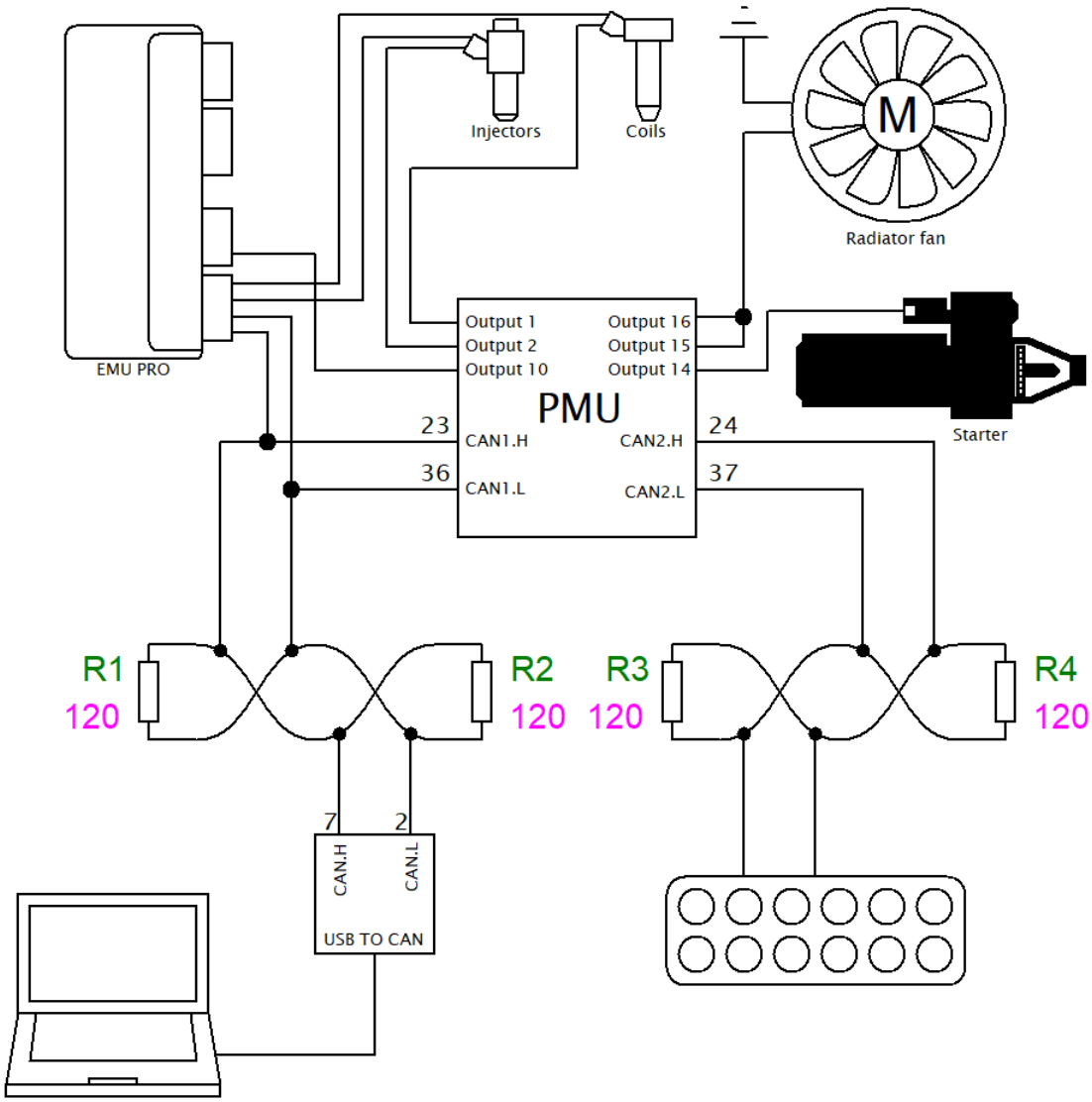
Example of activating the Power Output O6 through the Analog Input:





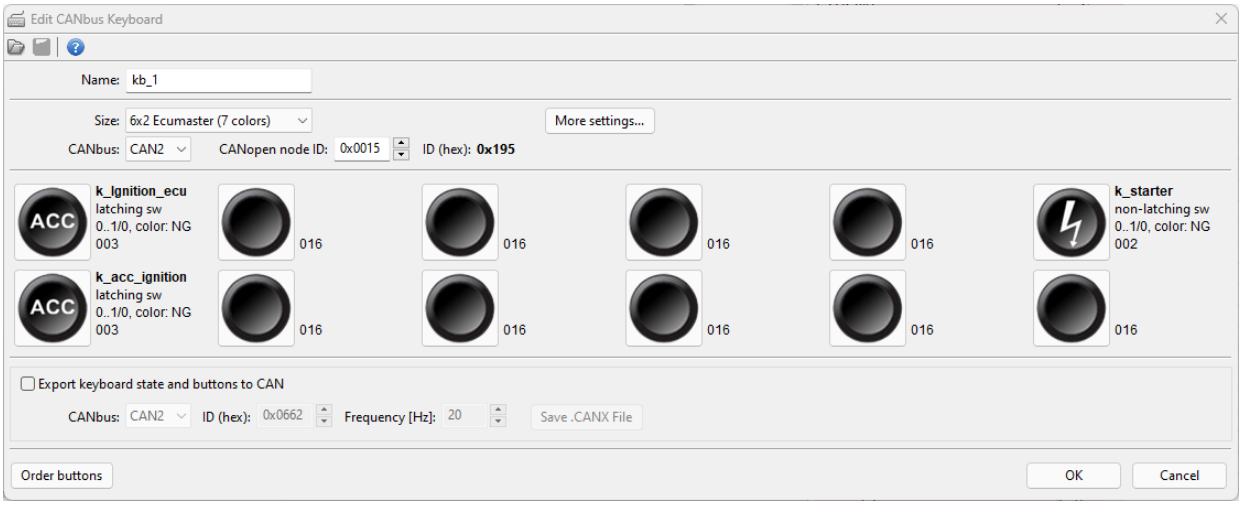


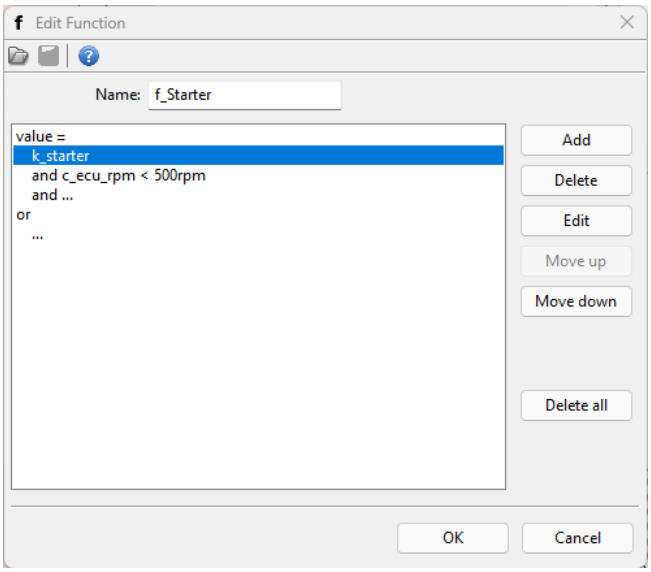
Connected Devices and Control in PMU: Configuration Example

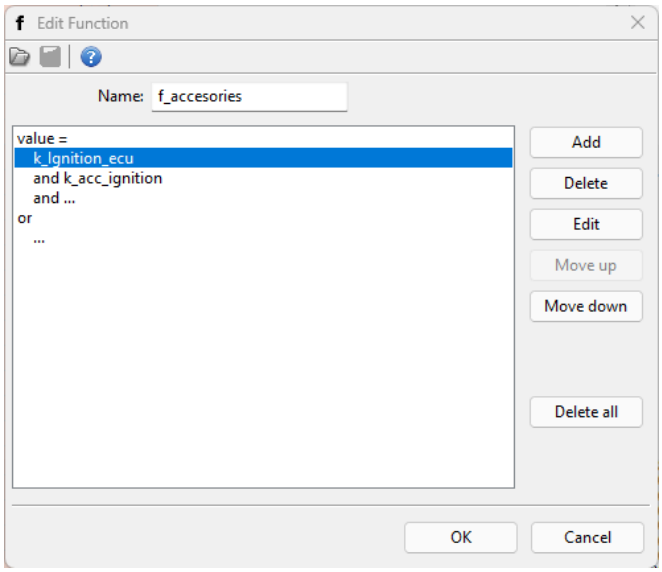


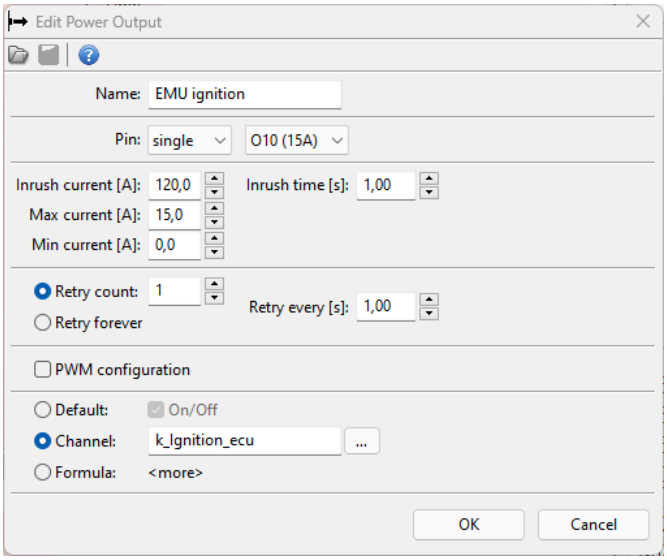
Project Tree

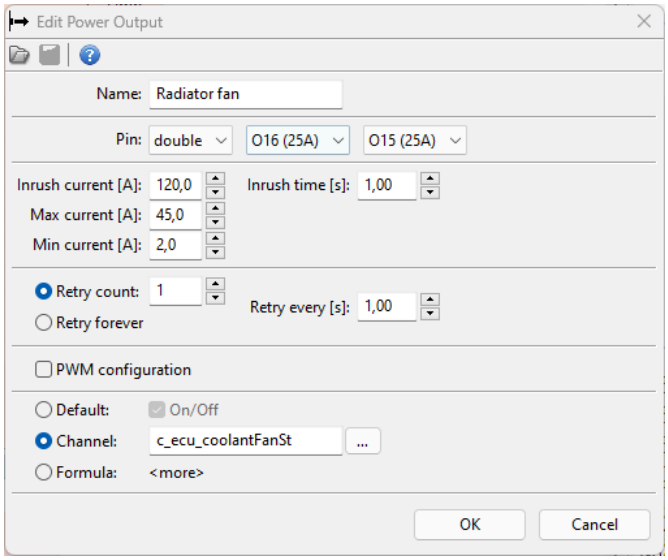
Name	Formula	Details
m_emublack	CAN1 0x600 - 8 frames	
kb_1	6x2 Ecumaster (7 colors)	CANopen node ID: 0x15, CAN2
f_Starter	(k_starter) and (c_ecu_rpm < 500rpm)	
f_accesories	(k_Ignition_ecu) and (k_acc_ignition)	
EMU ignition	k_Ignition_ecu	O10, max 15A (inrush 120A 1s): 1x / 1s
Radiator fan	c_ecu_coolantFanSt	O16,O15, min 2A, max 45A (inrush 120A 1s): 1x / 1s
Starter	f_Starter	O14, max 25A (inrush 120A 1s): 1x / 1s
Injectors	f_accesories	O2, max 20A (inrush 120A 1s): 1x / 1s
Coils	f_accesories	O1, max 25A (inrush 120A 1s): 1x / 1s

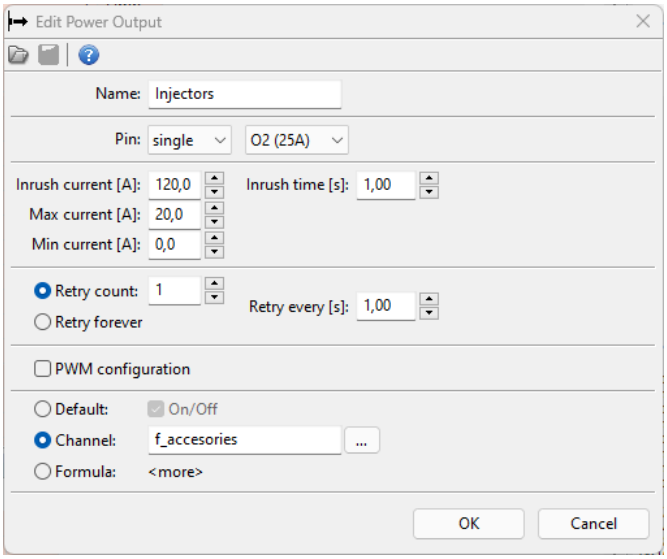

Edit CANbus Keyboard
Name: kb_1
Size: 6x2 Ecumaster (7 colors)
More settings...
CANbus: CAN2
CANopen node ID: 0x0015
ID (hex): 0x195
k_Ignition_ecu
latching sw
0..1/0, color: NG
003
k_starter
non-latching sw
0..1/0, color: NG
002
k_acc_ignition
latching sw
0..1/0, color: NG
003
016
Export keyboard state and buttons to CAN
CANbus: CAN2
ID (hex): 0x0662
Frequency [Hz]: 20
Save .CANX File
Order buttons
OK
Cancel

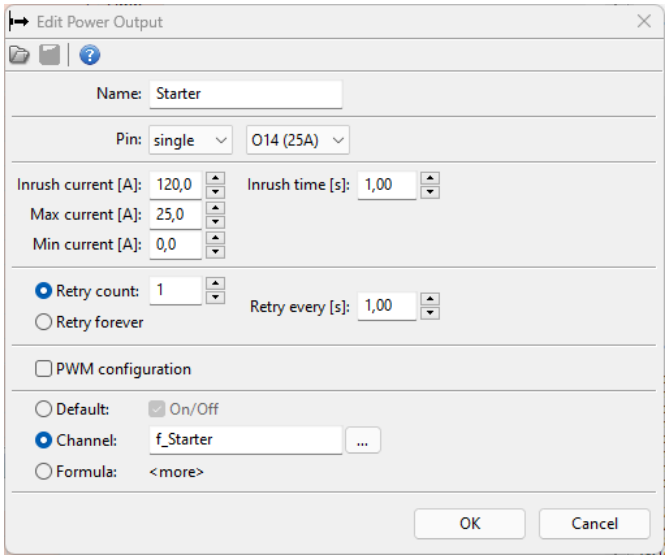

Edit Function
Name: f_Starter
value =
k_starter
and c_ecu_rpm < 500rpm
and ...
or
...
Add
Delete
Edit
Move up
Move down
Delete all
OK
Cancel

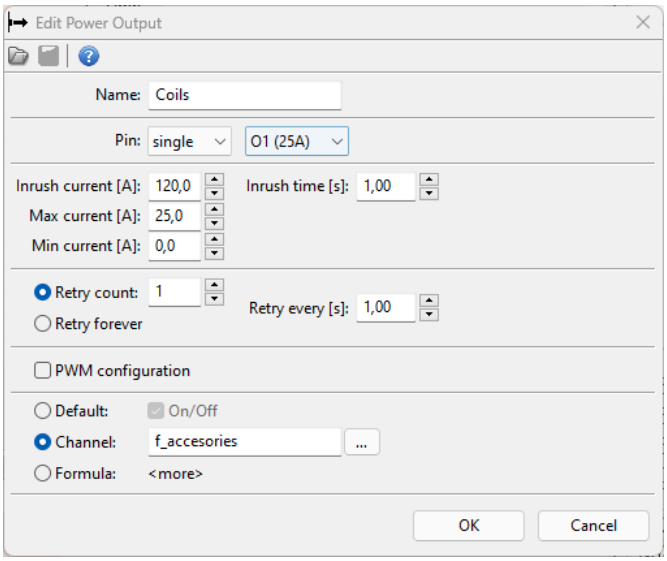

Edit Function
Name: f_accesories
value =
k_Ignition_ecu
and k_acc_ignition
and ...
or
...
Add
Delete
Edit
Move up
Move down
Delete all
OK
Cancel

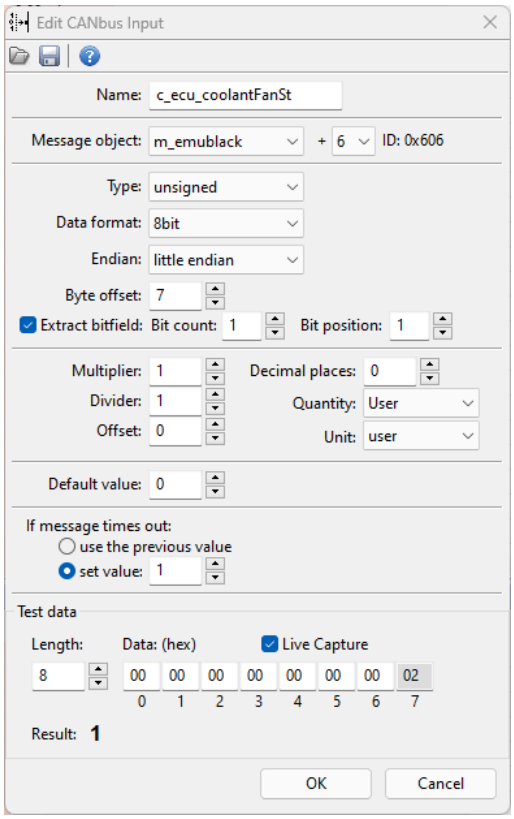

Edit Power Output
Name: EMU ignition
Pin: single
O10 (15A)
Inrush current [A]: 120,0
Inrush time [s]: 1,00
Max current [A]: 15,0
Min current [A]: 0,0
Retry count: 1
Retry forever
Retry every [s]: 1,00
PWM configuration
Default:
On/Off
Channel: k_Ignition_ecu
Formula: <more>
OK
Cancel

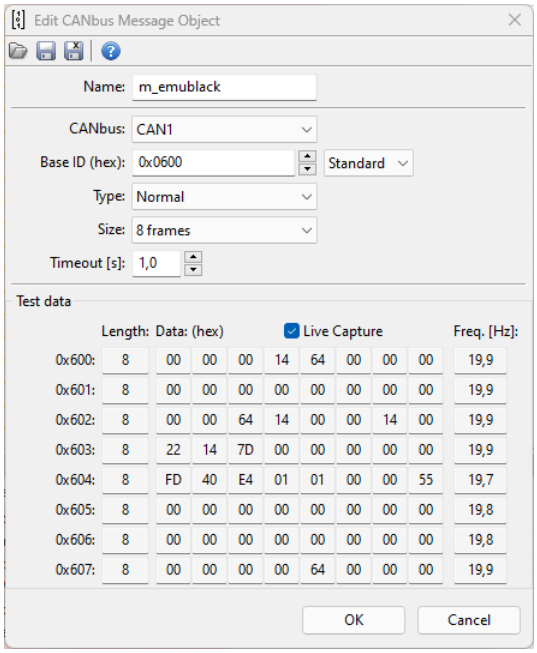

Edit Power Output
Name: Radiator fan
Pin: double
O16 (25A)
O15 (25A)
Inrush current [A]: 120,0
Inrush time [s]: 1,00
Max current [A]: 45,0
Min current [A]: 2,0
Retry count: 1
Retry forever
Retry every [s]: 1,00
PWM configuration
Default:
On/Off
Channel: c_ecu_coolantFanSt
Formula: <more>
OK
Cancel


Edit Power Output
Name: Injectors
Pin: single O2 (25A)
Inrush current [A]: 120,0
Inrush time [s]: 1,00
Max current [A]: 20,0
Min current [A]: 0,0
Retry count: 1
Retry forever
Retry every [s]: 1,00
PWM configuration
Default: On/Off
Channel: f_accesories
Formula: <more>
OK
Cancel


Edit Power Output
Name: Starter
Pin: single O14 (25A)
Inrush current [A]: 120,0
Inrush time [s]: 1,00
Max current [A]: 25,0
Min current [A]: 0,0
Retry count: 1
Retry forever
Retry every [s]: 1,00
PWM configuration
Default: On/Off
Channel: f_Starter
Formula: <more>
OK
Cancel


Edit Power Output
Name: Coils
Pin: single O1 (25A)
Inrush current [A]: 120,0
Inrush time [s]: 1,00
Max current [A]: 25,0
Min current [A]: 0,0
Retry count: 1
Retry forever
Retry every [s]: 1,00
PWM configuration
Default: On/Off
Channel: f_accesories
Formula: <more>
OK
Cancel


Edit CANbus Input
Name: c_ecu_coolantFanSt
Message object: m_emublack + 6 ID: 0x606
Type: unsigned
Data format: 8bit
Endian: little endian
Byte offset: 7
Extract bitfield: Bit count: 1 Bit position: 1
Multiplier: 1
Divider: 1
Offset: 0
Decimal places: 0
Quantity: User
Unit: user
Default value: 0
If message times out:
use the previous value
set value: 1
Test data
Length: 8
Data: (hex) 00 00 00 00 00 00 00 02
0 1 2 3 4 5 6 7
Live Capture
Result: 1
OK
Cancel



The following devices are connected to the PMU:

- Keyboard to CAN2 bus,
- ECU to CAN1 bus and output number 5,
- Radiator fan to output numbers 16 and 15,
- Injector power to output number 2,
- Coil power to output number 1, and
- Starter solenoid to output number 14.

The keyboard defines three buttons:

- Toggle button *k_ignition* (upon pressing, it changes the state to "true" (value 1), and upon the next press, to "false" (value 0)),
- Toggle button *k_acc_ignition*, and
- Momentary button *k_starter* (state "true" (value 1) only while pressed).

The ECU is connected to output number 10, which is powered when the k_ignition button is pressed, supplying power to the engine computer.

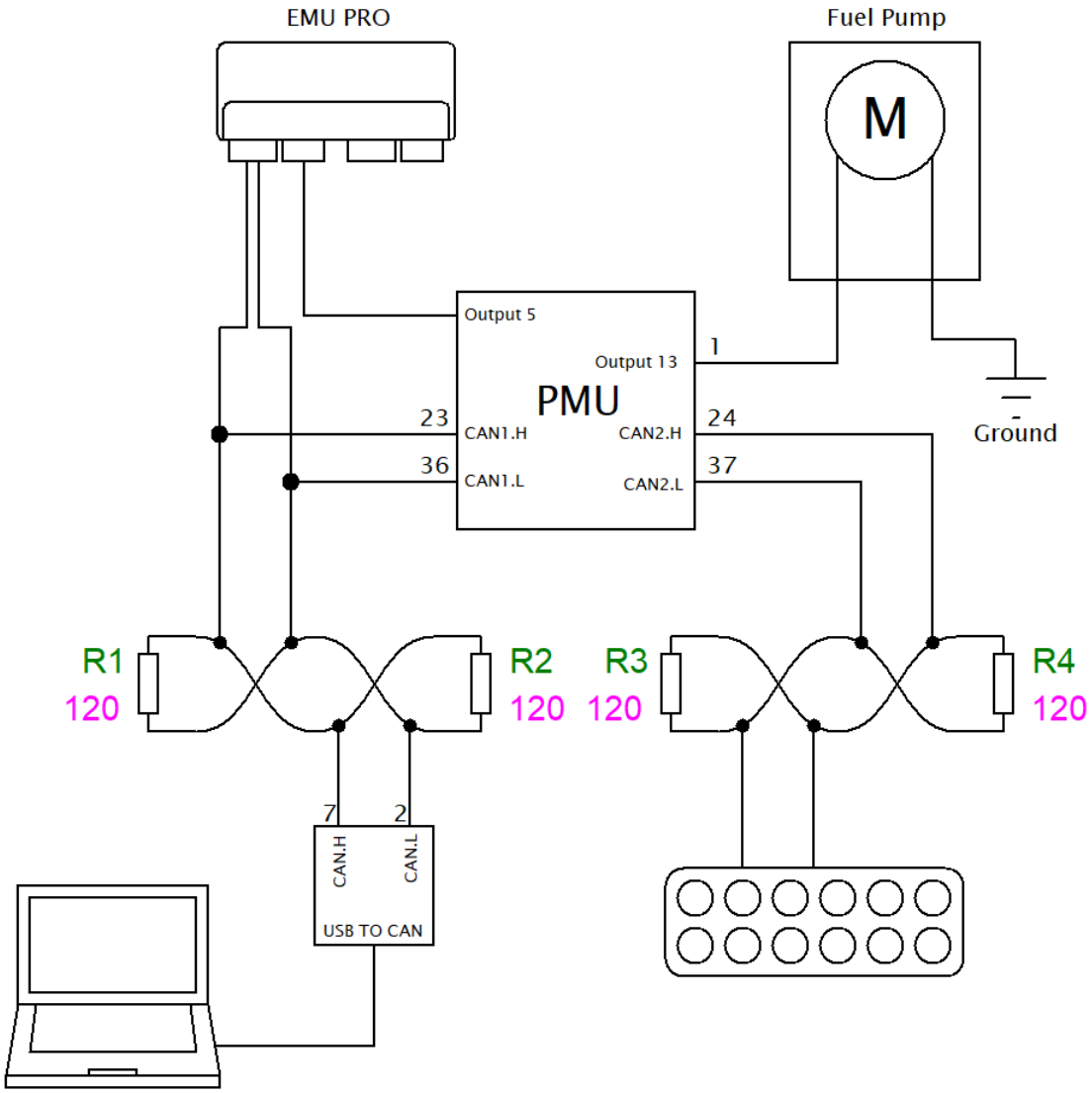
The radiator fan is activated through the c_ecu_coolantFanSt channel. This channel is broadcasted in CANstream from the ECU. When the channel is "true" (value 1), outputs numbers 15 and 16 are activated, powering the fan.

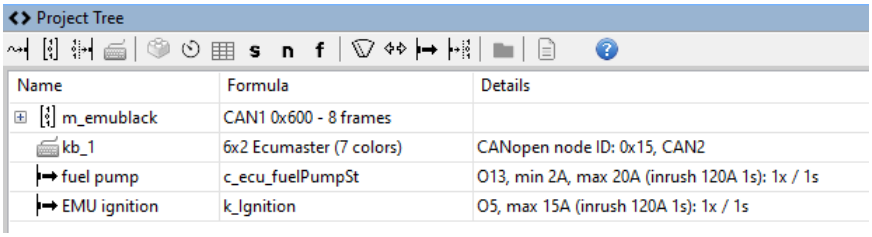
If the communication with the ECU times out (after 1 second), c_ecu_coolantFanSt channel will be set to default value 1 what will activate coolant fan (*If message times out* parameter).

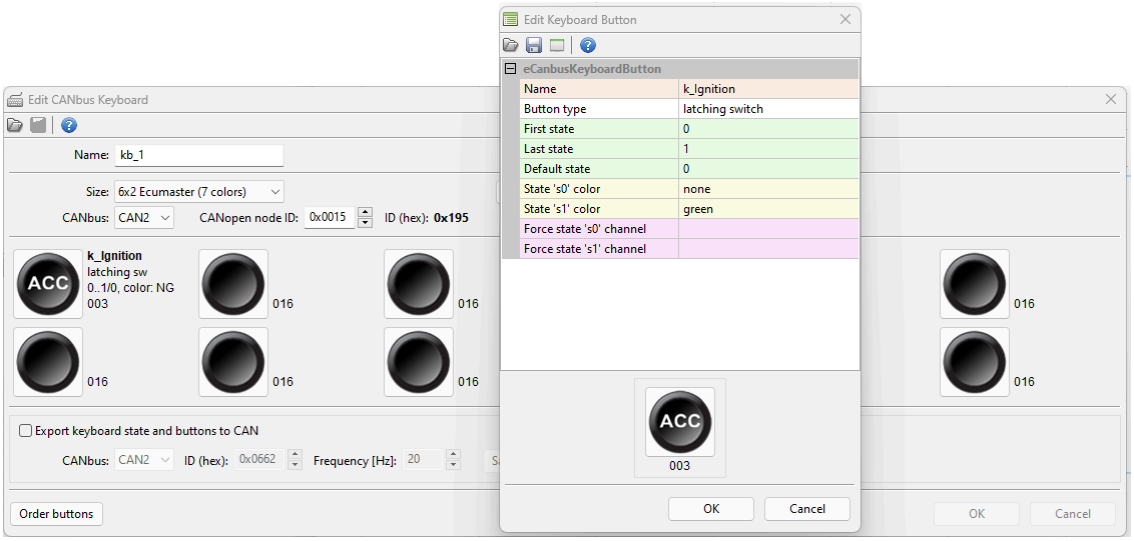
The outputs powering injectors and coils are activated when the conditions of the f_accessories function are met, meaning both the k_ignition and k_acc_ignition buttons are pressed.

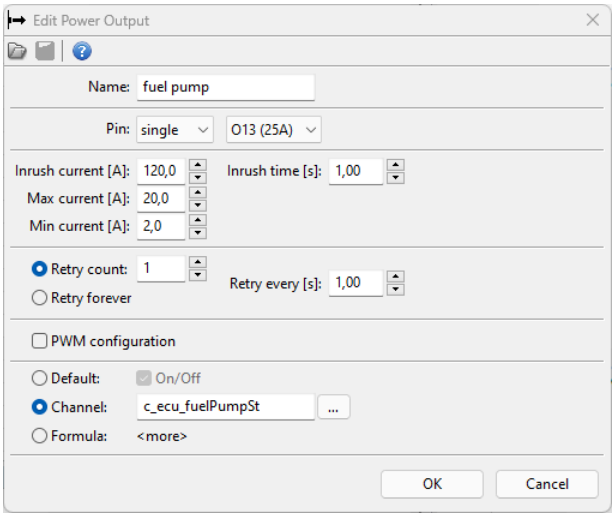
Output 14, controlling the starter solenoid, is powered when the conditions of the f_starter function are met. To satisfy this function, the engine RPM from the c_ecu_rpm channel must be below 500 RPM, and the k_starter button must be pressed. Thanks to the condition c_ecu_rpm < 500, the output won't be activated during engine operation."

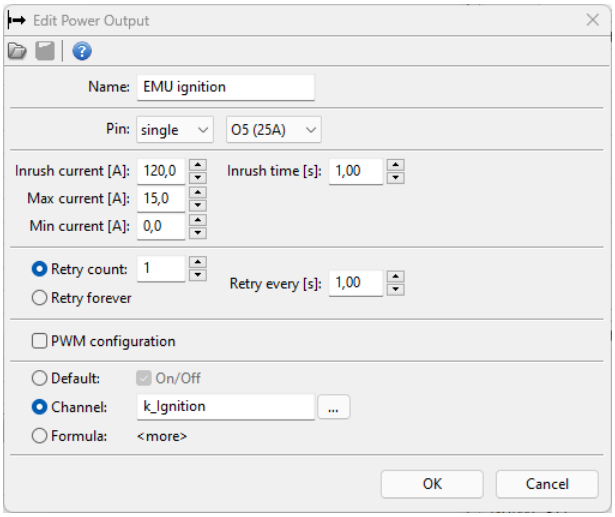
Example of Connecting a Keyboard, ECU, and Fuel Pump to the PMU

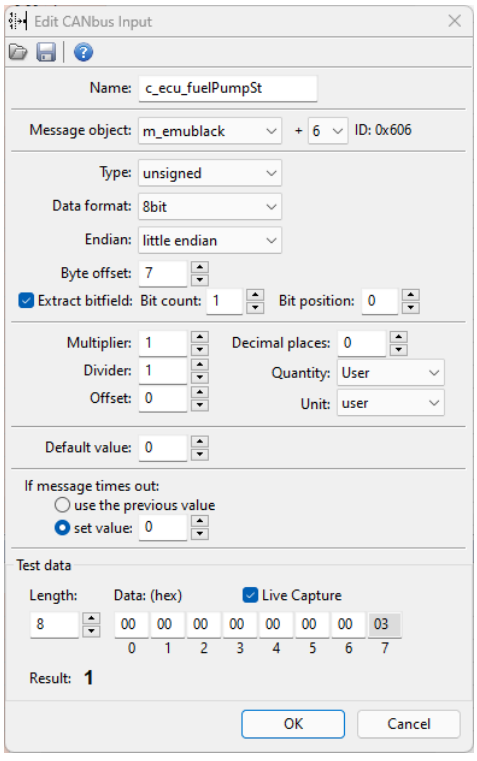





Edit Keyboard Button
eCanbusKeyboardButton
Name k_Ignition
Button type latching switch
First state 0
Last state 1
Default state 0
State 's0' color none
State 's1' color green
Force state 's0' channel
Force state 's1' channel
ACC
003
OK
Cancel
Edit CANbus Keyboard
Name: kb_1
Size: 6x2 Ecumaster (7 colors)
CANbus: CAN2
CANopen node ID: 0x0015
ID (hex): 0x195
k_Ignition
latching sw
0..1/0, color: NG
003
016
Export keyboard state and buttons to CAN
CANbus: CAN2
ID (hex): 0x0662
Frequency [Hz]: 20
Order buttons
OK
Cancel


Edit Power Output
Name: fuel pump
Pin: single
O13 (25A)
Inrush current [A]: 120,0
Inrush time [s]: 1,00
Max current [A]: 20,0
Min current [A]: 2,0
Retry count: 1
Retry forever
Retry every [s]: 1,00
PWM configuration
Default:
On/Off
Channel: c_ecu_fuelPumpSt
Formula: <more>
OK
Cancel


Edit Power Output
Name: EMU ignition
Pin: single
O5 (25A)
Inrush current [A]: 120,0
Inrush time [s]: 1,00
Max current [A]: 15,0
Min current [A]: 0,0
Retry count: 1
Retry forever
Retry every [s]: 1,00
PWM configuration
Default:
On/Off
Channel: k_Ignition
Formula: <more>
OK
Cancel


Edit CANbus Input
Name: c_ecu_fuelPumpSt
Message object: m_emublack
+ 6
ID: 0x606
Type: unsigned
Data format: 8bit
Endian: little endian
Byte offset: 7
Extract bitfield:
Bit count: 1
Bit position: 0
Multiplier: 1
Divider: 1
Offset: 0
Decimal places: 0
Quantity: User
Unit: user
Default value: 0
If message times out:
use the previous value
set value: 0
Test data
Length: 8
Data: (hex)
Live Capture
00 00 00 00 00 00 00 03
0 1 2 3 4 5 6 7
Result: 1
OK
Cancel

The following devices are connected to the PMU:

- Keyboard to CAN2 bus,
- ECU to CAN1 bus and output number 5, and
- Fuel pump to output number 13.

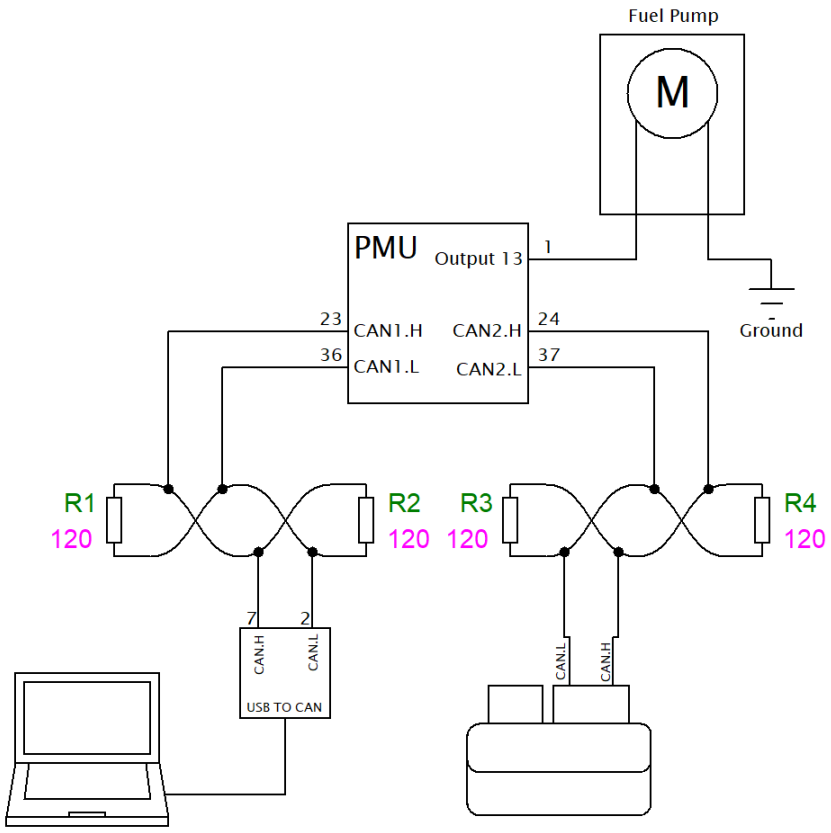
In the keyboard, a latching switch k_ignition has been defined. Upon pressing, it changes the state to "true" (value 1), and upon the next press, to "false" (value 0).

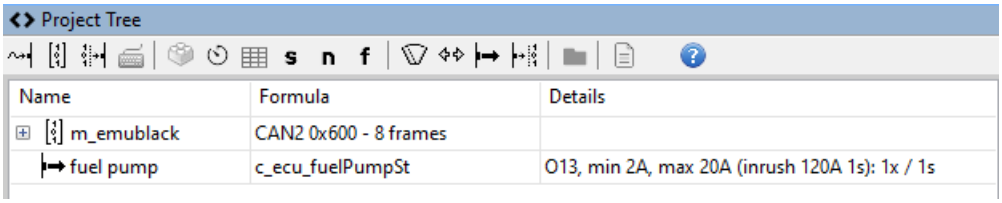
The ECU is connected to output number 5, which is powered when the k_ignition button is pressed, supplying power to the engine's computer.

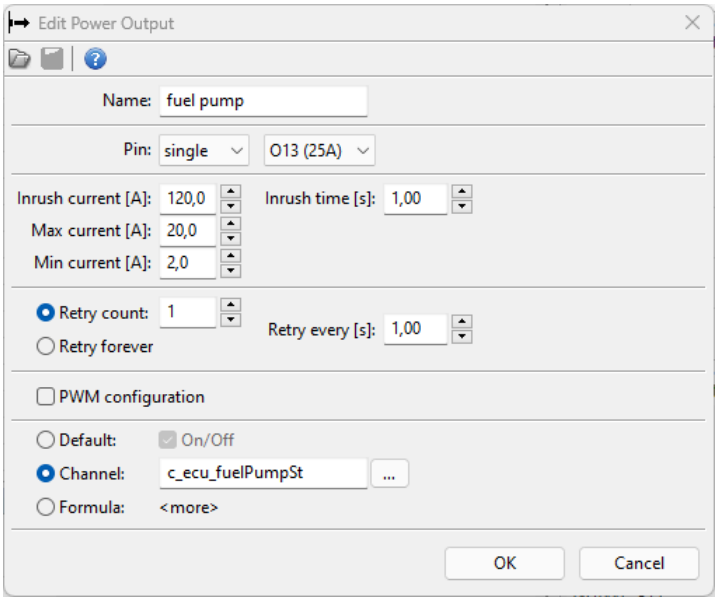
The fuel pump is activated by the c_ecu_fuelPumpSt channel, which is transmitted via CANstream from the ECU. When the channel has a status of "true" (value 1), output number 13 is activated, powering the fuel pump.

If the communication with the ECU times out (after 1 second), c_ecu_fuelPumpSt channel will be set to default value 0 what will deactivate fuel pump (*If message times out* parameter) for safety reasons (ie. crash).

Example of Fuel Pump powered by the PMU, controlled by the ECU via a CAN Signal



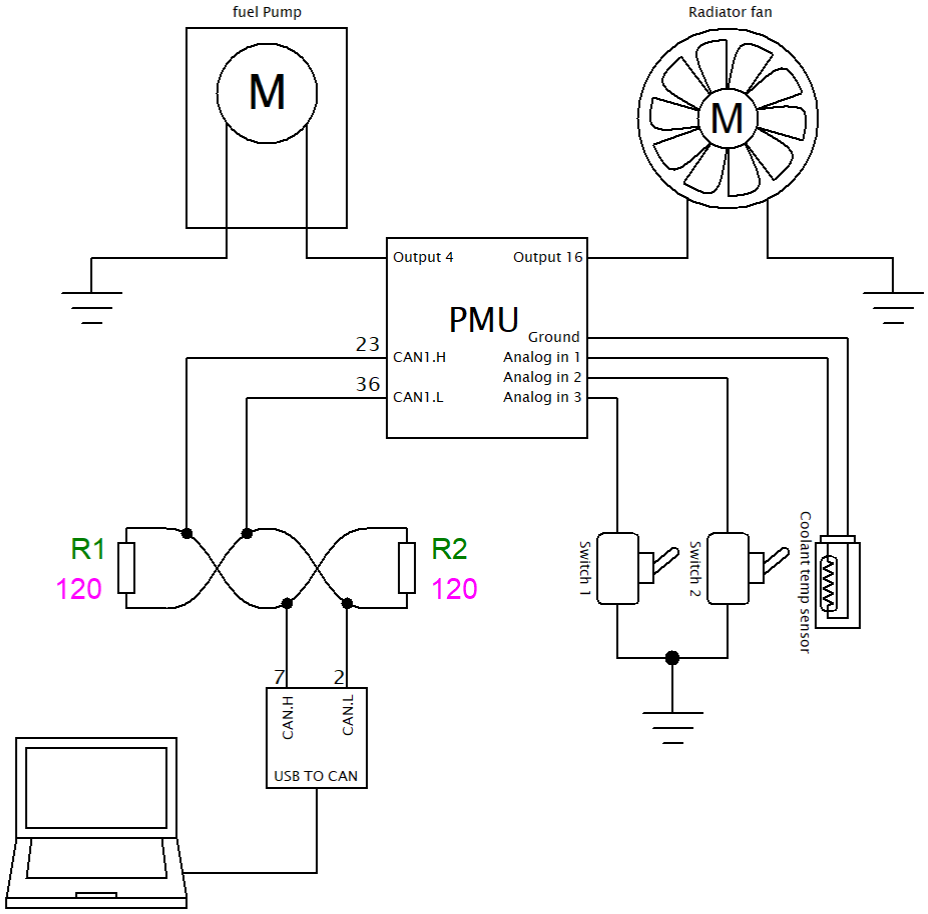




The ECU controls the fuel pump output. The output state is transmitted over the CAN bus from the engine computer.

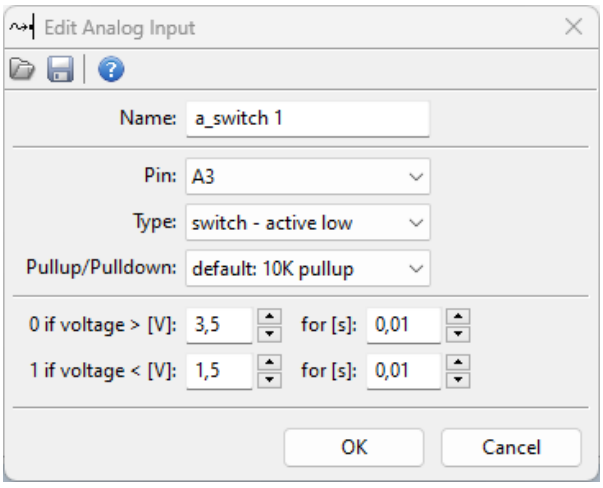
The parameter c_ecu_fuelPumpSt is assigned to output number 13, to which the fuel pump is connected. When the parameter is in the "true" state (value 1), the output powers the fuel pump.

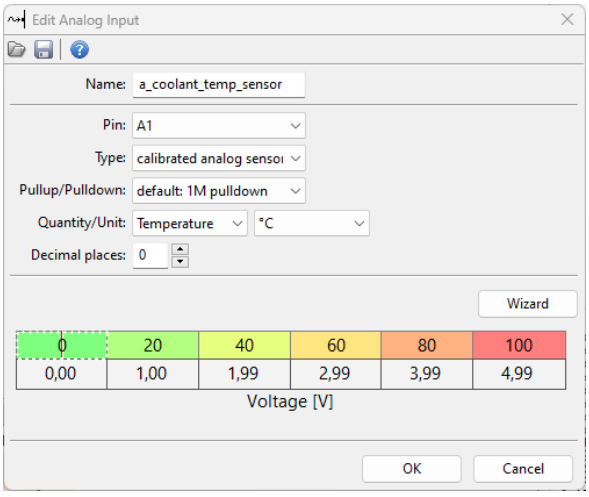
Example of using switches

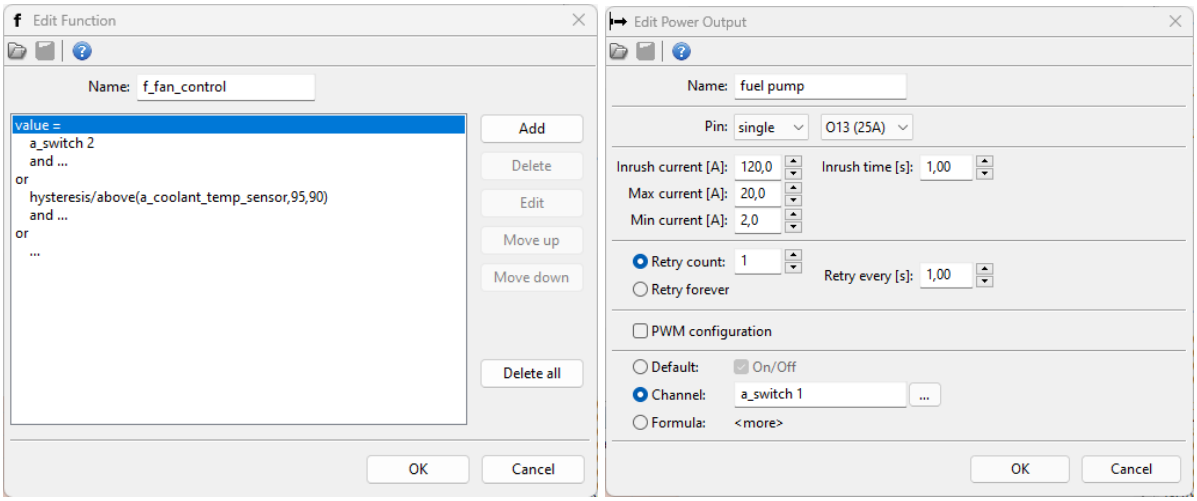


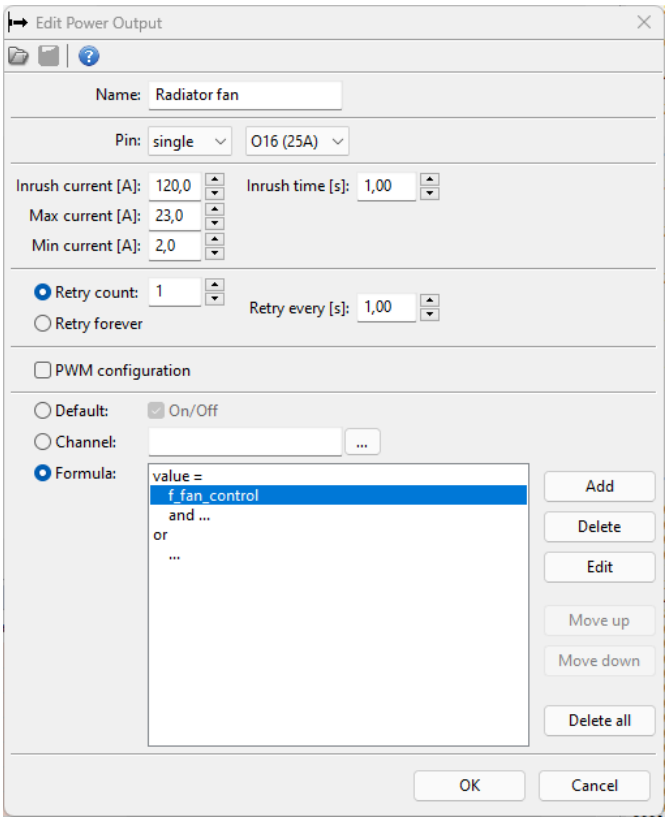
Project Tree

Name	Formula	Details
a_switch 1	switch: active low	A3, pu 10K
a_switch 2	switch: active low	A2, pu 10K
a_coolant_temp_sens	calibrated sensor	A1, pd 1M
f_fan_control	(a_switch 2) or (hysteresis/above(a_coolant_temp_sensor,95,90))	
fuel pump	a_switch 1	O13, min 2A, max 20A (inrush 120A 1s): 1x / 1s
Radiator fan	f_fan_control	O16, min 2A, max 23A (inrush 120A 1s): 1x / 1s









To the analog inputs are connected two switches grounded and a coolant temperature sensor. Both switches are grounded, so a 10k pull-up resistor to +5V is selected in the settings.

A +5V input means the switch is off, and a 0V input means the switch is on. The input to which the coolant temperature sensor is connected is set to 'Calibrated Sensor' mode, and the voltage from the sensor is converted to temperature.

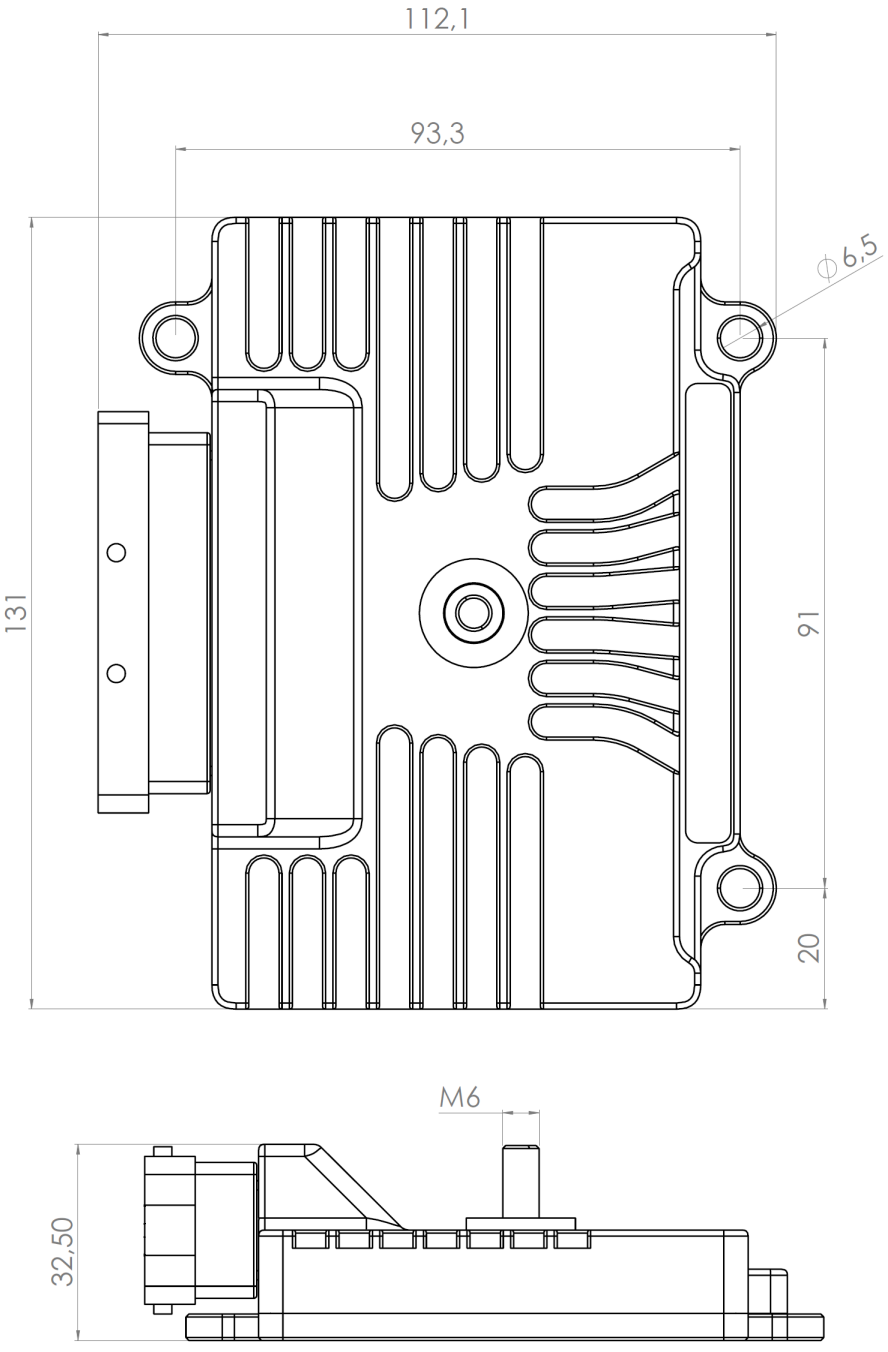
The fuel pump starts after pressing Switch 1. When the channel 'a_switch 1' has a status of 'true' (value 1), Output number 4 powers the fuel pump.

The radiator fan is controlled by the f_fan control function. The function achieves a status of 'true' (value 1) when one of two conditions is met:

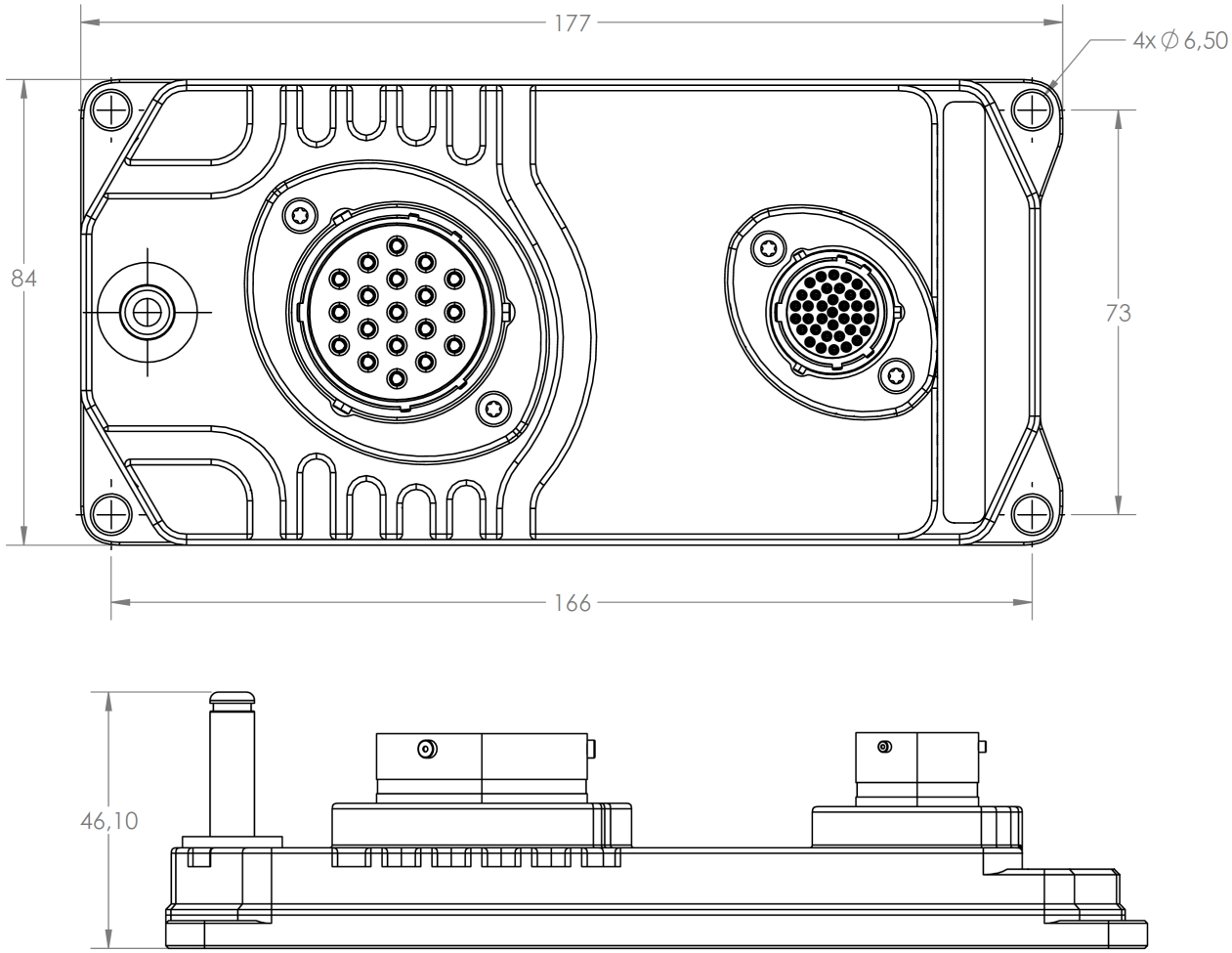
1. Input a_switch 2 has a status of 'true' (value 1).
2. The coolant temperature reaches 95°C. In this case, hysteresis is used. After the coolant cools to 90 degrees, the status of 'true' (value 1) is turned off.

18. Technical drawings

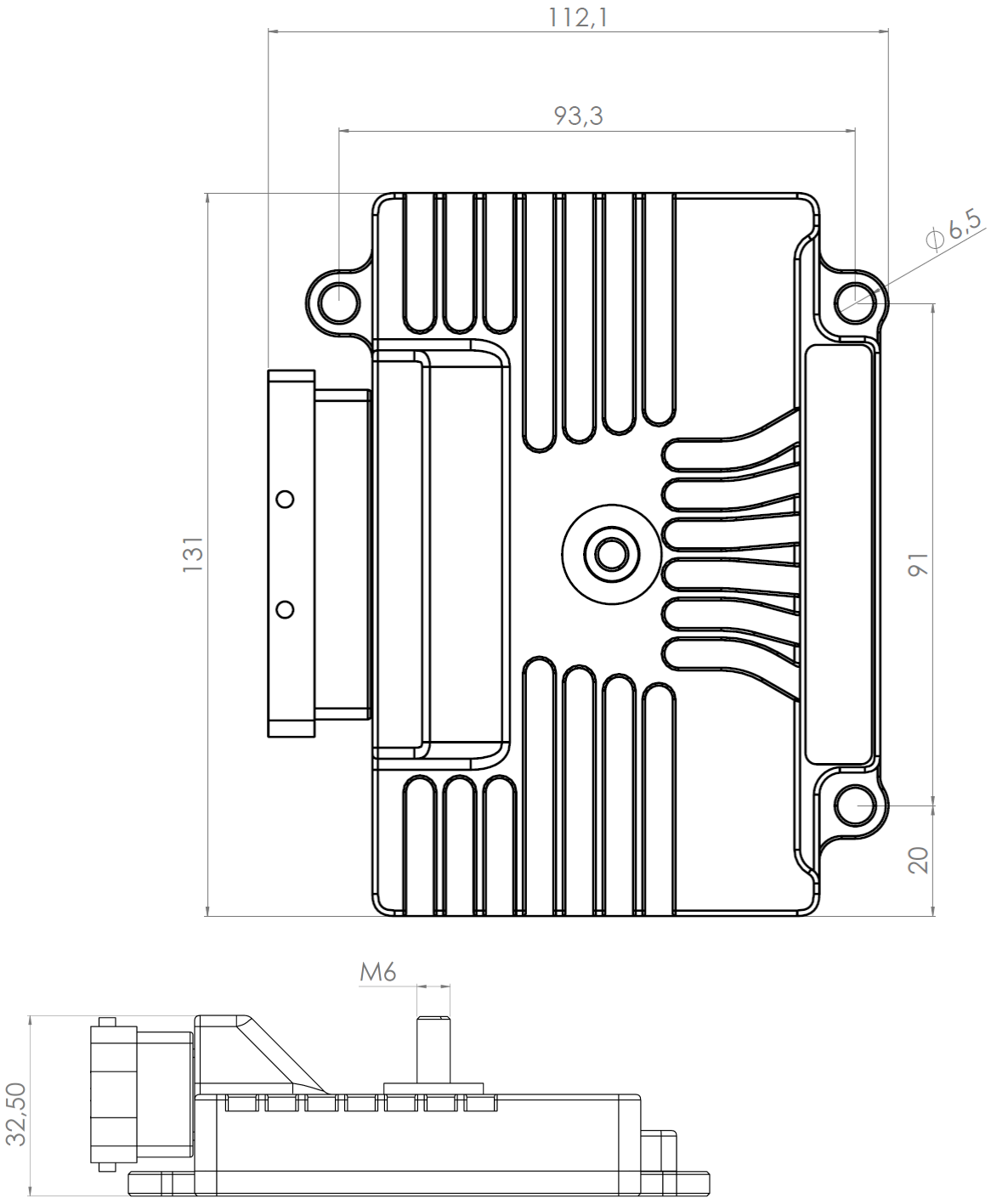
PMU 16 (all dimensions are in mm)



PMU 16 AS (all dimensions are in mm)



PMU 24 (all dimensions are in mm)



19. Document history

Revision	Date	Changes
1.00	2017.05.01	Initial release
1.01	2017.07.10	Added information about ground connection with USBtoCAN
1.02	2018.06.04	Clarified outputs specification
1.03	2020.09.02	Terminal current rating and derating
1.04	2022.09.05	Standard CAN Stream: added states of low side outputs in PMU AS Standard CAN Stream: frequencies have been corrected to match the actual device values Standard CAN Stream: PMU Status and status of each outputs explained
92.0	2023.09.27	Significant structural changes to the document along with the expansion of content
92.0.4	2024.03.13	Minor spelling correction Extension of LED status descriptions Addition of information about current thresholds for outputs Extension of the chapter on using multiple PMUs Addition of information about merging projects and changing target device
101.1	2024.09.09	New Ecumaster standard layout applied Added description of *Configuration, Autosaved channels* and *Standard channels assignment* panels Added description of *Lookup5* operation Added description of CAN Stream for PMU-24 Added How-to documents as appendices: "How-to Merge Projects" "How-to Configure 5x3MT Keyboard" "How-to Configure Autosaved Channels"
101.1.1	2024.10.08	Added information about CAN1 and CAN2 bus termination.
101.1.2	2025.02.05	Detailed information about CAN added to the *Specification* table.
101.1.3	2025.04.17	Corrections in the *Standard CAN Stream* section: changed the range, resolution, and frequency of output voltage channels

20. Appendix A - How-to Merge Projects

20.1. Description

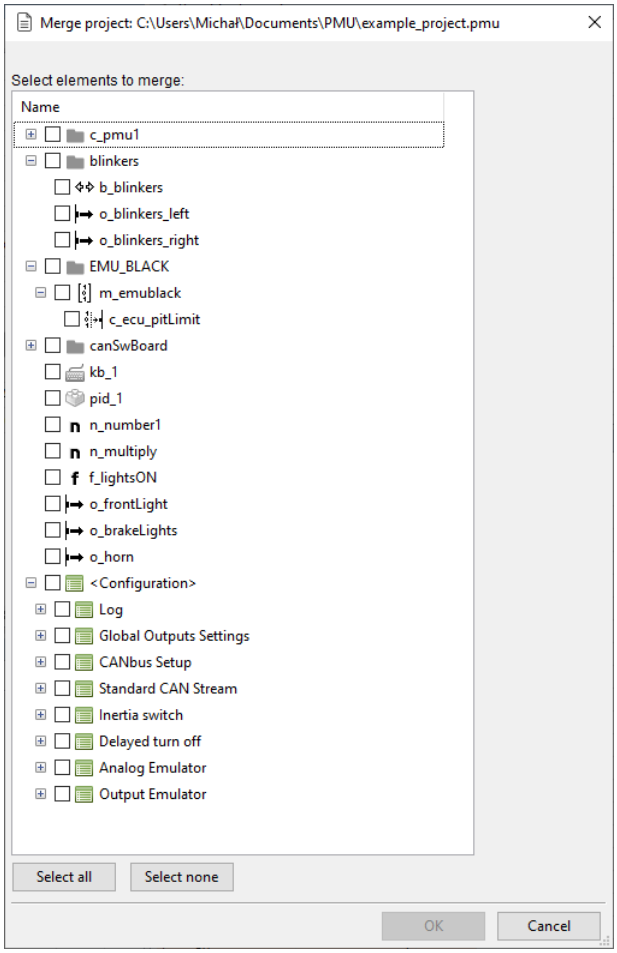
Merging projects enables you to pick specific elements from one project and integrate them into another.

In the document, the project you're currently working on will be named 'base'. The project you want to load the elements from, will be called 'incoming'.

All items from the Project Tree and the configurations can be merged between projects. This means all of the following:

- Analog inputs
- Digital inputs
- CANbus message objects
- CANbus inputs
- CANbus keyboards
- Enumerations
- PID controllers
- Timers
- Tables
- Switches
- Numbers
- Functions
- Wipers modules
- Blinkers modules
- CANbus exports
- Alarms
- Groups
- Configurations

To merge project select ***File/Merge project*** or press **Ctrl+M**. After selecting the incoming project, a dialog window will appear, displaying all elements from the incoming project available for merging:



After selecting individual elements or choosing all using the 'Select All' button, confirm your selection by clicking 'OK.' The selected elements will then be added to the Project Tree, and the chosen configuration settings will be applied.

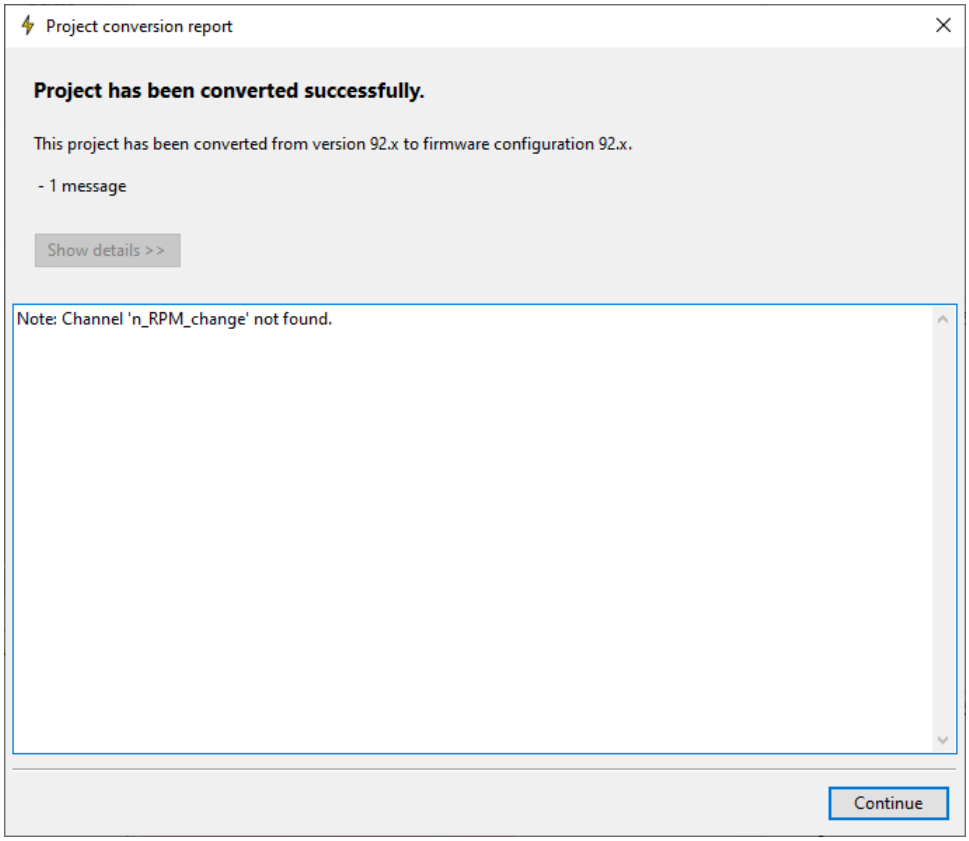
20.2. Resolving potential errors

Name conflict

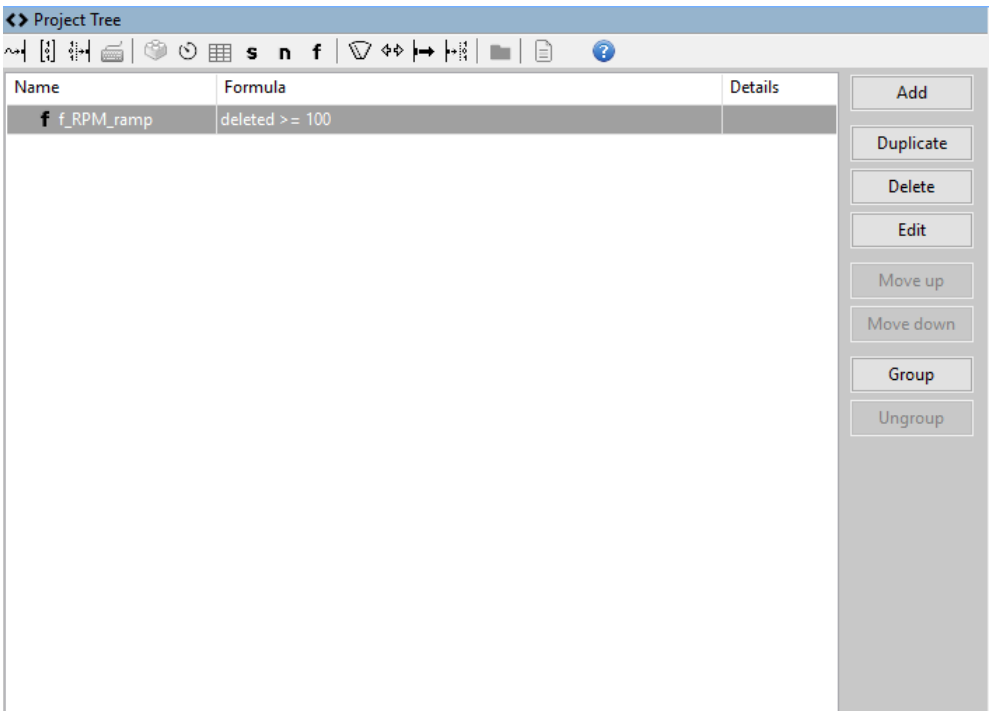
Name conflict is a situation where an incoming element has the same name as any element from the base project. In this case, the incoming element will be assigned a name with a '2' appended at the end.

Missing dependency

During the project merging process, you may encounter a missing dependency error. This occurs an incoming element (e.g., a calculation) relies on another item not present in the base project. If such an error arises, you will be notified through the following window.



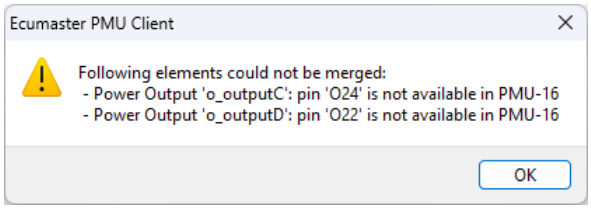
Despite this error, the affected element will still be loaded, and you have the option to manually resolve the missing dependency.



To prevent the **missing dependency error**, organize your projects using groups. Grouping related elements together makes sharing across projects easier. Choosing the entire group from the incoming project simplifies the process and reduces the chance of errors.

Output Mismatch Error (PMU-24 to PMU-16)

Project merging between PMU-24 and PMU-16 devices is supported in both directions. However, it's important to note that elements relying on outputs unavailable on the smaller PMU-16 device **will trigger a notification.**



The element will still be loaded. You'll then have the flexibility to manually assign it to one of the available outputs on the PMU-16.

20.3. Document history

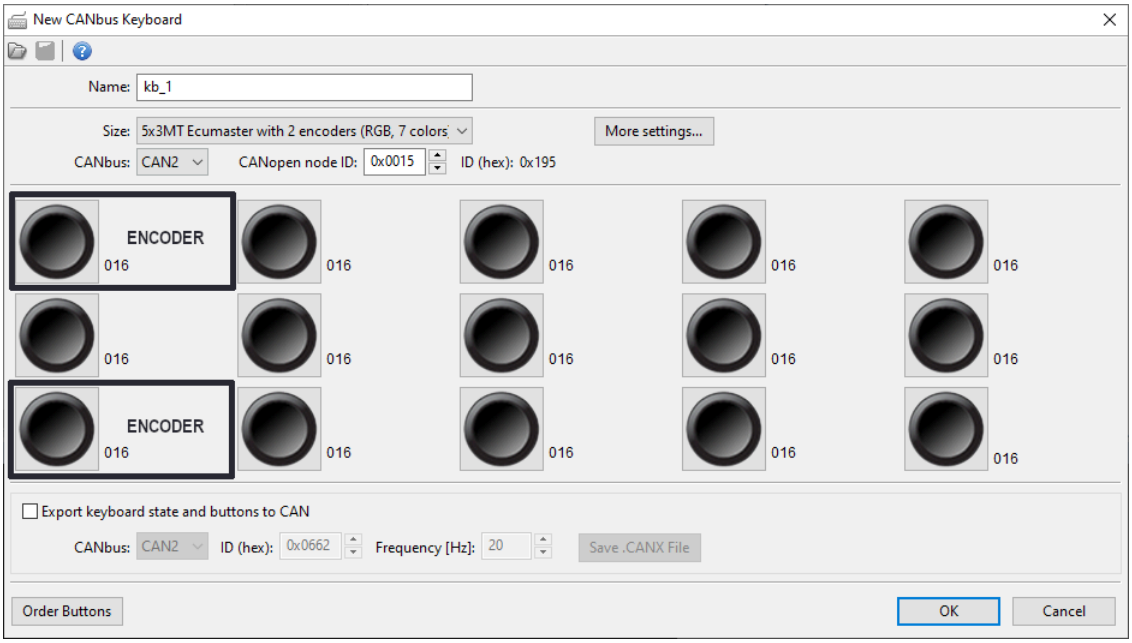
Version	Date	Changes
1.0	2024.02.15	Initial release

21. Appendix B - How-to Configure 5x3MT Keyboard

21.1. Description

This guide explains how to set up the 5x3MT Ecumaster keyboard, which is equipped with 13 ordinary buttons and 2 rotary encoders. This guide focuses mainly on the configuration options for the encoders. For more general information on CAN bus keyboard support, see the PMU user manual.

To configure a keyboard, click Add in the Project Tree and then select CANbus Keyboard from the list or click the keyboard icon in the toolbar. Select the 5x3MT from the *Size* menu. The window will look as follows, with encoder positions marked with a frame:



Knowing the three types of encoder operations is essential for getting the best configuration:

1. ***encoder***

   Rotating the encoder adjusts the associated channel's value. Unlike a rotary switch, you have the flexibility to configure the encoder's range of values.

   - *First state* - the lowest encoder state
   - *Last state* - the highest encoder state
   - *Default state* - after startup, the encoder will adopt its default state

2. ***changer***

   Operation in *changer* mode is similar to *encoder* mode but its main use is to change pages in the ADU. Turning the encoder clockwise moves to the next page, while turning counter-clockwise goes back to the previous page.

   In changer mode, the LEDs illuminate in a fan-like pattern, offering a visual indication of the operating mode.

   - *First state* - the number of the first page
   - *Last state* - the number of pages in the ADU
   - *Default state* - after startup, the page you want to start on
   - *Wrap pages* - decide if the encoder should jump from the last page to the first and vice versa

   For more information about the operation of the *changer* mode, see Example *(on page 151)*.

3. ***parameter controller*** and ***parameter selector***

You can use the keyboard encoder to control multiple settings. This is possible by setting it to *parameter controller* mode. For each setting you want to manage, select a button and set its mode to *parameter selector* mode. These selector buttons function like radio buttons, allowing you to choose one active parameter at a time that is modified by the knob.

Each *parameter selector* comes with its own range and state, updating the *parameter controller* when pressed.

a. If there are no button presses defined as a *parameter selector*, *parameter controller* remains neutral and rotation has no effect.
b. Pressing the button defined as a *parameter selector*, conveys the parameter information to the encoder. Encoder can change the state of the *parameter selector*.
c. After pressing the currently selected *parameter selector* button again or pressing the *parameter controller*, the encoder returns to the neutral behaviour. (Pressing any other button (not defined as a *parameter selector*) will not cause the encoder to lose control of the *parameter selector*.)

Surrounding the encoder are 16 LEDs that visually represent the current state of the encoder. Default starting LED is at 9 o'clock, numbered clockwise. To change, go to *More settings*, adjust *Start offset for encoder LEDs'* from 0 to 15, with 0 as the 9 o'clock position.

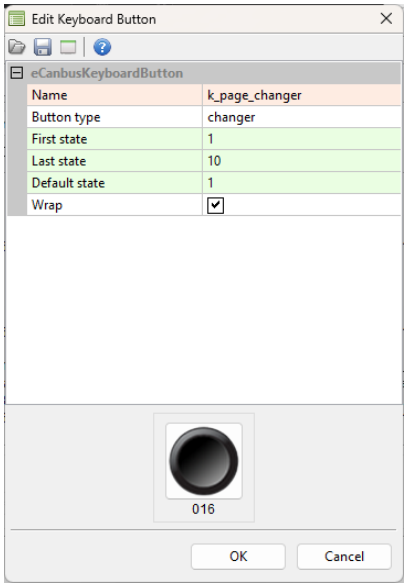
21.2. Example

We're configuring the 5x3MT to quickly navigate pages in ADU and finely adjust four parameters with the two rotary encoders. In this example, we'll use the top encoder for pages and the bottom one to control parameters.

Changing pages in the ADU via the PMU

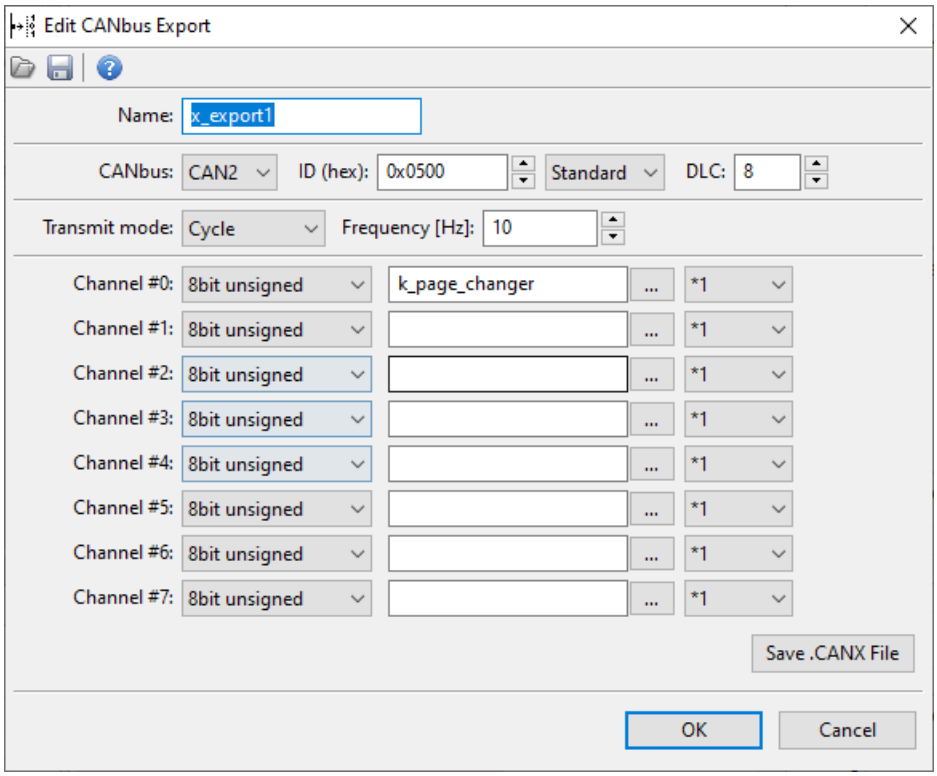
In the keyboard settings, select the top-left box for the rotary encoder. Set it as a *changer*. Determine the following parameters:

- *First state* - set to 1, to avoid confusion in page counting
- *Last state* - the number of pages in the ADU (remembering that it is not updated automatically - when you add or remove a page in ADU, it must be changed)
- *Default state* - after startup, the page you want to start on
- *Wrap pages* - decide if the encoder should jump from the last page to the first and vice versa

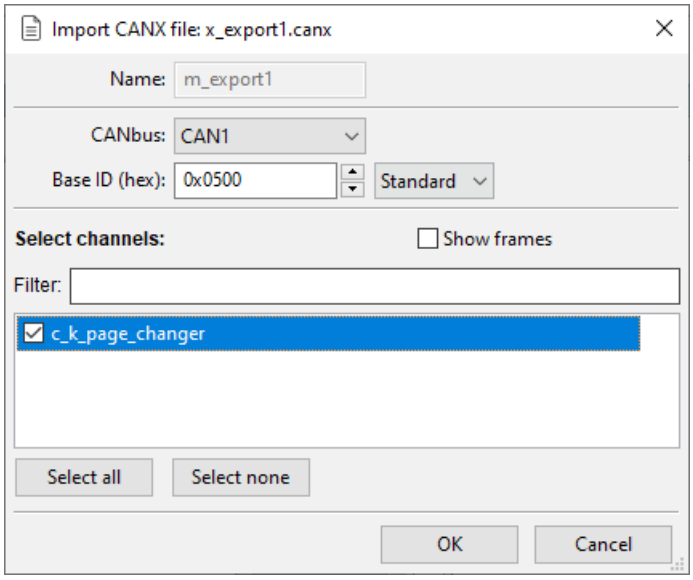


To enable page changes in the ADU, both the ADU and the PMU must be connected to the same CAN bus. Start in PMU, by adding CAN export in the Project tree. Remember to set correct CAN bus and an unoccupied ID. Select the channel that corresponds to the keyboard encoder set up in the previous step (*k_page_changer* in our example).

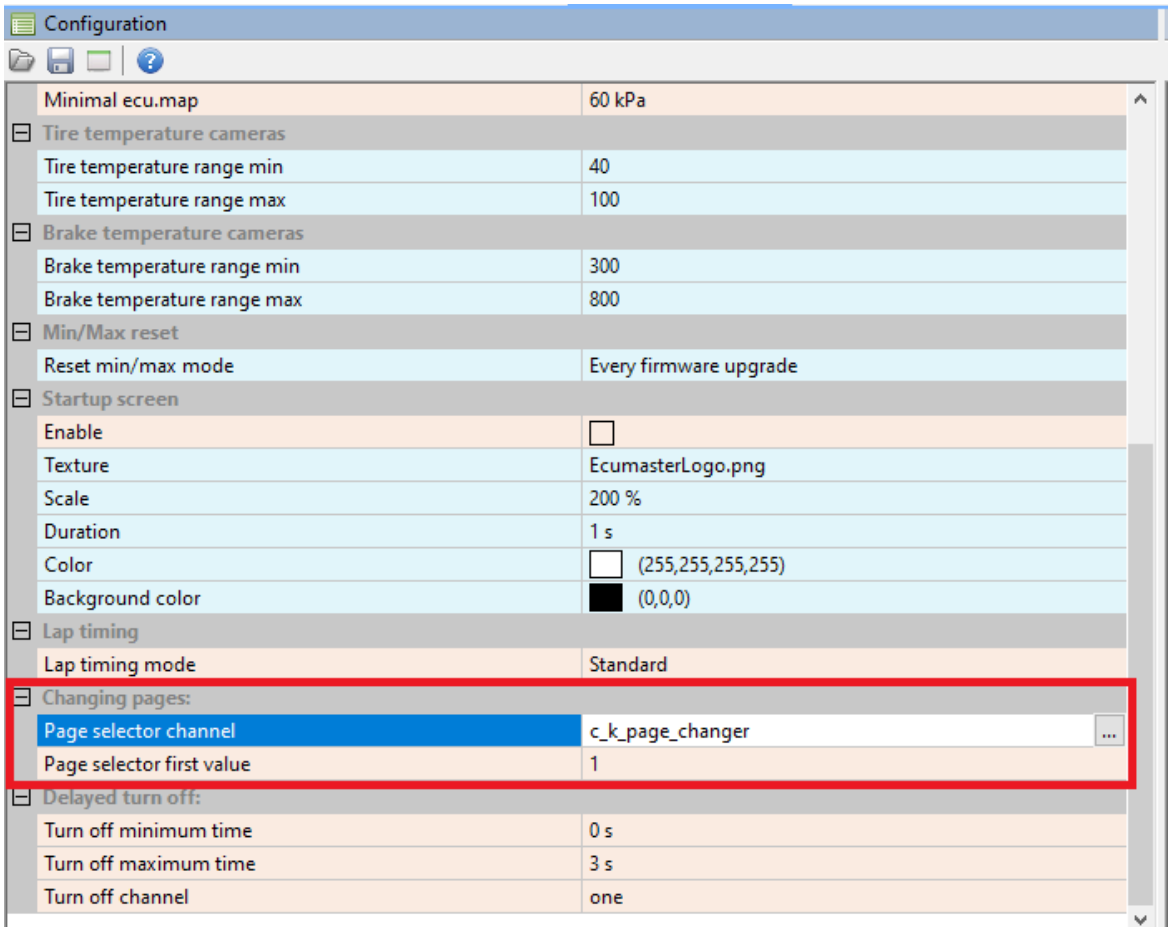
To streamline the import in the ADU, use *Save .CANX File* button.



In the next step, import the .CANX file created in the previous step into the ADU project.

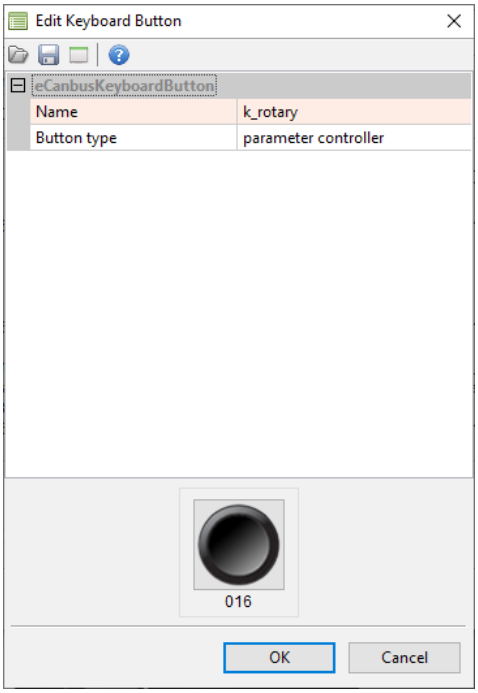


Then assign this imported channel in the *Configuration* panel to the *Page selector channel* and set *Page selector first value* as in the PMU configuration, equal to 1:

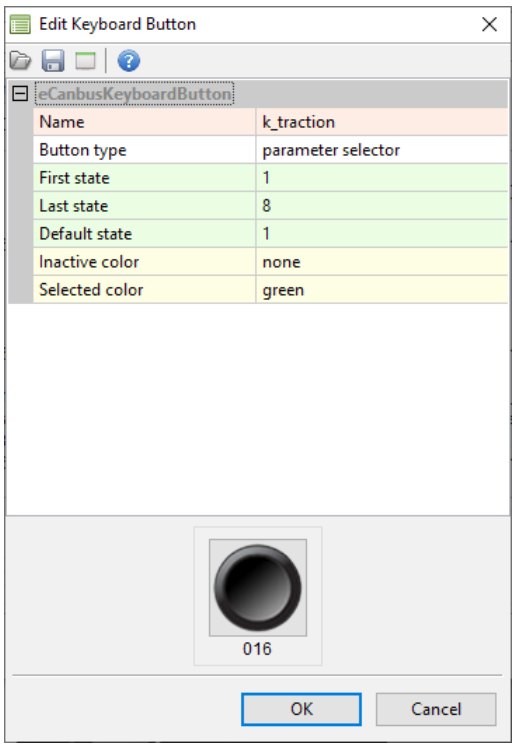


Controlling multiple parameters

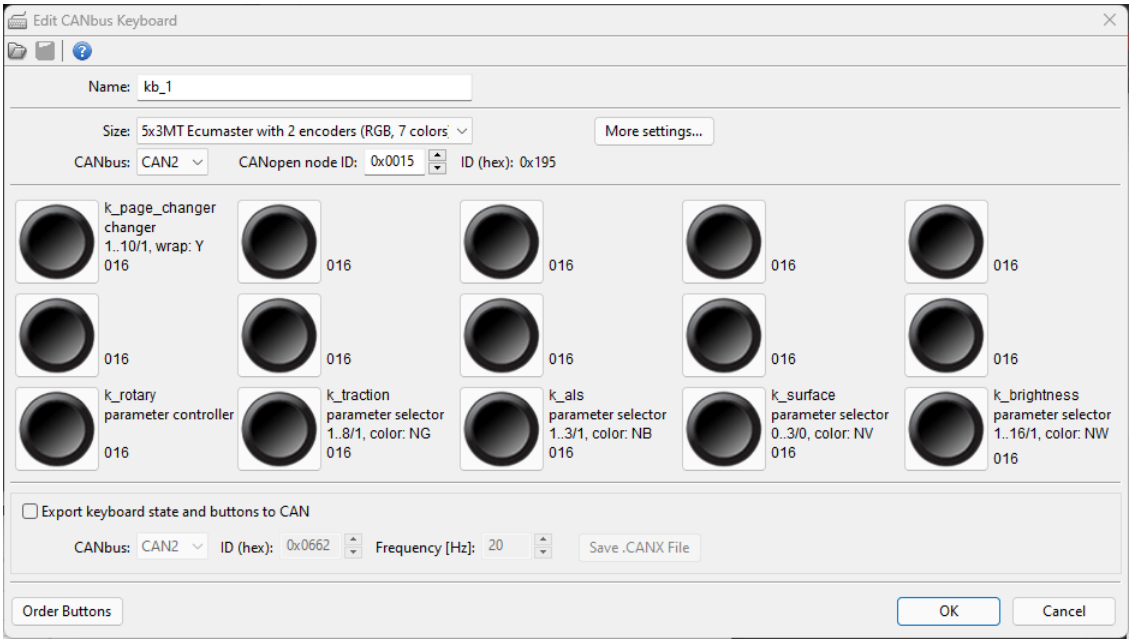
Back to the PMU software, we aim to use the other encoder for controlling four parameters. Begin by selecting the bottom-left box and designate it as a *parameter controller*.



Now proceed to configure each parameter. For each of the four, choose one button, change its type to *parameter selector* and set the first, last, and default states to adjust their value range.



The final setup of our example keyboard looks as follows:



When the driver wishes to adjust a setting, they can press the corresponding button. The rotary encoder will activate, enabling them to change the parameter by turning it.

21.3. Document history

Version	Date	Changes
1.0	2024.04.05	Initial release

22. Appendix C - How-to Configure Autosaved Channels

22.1. Description

The Autosave Channels feature allows users to store values for up to 20 channels, which are loaded when the device starts. These values are saved automatically.

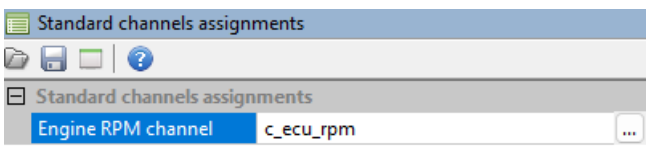
Below is a list of elements from Project Tree that can save their state:

1. Keyboards buttons
2. Switches
3. Logical functions
4. Numbers

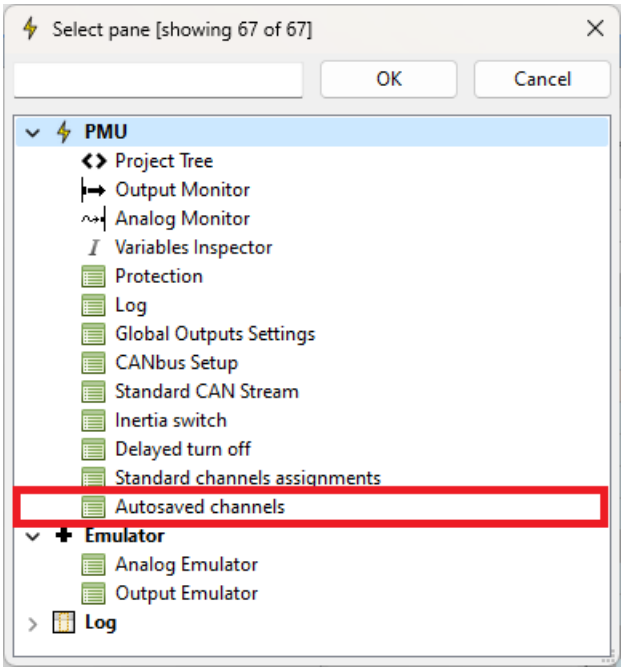
The Autosave Channels prove useful in various scenarios, such as preserving the state of any encoder. By doing so, users can easily recover the encoder's state after the device is turned on.

Auto-save operates based on three independent paths. These paths are pre-programmed and not configurable by the user:

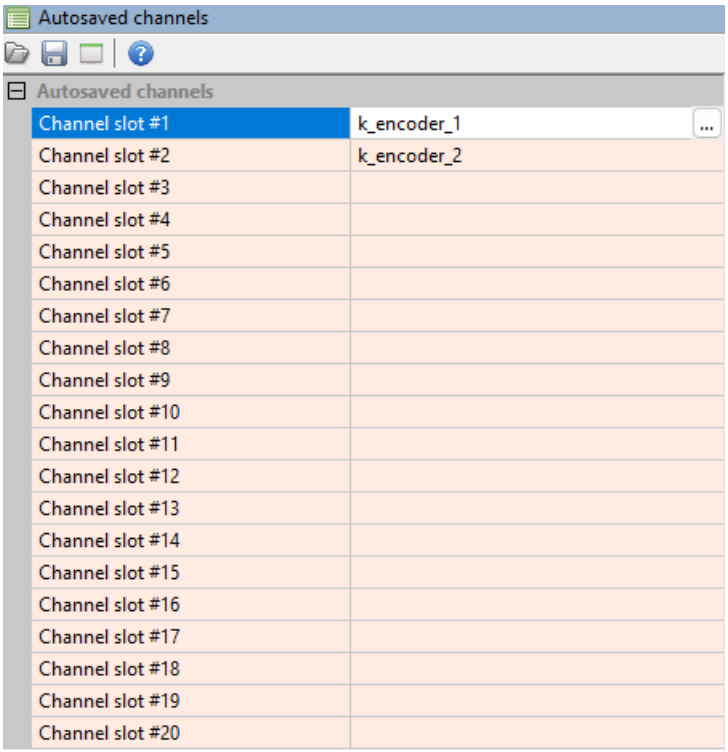
- **Ignition switch** – the save of all channels occurs when the ignition switch +12V is disconnected, but the constant battery +12V remains powered (requires separate wiring for battery and ignition switch). This method provides most reliable results.
- **On change** – after detecting a change in any of the autosaved channels, a save occurs, followed by a 2-minute period without further saving (cooldown). If none of the selected channels change their values, no save occurs, and the system awaits a channel change. The 2-minute intervals (cooldowns) without saving are implemented to prevent memory wear as it is not designed to handle rapid changes.
- **RPM drop** – the save of all channels occurs when the engine RPM drops to zero, provided the engine speed was greater than zero for at least 10 seconds. The RPM channel has to be defined in ***Standard channels assignments*** panel in ***Engine RPM channel***.



To open the *Autosave Channels* panel, press F9 and select it from the list or locate it in the *Configuration* panel.

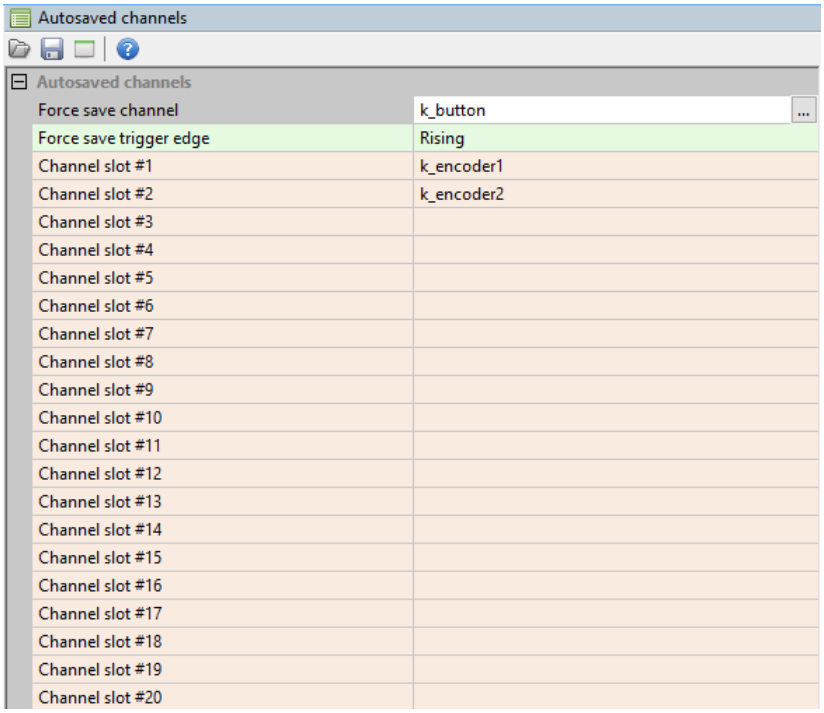


In the panel, select the channels you want to be autosaved.



Upon turning on the device, the autosaved channel values will be loaded into their respective channels. Subsequently, the evaluation proceeds as usual, meaning the loaded value will be replaced with any new data.

New in version 103.0



A new method for saving **Autosaved channels** has been introduced: the **"Force save channel"** feature. This adds to the existing save methods (Ignition switch, On change, and RPM drop) by allowing an immediate save when triggered.

When triggered by the rising or falling edge of the **Force save channel** (as configured by the user), a save occurs even if the standard 2-minute cooldown has not elapsed.
However, triggering an instant save starts a new 2-minute cooldown, during which further triggers will be ignored.

22.2. Document history

Version	Date	Changes
1.0	2024.04.09	Initial release
1.1	2024.04.11	Improved screenshot of Autosaved Channels panel
1.2	2025.03.18	Added "Force save channel" description