



USER MANUAL

ADU Manual

Document version 104.1.5

Software version: 104.1 or later

Published on: 05 February 2025



Contents

| | |
|--|-----------|
| 1. Ecumaster ADU..... | 7 |
| 1.1. Characteristics..... | 8 |
| 1.2. Technical drawings..... | 10 |
| 1.3. Device description..... | 15 |
| 1.4. Connector..... | 16 |
| 1.5. Description of the ADU AS socket | 18 |
| 1.6. Description of the ADU socket..... | 19 |
| 2. Installation..... | 21 |
| 2.1. CAN bus..... | 22 |
| 2.2. Connecting to ECU..... | 24 |
| 2.3. Connecting via a CAN bus..... | 24 |
| 2.4. Connecting via an RS232 serial bus..... | 25 |
| 2.5. OBD 2..... | 27 |
| 2.6. GPS module..... | 28 |
| 2.7. Ecumaster PMU..... | 32 |
| 2.8. Digital Inputs..... | 33 |
| 2.9. Analog Inputs..... | 34 |
| 2.10. USB Flash drive (pendrive)..... | 35 |
| 2.11. Low side outputs..... | 36 |
| 3. ADU Client software for Windows..... | 36 |
| 3.1. First connection with the device..... | 37 |
| 3.2. Firmware update..... | 40 |
| 3.3. Loading a project..... | 40 |
| 3.4. Appearance of the application..... | 43 |
| 4. Status field..... | 55 |

| | |
|---|-----------|
| 5. Switching the CAN adapter between ADU, PMU and Light Client programs..... | 57 |
| 6. Pages..... | 58 |
| 6.1. Default settings pg_page1..... | 58 |
| 6.2. Creating a page..... | 61 |
| 6.3. Page elements..... | 63 |
| 6.4. Adding elements to pages..... | 65 |
| 6.5. Grouping objects..... | 75 |
| 6.6. Toggling between pages..... | 78 |
| 6.7. Startup screen..... | 81 |
| 7. Graphic objects..... | 82 |
| 7.1. Gauge..... | 82 |
| 7.2. Classic gauge..... | 84 |
| 7.3. Bar graph..... | 86 |
| 7.4. Simple indicator..... | 89 |
| 7.5. Text..... | 90 |
| 7.6. Time..... | 92 |
| 7.7. Image..... | 93 |
| 7.8. RPM Bar..... | 94 |
| 7.9. Gear indicator..... | 95 |
| 7.10. G-Force..... | 96 |
| 7.11. Predictive time graph..... | 96 |
| 7.12. Tire temperature graph..... | 97 |
| 7.13. Temperature color scale..... | 98 |
| 7.14. Session results..... | 99 |
| 7.15. Track record table..... | 99 |
| 7.16. Rectangle..... | 100 |
| 7.17. Line..... | 101 |

| | |
|--|------------|
| 7.18. Circle..... | 102 |
| 7.19. Grid..... | 103 |
| 7.20. Textures..... | 104 |
| 8. Inputs..... | 107 |
| 8.1. Analog Inputs..... | 107 |
| 8.2. Digital Inputs..... | 110 |
| 9. Outputs..... | 114 |
| 9.1. <i>Low side</i> outputs..... | 114 |
| 9.2. Analog output..... | 114 |
| 10. Working with CAN buses in ADU..... | 115 |
| 10.1. Using pre-defined streams from .CANX and .DBC files..... | 115 |
| 10.2. Built-in support for PMU..... | 117 |
| 10.3. Own CAN streams – CANbus Message Object..... | 120 |
| 10.4. Own CAN streams – CANbus Input..... | 123 |
| 10.5. Own CAN streams – saving to a .CANX file..... | 126 |
| 10.6. Sending frames using the CAN bus (CANbus Export) | 126 |
| 10.7. Reserved CAN ID | 130 |
| 11. CAN bus keypad support..... | 130 |
| 12. Enumeration..... | 135 |
| 13. Processing information in the ADU..... | 142 |
| 13.1. Timers – counting time..... | 142 |
| 13.2. Tables – 2D / 3D lookup tables..... | 144 |
| 13.3. Switches virtual switches, counters..... | 146 |
| 13.4. Numbers – mathematical channels..... | 147 |
| 13.5. Functions – logical functions..... | 155 |
| 14. Alarms..... | 160 |
| 15. Logging channels..... | 163 |

| | |
|--|------------|
| 16. Logging to a USB storage device..... | 165 |
| 17. Permanent meters..... | 167 |
| 17.1. Resetting / changing meter status..... | 168 |
| 18. Min / max values for ECU channels..... | 169 |
| 19. Panels..... | 170 |
| 19.1. Buttons..... | 170 |
| 19.2. Shift light..... | 171 |
| 19.3. User lights..... | 174 |
| 19.4. User tracks..... | 175 |
| 19.5. User track defined by the button..... | 176 |
| 19.6. Fuel level filter..... | 178 |
| 19.7. OBD 2 | 179 |
| 19.8. Outputs..... | 181 |
| 19.9. Autobrightness..... | 181 |
| 19.10. Virtual fuel tank | 181 |
| 19.11. Configuration..... | 183 |
| 19.11.1. Delayed turning off..... | 186 |
| 19.12. Protection..... | 187 |
| 19.13. Log..... | 188 |
| 19.14. CANbus / Serial Setup..... | 188 |
| 19.15. Autosaved channels..... | 190 |
| 19.16. Channel Simulator..... | 190 |
| 19.17. Channel report..... | 190 |
| 20. Timing..... | 191 |
| 20.1. Configuration of time measurement with a beacon..... | 192 |
| 20.2. Configuration of time measurement with a GPS..... | 192 |
| 21. Data analysis..... | 193 |

| | |
|--|------------|
| 22. Appendix A - List of built-in tracks..... | 202 |
| 23. Appendix B - How-to Configure the Hill Climb..... | 204 |
| 23.1. Description..... | 204 |
| 23.2. Document history..... | 208 |
| 24. Appendix C - How to Configure PMU CAN Stream..... | 209 |
| 24.1. Description..... | 209 |
| 24.2. Document history..... | 212 |
| 25. Appendix D - How-to Merge Projects..... | 213 |
| 25.1. Description..... | 213 |
| 25.2. Resolving potential errors..... | 214 |
| 25.3. Document history..... | 216 |
| 26. Appendix E - How-to Configure 5x3MT Encoders..... | 217 |
| 26.1. Description..... | 217 |
| 26.2. Example..... | 219 |
| 26.3. Document history..... | 222 |
| 27. Appendix F - How-to Use Virtual Fuel Tank..... | 223 |
| 27.1. Description..... | 223 |
| 27.2. Example..... | 227 |
| 27.3. Document history..... | 233 |
| 28. Appendix G - How-to Configure Autosaved Channels..... | 233 |
| 28.1. Description..... | 233 |
| 28.2. Document history..... | 235 |
| 29. Document history..... | 236 |

1. Ecumaster ADU

ECUMASTER ADU (Advanced Display Unit) is an universal motorsport display. Unlike other similar devices on the market, this ADU delivers very high adaptability of user displayed information and a flexible input processing system. Featuring two CAN buses, the display is able to easily communicate with other devices (e.g. ECU, GPS, ABS, etc.), and it allows external sensors to be connected to eight analog inputs (e.g. pressure sensors, temperature sensors) and eight digital inputs (e.g. frequency sensors, beacons etc.).

Furthermore, the device can transmit information about the condition of the vehicle via a set of 15 RGB LEDs, fully controlled by the driver (e.g. information about low oil level, progressive shift light, etc.).

To improve user comfort in various lighting conditions, a high-quality LCD display has been used with a brightness of 600 cd/m² (for the 5" display) and 1000 cd/m² (for the 7" display). To reduce reflections and glare, the display is coated with an anti-reflective layer. To adapt to changing light conditions, a light intensity sensor is mounted on the front of the housing to automatically adjust the brightness of the display to the surroundings.

The ADU can also be used as a central data logger. All information collected by the display can be saved on a USB storage device (Flashdrive, Pendrive) with speeds of up to 500 Hz/channel. The dates and times of the files are provided by a built-in real-time clock. Data saved to the USB storage device can then be analysed using ADU Client software or Ecumaster DATA MASTER software.

The device also allows track lap times to be recorded (using an external GPS), it features predictive timing and facilitates subsequent analysis of times across different sectors using data analysis software.



1.1. Characteristics

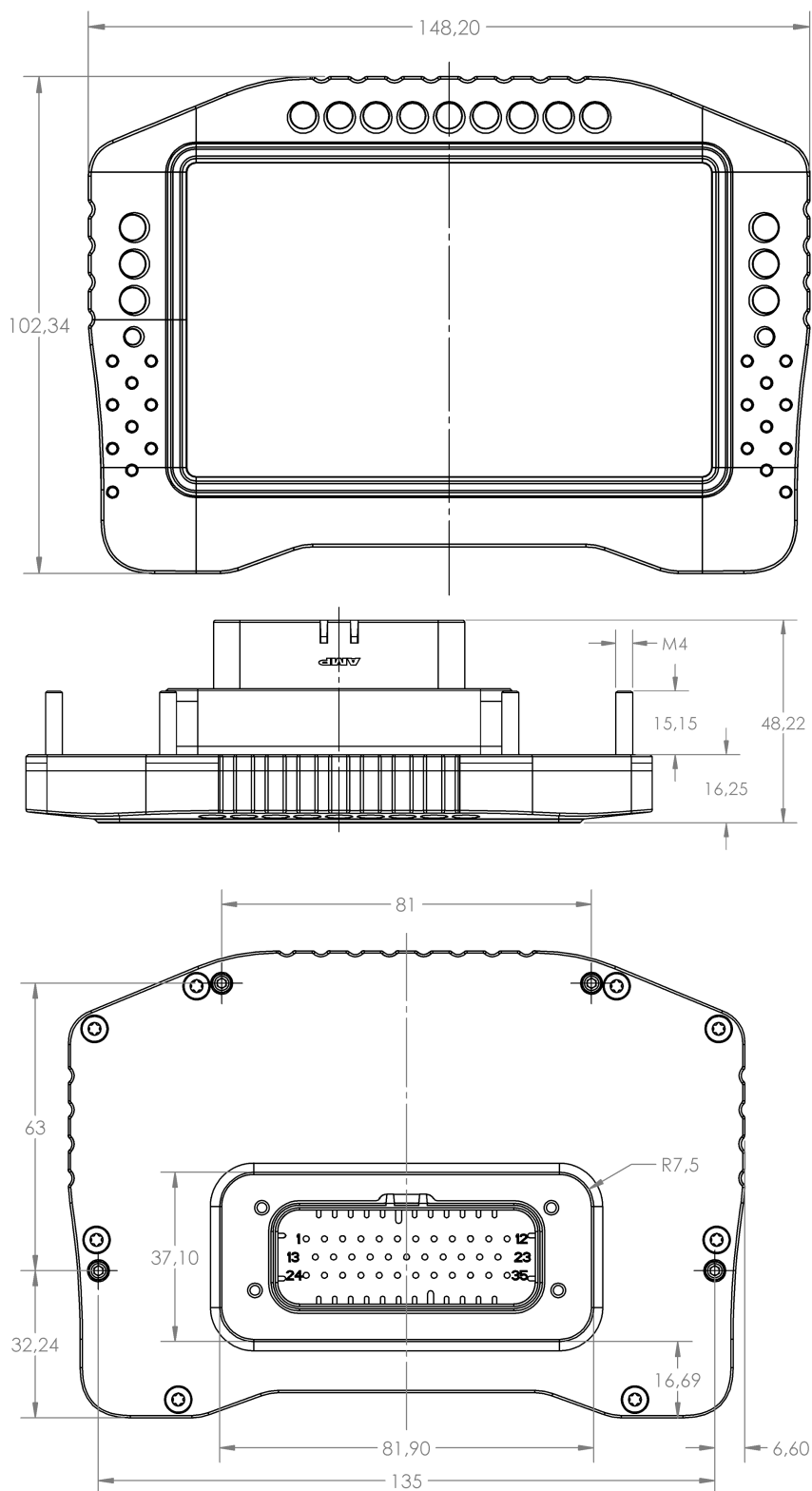
| General | |
|-----------------------------|---|
| Working temperature range | ACEQ100 (-40 – 85°C) |
| CPU | 32 bit, <i>automotive</i> , 90 MIPS |
| Reverse polarity protection | Yes, internal |
| Working voltage range | 6– 22 V, robust supply protection compliant with ISO 7637 |
| Housing | Anodized aluminium, CNC-manufactured |
| Tightness class | IP60 for ADU 5/7 Rev.1 (manufactured 2018-2020) IP65 for ADU 5/7 Rev.2 (manufactured from 2021-) IP60 for ADU7" AS (manufactured from 2018-) |
| Connector | 1 x 35 AMP <i>automotive</i> |
| Communication with PC | Via CAN bus. Required USB - CAN interface: <ul style="list-style-type: none"> • ECUMASTER USBtoCAN • Peak PCAN-USB or Peak PCAN-LAN, • Kvaser |
| Display type | TFT 800x600 |
| Display brightness | 5" - 600 cd/m ² , 7" - 1000 cd/m ² |
| Inputs / Outputs | |
| Analog inputs | 8 inputs, 10 bit resolution, 0-5 V, (programmed) switched pull-up / pull-down 10K resistor |
| Digital inputs | 8 digital inputs, programmed input sensitivity (VR, Hall), switched internal pull-up 4K7 resistors used for engine speed sensors, Flex Fuel, wheel speeds, turbocharger rotor speed |
| Outputs | 2 low side type outputs (switch to ground), up to 2 A current limit |
| +5V output | 500 mA, monitored |
| Communication | |
| CAN Interface | 2 x CAN2.0 A/B, 125, 250, 500 (default for CAN2), 1000 kbps (default for CAN1) |
| CAN termination | CAN1 - none, CAN2 - software-selectable |
| CAN streams | User-defined |

| | |
|---------------------------|--|
| Serial connector | RS232 Rx/Tx Protocols: AiM, Ecumaster, Hondata Kpro, AEM, GEMS, Athena GET, AUTRONIC, MoTec M4 DATA SET 5 |
| USB | For logging to external USB storage device |
| Other | |
| Light Emitting Diodes | 15 very bright RGB LEDs |
| Accelerometer / Gyroscope | 3D accelerometer + 3D gyroscope for vehicle dynamics analysis |
| Real-time clock | Yes, supported by a built-in battery |
| Light sensor | Yes, for automatic correction of brightness |
| Temperature sensor | Yes, for monitoring device temperature |
| Keyboard support | Yes, Ecumaster, Grayhill, MoTeC / RaceGrade, Emtron |

1.2. Technical drawings

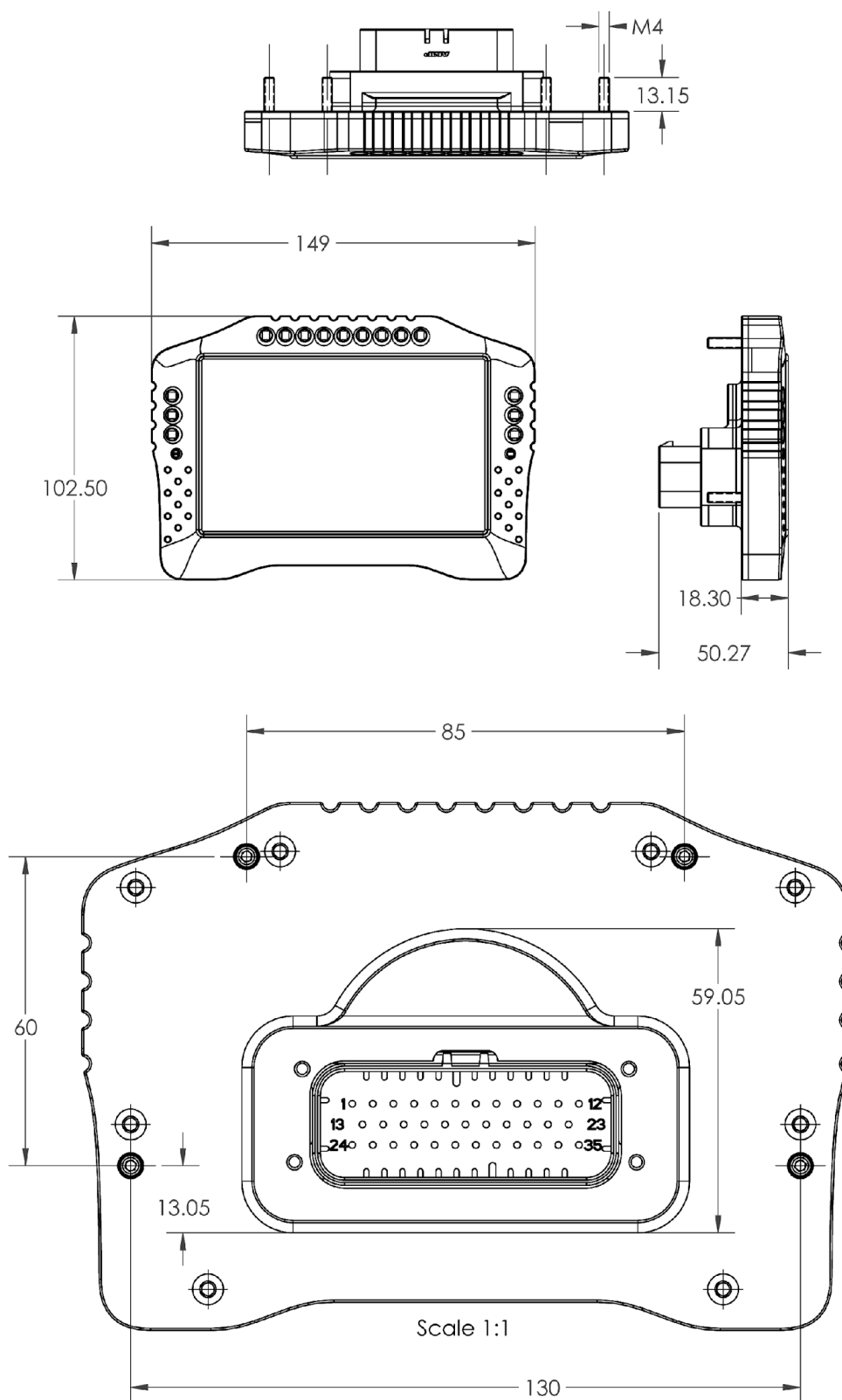
ADU 5 Rev.2

(all dimensions in mm)



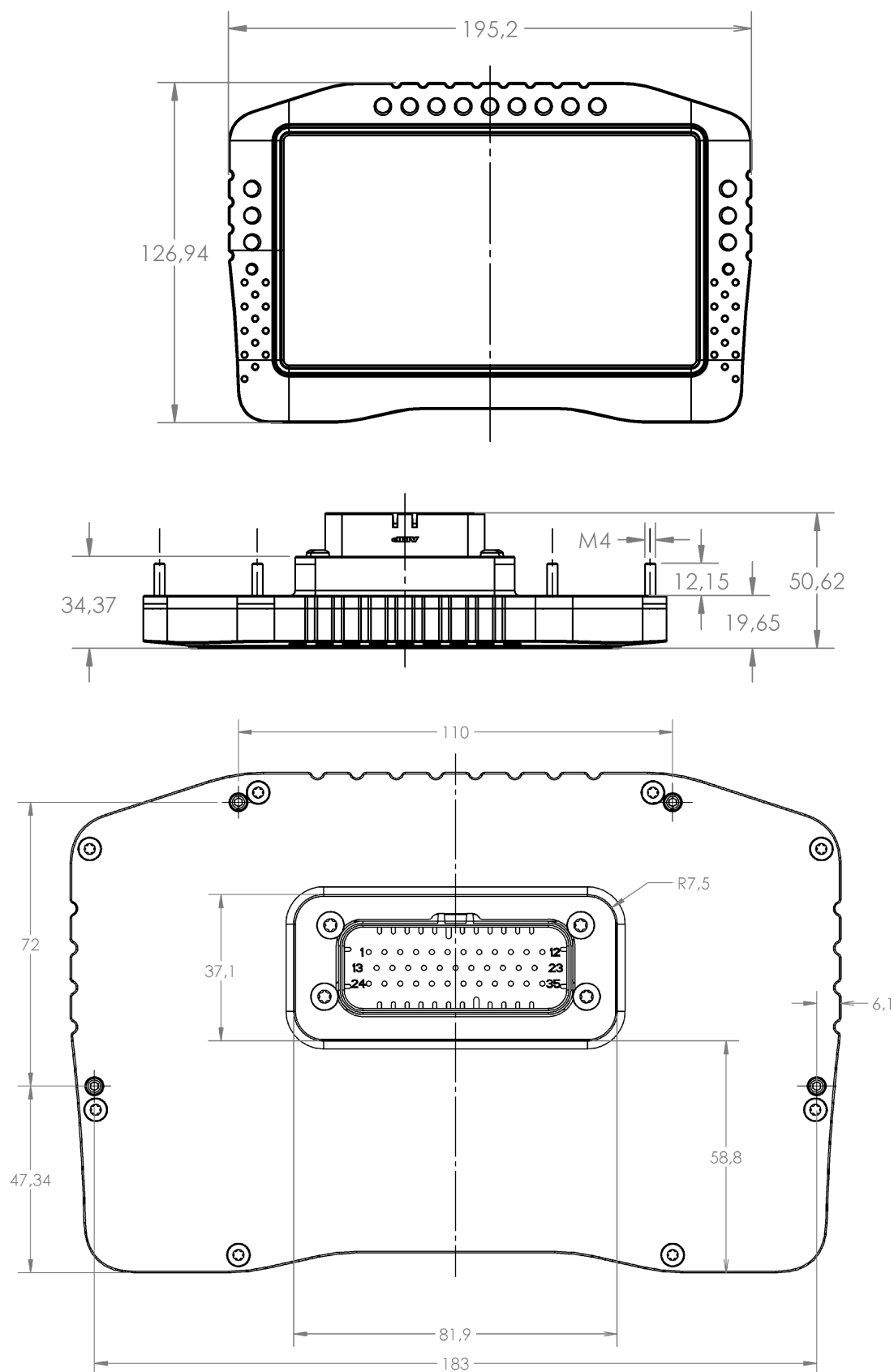
ADU 5 Rev.1 (manufactured 2018-2020, currently out of production)

(all dimensions in mm)



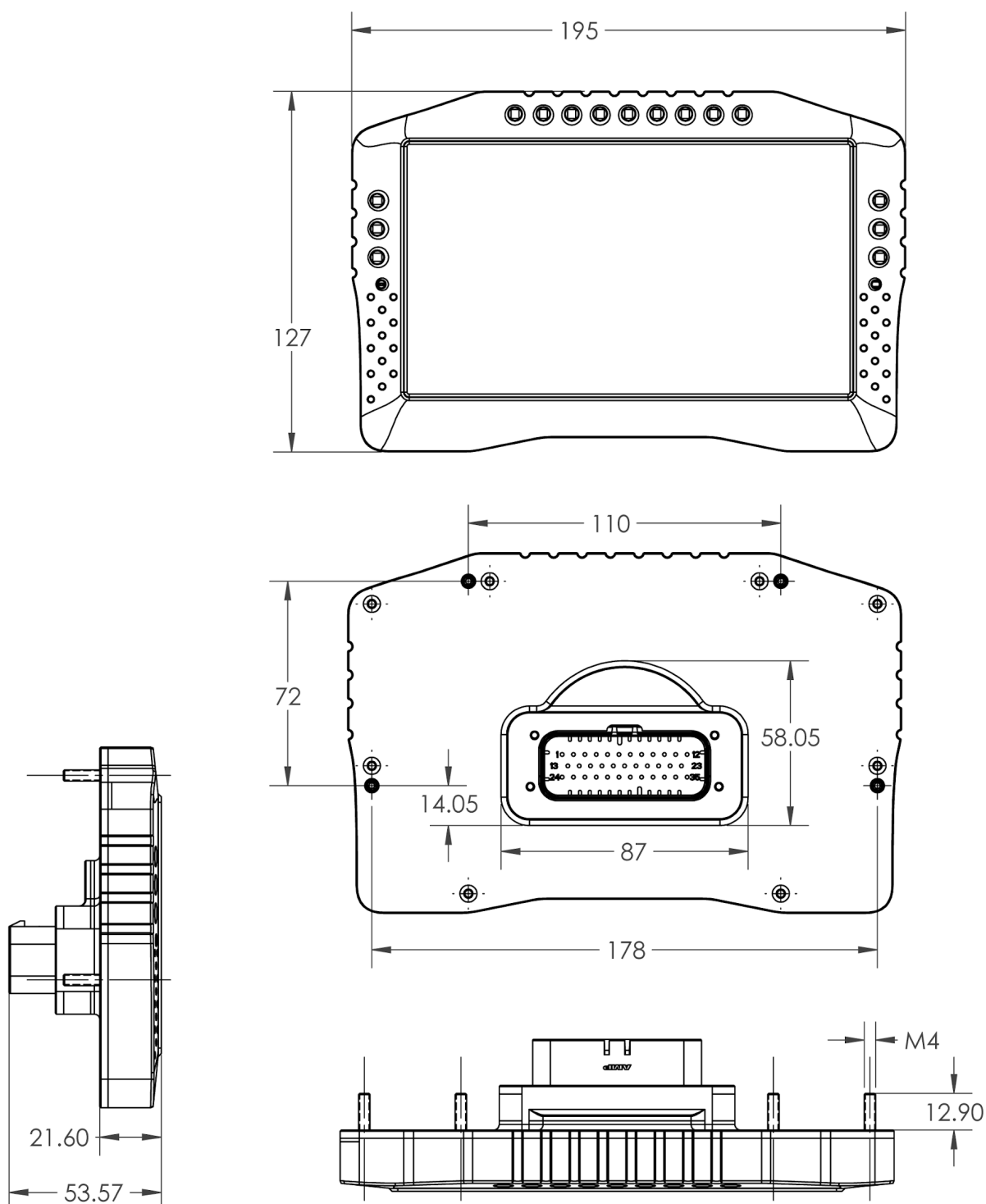
ADU 7 Rev.2

(all dimensions in mm)



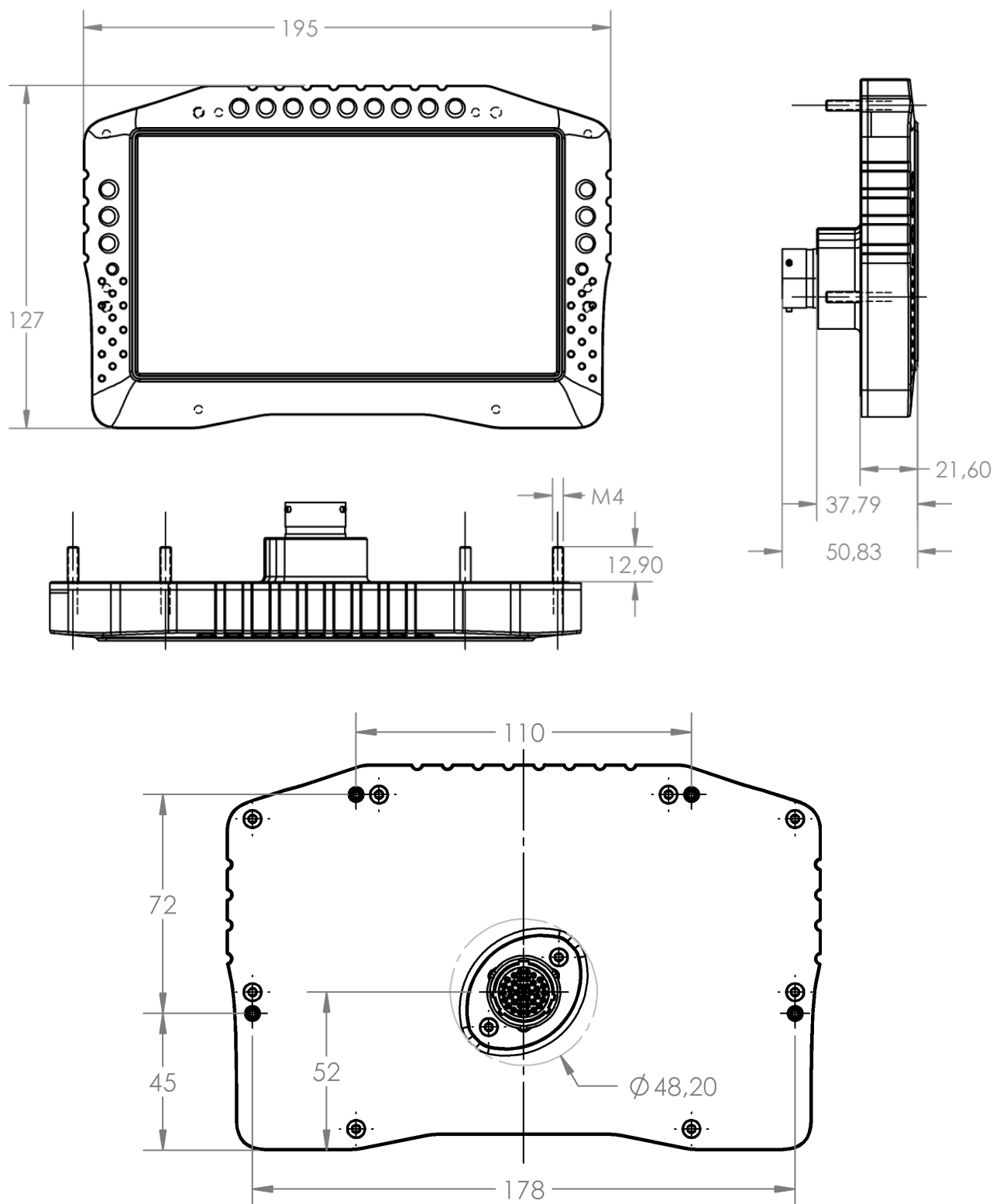
ADU 7 Rev.1 (manufactured 2018-2020, currently out of production)

(all dimensions in mm)



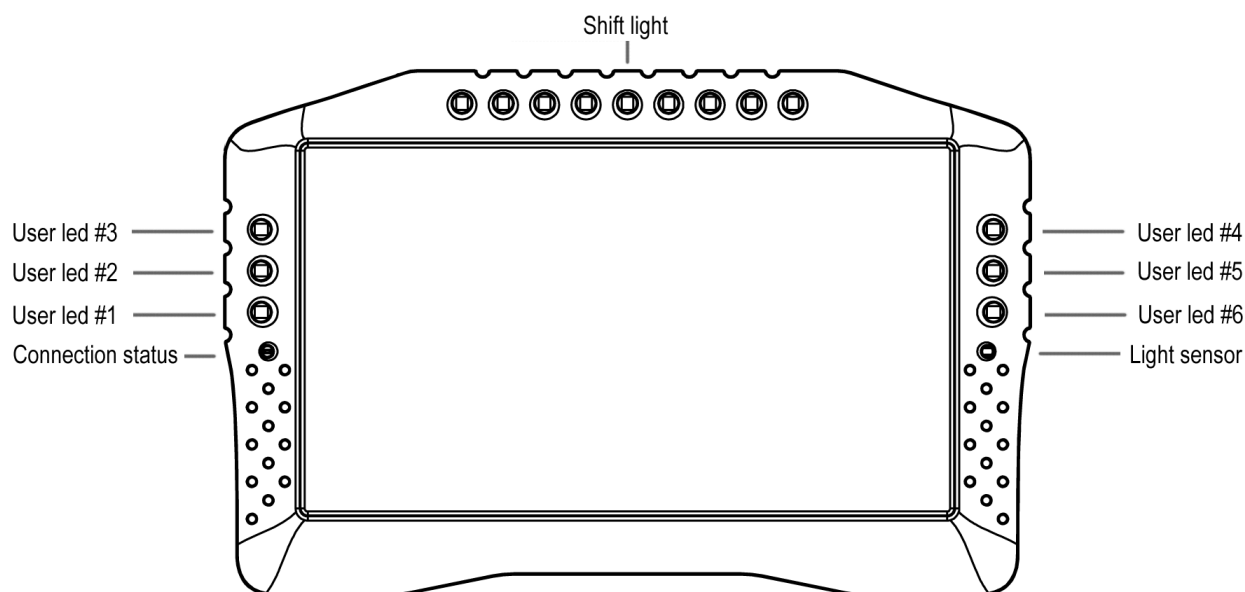
ADU 7 AS

(all dimensions in mm)



1.3. Device description

Front view



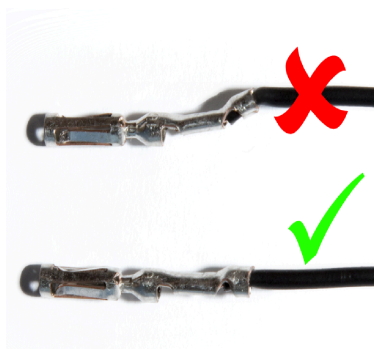
| Position | Description |
|--------------------------|--|
| Connection status | LED showing communication with a PC Flashes green when communicating. |
| Light sensor | The light sensor is used for automatic control of display and light-emitting diodes brightness |
| User LED #1 – #6 | RGB LEDs that can be controlled by user functions (e.g. alarms, indicators, etc.). |
| Shift light | Gear change indicator (user configurable) |

1.4. Connector

The display features a 35-pin AMPSEAL connector located at the rear of the housing. It is used to connect the power supply, CAN buses and additional sensors or buttons.

A complementary plug and terminals are included. Use the appropriate Crimper Tool to crimp the terminal's clamp around the cable. We do not recommend soldering the wire to the terminal! Compatible terminals: AMP 770520-1.

Step 1 – crimp the terminals



Correct crimping of the terminal's clamp is crucial to avoid problems when placing the terminal in a plug and its subsequent locking. When crimping, do not use excessive force and avoid deforming the terminal. The clamp part of the terminal which holds the cable's insulation should be straight, smooth and rounded and have a diameter equal to or smaller than the head of the terminal.

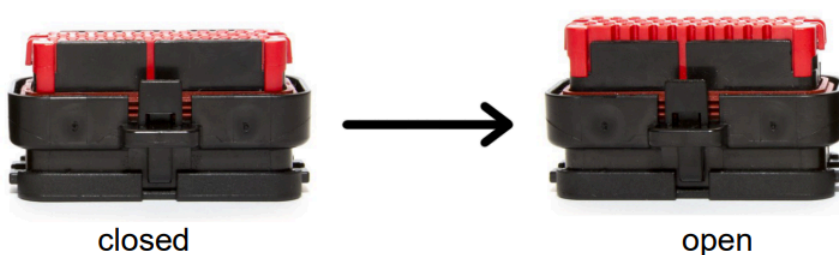
Step 2 – prepare the plug

To place the terminals into the plug, you must unlock the red part (the lock) and set it to open position. To unlock the red part of the plug, pry the black teeth on both sides of the plug using a sharp tool (Picture 1), and gently pull the lock. It should eject for about 0.5 cm (Picture 2). The teeth prevent total removal of the red lock from the plug (do not disassemble completely the plug!). In this position the plug is open and terminals can be plugged in.

Picture 1



Picture 2



Step 3 – plug in the terminals



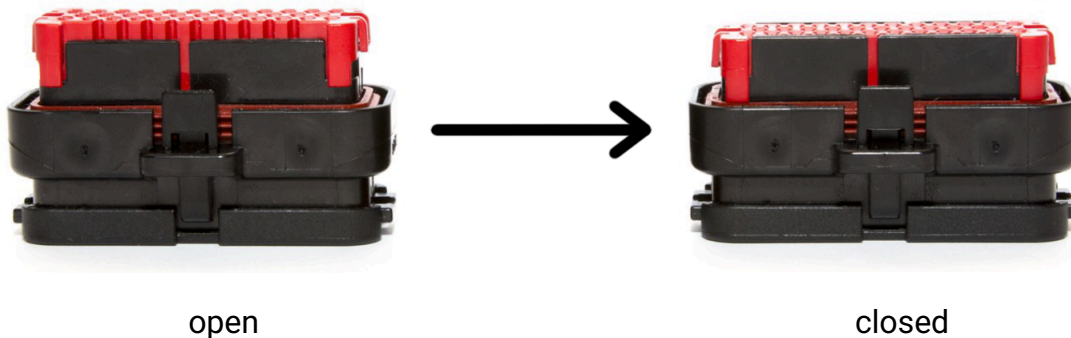
To place a terminal into the plug, insert a terminal until you hear and feel a click (the terminal should easily pinch through the silicon gasket).

**Warning:**

Do not force the terminal forward after you hear and feel the click – if it goes too deep, it will not be possible to close the red lock

Step 4 – close the lock

After all the terminals are correctly inserted, set the red lock to close position.



After the plug is locked inspect if any of the terminals don't stick out of the red lock, the tips of the terminals should be aligned with the surface of the lock.

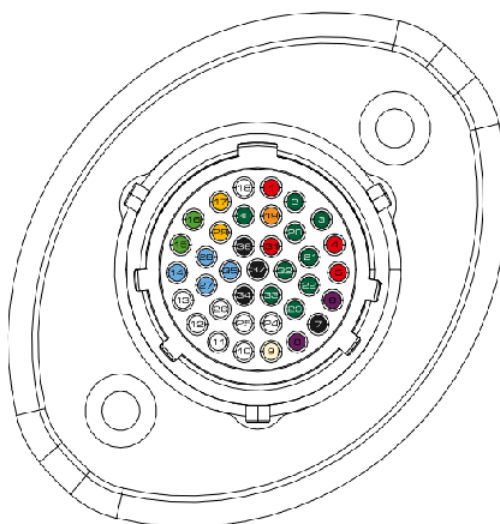
Removal of the terminal

To remove a terminal from the plug set the lock to open position (check Step 2). Then grasp the cable close to the plug, turn it left about half a turn, then turn it right while pulling gently outside.

Video reference presenting the correct installation of the terminals in the AMPSEAL connector:

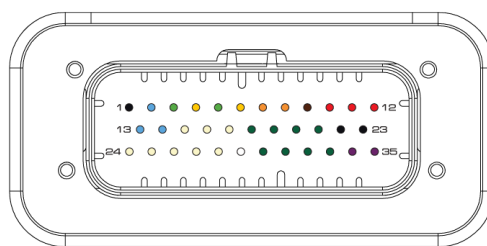
www.youtube.com/watch?v=uXTkm_XV20Y

1.5. Description of the ADU AS socket



| | | | | | |
|-----------|--------------------------|-----------|--------------------------|-----------|--------------------------|
| 1 | Battery 12V | 14 | USB.VBUS | 27 | USB.DM |
| 2 | Frequency/ digital in D1 | 15 | CAN1.H | 28 | USB.DP |
| 3 | Frequency/ digital in D2 | 16 | CAN2.H | 29 | CAN1.L |
| 4 | +5V output | 17 | CAN2.L | 30 | RS232.TXD |
| 5 | +5V output | 18 | Analog output | 31 | Switched 12V |
| 6 | AUX2 | 19 | RS232.RXD | 32 | Frequency/ digital in D6 |
| 7 | Power ground | 20 | Frequency/ digital in D3 | 33 | Frequency/ digital in D8 |
| 8 | AUX1 | 21 | Frequency/ digital in D4 | 34 | Sensor ground |
| 9 | Analog in A2 | 22 | Frequency/ digital in D5 | 35 | USB.GND |
| 10 | Analog in A1 | 23 | Frequency/ digital in D7 | 36 | Sensor ground |
| 11 | Analog in A3 | 24 | Analog in A4 | 37 | Ground |
| 12 | Analog in A5 | 25 | Analog in A6 | | |
| 13 | Analog in A7 | 26 | Analog in A8 | | |

1.6. Description of the ADU socket



| | | | | | |
|-----------|---------------|-----------|-------------------------|-----------|-------------------------|
| 1 | USB.GND | 13 | USB.DM | 24 | Analog in 8 |
| 2 | USB.VBUS | 14 | USB.DP | 25 | Analog in 7 |
| 3 | CAN1.H | 15 | Analog in 5 | 26 | Analog in 6 |
| 4 | CAN1.L | 16 | Analog in 3 | 27 | Analog in 4 |
| 5 | CAN2.H | 17 | Analog in 1 | 28 | Analog in 2 |
| 6 | CAN2.L | 18 | Frequency/ digital in 8 | 29 | Analog output |
| 7 | RS232.RXD | 19 | Frequency/ digital in 6 | 30 | Frequency/ digital in 7 |
| 8 | RS232.TXD | 20 | Frequency/ digital in 4 | 31 | Frequency/ digital in 5 |
| 9 | Sensor ground | 21 | Frequency/ digital in 2 | 32 | Frequency/ digital in 3 |
| 10 | +5V output | 22 | Power ground | 33 | Frequency/ digital in 1 |
| 11 | Switched 12V | 23 | Ground | 34 | Aux 1 |
| 12 | Battery 12V | | | 35 | Aux 2 |

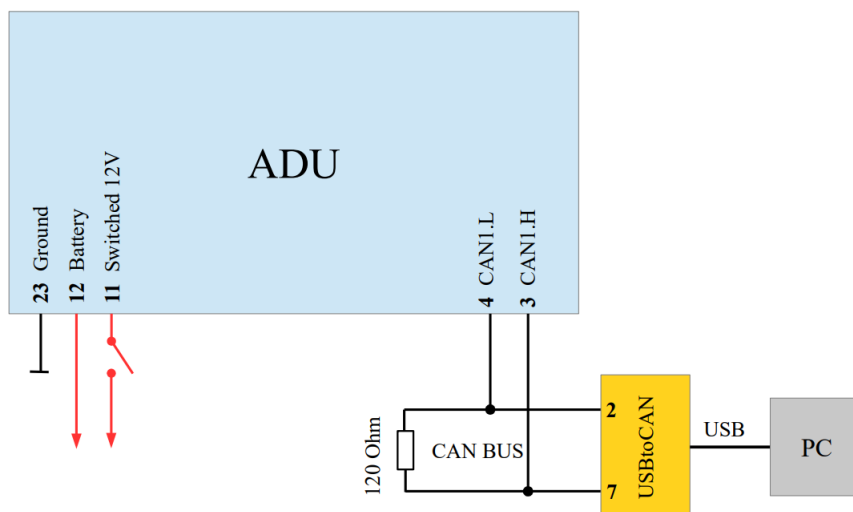
| Terminal | Description |
|-------------------------|---|
| 1. USB.GND | Ground cable for the USB port |
| 2. USB.VBUS | VBUS signal for the USB port |
| 3. CAN1.H | CAN H signal for CAN bus 1 |
| 4. CAN1.L | CAN L signal for CAN bus 1 |
| 5. CAN2.H | CAN H signal for CAN bus 2 |
| 6. CAN2.L | CAN L signal for CAN bus 2 |
| 7. RS232.RXD | RXD (reception) signal for serial RS232 bus Used for serial connection to EMU Classic, Classic Hondata, Autronic SM4, AEM, GEMS, Athena GE and other devices supporting the AIM protocol. |
| 8. RS232.TXD | TDX signal (transmission) for serial RS232 bus |
| 9. Ground sensor | Ground for external sensors (e.g. oil pressure sensor) |
| 10. +5V output | +5 V power supply for external sensors. Maximum current consumption 400 mA |
| 11. Switched 12V | +12 V signal to turn on the device. The device is supplied by terminal 12 (Battery 12 V). |

| Terminal | Description |
|------------------|--|
| 12. Battery 12V | Power supply for the device and real-time clock power when the device is turned off (no signal at the 12 V input 11). If the 12 V Battery fails to supply power, the real-time clock will be powered by an internal battery of the device. |
| 13. USB.DM | D- signal to the USB port |
| 14. USB.DP | D+ signal to the USB port |
| 15. Analog in 5 | Analog input 5. Measuring range 0–5 V |
| 16. Analog in 3 | Analog input 3. Measuring range 0–5 V |
| 17. Analog in 1 | Analog input 1. Measuring range 0–5 V |
| 18. Digital in 8 | Digital input 8 |
| 19. Digital in 6 | Digital input 6 |
| 20. Digital in 4 | Digital input 4 The input can also support a digital oil level / temperature sensor |
| 21. Digital in 2 | Digital input 2 The input can also support a Flex fuel sensor. |
| 22. Power ground | Device ground used by AUX outputs and LEDs |
| 23. Ground | Device ground |
| 24. Analog in 8 | Analog input 8. Measuring range 0–5 V |
| 25. Analog in 7 | Analog input 7. Measuring range 0–5 V |
| 26. Analog in 6 | Analog input 6. Measuring range 0–5 V |
| 27. Analog in 4 | Analog input 4. Measuring range 0–5 V |
| 28. Analog in 2 | Analog input 2. Measuring range 0–5 V |
| 29. Analog out | Analog output 0– 5 V |
| 30. Digital in 7 | Digital input 7 |
| 31. Digital in 5 | Digital input 5 |
| 32. Digital in 3 | Digital input 3 The input supports signal from an AIM beacon |
| 33. Digital in 1 | Digital input 1 The input supports signals from a crankshaft / camshaft position sensor (RPM) |
| 34. Aux 1 | <i>Low side</i> output. Maximum current 3 A |
| 35. Aux 2 | <i>Low side</i> output. Maximum current 3 A |

2. 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. 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.

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 section).

**Important:**

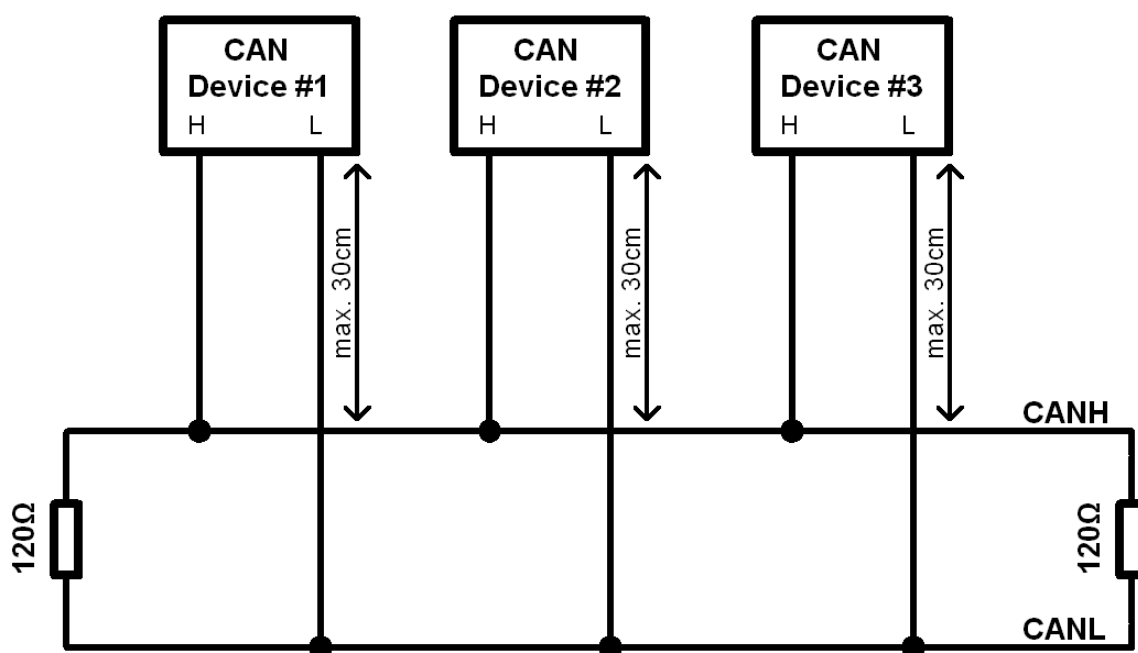
Do not connect the +5 V output from the CAN interface to +5 V ADU!

2.1. 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 ADU 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 nodes is 30

For 500kbps:

- 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 nodes 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 | CAL POT 2 | Switch mask | Heartbeat | | |

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

2.2. Connecting to ECU

To be able to display and log data from the control unit, connect the ADU to the ECU's CAN bus or serial port.

When using a CAN bus, you can connect the ECU to CAN1 (where the ECU 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.

Some engine managements used in motor sports do not have a CAN bus, but are equipped with a serial output (RS232). This is the case for EMU, CLASSIC EMU or Hondata. ADU supports the following serial formats: AIM, Ecumaster serial protocol, Ecumaster EMU Classic EDL1 protocol, Hondata serial protocol, AEM, GEMS, Athena GET, Autronic SM4, MoTec M4 DATA SET5. For additional information see the **CANbus / Serial Setup** chapter.

For OEM controllers equipped with an OBD 2 over a CAN bus (all cars after 2008) it is possible to use the OBD connector to read the basic engine operation parameters. For more information, see the **OBD 2** chapter.

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

2.3. Connecting via a CAN bus

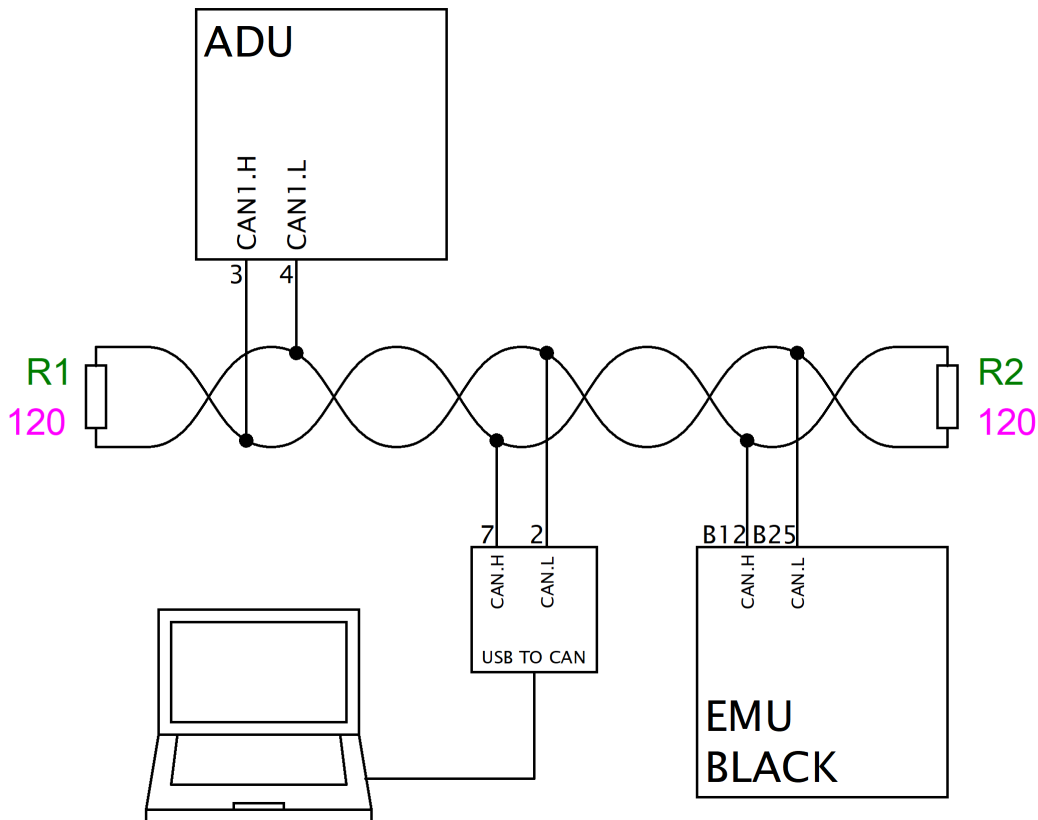
The following diagram shows an example connection of the ECU to the ADU using CAN1 bus.



Important:

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

If the CAN bus speed is different than 1 Mbps or if you want to isolate the ECU from the ADU 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* section.

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

Information about channel configuration with CAN stream data can be found in the **CANbus Inputs** section.

2.4. Connecting via an RS232 serial bus

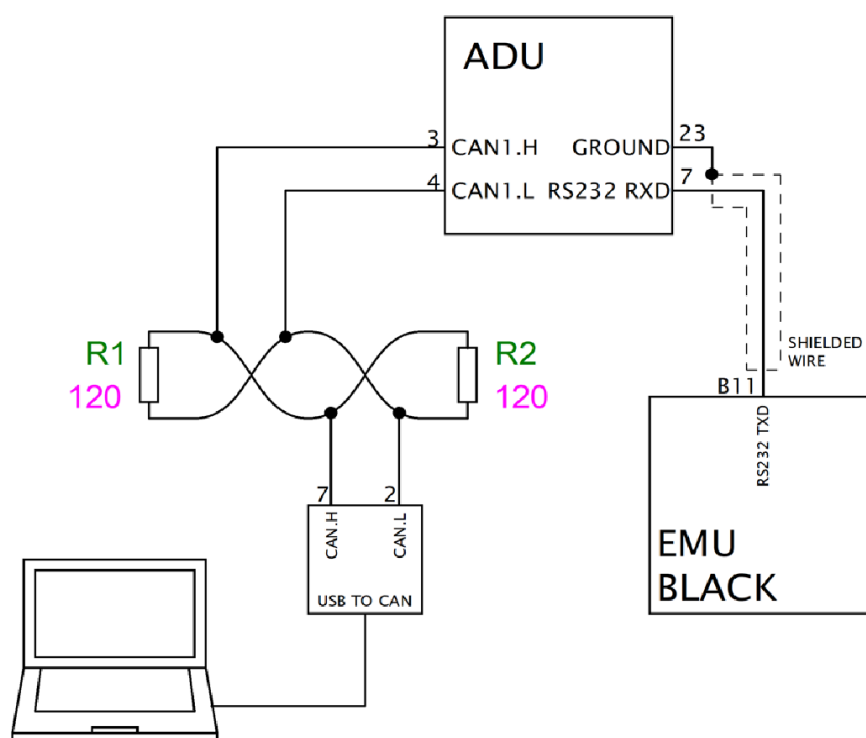
The ADU display features a built-in serial bus (RS232). In order to connect the ECU via a serial bus connect the ECU (Tx) output to the ADU (Rx) input.



Important:

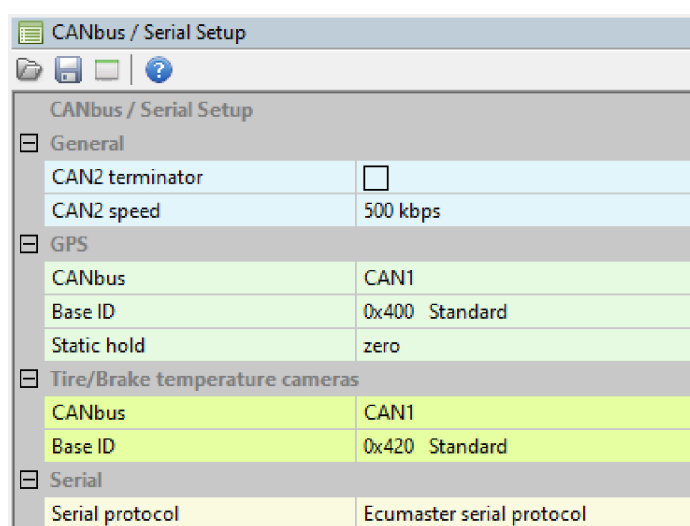
To transfer data via serial communication please use a shielded cable.

The diagram below shows an example of such a connection.



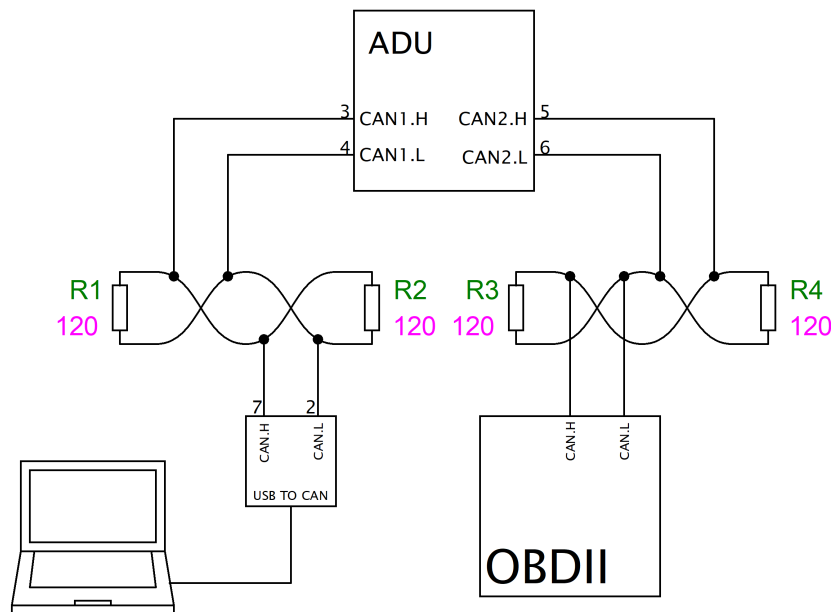
Detailed information on connecting specific ECU brands to ADU can be found in the application notes at www.ecumaster.com/products/adu/.

To configure the serial protocol (in accordance with the connected ECU), select the appropriate protocol in the *CANbus / Serial* window (*Ecumaster serial protocol*, *Ecumaster EMU Classic EDL 1 protocol*, *AIM*, *Hondata*, *Autronic SM4*, *AEM*, *GEMS*, *Athena GET*, *MoTec M4 DATA SET5*).

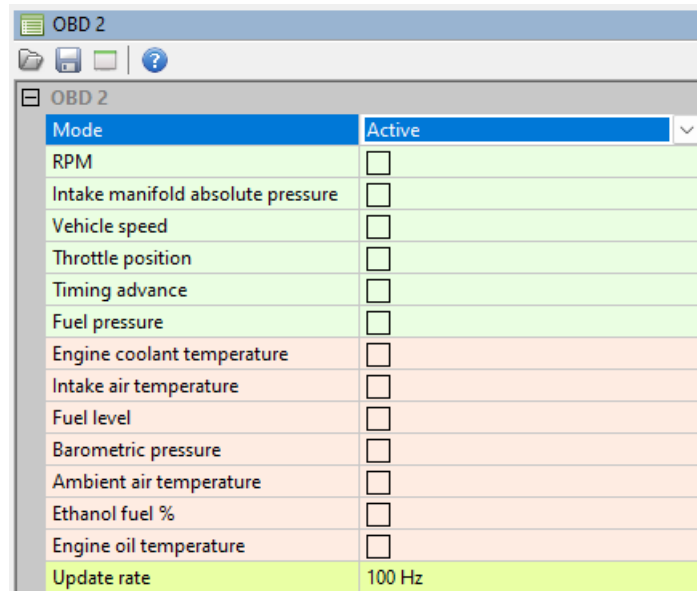


2.5. OBD 2

The following diagram shows a connection of the CAN2 bus to the OBD 2 bus of the vehicle.



In the *CANbus / Serial Setup* window, set the correct CAN bus speed (usually 500 kbps).



ADU offers compatibility with the ISO15765/SAE J2480 standard for OBD communication, providing 11-bit, 250 Kbaud, and 11-bit, 500 Kbaud variants.

Select active or passive mode. In the active mode, the device actively requests real-time data on selected parameters from the vehicle's systems. In passive mode, the device only reads information queried by another device operating in active mode. Passive mode is essential when multiple devices are connected to OBD, as allowing more than one device to send queries

simultaneously would disrupt communication. Therefore, only one device can actively query the vehicle's systems at a time, ensuring efficient and reliable communication.

Then, select the respective channels in the OBD 2 window. The channels are divided into two groups: fast (green) and slow (orange). Fast channels use 66% of the bandwidth (*update rate*) while slow channels use 34%. This means, for example, that with an Update rate of 100 Hz (data is downloaded by OBD 2 at 100 Hz), the RPM channel will be refreshed 66 times per second (66 Hz). If an additional channel is selected (e.g. *Throttle position*), the frame rate of both channels will be 33 Hz. It should be noted that not all ECUs are capable of refreshing data at 100 Hz and it may be necessary to reduce the *Update rate*.

**Important:**

Not all logging channels found in the OBD 2 window are available for all ECUs.

There is also the possibility of mixed parameter reading for serial controllers, e.g. *throttle position* and *RPM* can be read directly from the CAN bus and the other parameters via OBD 2. For more information see the **CANbus Inputs** section.

2.6. GPS module

The ADU display makes it possible to use the *Ecumaster GPS to CAN* module to measure lap times on a track and allows data analysis based on the vehicle's position on the track.

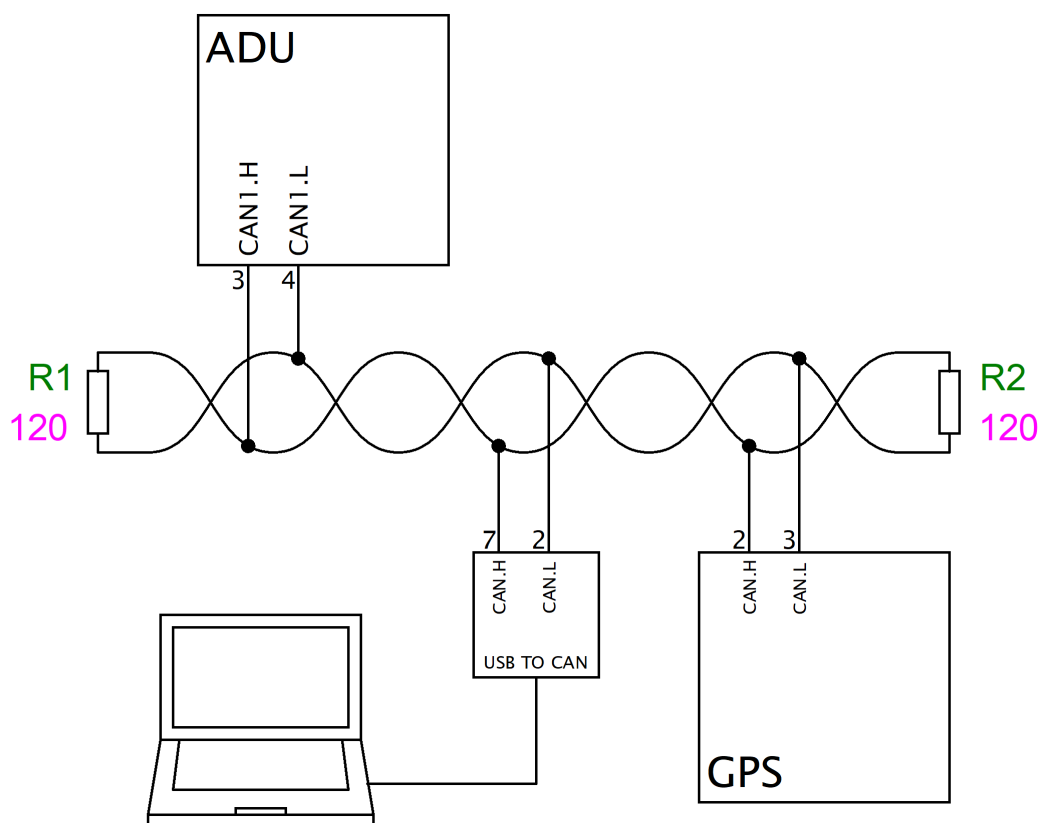
To fix the current position of the vehicle, the module uses information from GPS / Glonass satellites, a built-in accelerometer and a gyroscope. The position is refreshed with a frequency of 20 Hz or 25 Hz dependent on GPS module version.

**Note:**

The ADU only supports the **Ecumaster GPS to CAN** module and does not support other GPS modules or interfaces.

The CAN bus speed of the GPS module is factory set to 1 Mbps. This is why it is recommended to connect the GPS module to CAN1 bus.

A sample connection diagram is shown below:



In the *CANbus / Serial Setup* options, select the CAN bus to which the GPS module is connected (*GPS / CANbus*).

The method of installing the GPS module to the vehicle's body and its subsequent calibration are very important. Due to the use of both gyroscope and accelerometer in order to obtain more accuracy the GPS module should be mounted using the supplied anti-vibration pads.

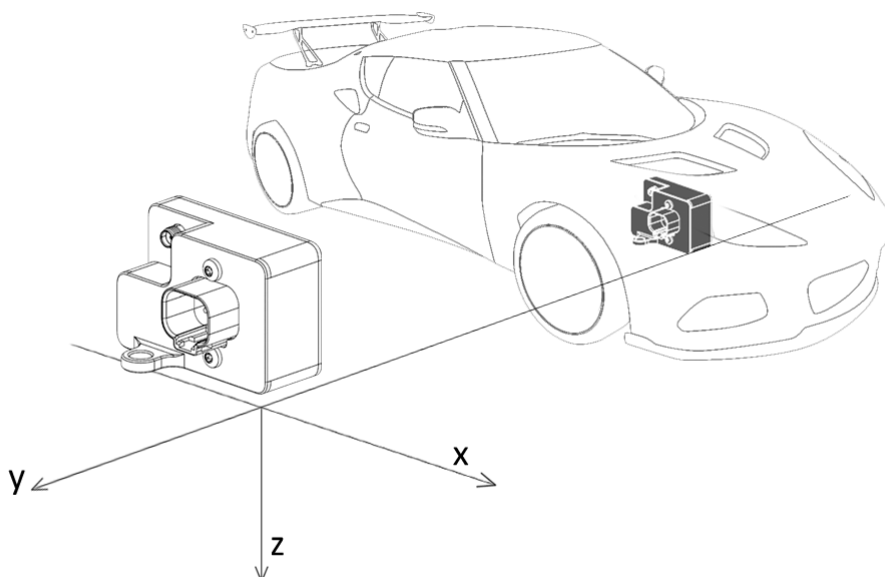
GPS V1 (20 Hz)

The orientation of the module does not matter, however, after installation, it must be calibrated. Calibration is automatic when the vehicle travels the first few hundred meters. If the position of a calibrated module is changed, a short calibration drive is required.

For more information about measuring lap times with the GPS module, see the **Lap Times** chapter.

GPS V2 (25 Hz)

The orientation of GPS V2 should be as presented:



No additional calibration is required.

Available GPS channels

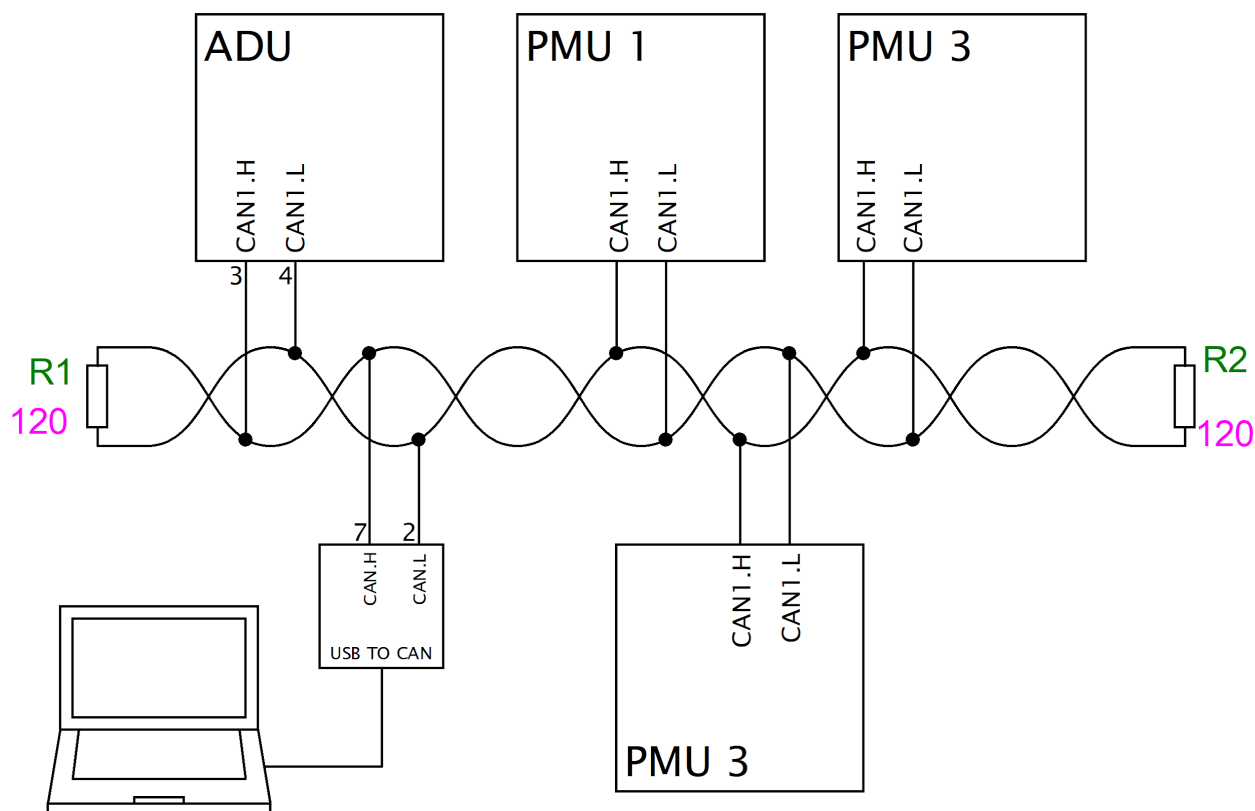
| Channel | Description | | |
|----------------------|----------------------------------|---------------------|---|
| gps.latitude | Latitude | | |
| gps.longitude | Longitude | | |
| gps.height | Height above sea level in metres | | |
| gps.status | 0 | Disconnected | no GPS module data |
| | 1 | No fix | unable to fix position |
| | 2 | IMU | position is fixed using the built-in accelerometer and gyroscope – (only GPS V1) |
| | 3 | GPS 2D | position is fixed in 2D space using GPS / Glonass satellite data |
| | 4 | GPS 3D | position is fixed in 2D space using data from GPS or Glonass satellites |
| | 5 | GPS + IMU | position is fixed in 3D space and corrected using the built-in accelerometer (highest accuracy) – (only GPS V1) |

| Channel | Description | | |
|--------------------|---|----------------|--|
| gps.fusionStatus | 0 | Initialisation | Initialisation and calibration of inertial sensors – (only GPS V1) |
| | 1 | Fusion | the device uses inertial sensors to correct the vehicle's position – (only GPS V1) |
| | 2 | Suspend | temporary error of inertial sensors – (only GPS V1) |
| | 3 | Disabled | error of inertial sensors; they are not taken into account when determining the vehicle's position |
| gps.speed | Vehicle speed in km/h | | |
| gps.headingMotion | Vehicle motion direction. When the vehicle is skidding, the <i>headingMotion</i> value will be different from the <i>headingVehicle</i> value. | | |
| gps.headingVehicle | The direction in which the front of the vehicle is facing. When the vehicle is skidding, the <i>headingMotion</i> value will be different from the <i>headingVehicle</i> value. | | |
| gps.accX | Longitudinal acceleration (longitudinal g) | | |
| gps.accY | Lateral acceleration (lateral g) | | |
| gps.accZ | Vertical acceleration (vertical g) | | |
| gps.gyroX | Angular velocity of the vehicle's longitudinal axis | | |
| gps.gyroY | Angular velocity of the vehicle's lateral axis | | |
| gps.gyroZ | Angular velocity of the vehicle's vertical axis | | |
| gps.noise | Average satellite signal noise level. The lower the value the better. | | |
| gps.numSatellites | Umber of satellites used to fix position | | |

2.7. Ecumaster PMU

If connecting the ADU to a system with a PMU, it is recommended to connect CAN1 of the PMU to CAN1 bus of the display. This will allow easy communication both with a PC and between the devices.

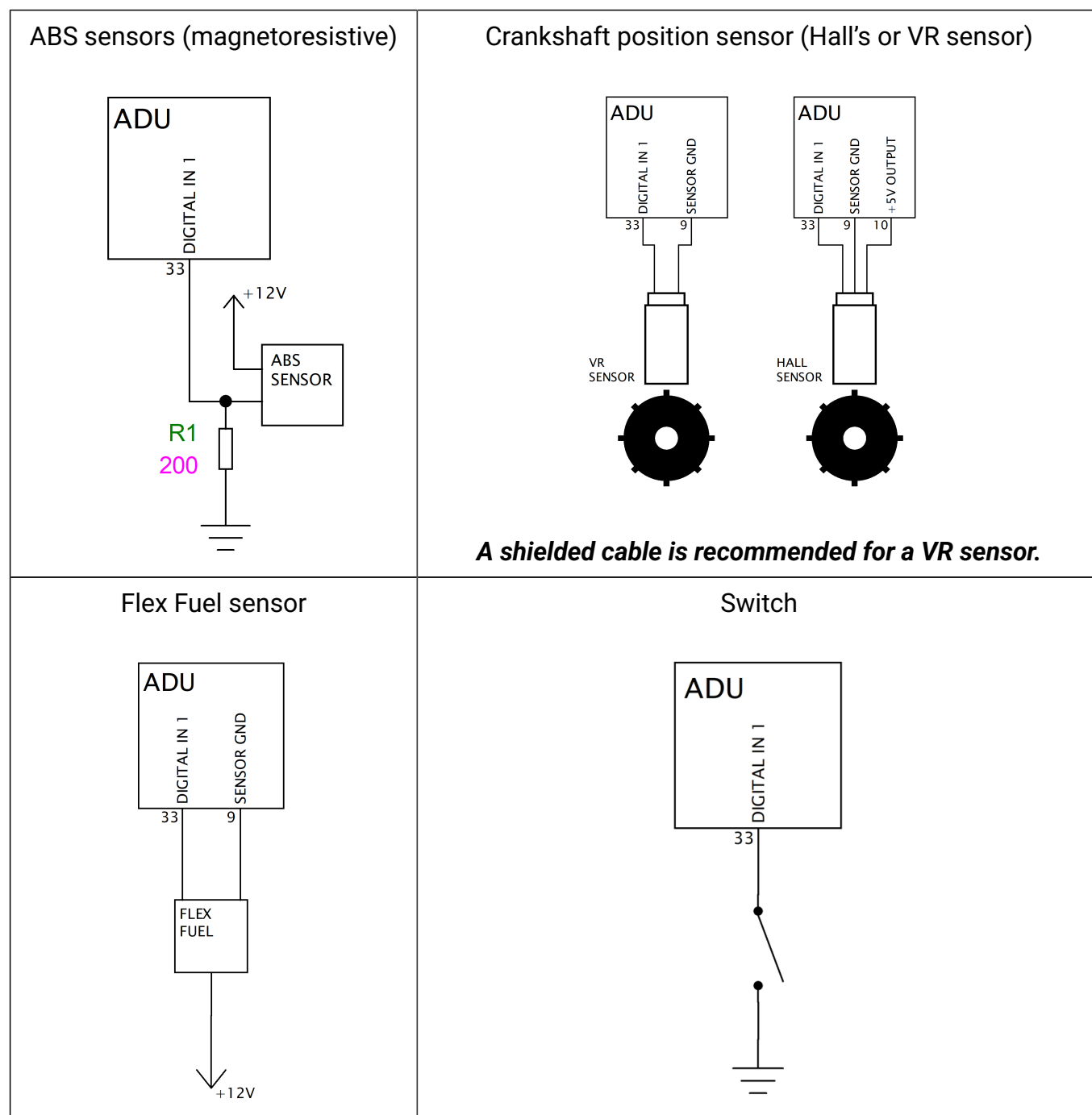
A sample connection diagram is shown below.



2.8. Digital Inputs

Digital inputs are used to process frequency signals (e.g. from the engine speed sensor, the turbocharger sensor, the Flex Fuel sensor), beacon signals and to connect switches.

It is possible to read signals from inductive sensors (VR sensors), Hall's sensors, optical sensors. These inputs are equipped with built-in switchable 2k2 (up to +5 V) pull-up resistors, which enables connecting Hall sensors or switches shorted to ground.



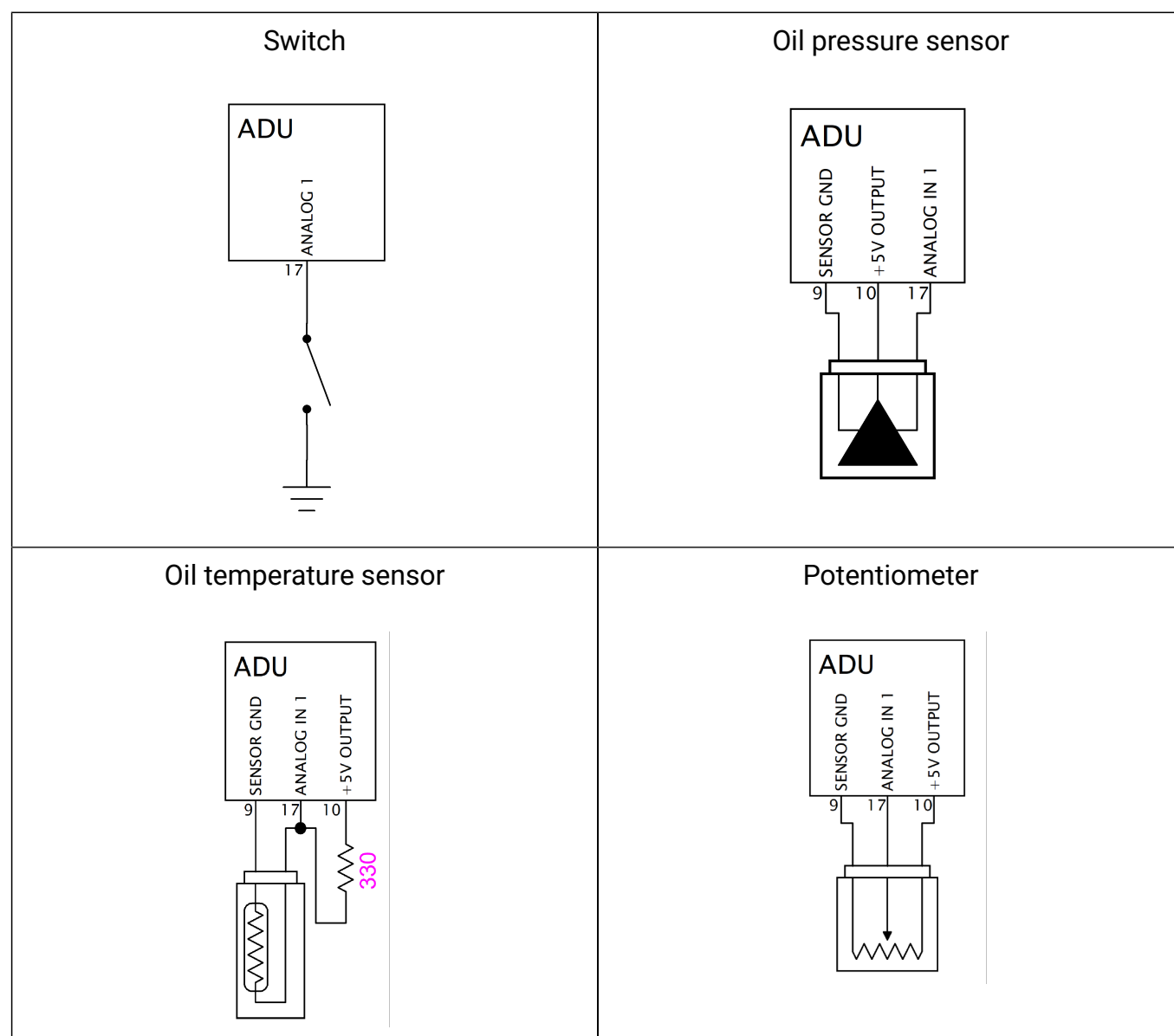
2.9. Analog Inputs

Analog ADU inputs are able to process voltage in the range of 0-5 V (voltages above 5 V are read as 5 V) with a frequency of 500 Hz.

The primary use of these inputs is to display and log signals from analog sensors such as pressure sensors (oil, fuel, water, etc.) or RTDs (e.g. oil temperature sensor, fuel level sensor).

These analog inputs can also be used to connect *switches*.

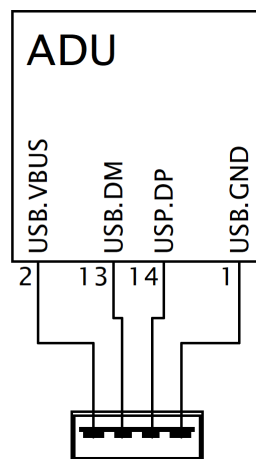
The voltage of all analog inputs can be transferred to other devices (e.g. EMU, BLACK) via the CAN bus.



2.10. USB Flash drive (pendrive)

To log parameters to an external storage device, connect a USB storage device to the ADU. This makes it possible to use popular pen drives. Built-in real time clock (battery powered) is used to set the file dates correctly. The supported file system is FAT32. File size depends on the number of channels defined and the save frequency. For more information on data logging go to the *Logging* chapter. It is recommended to use quality brand flash memory sticks compatible with the USB 3.0 standard (e.g. Sandisk Ultra, Sony USM8W3, USM16W3, USM32W3), with a maximum capacity of 32 GB. Using USB storage devices with poor parameters can lead to interrupt the recording of logged data.

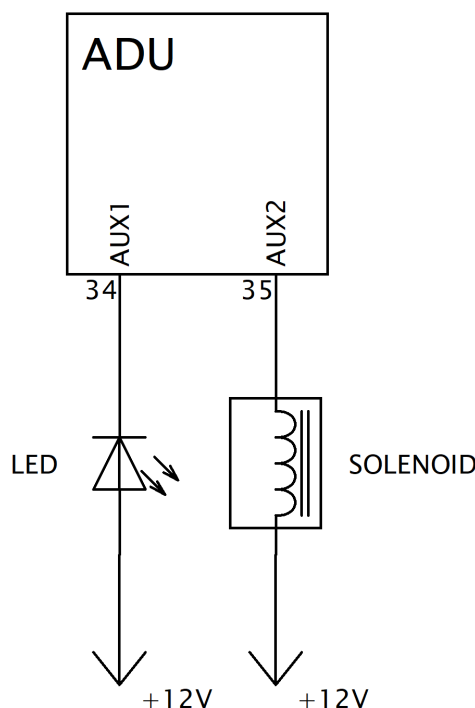
A sample USB socket connection diagram is shown below.

**Important:**

D+ and D- cables should be twisted pair wires and the entire USB bundle must be shielded.

2.11. Low side outputs

Built-in low side outputs (shorted to ground) can be used to connect external loads (e.g. LEDs, solenoid valves) depending on the functions defined by the user (e.g. radiator fan activation, signalling).



3. ADU Client software for Windows

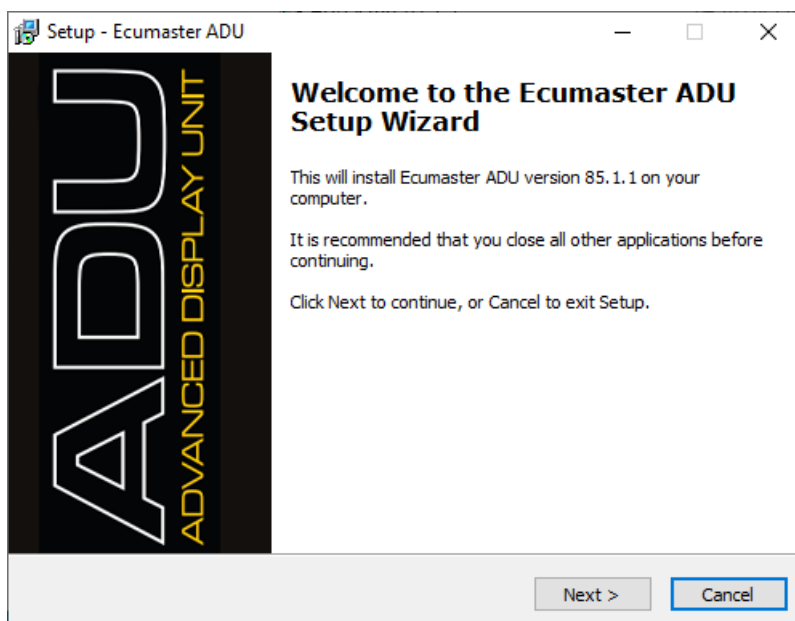
To set up the ADU dashboard, it is necessary to install software for Windows. Official software can be found at www.ecumaster.com/products/adu/.

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:

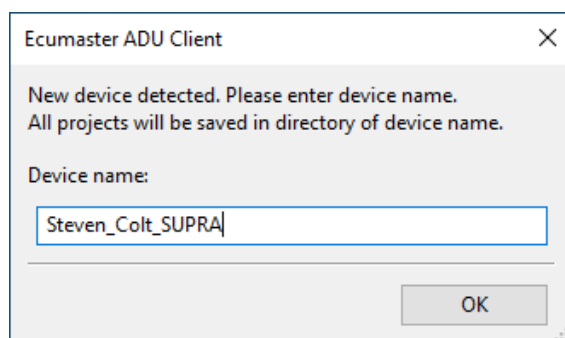
- Windows XP, VISTA, 7, 8, 10, 11 (32 and 64 bits)
- minimum screen resolution of 1366x768
- Open GL compatible graphics card
- 2GB RAM
- USB port

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



3.1. 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 / ADU / folder will be created under this name, where all settings relating to the device will be stored.

Client and Firmware:

Each ADU 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 ADU using a PC.

The Client software installation file contains the latest Firmware versions for the ADU.

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 ADU Client.
- FirmwareMinor, i.e. the minor version - changes when a Firmware fix is published, but WITHOUT making new settings in the ADU 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:

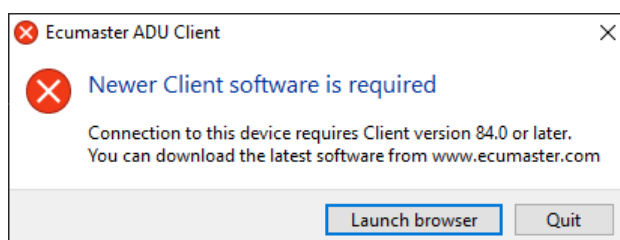
- supports any Firmware from the 83.x family: 83.0, 83.1, 83.2 etc.;
- supports all older Firmware versions, e.g. 50.0, 51.1, 82.1, 82.0, 81.0 etc.;
- It does NOT support higher FirmwareConfiguration version numbers e.g. 84.0, 84.1.

In other words, the latest ADU Client version supports all ADUs 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 first connection:

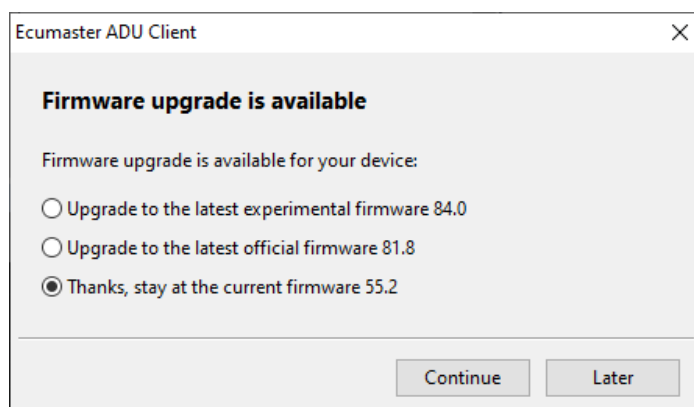
Depending on the Firmware version on the ADU and the Client software version installed on the PC, messages recommending a firmware update may appear the first time you connect to the device:

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.

- 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 ADU (***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.



3.2. Firmware update

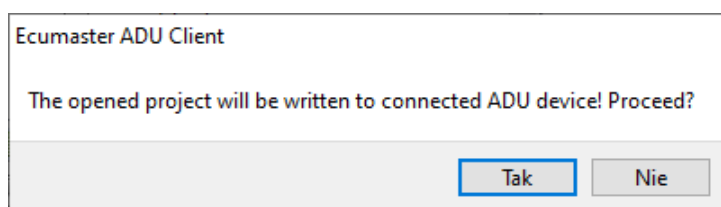
If necessary, the Firmware version can be changed at any time in the ADU. To do this, select **File / Upgrade firmware...** from the main menu and then select the relevant firmware version from the list.

3.3. Loading a project

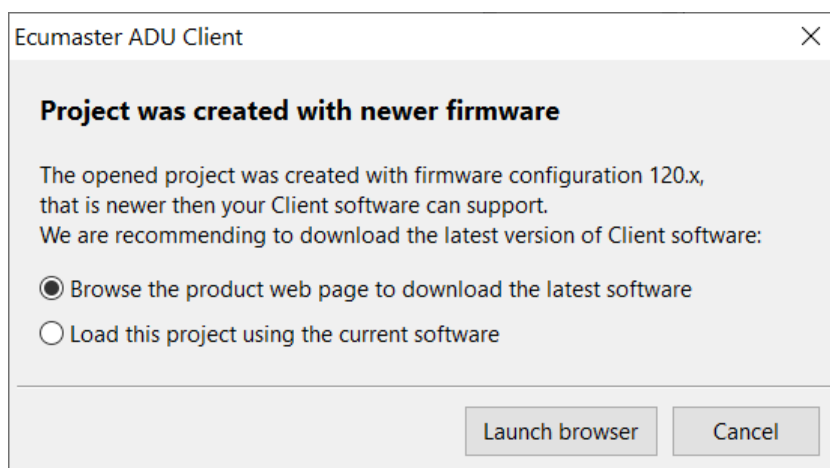
When opening a finished project, different messages may appear depending on:

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

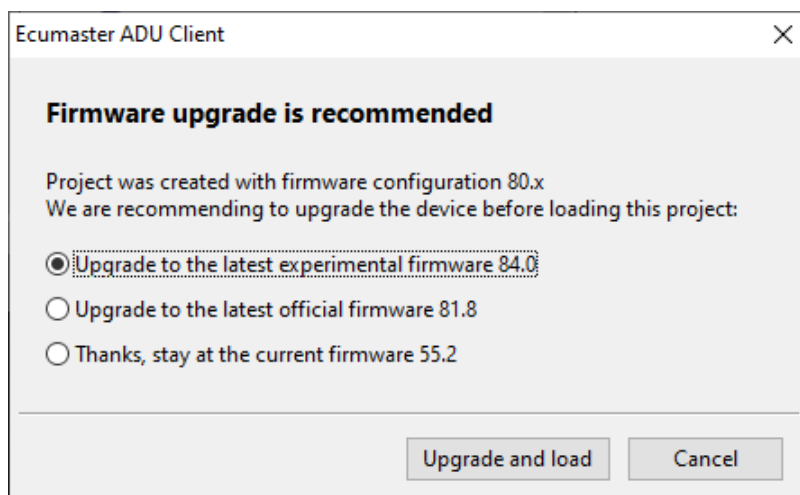
If the device is connected while opening a project, a message may appear indicating that the project being opened will be saved on the connected ADU.



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 ADU is connected and when working offline.



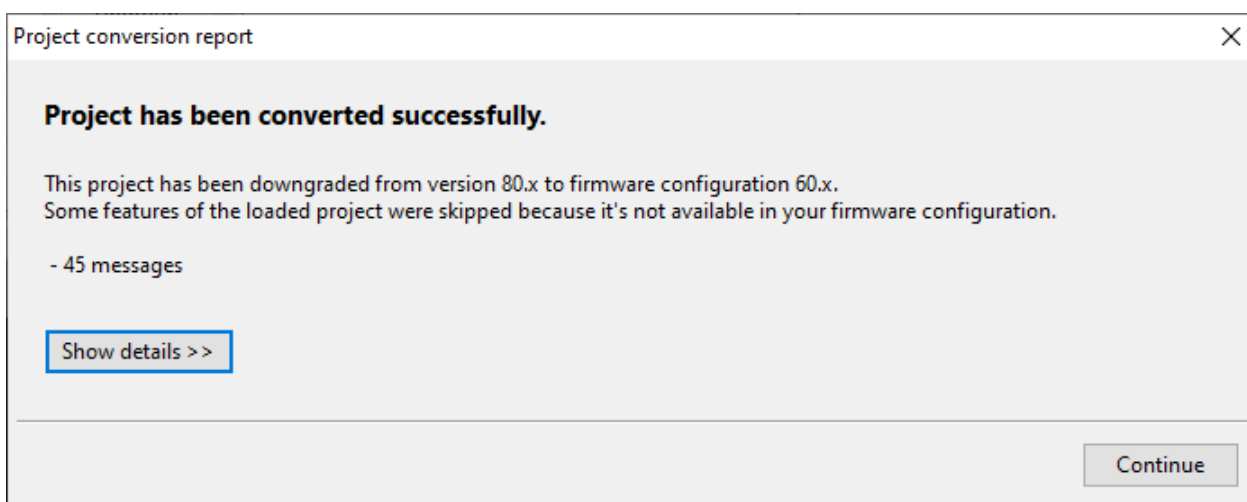
If a project was created using a firmware configuration not supported by the version of Firmware on the ADU, a window may appear recommending a Firmware update before loading the project.



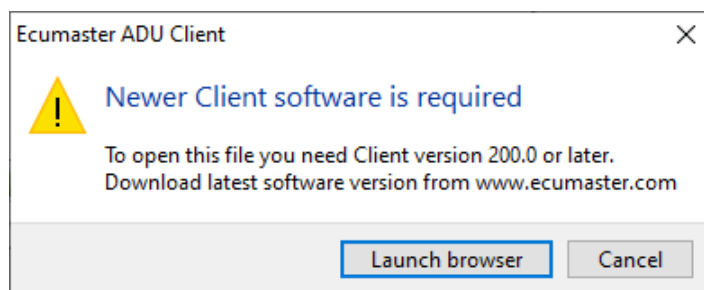
If the software was installed in an experimental (test) version, the message window will offer three choices: update to the latest experimental version, update to the latest official version or keep the existing version.

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.

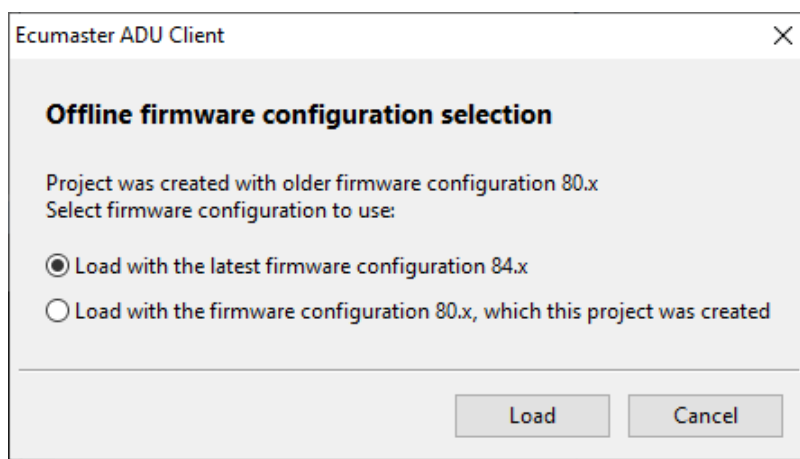
If a project created using a newer configuration is loaded into a device with older configuration software, some functions of the project will not be available because they are not supported by the older configuration.



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 ADU is connected and when working offline.

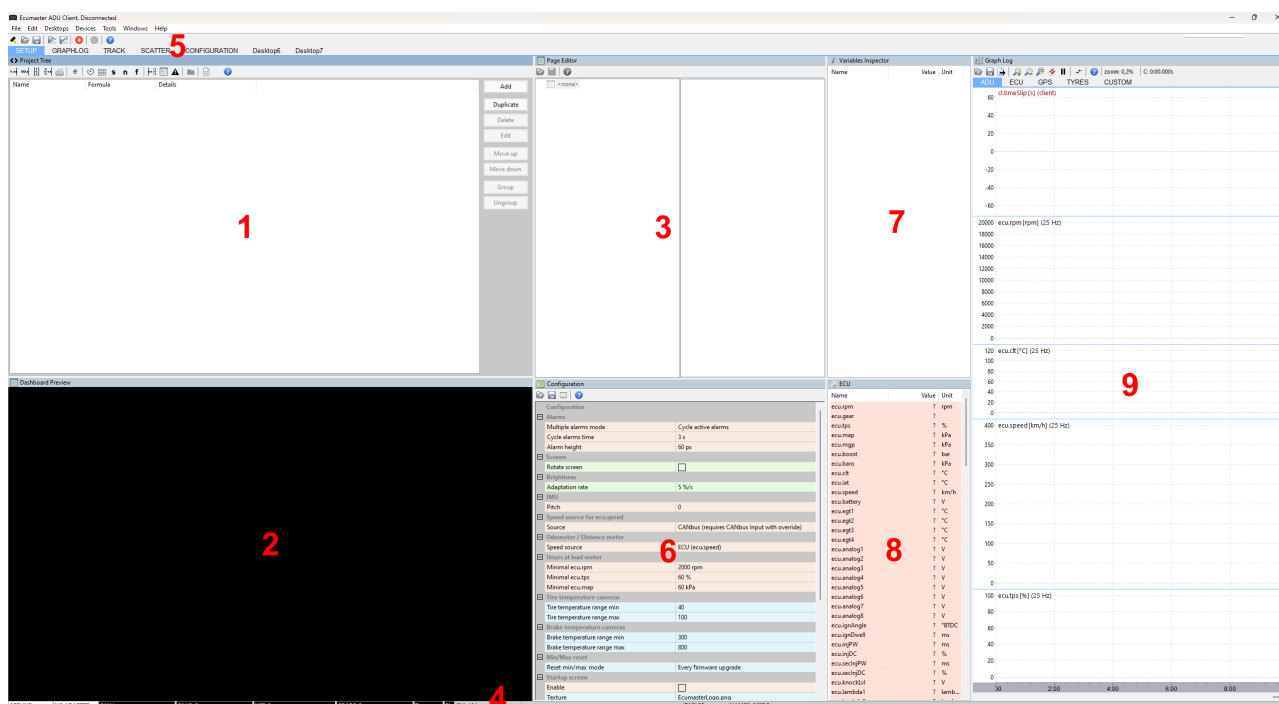


If there is no connection to the ADU and an attempt is made to open a project created using an older configuration than the one supported by the version of the Client software installed on the PC, a window will appear, prompting the user to select a software configuration.



3.4. 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:

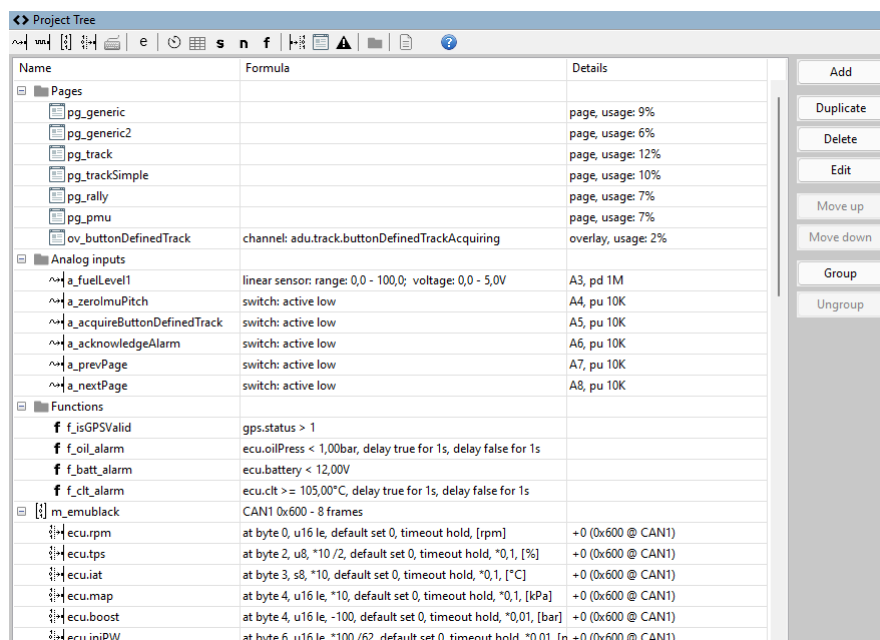
- **Analog Input** – an analog input, where all of its Parameters can be defined, e.g. the input number, the name of the variable, the type, pull-up etc.
- **Digital Input** – a digital input, where its properties can be defined, e.g. the type, sensitivity, name of the variable, pull-up, etc.
- **CANbus 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
- **Enumeration** – an element to assign a numeric value to the desired text or color, in order to display it on the screen
- **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
- **CANbus Export** – an element for sending CAN frames with variable and constant values
- **Page** – a single page displaying data. If more pages are defined, they can be toggled between using both functions and the buttons.
- **Alarm** – an element displaying the defined alarms, irrespective of the current page
- **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.

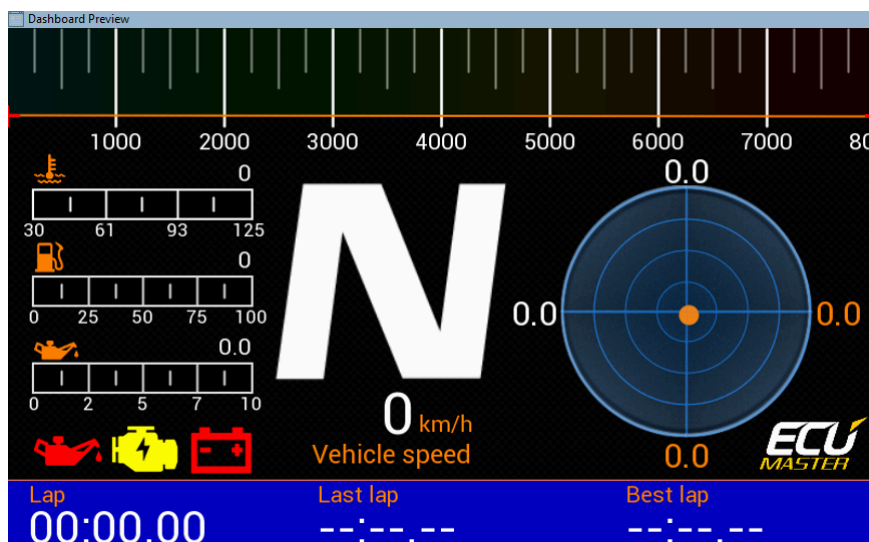


| Name | Formula | Details |
|-----------------------------|---|--------------------|
| Pages | | |
| pg_generic | | page, usage: 9% |
| pg_generic2 | | page, usage: 6% |
| pg_track | | page, usage: 12% |
| pg_trackSimple | | page, usage: 10% |
| pg_rally | | page, usage: 7% |
| pg_pmu | | page, usage: 7% |
| ov_buttonDefinedTrack | channel: adu.track.buttonDefinedTrackAcquiring | overlay, usage: 2% |
| Analog inputs | | |
| a_fuelLevel1 | linear sensor: range: 0,0 - 100,0; voltage: 0,0 - 5,0V | A3, pd 1M |
| a_zeroImuPitch | switch: active low | A4, pu 10K |
| a_acquireButtonDefinedTrack | switch: active low | A5, pu 10K |
| a_acknowledgeAlarm | switch: active low | A6, pu 10K |
| a_prevPage | switch: active low | A7, pu 10K |
| a_nextPage | switch: active low | A8, pu 10K |
| Functions | | |
| f_isGPSValid | gps.status > 1 | |
| f_oil_alarm | ecu.oilPress < 1,00bar, delay true for 1s, delay false for 1s | |
| f_batt_alarm | ecu.battery < 12,00V | |
| f_clt_alarm | ecu.clt >= 105,00°C, delay true for 1s, delay false for 1s | |
| m_emublack | | |
| ecu.rpm | at byte 0, u16 le, default set 0, timeout hold, [rpm] | +0 (0x600 @ CAN1) |
| ecu.tps | at byte 2, u8, *10 /2, default set 0, timeout hold, *0,1, [%] | +0 (0x600 @ CAN1) |
| ecu.iat | at byte 3, s8, *10, default set 0, timeout hold, *0,1, [°C] | +0 (0x600 @ CAN1) |
| ecu.map | at byte 4, u16 le, *10, default set 0, timeout hold, *0,1, [kPa] | +0 (0x600 @ CAN1) |
| ecu.boost | at byte 4, u16 le, -100, default set 0, timeout hold, *0,01, [bar] | +0 (0x600 @ CAN1) |
| ecu.injPW | at byte 6, u16 le, *100 /62, default set 0, timeout hold, *0,01, [n +0 (0x600 @ CAN1) | |

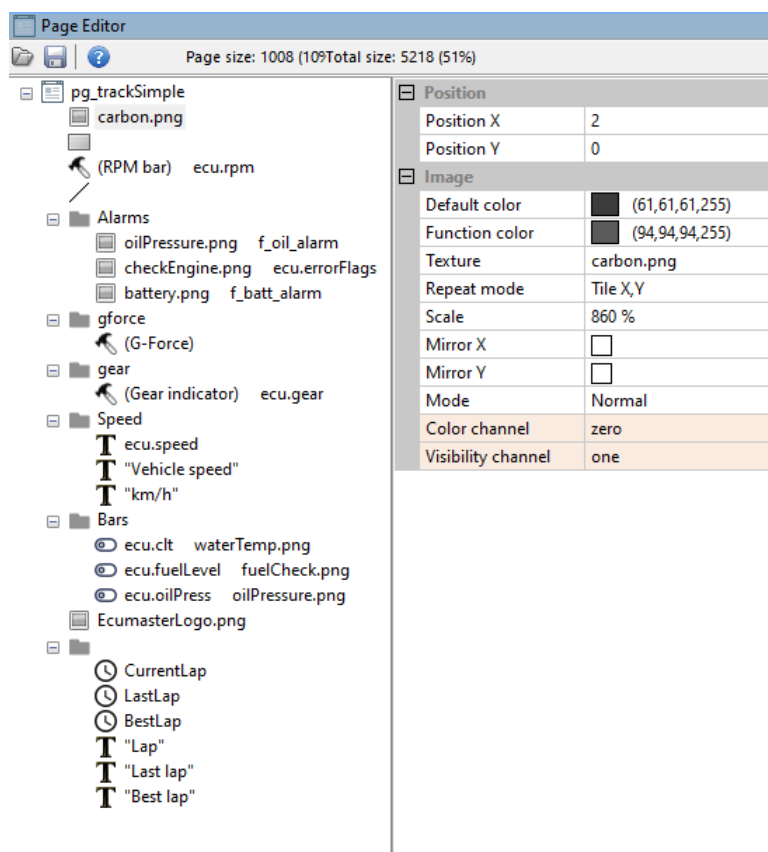
A selected page may be displayed and edited using the page preview screen (2).

When the application is connected via the CAN to USB interface to the ADU, all changes made to a page are transmitted to the device in real time.

The **CTRL+K** shortcut pressed in an active Dashboard Preview screen saves the current page image (800x480) to a png file.

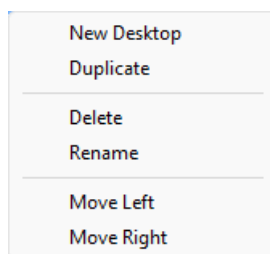


A page editor (3) is available in the page preview screen. It may be used to add and parametrize graphic page elements. For more information on pages see the [Pages \(on page 58\)](#) chapter.



The status bar is another important element of the interface (4). It contains useful information on the status of the connected device. For a detailed description see the *Status field* chapter.

Tabs are a very important element of the application (5). 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:

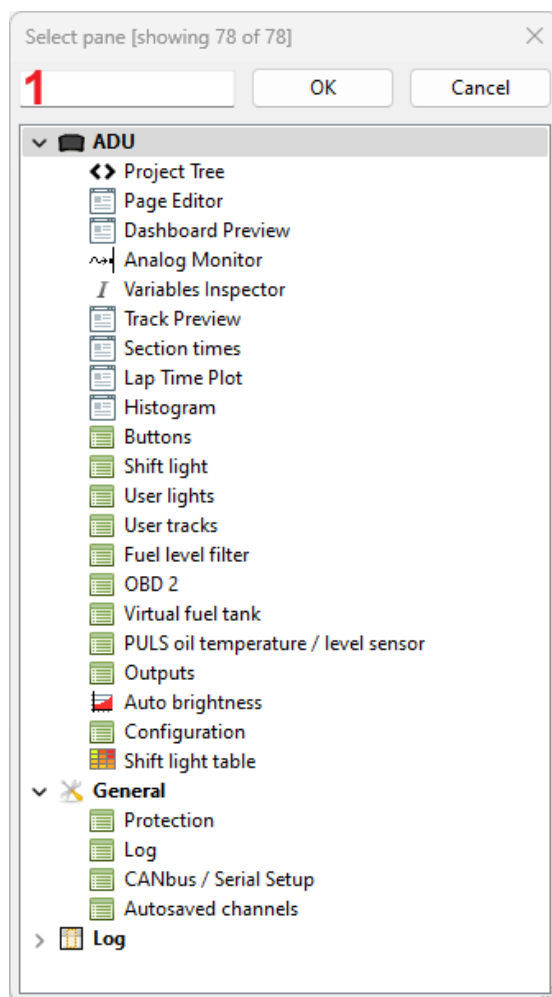


| | |
|--------------------|---|
| New Desktop | Create a new tab. A tab will always be created at the end. |
| Duplicate | Duplicate a tab. This option creates a new tab and copies the content of a highlighted tab into it. |
| Delete | Delete a tab. |
| Rename | Rename a tab. |
| Move Left | Move a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

Tabs are saved on the disc after pressing the **F2** button (*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.adulayout* file.

Panels, together with all types of parameters, are another element of the interface. The device may be configured using these. To add a new parameter panel, press **F9** (or click on the + icon on the toolbar). A window with all available parameters will pop up.



The search can be accelerated by entering the desired option in the filter field (1).

Clicking on the selected panel will open it in the desktop. New panels are always displayed on the right side of the desktop. They can be moved by left-clicking on the title bar and dragging it to a new position. To remove a panel from the desktop, right-click its bar. A menu will appear from which the delete option (*Close panel*) must be selected.

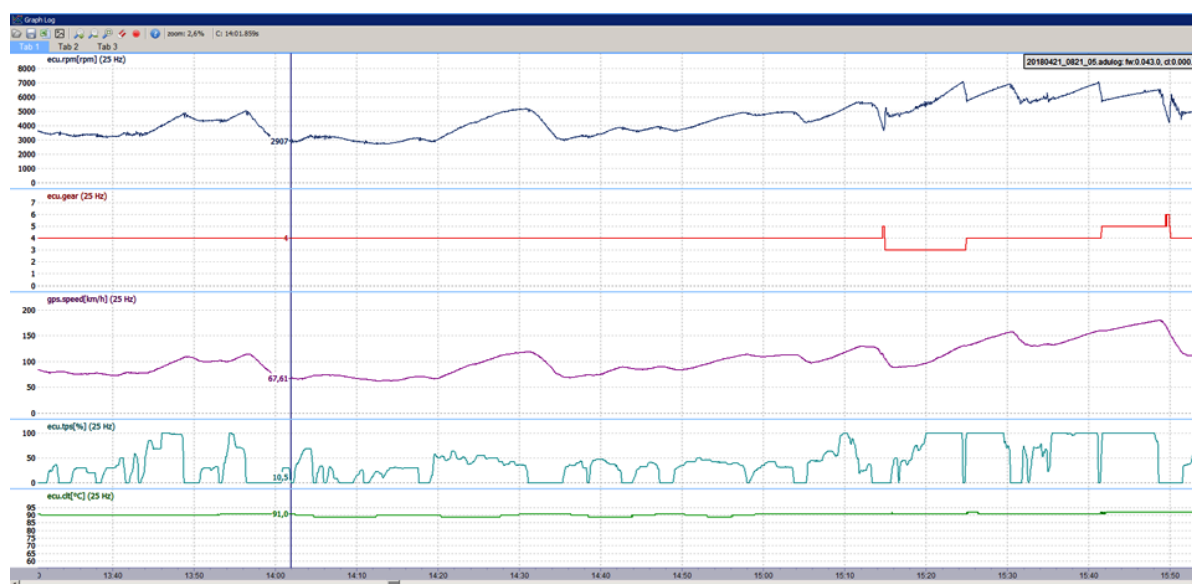
There are different panel types. The first type are the configuration panels discussed above. Another type are panels for previewing variables, such as the **Variables Inspector** (7), log channel preview (8) or the graphic log (9) showing the course of logging channels over time.

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.

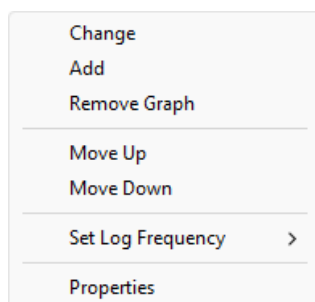
| Variables Inspector | | |
|-----------------------------|-------|------|
| Name | Value | Unit |
| a_fuelLevel1 | 0,0 | |
| a_zeroImuPitch | 0 | |
| a_acquireButtonDefinedTrack | 0 | |
| a_acknowledgeAlarm | 0 | |
| a_prevPage | 0 | |
| a_nextPage | 0 | |
| c_ecu_dbwPos | 0,0 | % |
| c_ecu_dbwTrgt | 0,0 | % |
| c_ecu_tcDrpmRaw | 0 | |
| c_ecu_tcDrpm | 0,0 | bar |
| c_ecu_tcTrqRdc | 0 | % |
| c_ecu_pitLTrqRdc | 0 | % |
| c_ecu_boostTarget | 0 | kPa |
| c_ecu_pwm1DC | 0 | % |
| c_ecu_atMode | 0 | |
| f_isGPSValid | 0 | |
| f_oil_alarm | 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. It is possible to display values of variables of a given class by opening the following windows: **Analog Inputs** (displays the analog input values), **Digital Inputs** (displays digital input values), **CANbus Message Objects**, **CANbus Inputs** (variables from CAN bus), **Tables** (values from the tables), **Numbers** (mathematical function values) or **Functions** (logical function values).

The logging channel **Graph Log** is another panel type. It can be used to display the desired channels and preview them using the time function.

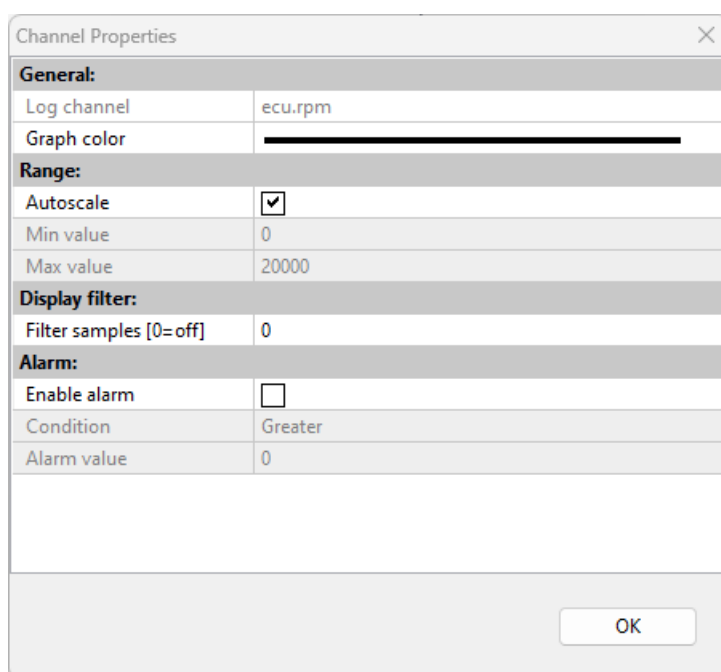


If you right-click the log area, the following menu will appear.



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**).

Just like the main desktop of the application, the log panel has tabs used to open different logging channel groups (e.g. engine, track, etc.). These tabs work exactly the same way as the main desktop tabs.



Channel Properties

| | |
|-------------------------|-------------------------------------|
| General: | |
| Log channel | ecu.rpm |
| Graph color | <div></div> |
| Range: | |
| Autoscale | <input checked="" type="checkbox"/> |
| Min value | 0 |
| Max value | 20000 |
| Display filter: | |
| Filter samples [0= off] | 0 |
| Alarm: | |
| Enable alarm | <input type="checkbox"/> |
| Condition | Greater |
| Alarm value | 0 |

OK

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 disk (**Open log**) - it is possible to append several logs for analysis (**Append data**) or replace the currently open logs with others (**Replace data**);
- saving the log file to disk (**Save log**);
- saving the displayed channels in files (**Export to text format**) with the following formats:

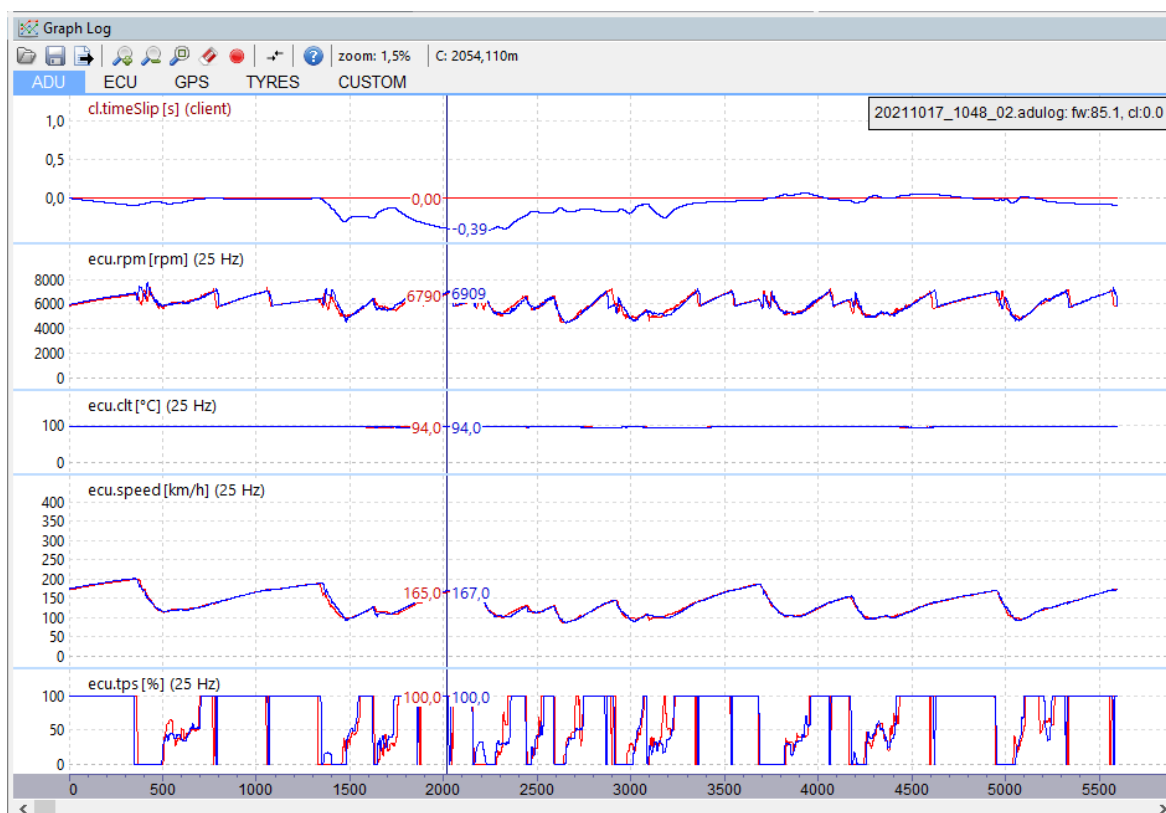
| | |
|----------------------|---|
| Export to CSV | <p>Export to a CSV file with the possibility of changing the settings:</p> <p>Export frequency: Same as channel– export frequency including actual logging frequency for individual channels 1Hz, 5Hz, 25Hz, 50Hz, 125Hz, 250Hz, 500Hz - selection of export 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).</p> <p>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.</p> <p>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.</p> |
| Export to PNG | Export to a PNG file |
| Export to VBO | Export to a VBO file (channels displayed in the log with the required channel package for the Vbox) |

- It is possible to save only the selected part of the log. Ensure that the log being saved (at the time of export) is not in **Compare laps** mode.
- (**Zoom In, Zoom Out, Zoom Extents**) change of scale;
- **Clear log**;
- **Pause / Resume log**;

- comparing data from two selected laps of the same track by overlapping them (**Compare laps**).

| | |
|---------------------|---|
| Enable | A checked tick box indicates that comparison mode is enabled. |
| Lap red | Lap selected for comparison - its graph is displayed in red on the <i>Graph Log</i> . |
| Lap blue | Lap selected for comparison - its graph is displayed in blue on the <i>Graph Log</i> . |
| X axis | The X axis is defined by the lap time Time or the distance from the start (track distance travelled) Distance . |
| Speed source | Calculations that take into account the speed of the vehicle are required to compare the two laps. Speed data can be taken from the ecu.speed channel (preferred option) or gps.speed . |

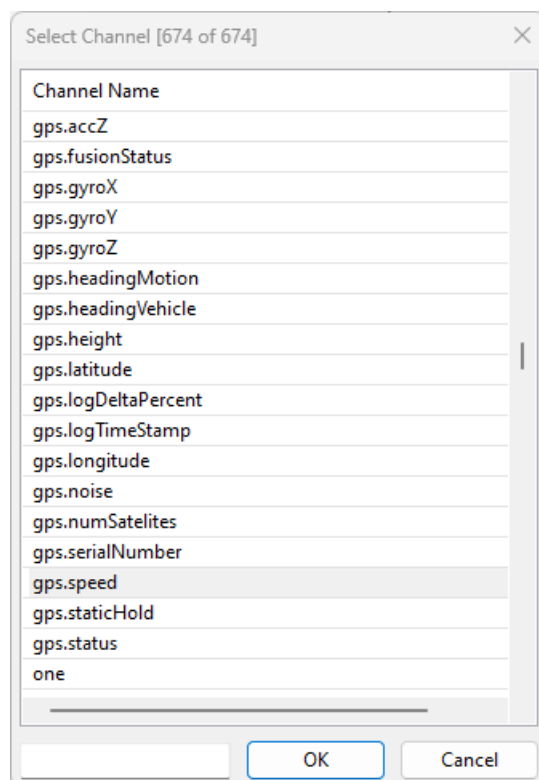
The **cl.timeSlip** channel is used to show the time differences (in seconds) of the laps being compared at each point on the track. The lap displayed in red is the reference point (marked as zero), while the lap displayed in blue takes on negative values when the section of the track is faster and positive values when it is slower.



The graphic log window 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 (increasing a log) |
| Down arrow or a | Increases the time base (decreasing a log) |
| Z | Adjusts the increase 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 **gps**, only channels containing the word **gps** 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 | Select specific elements from one project and integrate them into current project. For more details, see: Appendix D - How-to Merge Projects (on page 213) |
| Import log... | Import a log from a memory stick (SHIFT + F4) |
| Show full creen | Full screen mode. This increases the screen space available to the application (CTRL + F) |
| 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 <i>MyDocuments / ADU / DeviceName / QuickSave</i> directory (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: <i>MyDocuments / ADU / Default / desktops.adulayout</i> |
| Store desktops | Saves desktop configurations to the following file: <i>MyDocuments / ADU / Default / desktops.adulayout</i> |
| Open desktop templates... | Reads desktop configuration from a selected file. This allows to transfer configurations between PCs. |
| Save desktop templates... | Saves desktop configurations to a selected file. This allows to transfer configurations between PCs. |

| | |
|---------------------------------|--|
| Add new panel | Adds a new panel to the desktop (F9) |
| Replace panel | 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 + PGDWN) |
| Devices | |
| Device selector | If one or more ADU 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 ADU 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 ADU according to the current PC time. This time is used to date the files of a log saved into an external USB storage device. It can also be displayed on the screen of a device. |
| 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 |
| Send data to ADU | Sends data to ADU and restarts all functions. |
| Tools | |
| Texture manager dialogue | Displays a dialogue for managing the dashboard textures (graphics). You will find more information regarding texture management further in the manual. |
| Customize keys | Change the shortcut keys |

| | |
|------------------------------|--|
| Reset track data | Resets all times for a given track. It is also possible to delete track data using an external button connected to an ADU device. |
| Set meters | Sets the vehicle mileage |
| Analog sensor zeroing | Resets analog sensors |
| 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 window (SHIFT+F7). |
| Analog Monitor | Displays the analog channel monitoring window (SHIFT + F10). |
| Variables Inspector | Displays the variable monitoring window (SHIFT + F11). |
| Options | Displays a dialogue window with the application options 2D tables color – color map 2D 3D tables color scheme – color scheme for 3D maps Auto save logs – automatic saving of logs onto the disc Use mouse wheel to zoom on Graph Log – log scaling function using the mouse wheel |

4. 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: <ul style="list-style-type: none"> • USBtoCAN - ECUMASTER interface • PCAN-USB - Peak System interface • Kvaser - Kvaser interface |
| CAN1 status | The status of the CAN1 bus from the USB to CAN interface |
| CAN2 status | The status of the CAN2 bus read from CAN controller of the ADU display |
| USB logger state | Pendrive save status |

| | |
|-------------------------------|---|
| USB buffer usage | Information about the quality of the pendrive (from A to F) and the buffer status |
| Board temperature | Device temperature |
| Saving log in progress | Log auto-save status |
| FW Version | Firmware version |
| Device type | Device type - 5" or 7" |
| Used resources | The number of functions used |
| Tables | Memory available for 2D and 3D user maps |
| Names | Memory available for element names (<i>CANbus Inputs, Functions</i> 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 equipment. Or: CAN and CANH are interchanged. |
| Offline | The programme operates in Offline mode - there is no access to the CAN bus. |

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

Ecumaster devices connected to the CAN bus communicate with the PC via a CAN adapter: USBtoCAN, Peak or Kvaser.

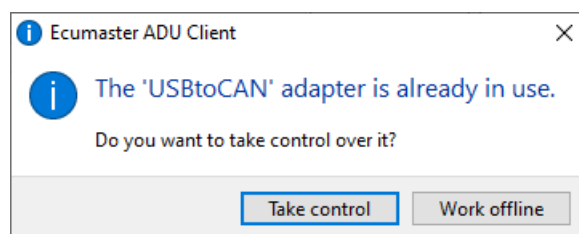
Ecumaster software used to configure the devices: ADU Client, PMU Client and Light Client.

From software version:

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

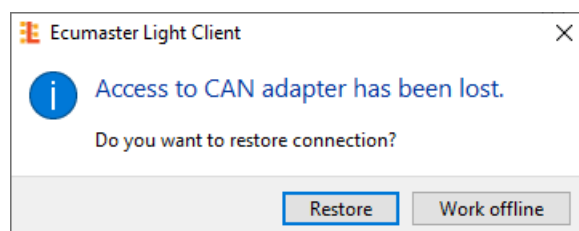
it is possible to switch easily between programmes without having to close one to open another.

When opening a programme (e.g. ADU Client) whilst another programme is already active (e.g. Ecumaster Light Client), a questions automatically appears on whether the new programme (ADU Client) should take control of the USBtoCAN adapter.



The programme that was in use (Ecumaster Light Client) will then lose connection and become inactive.

When the programme is called back to the foreground, a message will automatically appear indicating that the previously lost connection can be restored.



6. Pages

Pages are the basic element defining the image displayed. The ADU allows multiple pages to be defined (their number depends on their complexity and the amount of a device memory engaged). You can switch between pages using an external button, rotary switch or function (e.g. a different page is displayed when the car is stationary than when it is moving).

There are 3 types of pages:

- **Page** – a standard page.
- **Overlay** – a page which can be overlaid on another page.
- **Overlay with background** – a page which can be overlaid on any other page covering it up altogether.

Additionally, an alert or warning can be displayed irrespective of the page (more information in the *Alarms and Warnings* chapter).

6.1. Default settings pg_page1

In the ADU, you can use the prepared default settings stored as **pg_page1**.

This project defines pages, analog inputs, functions, CAN streams for emu black and PMU1 and alarms.

Analog inputs (*Analog inputs*):

- **A3** - fuel level sensor (**a_fuelLevel1**)
- **A4** - display tilt reset button (**a_zeroImuPitch**)
- **A5** - button marking the start/ finish line on a closed track (**a_acquireButtonDefinedTrack**)
- **A6** - alarm acknowledgement button (**a_acknowledgeAlarm**)
- **A7** - button to change the page to the previous page (**a_prevPage**)
- **A8** - button to change the page to the next page (**a_nextPage**)

Standard CAN Stream:

- CANbus Message Object for emublack (**m_emublack**)
- CANbus Message Object for PMU1 (**m_pmu1**)

Alarms:

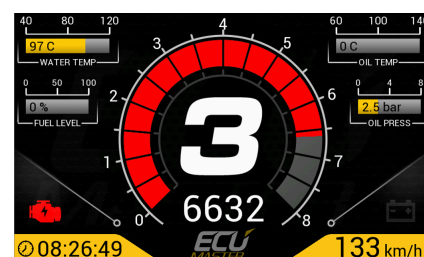
- **al_battery** - alarm indicating the state of charge of the battery with the engine running (if below 13 V a "Battery voltage too low" message will be displayed)

Functions:

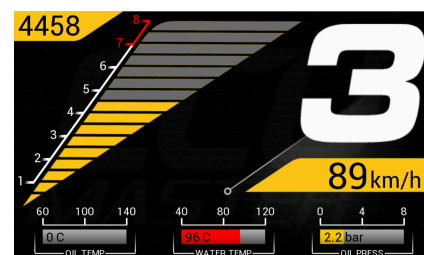
- **f_isGPSValid**- function used to change the color of the satellite icon located on the *pg_track* page depending on the quality of the GPS signal
- **f_oil_alarm**- function used to light up the oil pressure light located on *pg_trackSimple* when oil pressure drops below 1 bar
- **f_batt_alarm** - function used to light up the battery light located on *pg_generic* and *pg_trackSimple* when the charge drops below 12 V, and to light up the led
- **f_clt_alarm** - function that can be used to signal the coolant temperature alarm

Pages:

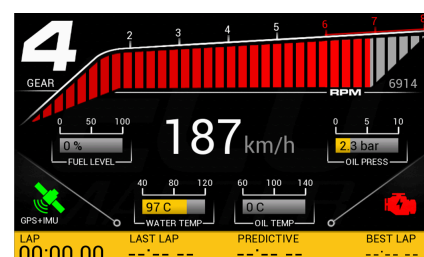
- **pg_generic** - general page



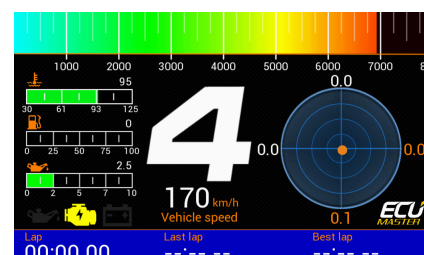
- **pg_generic2** - optional general page



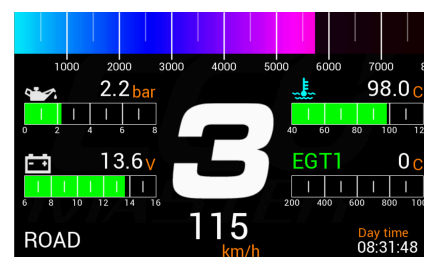
- **pg_track** - page prepared for track races, shows running times



- **pg_trackSimple** - optional page for track races



- **pg_rally** - page prepared for rallies



- **pg_pmu** - page displaying PMU information

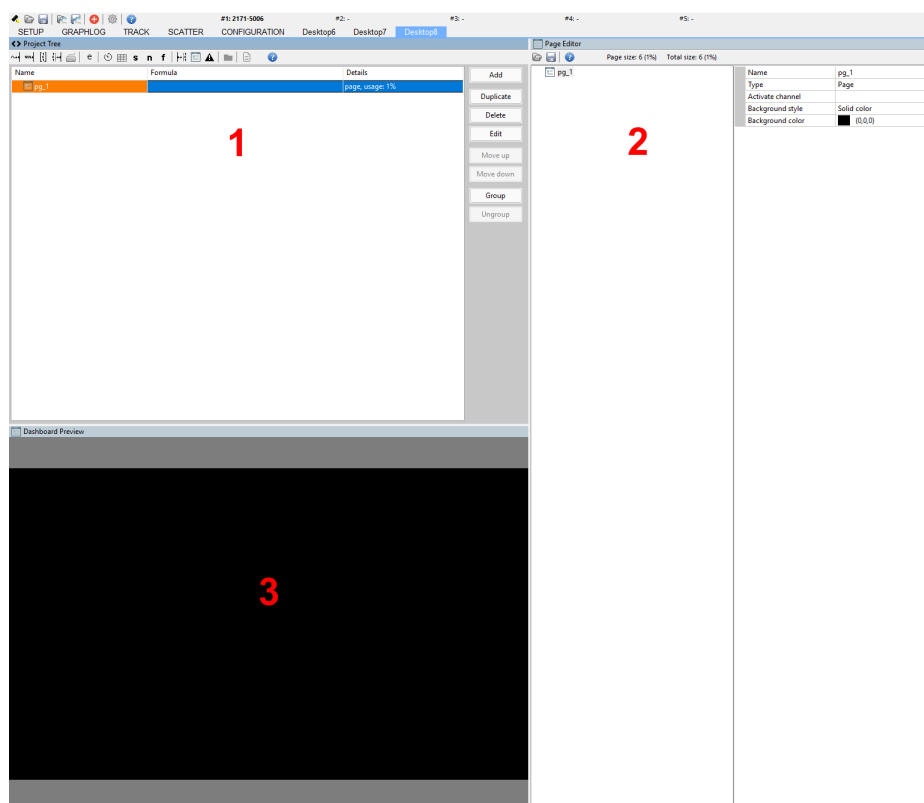
| PMU | | | |
|------------------------|-----|-----|------------------|
| TOTAL CURRENT: 15 A | O1 | ... | 0.0 A ON 13.9 V |
| | O2 | ... | 13.5 A ON 13.9 V |
| | O3 | ... | 0.0 A OFF 0.0 V |
| | O4 | ... | 0.0 A OFF 0.0 V |
| BATTERY: 13.9 V | O6 | ... | 0.7 A ON 13.9 V |
| | O7 | ... | 0.2 A ON 13.9 V |
| | O8 | ... | 0.0 A OFF 0.0 V |
| | O9 | ... | 0.0 A OFF 0.0 V |
| TEMPERATURE: 33 C | O10 | ... | 0.0 A OFF 0.0 V |
| | O11 | ... | 0.0 A OFF 0.0 V |
| | O12 | ... | 0.0 A OFF 0.0 V |
| | O13 | ... | 0.0 A OFF 0.0 V |
| STATUS: ON | O14 | ... | 0.0 A OFF 0.0 V |
| | O15 | ... | 0.0 A OFF 0.0 V |
| | O16 | ... | 0.0 A OFF 0.0 V |

- **ov_buttonDefinedTrack** - a page displayed for a short time after defining a new track, with information about the length of the track and setting the start / finish line



6.2. Creating a page

To add a page, click *Add* in *Project Tree* and select *Page*. A **pg_1** (1) page should appear in the project.

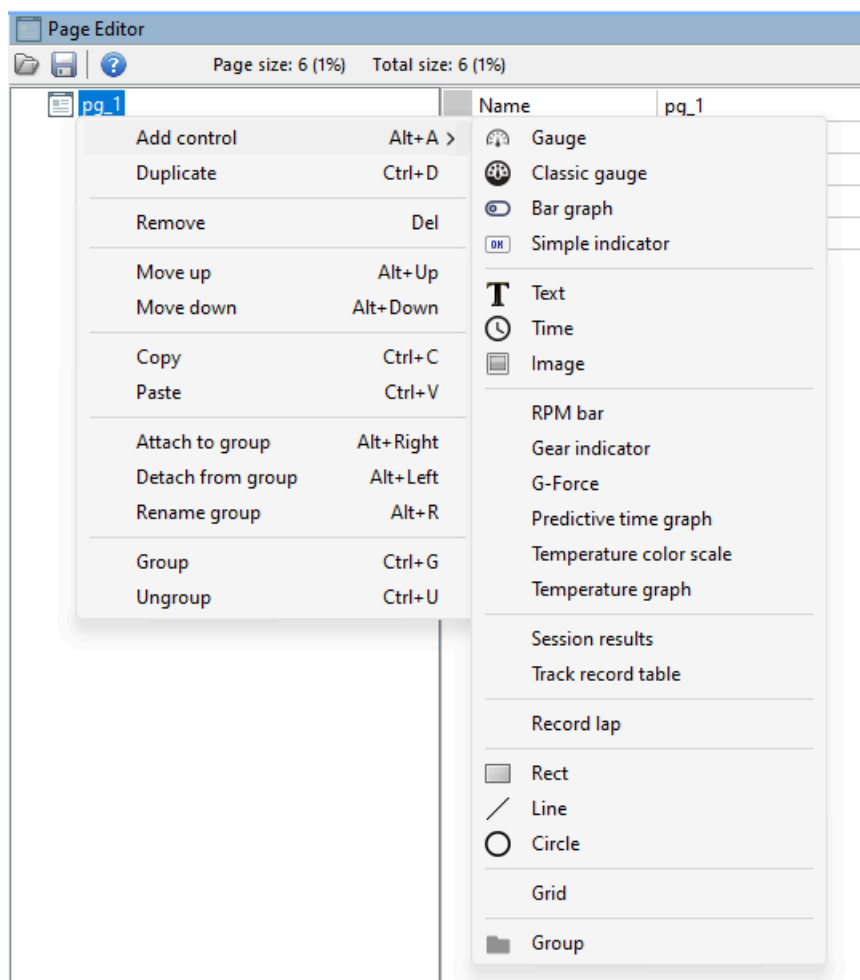


The currently selected page with its elements and configuration should appear in **Page Editor** (2). At this point we don't have any elements defined for an example page yet, but it is possible to set the following page attributes:

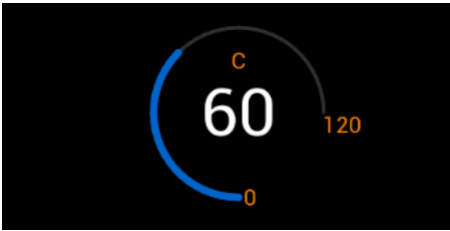

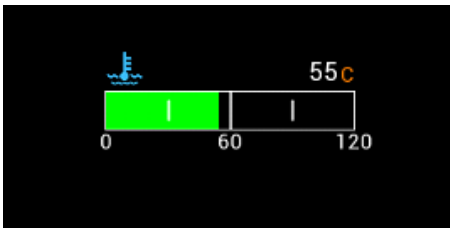

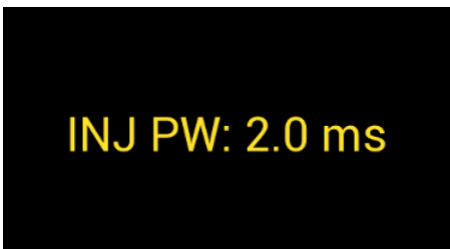
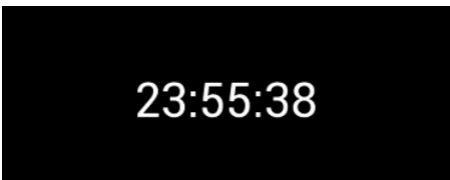

| Attribute | Description |
|---------------------------|---|
| Name | Page name in Project view |
| Type | Page type: <ul style="list-style-type: none"> • Page – a standard page • Overlay – a page than can be overlaid on another page • Overlay with background – a page similar to the Overlay page, but with a non-transparent background |
| Activation channel | Name of the channel or function that can automatically activate the page |
| Background style | Background style: <ul style="list-style-type: none"> • Solid color - Background filled with a defined Background color • Theme - graphic background, predefined in the device memory. |

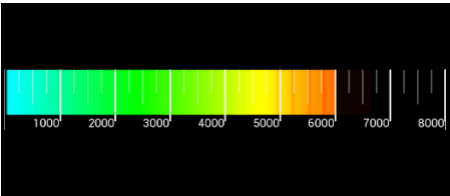

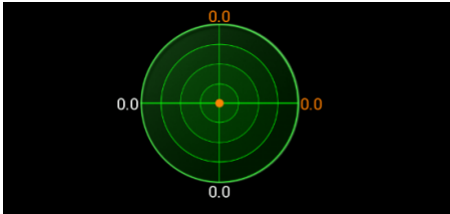
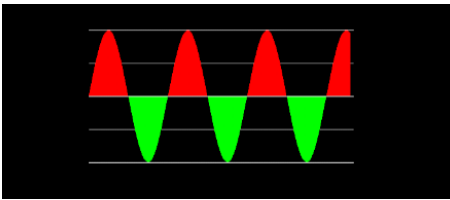
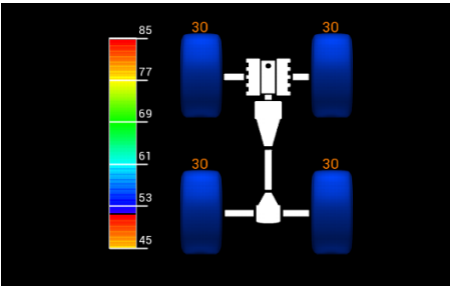
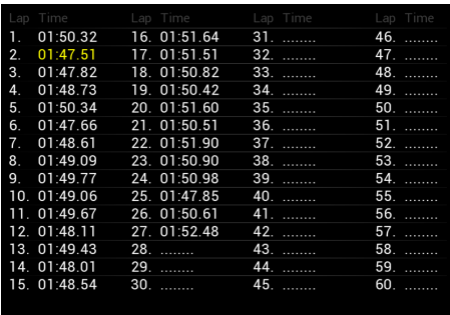
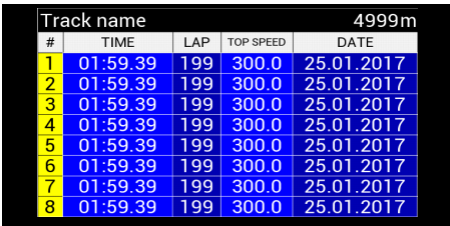
| Attribute | Description |
|-------------------------|---|
| | There are several types of backgrounds to choose from: Theme 1, Theme 2, Theme 3, Race track, Race track mirrored |
| Background color | The color of a page background |

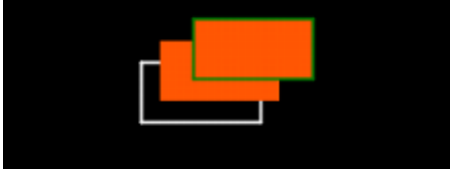

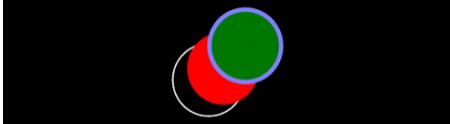
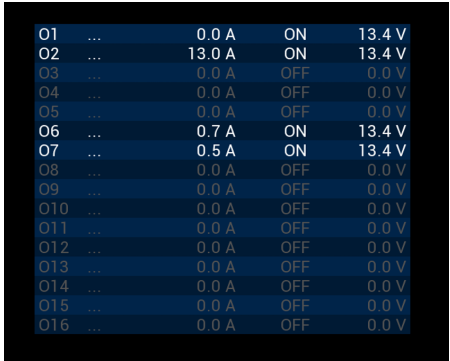
To add an element to a page right-click the page name in the page editor and select **Add control** and the desired element from the menu. You can also open the menu by pressing **Alt + A**.



6.3. Page elements

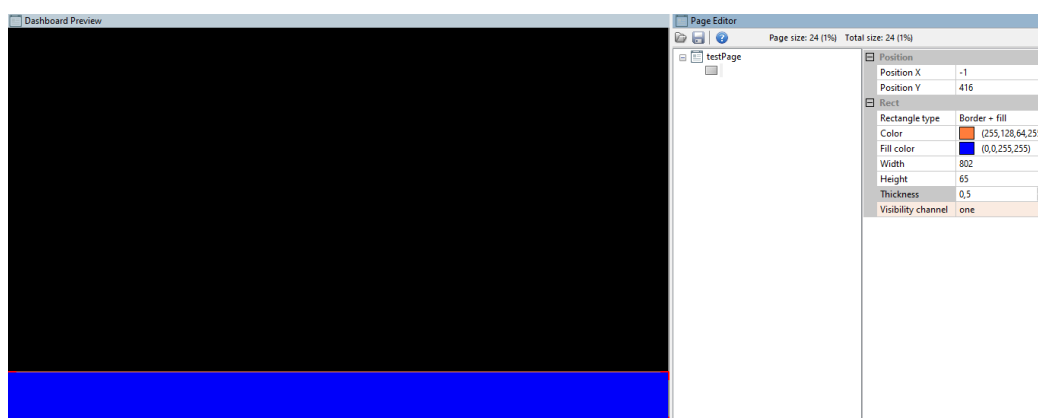
| Element | Description | Preview |
|-------------------------|--|---|
| Gauge | A round gauge to display a numerical value and its visualization using a circle segment. |  |
| Classic gauge | A classic gauge resembling a car gauge. It allows to display a numerical value, as well as to indicate it using a hand on its dial. |  |
| Bar graph | An indicator displaying values in the form of a moving bar (either horizontal or vertical). It is also possible to display an icon symbolising the measured value. |  |
| Simple indicator | It allows to display two texts in two different color sets that you can switch between using the log channel or functions. |  |
| Text | An indicator allowing to display text in different font sizes and colors. You can additionally attach a log channel (e.g. injection time, cooling liquid temperature, etc.). |  |
| Time | Allows to display formatted time. You can choose from the following times: real time, lap time, last lap time, best time and session time. |  |
| Image | Allows to add graphical elements such as background, logo, icon on a page. You can use the graphics available in the device or load one of your own from a file. |  |

| Element | Description | Preview |
|--|--|---|
| RPM bar | This indicator is dedicated for displaying the engine speed. There are four ways in which it can be displayed - a horizontal bar, a curved bar, a vertical triangular bar, and a round gauge. |  |
| Gear indicator | Indicates the currently selected gear and uses a special, enlarged font |  |
| G-Force | The g-force indicator uses an internal accelerometer of the device |  |
| Predictive time graph | A graph displaying the current difference between a lap time and the best time. Green indicates the better time, while red the worse one. |  |
| Temperature graph and Temperature color scale | Displays tire or brake disc temperature from thermal imaging cameras and indicates their maximum temperature. The indicator can also display the temperature in the form of horizontal bars with a temperature gradient (16 values for each tire). |  |
| Session results | It allows to display last 60 lap times, the best of which is highlighted in a different color. These times aren't saved in the device memory and reset after powering down. |  |
| Track record table | An indicator showing 8 best times scored on a given track. These times are saved in the device memory. |  |

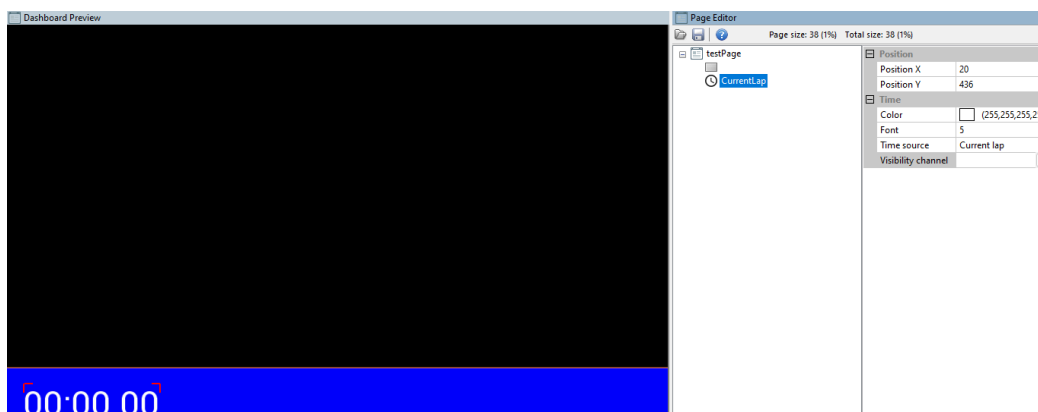
| Element | Description | Preview |
|---------------|---|--|
| Rect | Displays a rectangle in the form of a frame or a filled-in shape. |  |
| Line | Displays a line with the selected color and thickness. |  |
| Circle | Draws a circle of any color, with or without a filling. |  |
| Grid | A table displaying values defined by separate channels in each cell |  |

6.4. Adding elements to pages

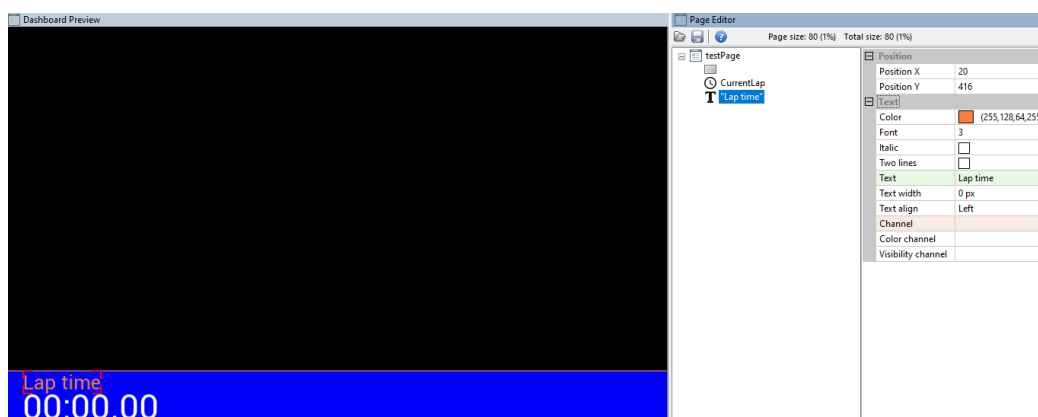
The following text will discuss the process of creating a sample page. Let's start with adding a **Rect** type object to a page. By clicking the object with the left mouse button, we can move it. If you are connected to an ADU device, the object will also move around on the display. You can change the **Rect** object parameters in the **Page Editor** window. After entering the parameters as shown in the image below, the page should look as follows:



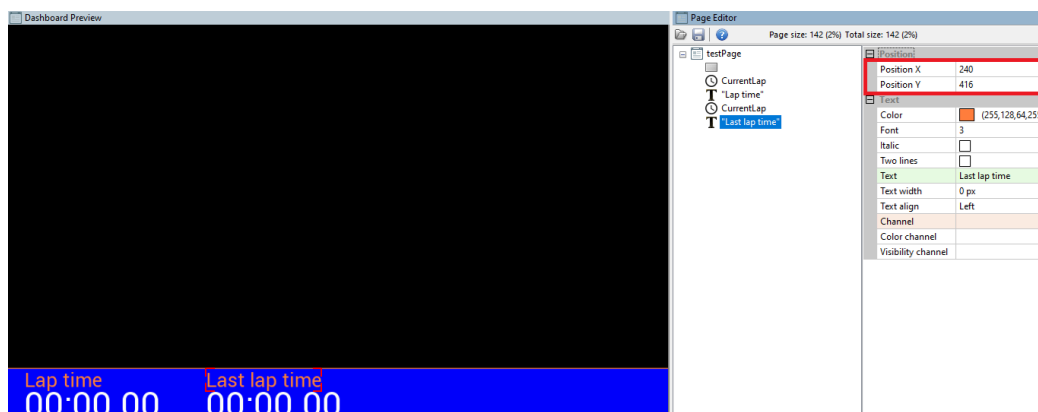
We will add information on the current lap time in the next step. To do this, a **Time** object should be added.



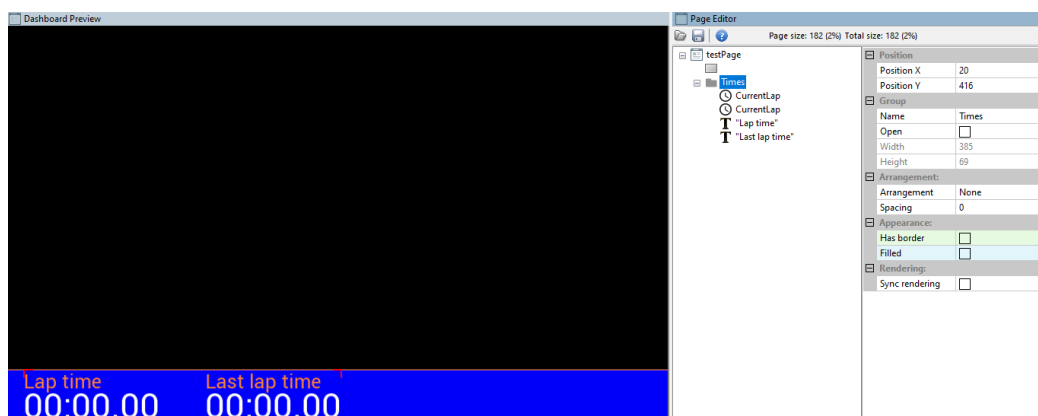
We will use a **Text** object to add a description explaining what the inserted time indicator relates to. You can enter the text to be displayed on the screen in the **Text** field. You can also choose its color. This object allows to display both the static text and the channel and variable values, which will be discussed in more detail further in the chapter.



It is possible to duplicate objects on a page. To do this, select an object in the page editor and press CTRL+D. In this example we will duplicate the *Current lap* and the *Lap time* objects to show the last lap time next to it.



To accurately set position on a page, you can enter values into the *Position X* and *Position Y* fields. Grouping objects is a very useful function. Thanks to it, it is possible to work on an entire group (e.g. by moving it around the page). It is also possible to automatically arrange the position of objects relative to each other (it will be discussed in more detail further on). To group objects you should select them in the editor and press **CTRL+G**. You can name a group to facilitate its further editing. In our example we will create a group out of text and times and let's name it *Times*.

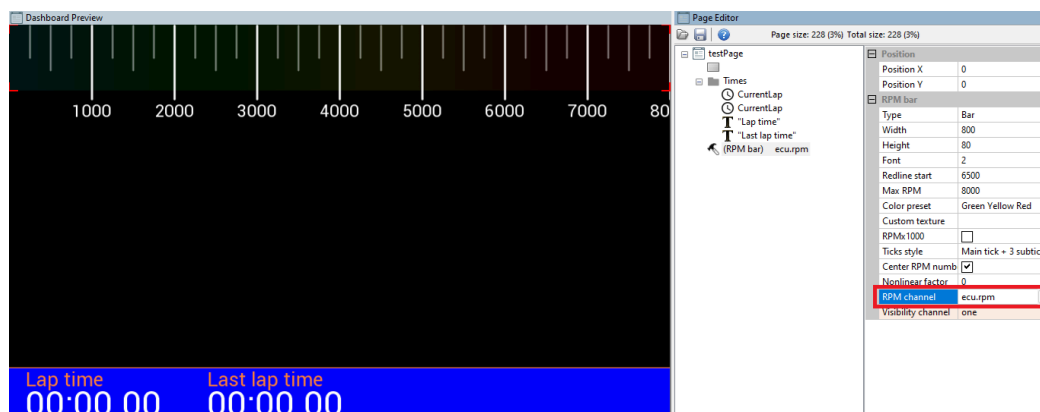


Another element that we will add is the engine speed indicator. To do this, the **RPM Bar** object should be added to a page.

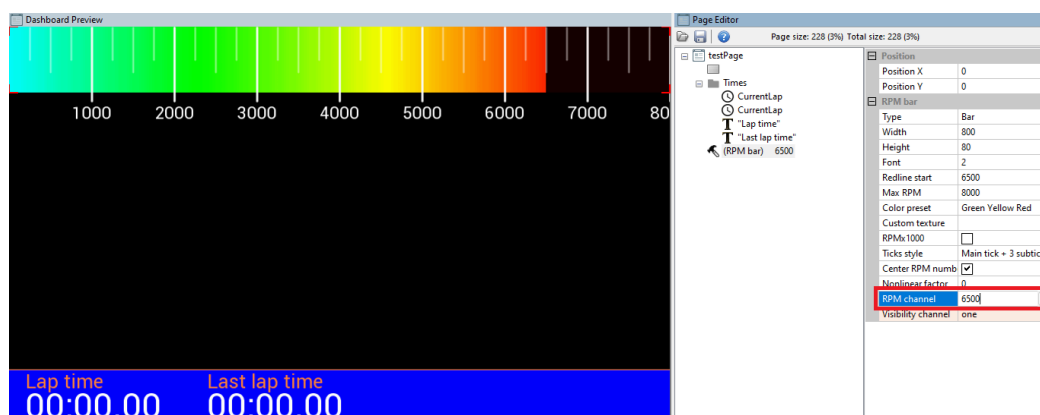
When adding an object that will display a value that changes over time, an important configuration element is the choice of channel that defines that value. This could be information from the CAN bus, a function, an analog or digital input value or a constant value. In the case of RPM, the information will come from the engine management sent via the CAN bus or serial communication. Most channels for ECUs available on the market are similar (such as RPM, lambda, coolant temperature, etc.), and they are represented in the ADU by *ecu.** channels.

Upon loading a CANX file suitable for the connected ECU or configuration of a serial connection, some **ecu.*** channels will display the values sent by the ECU. For more information about connecting engine control units available on the market see application notes. There is also information about the available log channels. If the ECU sends specific channels not included in the **ecu.*** channels, they are assigned unique names. They can be previewed in **Project Tree** or found in an application note. One of the parameters of the **RPM Bar** object is the **RPM channel** field. The name of the channel must be entered there. You can also click the '...' icon or press the **F4** button to open a window showing all available channels. Entered the '**ecu**' phrase in the search field will cause all available channels containing the word **ecu** to be display.

A set of **ecu.*** channels has the advantage that previously prepared pages will operate properly independently of the connected ECU (RPMs will always be **ecu.rpm**, the cooling liquid temperature will be **ecu.clt**, etc.).

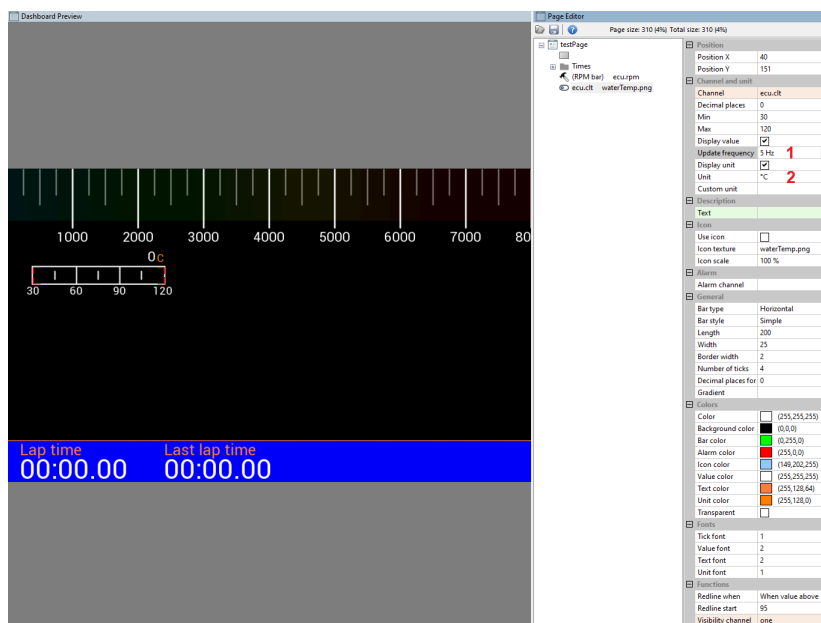


To test an indicator you can enter a numerical value (only integer values) in the **Channel** field. In our case, when we type the value 6500 in the **RPM channel** field, the engine speed indicator should look as in the picture below.

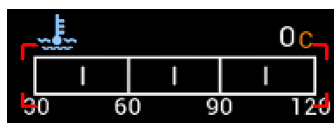
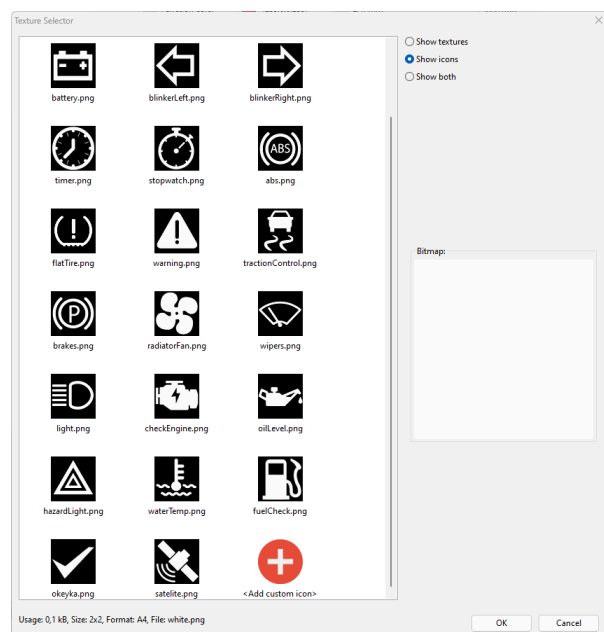


In the next step we will add a horizontal cooling liquid temperature (CLT) indicator to our page. For this purpose we will use the **Bar graph** control, which has much more options than those used so far. **Update frequency** (1) is a very important parameter. It limits the refresh rate of the value on the screen to a predetermined frequency. An ADU device refreshes the screen at a frequency of 50 Hz. The value of the channels which display quickly changing values (e.g. pulsating oil pressure) are very difficult to read as they change very frequently. Limiting the refresh rate allows you to impose a kind of filter on them. Measurement units are another important function. In our case, we want to display temperature in degrees Celsius. If we change the units to Fahrenheit in the **Units** field, the displayed values will immediately be converted accordingly. Another important function is "redline" (3), which allows to change the indicator appearance when a predetermined value is exceeded. In the **redlineWhen** field you can select the conditions, while in the **redlineStart**

field - the value at which the appearance of the indicator will change. The “redline” appearance is defined in the **colorRedline** field.

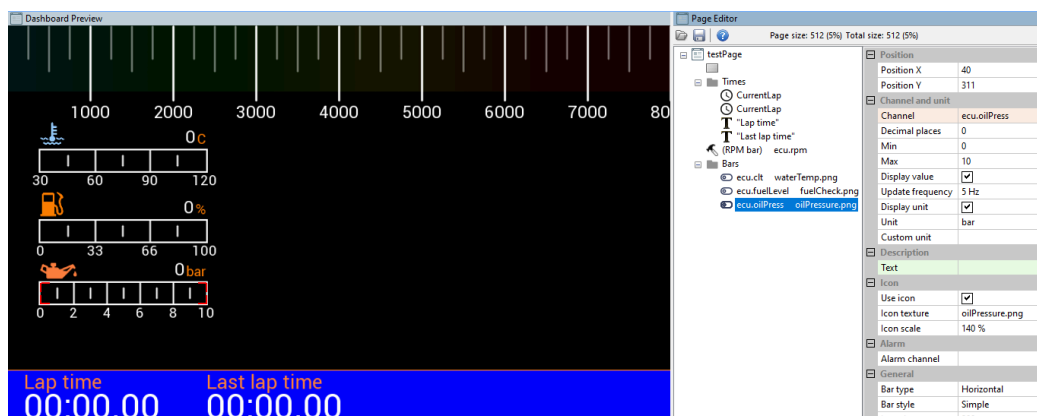


It is possible to display an icon in the **Bar graph indicator field**. To that end, select the **Use icon** option and select the desired icon from the **Icon texture** field (by clicking the ‘...’ icon). A graphics selection window will pop up from which you will be able to select or load your own icon (you can find more information on icons further in the manual). After selecting the *waterTemp.png* icon the indicator will look as shown below.

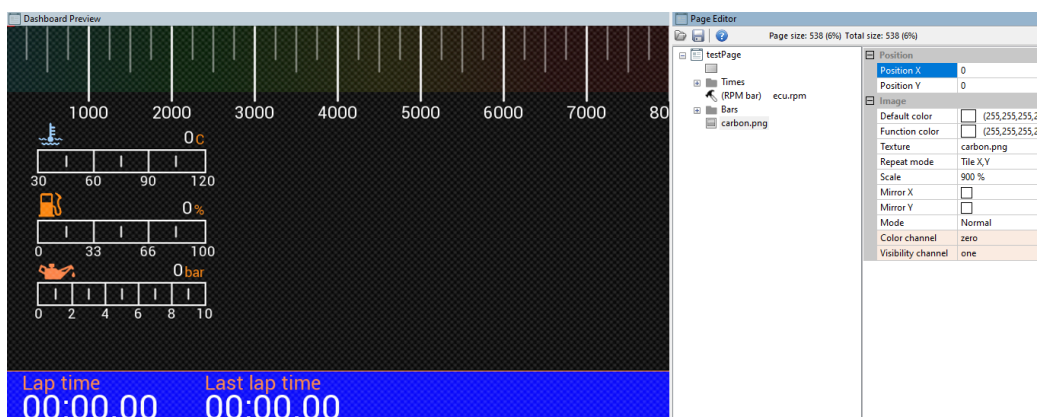


In the next step we will copy the created indicator twice in order to create a fuel level and an oil pressure indicator. After copying, set the new position of the indicator, select the appropriate

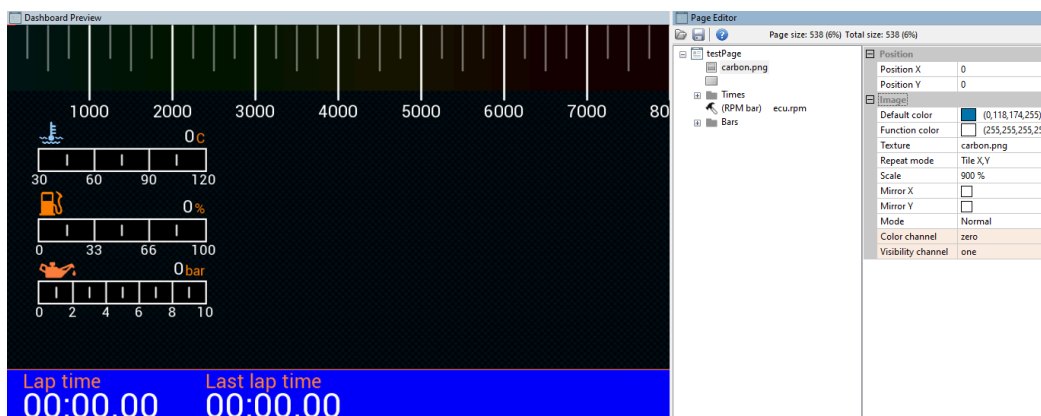
icons and assign the appropriate channels in the Channel field. These will be **ecu.fuelLevel** for the fuel level and **ecu.oilPress** for the oil pressure, respectively. It is also worthwhile to group the indicators to facilitate their editing in the future. After copying the page should like as below.



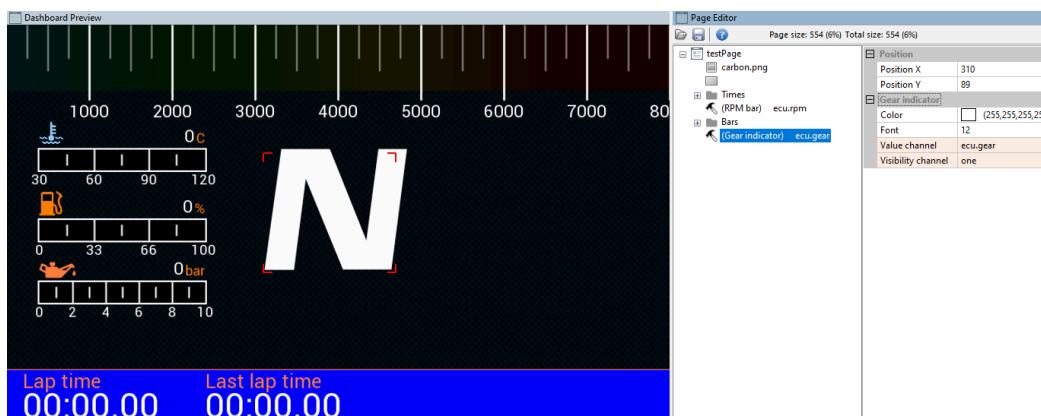
It is also possible to add background texture. For the purposes of our example we will create a carbon fibre-style background. To do this, use the **Image** object, and then select the *carbon.png* texture for it. To cover the entire surface with it, select the **Repeat mode Tile X, Y** option and set **Scale** to 900% to spread the texture over the entire screen. Another option to create a background is to select a predefined background in the page configuration (**Background style**).



Note that our texture is displayed “over” the existing elements. To move it “under” it should be placed as the first element of the page. The order of drawing the elements is determined by their order on the list. To change the object position on the list you should select the element and then move it using the combination of ALT + up arrow or ALT + down arrow. Let's also change the background color to dark blue.



To insert the indicator for the currently displayed gear, use **Gear indicator**. You should select a channel (**ecu.gear** in our case) and a font size. The indicator for the currently selected gear comes with a special font for displaying large digits.

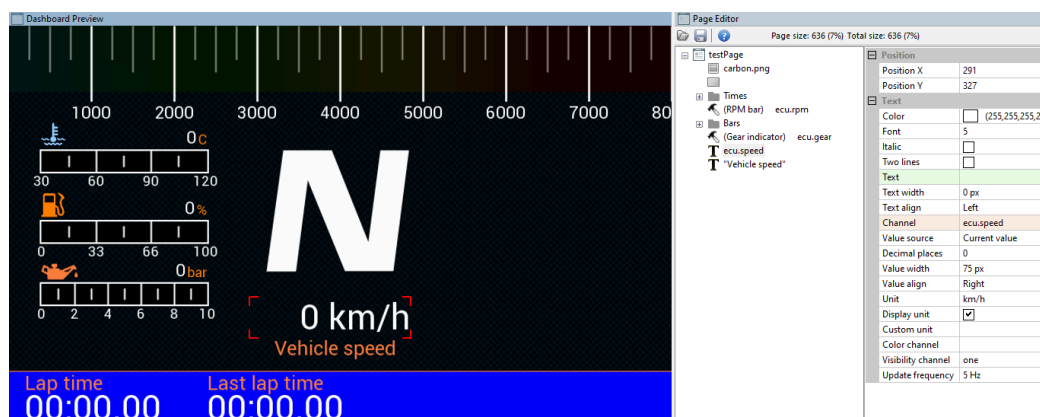


Under the gear indicator we will now display a numerical value representing the speed of the vehicle.

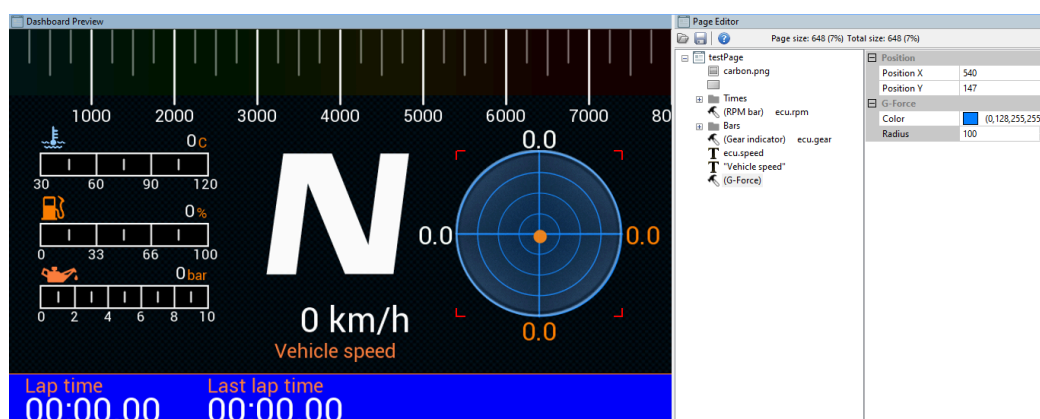
| | |
|----------------|-------------------------------------|
| Channel | ecu.speed |
| Value source | Current value |
| Decimal places | 0 |
| Value width | 75 |
| Value align | Right |
| Unit | km/h |
| Display unit | <input checked="" type="checkbox"/> |
| Custom unit | |

We will add the **Text** indicator for this purpose. In addition to displaying static texts, it allows to display the values of the channels and the variables. Enter the name of the channel or a variable in the **Channel** field. When a channel is selected, its unit will be shown, which can also be displayed together with a numerical value. Defining the maximum width (in pixels) of the displayed value is very important when it comes to formatting text. In the example below, the speed will be displayed in km/h. To prevent the text from shifting after changing the numerical value (e.g. From 7 km/h

to 11 km/h) we define the text width of the displayed value at 75 pixels. To change the unit from km/h to mph, select it from the **Unit** field. The conversion of the value to adjust it to the selected units will be done automatically. In the example below, let's also add a description under the speed explaining what the indicator shows.



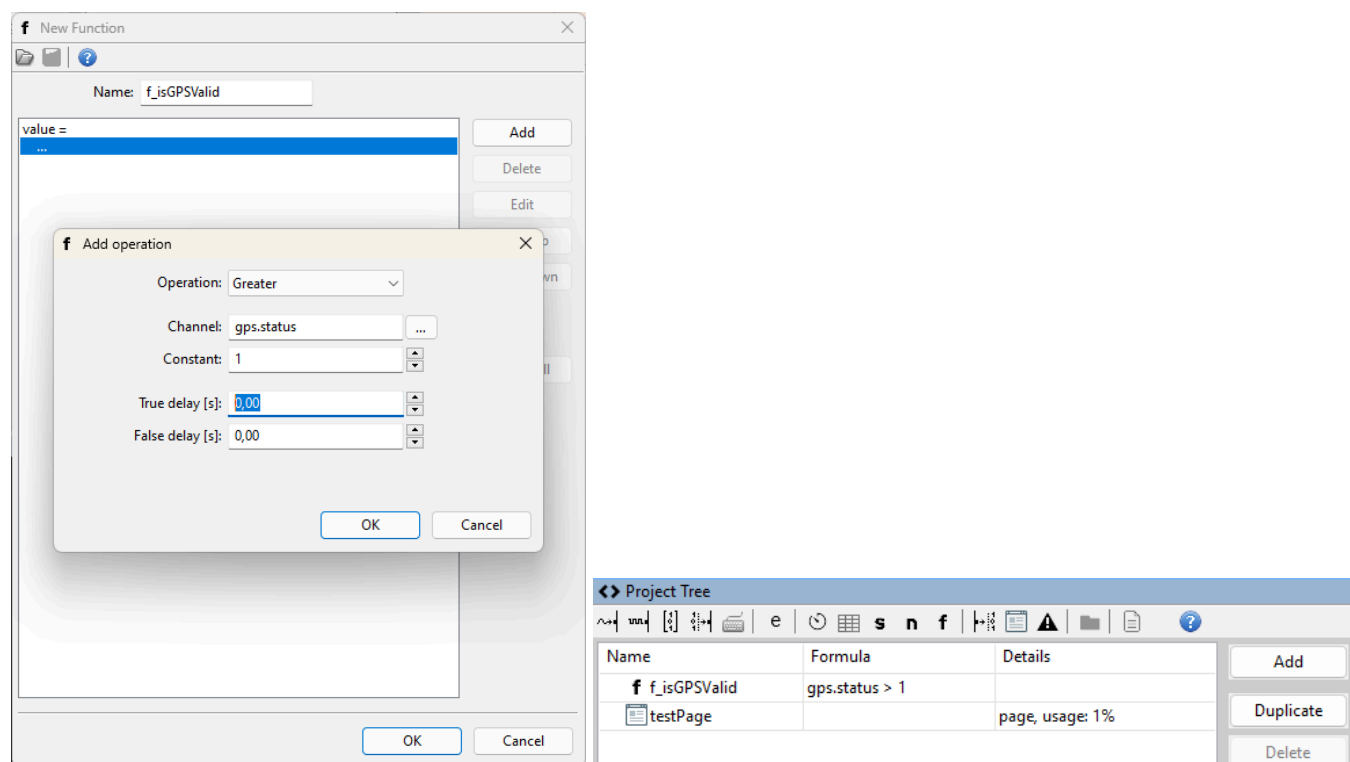
Another element that we will place on the page will be a G-force indicator (**G-Force**). It shows the current G-force acting on the vehicle. The ADU is equipped with a built-in accelerometer which, when installed in a vehicle, may require calibration (setting the 0 g point). You will find more information about the accelerometer further in the manual.



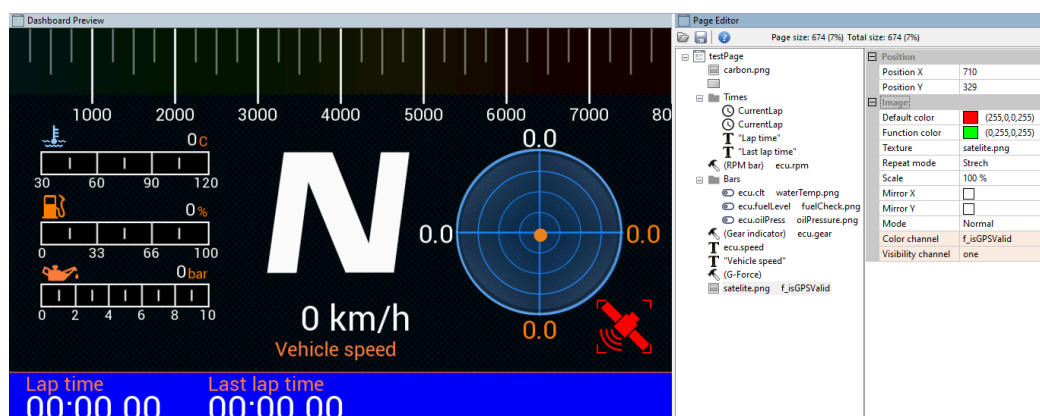
When using a GPS module, it is worth ensuring that the user can see its status. To this end, we will build a GPS status module. We will use two channels: *gps.status*, showing the mode of operation of the GPS, and *gps.numSatellites*, showing how many satellites the module is currently using.

To create the indicator, we will first use the satellite icon (*satellite.png*) and place it on the page using the **Image** indicator. The indicator can also be used to change colors depending on the channel / function assigned. Let's set red as the default color when the GPS is not working and green when the GPS is working correctly. We also have to create a function that will check if the value of the *gps.status* channel is greater than 1. 0 value means that the GPS is disconnected, and value 1 means that it is not synchronized with the satellites. Value 2 means that the GPS has

fixed the position. We add the function in the **Project Tree** using the **Add** (function) button. Let's call it *f_isGPSValid* and add a condition using the **Add** button. In the **Operation** field we choose **Greater**, and in the **Channel** value - *gps.status*. Value to be compared (**Constant**) is set as 1. In this way, we have created the condition $f_isGPSValid = gps.status > 1$. You can find more information regarding the functions further in the manual.

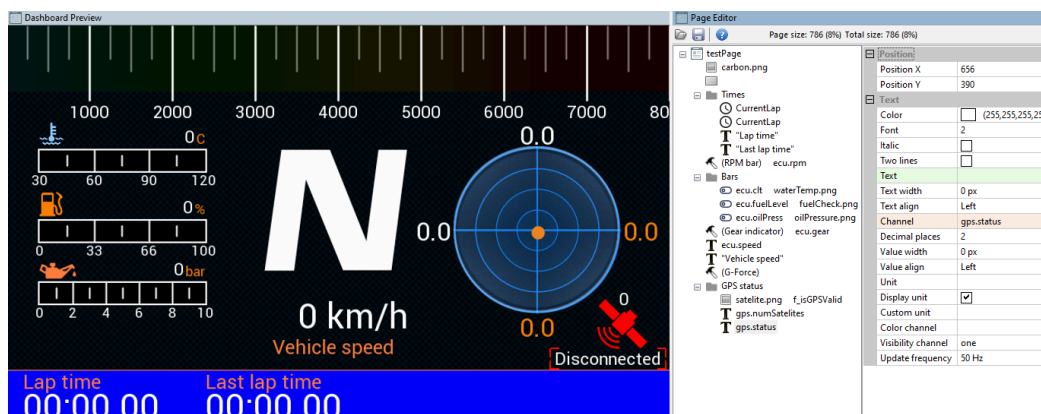


Enter *f_isGPSValid* in the **ColorChannel** field of our satellite icon. We can also test the color change option by entering there the value 0 (basic color) or 1 (alternative color).



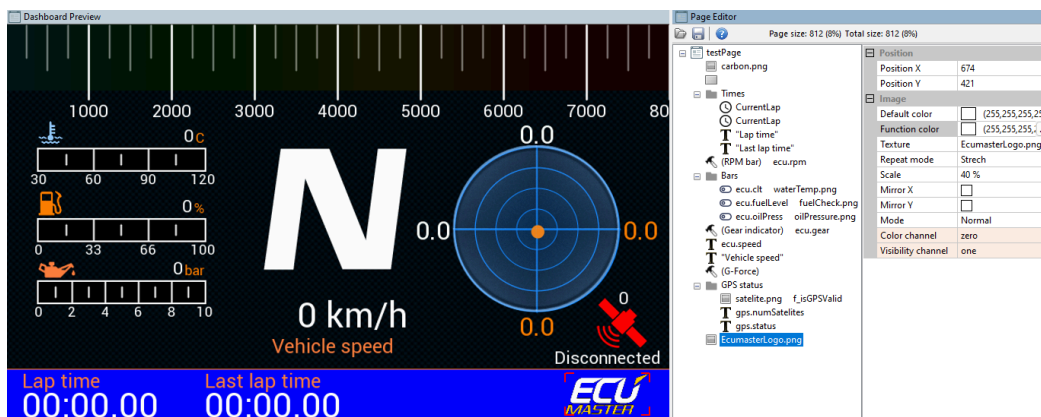
We will place two text fields over and under the icon. The upper field will indicate the number of satellites (*gps.numSatellites* channel), while the lower field will display the current GPS module status. In the case of the *gps.status* channel, the text field will automatically transform its value

into text (e.g. disconnected, gps 3D, etc.). We will additionally group all elements of indicators into the GPS status group.



The last element to be added to the page will be the company logo. In the example we will use the logo of Ecumaster embedded in the device, however, you can add your own graphics and

display the logo of your own company. You can find more information about this further in the manual. In order to add graphics, select the **Image** indicator and then the desired logo (in the **Texture** field).



6.5. Grouping objects

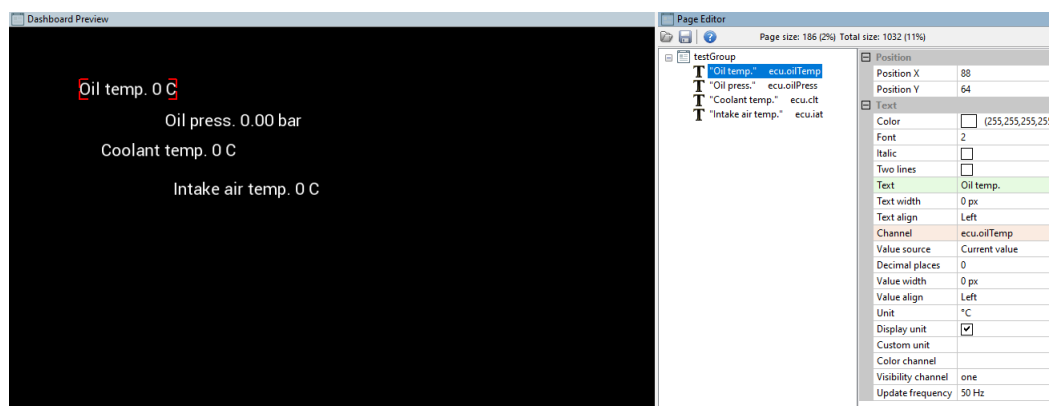
The **Group** object has two functions. It makes it possible to group objects for ease of management, and also makes it possible to automatically arrange objects within a group. All group object options are shown on the adjacent figure. Some settings are hidden by default and appear after activating a given option (e.g. when **Has border** is selected)

| | |
|------------------------|-------------------------------------|
| Position | |
| Position X | 100 |
| Position Y | 100 |
| Group | |
| Name | |
| Open | <input type="checkbox"/> |
| Width | 0 |
| Height | 0 |
| Arrangement: | |
| Arrangement | None |
| Spacing | 0 |
| Appearance: | |
| Has border | <input checked="" type="checkbox"/> |
| Border color | (141,141,141,255) |
| Border thickness | 1 |
| Filled | <input checked="" type="checkbox"/> |
| Fill color | (0,37,74,255) |
| Alternating fill color | <input checked="" type="checkbox"/> |
| Fill color #2 | (0,27,53,255) |
| Rendering: | |
| Sync rendering | <input type="checkbox"/> |

| Parameter | Description |
|----------------------|--|
| Position X, Y | Group position on the page Changing the position value affects the position of all objects in the group. |
| Name | Group name |
| Open | A group that is open (<i>Open</i> box checked) allows manipulating the position of the sub-objects on the preview screen. When a group is closed, it is only possible to move the entire group. |
| Width | Shows the width of the entire group (in pixels). |
| Height | Shows the height of the entire group (in pixels). |
| Arrangement | Parameter enabling automatic arrangement of group elements: <ul style="list-style-type: none"> • None – no arrangement of elements • Horizontal – horizontal arrangement of elements • Vertical – vertical arrangement of elements |
| Spacing | Additional distance between group objects being arranged (in pixels) |
| Has border | Enables a frame to be drawn around the group. |
| Border color | Frame color around the group |

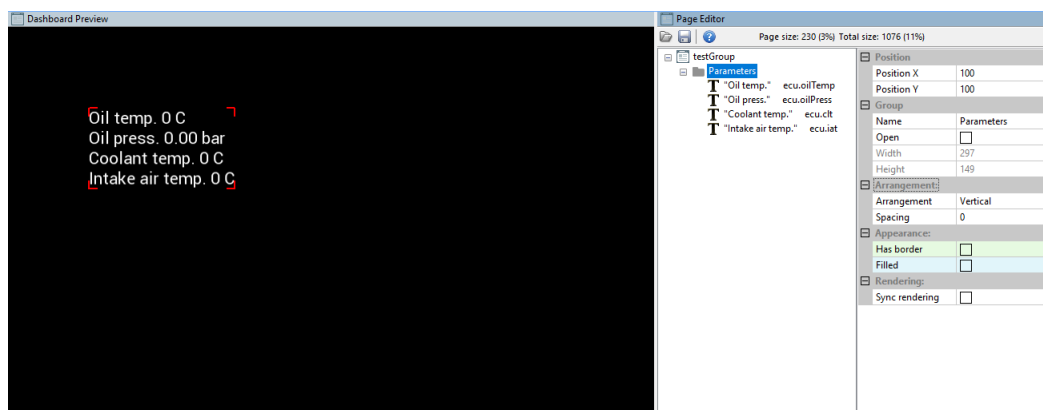
| Parameter | Description |
|-------------------------------|--|
| Border thickness | Frame thickness in pixels |
| Filled | Enables the filling of fields created when arranging group elements. |
| Fill color | The fill color |
| Alternating fill color | Enables an alternative fill color. The fill will alternately use the color defined in Fill color and in Alternating fill color . |
| Fill color #2 | Alternative fill color |
| Sync rendering | |

To demonstrate the possibilities of arranging elements using a group, we will create an example table displaying engine performance. In our example, we will display values for oil pressure (*ecu.oilPress*), oil temperature (*ecu.oilTemp*), coolant temperature (*ecu.clt*) and intake air temperature (*ecu.iat*).

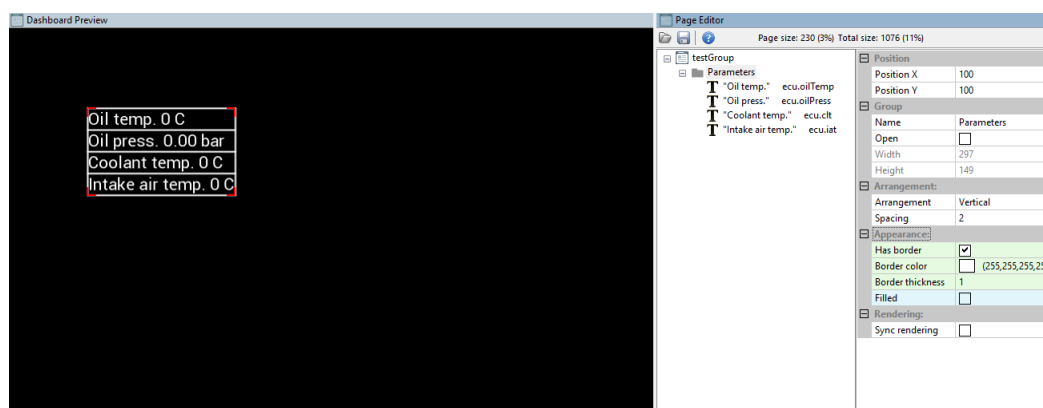


| | | | |
|--|---|--|---|
| Position Position X 88 Position Y 64 Text Color (255,255,255,255) Font 2 Italic Text Oil temp. Text width 0 Text align Left Channel ecu.oilTemp Value source Current value Decimal places 0 Value width 0 Value align Left Unit °C Display unit <input checked="" type="checkbox"/> Custom unit Visibility channel Update frequency 50 Hz | Position Position X 194 Position Y 103 Text Color (255,255,255,255) Font 2 Italic Text Oil press. Text width 0 Text align Left Channel ecu.oilPress Value source Current value Decimal places 2 Value width 0 Value align Left Unit bar Display unit <input checked="" type="checkbox"/> Custom unit Visibility channel Update frequency 50 Hz | Position Position X 115 Position Y 140 Text Color (255,255,255,255) Font 2 Italic Text Coolant temp. Text width 0 Text align Left Channel ecu.clt Value source Current value Decimal places 0 Value width 0 Value align Left Unit °C Display unit <input checked="" type="checkbox"/> Custom unit Visibility channel Update frequency 50 Hz | Position Position X 204 Position Y 188 Text Color (255,255,255,255) Font 2 Italic Text Intake air temp. Text width 0 Text align Left Channel ecu.iat Value source Current value Decimal places 0 Value width 0 Value align Left Unit °C Display unit <input checked="" type="checkbox"/> Custom unit Visibility channel Update frequency 50 Hz |
|--|---|--|---|

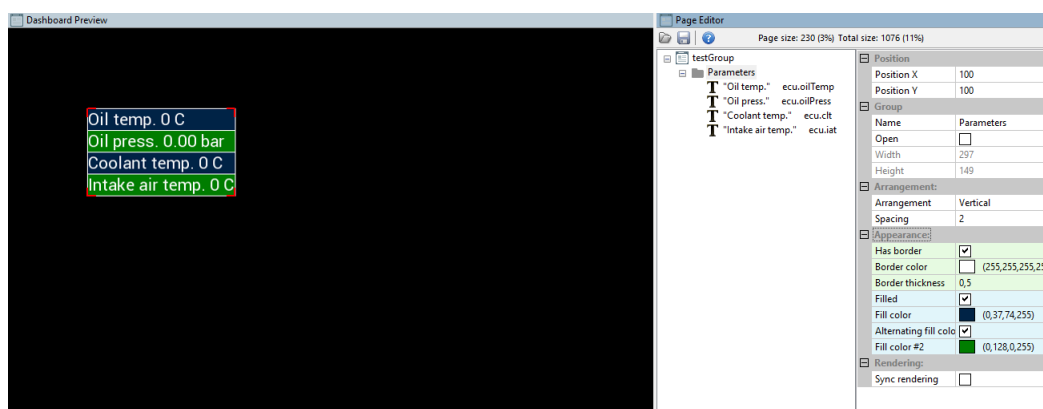
Let's select all 4 text objects and choose **Group** (CTRL+G). In the group options, let's set the name to 'Parameters' and in the **Arrangement** option select **Vertical**. This will automatically arrange the objects along the vertical axis. Using the **Spacing** parameter, we can increase the distance between elements. The page should look as follows.



As you can see, the text has been tidied up, but it is still not readable because the displayed numerical values are in different places. To solve this problem, set a fixed text width in all **Text** objects, creating two columns: one for the description and one for the value. To do this, select all four **Text** objects and enter 160 in the **Text width** field. This will change the width in all selected objects. In addition, let's set the **Has border** parameter in our group and select white as border color.



Using the **Filled** option, we can change the background color of our 'table'.

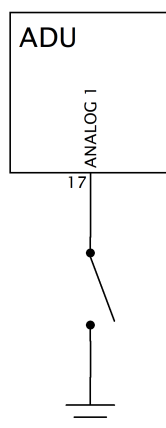


To remove a group from the page (leaving the objects), select it and choose **Ungroup** (CTRL+U) from the context menu (right mouse button). To exclude objects from a group, select them and choose **Detach from group** from the context menu (ALT+Left arrow). To attach an object to an

existing group, move the object in the hierarchy under the group of interest and then select **Attach to group** (ALT + Right arrow).

6.6. Toggling between pages

There are two methods for toggling pages. The first one employs a button than can be connected directly to the ADU (using analog or digital inputs) or to another device (e.g. a CAN switch board) and then sent to ADU using the CAN bus.



The buttons that will be used to toggle pages (*Next page* and *Previous page*) are defined in the *Buttons* panel.

A second method of toggling pages is to use a rotary switch (*Rotary switch*). You can define a function and assign it to the Activation channel attribute for each of its positions.

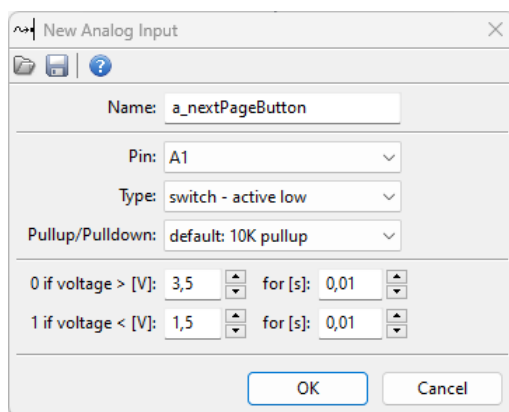
Page toggler (*Rotary switch*) can be defined in the *Configuration > Changing pages > Page selector channel panel*.

Page selector takes control of the keys *Next page* and *Previous page* defined in the *Buttons* panel.

The following example shows how to connect and configure a page toggle button. The button is connected to the analog input Analog #1 and shortened to ground.

In the *Project Tree* window the button must be defined as an analog input. To do this, press **Add** and then select *Analog Input*.

The following window should pop up:

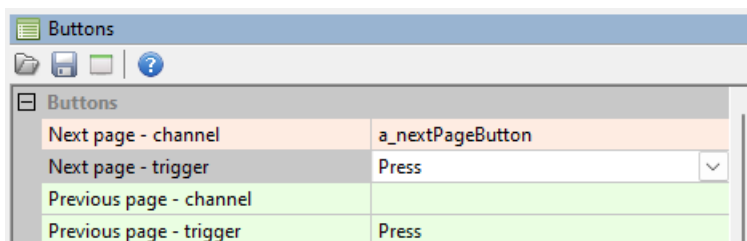


Name - determines the name of the input that will be visible in the project. In the Pin field define the analog input to which the button is connected (Analog #1 in our case). Choose *Switch - active low* (which means that the button is activated at a low state). You also need to connect a 10K pull-up resistor.

To preview the button state, open the *Analog Inputs* panel, in which it is possible to track all *Analog Inputs*). The value of the button should be 0 for an unpressed button and 1 for a pressed button.

| Analog inputs | | |
|------------------|-------|------|
| Name | Value | Unit |
| a_nextPageButton | ? | |

A button defined in this way should be assigned in the *Buttons* panel to the option of toggling a page to the next one (*Next page - channel*). It should be noted that when the button on the last page is pressed, the view will go to page one. You can use the buttons only to switch between the Page type pages. *Overlay* pages are ignored.



Procedure for connecting a second button and assigning it to the *Previous page* switch functions the same.

Another method for toggling between the pages is to use the *Activation channel* page attribute. It is available after selecting a page in the *Page editor* together with the attributes such as page name, background color or type.

Overlays are the main application of this type of toggling. They make it possible to overlay (another page) over the displayed page whenever another event occurs. To make use of this possibility, a function must be created that activates the overlay. It will be visible as long as the function result differs from 0.

6.7. Startup screen

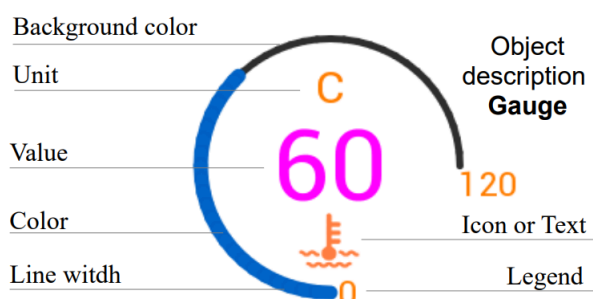
A user can create a startup screen that will appear immediately after switching the device on and will be displayed for a time defined by the user. To do this, it is required to configure the *Startup Screen* parameters in the *Configuration* panel.

| Parameter | Description |
|--------------------------------|---|
| <i>Enable</i> | It activates the startup screen. |
| <i>Texture</i> | The texture displayed on the startup screen (centred) |
| <i>Scale</i> | The scale of the displayed texture |
| <i>Duration</i> | Time during which the start-up screen will be displayed |
| <i>Color</i> | The color of the displayed texture |
| <i>Background color</i> | The color of the background |

7. Graphic objects

7.1. Gauge

This object allows to display a numerical value using a circular segment. You can also add a numerical value, a description, an icon and the units in which values will be expressed to the gauge.

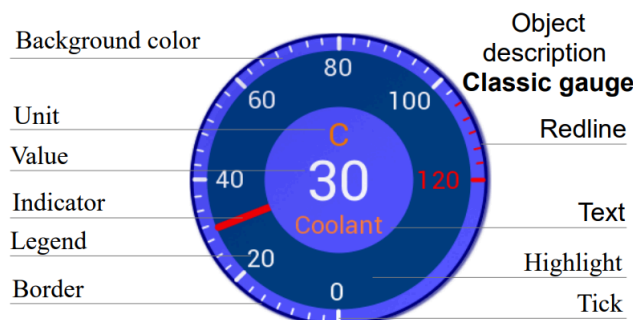


| Parameter | Description |
|-------------------------|---|
| Position X,Y | Object position on the page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Channel | Name of a channel or a variable that will be displayed in the Value field and displayed on the gauge. In this field you can also enter a numerical value without a decimal separator to test the operation of the gauge. |
| Decimal places | The number of decimal places displayed for Value and Legend |
| Min, Max | The minimum and maximum value displayed by the indicator |
| Display value | This parameter allows to hide the Value displayed |
| Update frequency | The update frequency on the Value parameter screen. The screen is updated 50 times per second (50 Hz). With fast-changing variables, reading is very difficult at this frequency. The parameter allows to decrease the update frequency (e.g. to 5 Hz), which makes the Value parameter easier to read. |
| Display unit | Option for displaying units (Unit). The units will be displayed above the Value parameter. |
| Unit | Selection of in which the value will be presented (e.g. kPa, Bar, Psi). You can define your own units in the Custom unit field. To this end, select User in the Unit field. |

| Parameter | Description |
|-------------------------|--|
| Custom unit | This field is used for entering a user-defined measurement units. To use it, select User in the Unit field. |
| Text | Defines the text displayed below the Value parameter. |
| Use icon | Parameter for displaying an icon instead of text |
| Icon texture | Allows to select an icon from the texture menu. You can add a custom icon (more information about managing textures can be found further in the manual). |
| Icon scale | The icon size is automatically adapted to the indicator size. It is possible to change the size using the Icon Scale parameter. |
| Alarm channel | A channel or allowing to change the indicator color when its value differs from 0. The alarm color is defined in the Alarm Color field . |
| Radius | The indicator diameter in pixels. |
| Line width | The indicator line width in pixels. Decimal values are possible. |
| Angle offset | The initial angle of the first indicator (0 in the above example). |
| Angle span | The range in which the indicator is to be divided - in degrees (in the example above it is 270 degrees). |
| Line Color | Indicator line color |
| Value color | Displayed value color |
| Text color | Text or icon color |
| Unit color | Measurement units color |
| Alarm color | Color to which the indicator and the displayed value will change if the value of the variable used in Alarm channel differs from 0. |
| Background color | Indicator background line color |
| Value font | Font size used for displaying values |
| Text font | Font size used for displaying text |
| Unit font | Font size used for displaying measurement units |
| Legend font | Font size of the legend |

7.2. Classic gauge

This object allows to display a numerical value in a similar way as in classic car gauges. The object displays a numerical value and also indicates it using a hand and its position on a clock face.



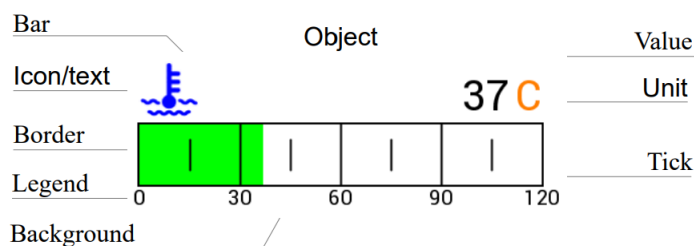
| Parameter | Description |
|-------------------------|---|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Channel | Name of a channel or a variable that will be displayed in the Value field and shown on the gauge. In this field you can also enter a numerical value without a decimal separator to test the operation of the gauge. |
| Decimal places | The number of decimal places displayed for Value and Legend |
| Min, Max | The minimum and maximum value displayed by the indicator |
| Display value | This parameter allows to hide the Value displayed |
| Update frequency | The refresh frequency on the Value parameter screen. The screen is refreshed 50 times per second (50 Hz). Reading fast-changing variables is very difficult at this frequency. The parameter allows to decrease the refresh frequency (e.g. to 5 Hz), which makes the Value parameter easier to read. |
| Display unit | Option for displaying units (Unit). The units will be displayed above the Value parameter. |
| Unit | Selection of the unit in which the value will be expressed (e.g. kPa, Bar, Psi). You can define your own unit in the Custom unit field. To this end, select User in the Unit field. |
| Custom unit | This field is used for entering a user-defined measurement units. To use it, select User in the Unit field |
| Text | Defines the text displayed below the Value parameter. |

| Parameter | Description |
|--------------------------|---|
| Use icon | Parameter for displaying an icon instead of text |
| Icon texture | Allows to select an icon from the texture menu. You can add your own icon (for more information regarding texture management see further sections of this manual). |
| Icon scale | The icon size is automatically adapted to the indicator size. It is possible to change the size using the Icon Scale parameter |
| Alarm channel | A channel or allowing to change the indicator color when its value differs from 0. The alarm color is defined in the Alarm Color field . |
| Radius | The indicator diameter in pixels. |
| Border Size | The width of the line surrounding the indicator in pixels. Decimal values are possible. |
| Highlight percent | Determines how much of the indicator will be filled with the Highlight color. 0 means that the entire indicator will be filled with the Background color . |
| Num ticks | Value defining the number of main parts into which the indicator area will be divided. For each part a respective value and indicator in the form of a section will be displayed. |
| Num subticks | Value defining the number of divisions between the main indicators |
| Indicator width | Width of pointer showing value at a given time |
| Angle span | The range in which the indicator is to be divided - in degrees (in the example above it is 270 degrees). |
| Angle offset | The initial angle of the first indicator (0 in the above example). |
| Color | The color of the value, legend and indicators displayed |
| Border color | The indicator frame color |
| Highlight color | The highlight area color |
| Indicator color | The hand color |
| Background color | The color of the background |
| Text color | Text or icon color |
| Unit color | Measurement units color |
| Alarm color | Color to which the indicator and the displayed value will change if the value of the variable used in Alarm channel differs from 0. |

| Parameter | Description |
|----------------------|--|
| Tick font | The size of the font used for displaying the legend |
| Value font | Font size used for displaying values |
| Text font | Font size used for displaying text |
| Unit font | Font size used for displaying measurement units |
| Redline when | The condition for displaying the indicator in a “redline” mode. Never - never display in the “redline” mode. When value above – display the bar in Redline color when the Value parameter is greater than the Redline start value. When value below – display the bar in Redline color when the Value parameter is lower than the Redline start value. |
| Redline start | This parameter defines the value for a condition defined by the Redline when parameter |

7.3. Bar graph

This indicator allows to display values in the form of a moving bard (either horizontal or vertical). It is also possible to display an icon symbolising the measured value.



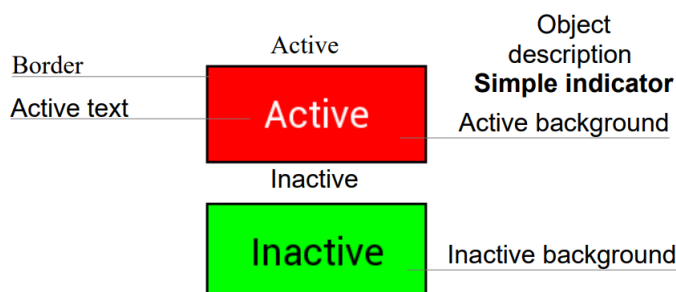
| Parameter | Description |
|-----------------------|---|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Channel | Name of channel or variable that will be displayed in the Value field and shown on the gauge. In this field you can also enter a numerical value without a decimal separator to test the operation of the gauge. |
| Decimal places | The number of decimal places displayed for Value and Legend |
| Min, Max | The minimum and maximum value displayed by the indicator |
| Display value | This parameter allows to hide the Value displayed |

| Parameter | Description |
|-------------------------|---|
| Update frequency | The refresh frequency on the Value parameter screen. The screen is refreshed 50 times per second (50 Hz). Reading fast-changing variables is very difficult at this frequency. The parameter allows to decrease the update frequency (e.g. to 5 Hz), which makes the Value parameter easier to read |
| Display unit | Option for displaying units (Unit). The units will be displayed above the Value parameter. |
| Unit | Selection of the unit in which the value will be expressed (e.g. kPa, Bar, Psi). You can define your own unit in the Custom unit field. To this end, select User in the Unit field. |
| Custom unit | This field is used for entering a user-defined measurement units. To use it, select User in the Unit field |
| Text | Defines the text (description) displayed along the indicator. |
| Use icon | Parameter for displaying an icon instead of text |
| Icon texture | Allows to select an icon from the texture menu. You can add a custom icon (more information about managing textures can be found further in the manual). |
| Icon scale | The icon size is automatically adapted to the indicator size. It is possible to change the size using the Icon Scale parameter |
| Alarm channel | A channel or allowing to change the indicator color when its value differs from 0. The alarm color is defined in the Alarm Color field. |
| Bar type | Indicator type Horizontal , Vertical |
| Bar style | Indicator style: Simple - in the form of a movable bar with graduation, Simple without decoration - in the form of a movable bar without graduation, Style 1 - in the form of a movable bar under the number axis |
| Length | Indicator length |
| Width | Indicator width |
| Border width | The width of the line surrounding the indicator (in pixels) |
| Number of ticks | Value defining the number of main parts into which the indicator area will be divided. For each part a respective value and indicator will be displayed. An additional section is drawn automatically between the indicators. |

| Parameter | Description |
|--------------------------------|---|
| Decimal places for tick | This value defines the number of decimal places drawn for the indicator description |
| Color | Indicator frame color |
| Background color | Indicator background color |
| Bar color | Indicator bar color |
| Alarm color | Color of the indicator if the value of the variable used in Alarm channel differs from 0. |
| Icon color | Icon color |
| Value color | Displayed value color |
| Text color | Text or icon color |
| Unit color | Measurement units color |
| Redline color | The indicator bar color when the displayed value meets the Redline condition . |
| Transparent | Decides if the indicator background is to be displayed. |
| Tick font | The size of the font used for displaying the legend |
| Value font | Font size used for displaying values |
| Text font | Font size used for displaying text |
| Unit font | Font size used for displaying measurement units |
| Redline when | <p>The condition for displaying the indicator in “redline” mode.</p> <p>Never - never display in the “redline” mode.</p> <p>When value above – display the bar in Redline color when the Value parameter is greater than the Redline start value.</p> <p>When value below – display the bar in Redline color when the Value parameter is lower than the Redline start value.</p> |
| Redline start | This parameter defines the value for a condition defined by the Redline when parameter |
| Visibility channel | Name of channel or variable that will control object visibility. 0 means that the object will be hidden, a value different from zero or no assigned channel means that the object will be visible. |

7.4. Simple indicator

This indicator allows to display the indicator of the function attached (e.g. ALS). It makes it possible to assign two different texts in two different background colors depending on the value of the function attached in the **Channel active** field.

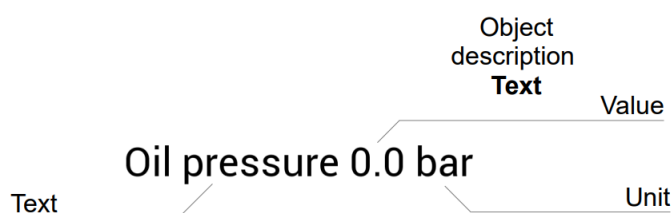


| Parameter | Description |
|---------------------------------|---|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Channel | Name of channel or variable that will be displayed in the Value field and shown on the gauge. In this field you can also enter a numerical value without a decimal separator to test the operation of the gauge. |
| Width | Indicator width |
| Height | Indicator height |
| Border width | The width of the line surrounding the indicator in pixels |
| Style | Background type: solid or textured of your choice |
| Texture | Texture / icon selection |
| Background color default | Inactive indicator background color |
| Background color active | Active indicator background color |
| Text color default | Color of the text displayed by the inactive indicator |
| Text color active | Color of the text displayed by the active indicator |
| Border color | The indicator frame color |
| Text | Text displayed by the inactive indicator |
| Active text | Text displayed by the active indicator |
| Text font | Font size used for displaying text |

| Parameter | Description |
|---------------------------|--|
| Active channel | Channel or function or variable defining whether the indicator is to be displayed in inactive mode (0) or active mode (a value different from zero). |
| Visibility channel | Name of channel or variable that will control object visibility. A value of 0 and no channel assigned means that the object will be hidden, a value different means that the object will be visible. |

7.5. Text

This indicator allows to display text with content read from a log channel or function together with a measurement unit. The user can also hide the object using a function or a log channel.



| Parameter | Description |
|---------------------|---|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Color | Displayed text color |
| Font | Displayed font size |
| Italic | Enables italics. |
| Two lines | Text can be displayed on two lines |
| Text | A field containing the displayed text |
| Text width | The area width used by the Text align functions. For example, by entering 800 (screen width), and setting the position X as 0 and selecting appropriate alignment, you can display the text to the left side of the screen, in the centre of the screen and to the right side of the screen. |
| Text align | Defines text alignment in the Text width area. |

| Parameter | Description |
|---------------------------|---|
| | Left – align the text to the left Center – text centering Right – align the text to the right |
| Channel | Name of channel or variable that will be displayed in the Value field and shown on the gauge. In this field you can also enter a numerical value without a decimal separator to test the operation of the gauge. |
| Value source | Specifies the value displayed for the selected channel: Current – the current channel value Min value – the minimum registered value Max value – the maximum registered value |
| Decimal places | The number of decimal places displayed for the Value parameter |
| Value width | The width of the area of the displayed value used by the Value align function |
| Value align | Defines text alignment in the Value width area . Left – align the text to the left Center – text centering Right – align the text to the right |
| Display unit | Option for displaying units (Unit). The units will be displayed above the Value parameter. |
| Unit | Selection of the unit in which the value will be presented (e.g. kPa, Bar, Psi). You can define your own unit in the Custom unit field. To this end, select User in the Unit field. |
| Custom unit | This field is used for entering a user-defined measurement units. To use it, select User in the Unit field. |
| Color channel | Name of the channel or variable that will control the color of the text displayed. If the channel / function value is 0, the color will be defined in the Color field . Otherwise red will be used. If the selected channel has colors assigned by enumeration, these will be the master colors. |
| Visibility channel | Name of the channel or variable that will control text visibility. 0 means that the text will be hidden, a value different from zero or no assigned channel means that the text will be visible. |

| Parameter | Description |
|-------------------------|---|
| Update frequency | The update frequency on the Value parameter screen. The screen is refreshed 50 times per second (50 Hz). Reading fast-changing variables is very difficult at this frequency. The parameter allows to decrease the update frequency (e.g. to 5 Hz), which makes the Value parameter easier to read. |

7.6. Time

This indicator allows to display the built-in “clocks” of the device, such as the real time, lap time, best lap time etc.

| Parameter | Description |
|---------------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Color | Displayed text color |
| Font | Displayed font size |
| Time source | <p>RTC – the device internal clock time. To set the RTC clock time, select the Devices / Set real time clock from the application menu.</p> <p>Current lap – current lap time</p> <p>Last lap – last lap time</p> <p>Best lap – best lap time</p> <p>Session time – an internal clock measuring the current session time. You can reset it using the external button.</p> <p>Predictive time – predicted lap time based on the best lap time on a given track. Requires a GPS module.</p> |
| Visibility channel | Name of the channel or variable that will control text visibility. 0 means that text will be hidden, a value different from zero or no assigned channel means that text will be visible. |

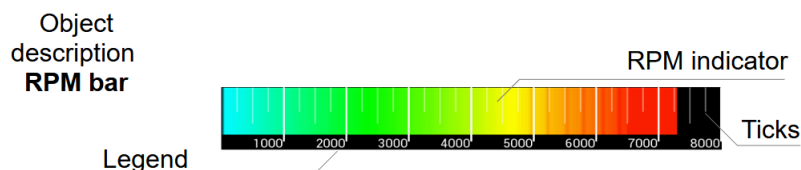
7.7. Image

This object allows a texture (image) or icon to be displayed on the screen and edited in terms of the following parameters: scaling, color selection, mirroring and repetition. In addition to the textures contained in the device memory, you can upload own textures / icons.

| Parameter | Description |
|---------------------------|---|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Default color | Displayed text color |
| Function color | Displayed font size |
| Texture | Allows to select a texture and an icon that will be displayed |
| Repeat mode | Texture repetition mode. Stretch – allows to stretch the texture out (Scale parameter). Tile X – repeats texture along the X axis. Tile Y – repeats texture along the Y axis. Tile X,Y – repeats texture along the X and Y axes. To replicate a texture for Tile mode twice, set the Scale parameter to 200%. To replicate it four times, enter 400% and so on. |
| Scale | This parameter determines the scale of a texture or the number of its copies for Tile modes |
| Mirror X/Y | It allows to produce a mirror image of a texture along the X and / or Y axis. |
| Mode | Texture drawing mode. In the Screen mode a texture is displayed by adding it to the current image. In the Normal mode a texture overwrites the current image, having regard for the alpha channel. You can find more information regarding the textures further in the manual. |
| Color channel | Name of the channel or variable that will control the color of the texture displayed. If the channel / function value is 0, the color will be defined in the Default color field. Otherwise the color of the Function color will be used. |
| Visibility channel | Name of the channel or variable that will control text visibility. 0 means that text will be hidden, a value different from zero or no assigned channel means that text will be visible. |

7.8. RPM Bar

This object allows to display the engine speed in the form of a horizontal bar or a round indicator.

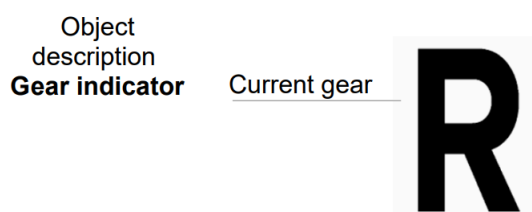


| Parameter | Description |
|----------------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Type | The type of the engine speed indicator displayed: Classic – displayed in the form of a “curved” bar Bar – displayed in the form of a horizontal bar (as shown in the above visualisation) Round – displayed in the form of a round gauge Vertical - displayed as a vertical "triangular" bar |
| Color | Determines the color of the bar indicating the engine speed for Classic , Round and Vertical type indicators. |
| Redline color | Determines the color of the bar indicating the engine speed for Classic , Round and Vertical type indicators, when revs exceed the Redline start value. |
| Background color | For a Classic and Vertical type indicators determines the background color of the indicator. |
| Show redline on bar | For a Classic and Vertical indicator types highlights the redline zone. |
| Width | For a Bar type indicator determines the length of the indicator. |
| Height | For a Bar type indicator determines the height of the indicator. |
| Font | For a Bar type indicator determines the font size of the graduation description. |
| Redline start | A value specifying the beginning of the engine speed threshold area |
| Max RPM | The maximum engine speed value displayed on the indicator |
| Color preset | For a Bar type indicator, it determines the color gradient that will be used for the pointing bar. |

| Parameter | Description |
|---------------------------|---|
| Custom texture | For a Bar type indicator, if Custom texture is selected in Color preset , it determines the texture selection for the pointing bar. |
| RPM x 1000 | For a Bar type indicator, it determines whether the engine speed is displayed in the legend in full or divided by one thousand. |
| Ticks style | For a Bar type indicator determines whether additional graduation lines are to be displayed (Ticks). No ticks - No division Main tick only - main division lines only Main tick + 1 subtick - one additional line between each compartment Main tick + 3 subticks - three additional lines between each compartment |
| Center RPM number | For a Bar type indicator displays each graduation value centrally under the dividing lines |
| Nonlinear factor | Non-linear graduation for a Classic iBar indicator |
| RPM Channel | Channel or function containing the current speed |
| Visibility channel | Name of channel or variable that will control object visibility. A value of 0 and no channel assigned means that the object will be hidden, a value different means that the object will be visible. |

7.9. Gear indicator

This object displays the currently selected gear. The object has been provided with a specially prepared font of increased size containing numbers and the letters R (Reverse) and N (Neutral). The respective displayed gear values are as follows: -2 for park, -1 for reverse, 0 for idle, 1 for first gear, 2 for second gear and similarly up to gear 8.

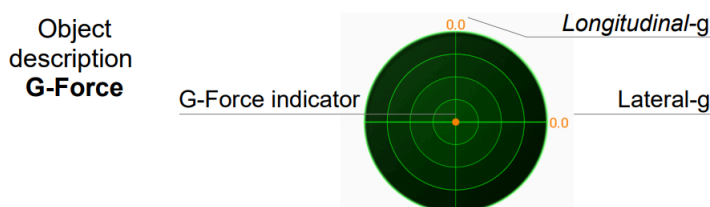


| Parameter | Description |
|---------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |

| Parameter | Description |
|---------------------------|--|
| Color | Displayed gear color |
| Font | Displays font size (maximum size is: 15) |
| Value channel | Channel or function containing the currently selected gear |
| Visibility channel | Name of channel or variable that will control object visibility. 0 means that the object will be hidden, a value different from zero or no assigned channel means that the object will be visible. |

7.10. G-Force

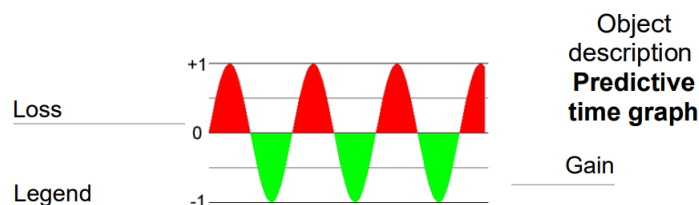
The G-Force object indicates the current G-force acting on the vehicle by means It uses the accelerometer built into the display to do this. The *adu.latG* channel records lateral overloads acting on the vehicle (left / right), and the *adu.longG* channel records longitudinal overloads (braking / acceleration). The accelerometer requires calibrating after installing the device. This can be done using a button defined in the *Buttons / IMU* pitch zeroing panel or manually in the *Configuration / IMU Pitch* panel.



| Parameter | Description |
|---------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Color | Displayed gear color |
| Size | Indicator diameter |

7.11. Predictive time graph

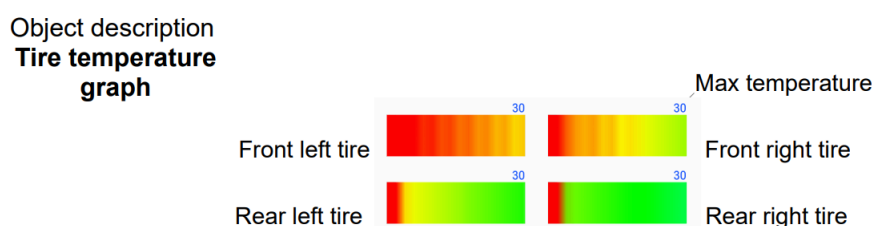
The **Predictive time graph** object indicates the time difference between the best reference lap and the current position on the track. This object requires a GPS module and a correct configuration of the racing track.



| Parameter | Description |
|---------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Width | Object width |
| Height | Object height |
| Gain color | The color of the graph when the current time is better than the reference time |
| Loss color | The color of the graph when the current time is worse than the reference time |
| Lines color | The graph line color |
| Font color | The color of the font for describing the axes |
| Font size | The font size |
| Time range | Time range: +/-0.5; +/-1 or +/-2 |

7.12. Tire temperature graph

The **Temperature graph** object displays the temperature gradient of tyres or brake discs recorded by thermal imaging cameras. The tire temperature can be presented as gradients or tires. Configuration of the measuring range of the cameras can be found in the **ADU / Configuration / Tire temperature cameras** panel and in the **ADU / Configuration / Brake temperature cameras panel**.

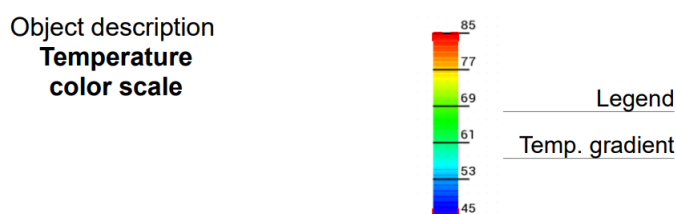


| Parameter | Description |
|---------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |

| Parameter | Description |
|------------------|--|
| Style | Object display style: Tires – temperature displayed as tires Bar – temperature displayed as gradients |
| Source | Tire temperature cameras Brake temperature cameras |
| Length | Object length (parameter available only for the Bar style) |
| Breadth | Object width (parameter available only for the Bar style) |
| Spacing X | The distance on the horizontal axis between gradients representing tires (parameter available only for the Bar style) |
| Spacing Y | The distance on the vertical axis between gradients representing tires (parameter available only for the Bar style) |
| Scale | Object size (parameter available only for the Tires style) |
| Rotate | For the Bar style, displays temperature gradients as vertical bars |

7.13. Temperature color scale

The **Temperature color scale** object displays the temperature gradient and the temperature values assigned to a given color. Configuration of the measuring range of the cameras can be found in the **ADU > Configuration > Tire temperature cameras** and **ADU > Configuration > Brake temperature cameras panels**.



| Parameter | Description |
|---------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Source | Tire temperature cameras Brake temperature cameras |
| Scale | Defines the gradient size. |
| Legend color | Defines the description color. |

7.14. Session results

The **Session results** object allows the times of sixty individual laps to be displayed, highlighting the best laps in a different color.

Object description

Session results

| Lap | Time | Lap | Time | Lap | Time | Lap | Time |
|-----|----------|-----|----------|-----|-------|-----|-------|
| 1. | 01:50.32 | 16. | 01:51.64 | 31. | | 46. | |
| 2. | 01:47.51 | 17. | 01:51.51 | 32. | | 47. | |
| 3. | 01:47.82 | 18. | 01:50.82 | 33. | | 48. | |
| 4. | 01:48.73 | 19. | 01:50.42 | 34. | | 49. | |
| 5. | 01:50.34 | 20. | 01:51.60 | 35. | | 50. | |
| 6. | 01:47.66 | 21. | 01:50.51 | 36. | | 51. | |
| 7. | 01:48.61 | 22. | 01:51.90 | 37. | | 52. | |
| 8. | 01:49.09 | 23. | 01:50.90 | 38. | | 53. | |
| 9. | 01:49.77 | 24. | 01:50.98 | 39. | | 54. | |
| 10. | 01:49.06 | 25. | 01:47.85 | 40. | | 55. | |
| 11. | 01:49.67 | 26. | 01:50.61 | 41. | | 56. | |
| 12. | 01:48.11 | 27. | 01:52.48 | 42. | | 57. | |
| 13. | 01:49.43 | 28. | | 43. | | 58. | |
| 14. | 01:48.01 | 29. | | 44. | | 59. | |
| 15. | 01:48.54 | 30. | | 45. | | 60. | |

| Parameter | Description |
|-----------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Color | Defines the color of the text in the table. |
| Best lap color | Defines the color of the best lap time displayed. |

7.15. Track record table

The **Track record table** object allows to display eight best times for a given racing track. A track is recognized using a GPS position. You can find more information regarding the GPS further in the manual.

Object description

Track record table

| Track name | | 4999m | | | |
|------------|----------|-------|-----------|------------|--|
| # | TIME | LAP | TOP SPEED | DATE | |
| 1 | 01:59.39 | 199 | 300.0 | 25.01.2017 | |
| 2 | 01:59.39 | 199 | 300.0 | 25.01.2017 | |
| 3 | 01:59.39 | 199 | 300.0 | 25.01.2017 | |
| 4 | 01:59.39 | 199 | 300.0 | 25.01.2017 | |
| 5 | 01:59.39 | 199 | 300.0 | 25.01.2017 | |
| 6 | 01:59.39 | 199 | 300.0 | 25.01.2017 | |
| 7 | 01:59.39 | 199 | 300.0 | 25.01.2017 | |
| 8 | 01:59.39 | 199 | 300.0 | 25.01.2017 | |

First row

First column

| Parameter | Description |
|-------------------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| 1st column bkgrd color | First column background color |
| 1st column text color | First column text color |

| Parameter | Description |
|----------------------------|-----------------------------------|
| Table bkgrd color 1 | Background color for odd columns |
| Table bkgrd color 2 | Background color for even columns |
| Table text color | Table text color |
| 1st row bkgrd color | First row background color |
| 1st row text color | First row text color |

7.16. Rectangle

The **Rectangle** object allows to draw a rectangle on a page. You can define the width of the frame line and the fill color.

Object description

Rectangle



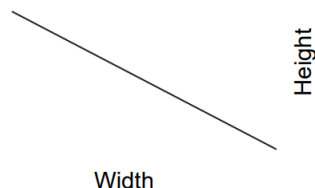
| Parameter | Description |
|---------------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Rectangle type | The type of the displayed rectangle. Border – displays only the rectangle border Border + fill – displays the rectangle border and fill Only fill - displays only the rectangle fill |
| Color | Border color |
| Fill color | Fill color |
| Width | Rectangle width |
| Height | Rectangle height |
| Thickness | Width of the border in pixels |
| Visibility channel | Name of channel or variable that will control object visibility. 0 means that the object will be hidden, a value different from zero or no assigned channel means that the object will be visible. |

7.17. Line

The **Line** object is used to draw lines on pages.

Object description

Line



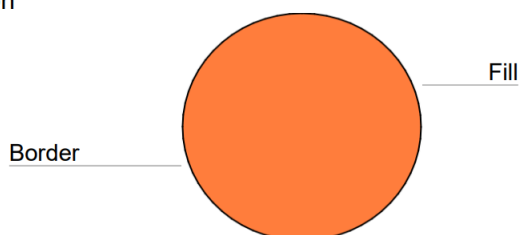
| Parameter | Description |
|---------------------------|---|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Color | Border color |
| Width | Distance between the beginning and the end of the line along the X axis |
| Height | Distance between the beginning and the end of the line along the X axis |
| Thickness | Width of the border in pixels |
| Style | Type of line: Simple - a line for which the beginning and end can be arbitrarily specified on the plane Horizontal gradient one side - horizontal line, colour graduation from the left side Horizontal gradient two sides - horizontal line, colour graduation from both ends Vertical gradient one side - vertical line, colour graduation from above Vertical gradient two sides - vertical line, colour graduation from both ends |
| Visibility channel | Name of channel or variable that will control object visibility. 0 means that the object will be hidden, a value different from zero or no assigned channel means that the object will be visible. |

7.18. Circle

The **Circle** object is used to draw circles on pages. You can define in it the width of the lines and the fill color.

Object description

Circle



| Parameter | Description |
|---------------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Circle type | The type of the displayed rectangle. Border – displays only the rectangle border Border + fill - displays border and fill Only fill - displays only the fill |
| Color | Border color |
| Fill color | Fill color |
| Thickness | Width of the border in pixels |
| Radius | Circle radius |
| Visibility channel | Name of channel or variable that will control object visibility. 0 means that the object will be hidden, a value different from zero or no assigned channel means that the object will be visible. |

7.19. Grid

The **Grid** object displays a table which shows values defined by separate channels in each cell.

Object description

Grid

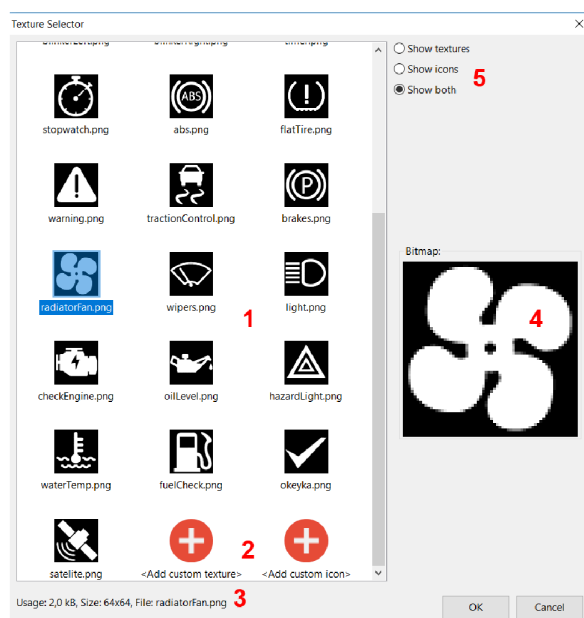
| | | | | |
|-----|-----|--------|-----|--------|
| O1 | ... | 0.0 A | ON | 13.9 V |
| O2 | ... | 14.0 A | ON | 13.9 V |
| O3 | ... | 0.0 A | OFF | 0.0 V |
| O4 | ... | 0.0 A | OFF | 0.0 V |
| O5 | ... | 0.0 A | OFF | 0.0 V |
| O6 | ... | 1.0 A | ON | 13.9 V |
| O7 | ... | 0.5 A | ON | 13.9 V |
| O8 | ... | 0.0 A | OFF | 0.0 V |
| O9 | ... | 0.0 A | OFF | 0.0 V |
| O10 | ... | 0.0 A | OFF | 0.0 V |
| O11 | ... | 0.0 A | OFF | 0.0 V |
| O12 | ... | 0.0 A | OFF | 0.0 V |
| O13 | ... | 0.0 A | OFF | 0.0 V |
| O14 | ... | 0.0 A | OFF | 0.0 V |
| O15 | ... | 0.0 A | OFF | 0.0 V |
| O16 | ... | 0.0 A | OFF | 0.0 V |

| Parameter | Description |
|-------------------------------|--|
| Position X,Y | Object position on a page The reference point is the upper-left corner of a rectangle inscribed into the object. |
| Row count | Number of rows |
| Font | The font size |
| Active | Specifies whether the column will be displayed |
| Column width | Width of column concerned |
| Prefix | Prefix for line numbering |
| Suffix | Suffix for line numbering |
| Start index | Initial value of line numbering |
| Name #1-#16 | Description of lines from 1 to 16 |
| Decimal places | Decimal places for the displayed values in a given column (A to C) |
| Text aling | Column text alignment |
| Channel #1-#16 | The channel from which the values displayed in the specified row (from 1-16) and column (from A-C) will originate |
| Default text color | Primary text color |
| Fill color | Row background color |
| Alternating fill color | Alternating row background colors |
| Fill color #2 | Alternative row background color |
| Channel #1-#16 | The name of the channel or variable that will control the color of the displayed text for the corresponding line (from 1-16). If the channel / function value is 0, the color defined in the Default text Color field will be |

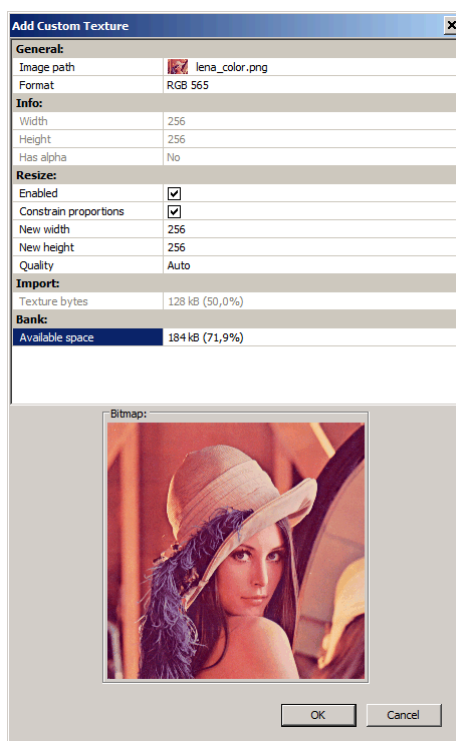
| Parameter | Description |
|-----------|---|
| | applied. Otherwise red will be used. If the selected channel has colors assigned by enumeration, these will be the master colors. |

7.20. Textures








Textures are bit maps (images) defining the graphics that we can display using the Image object. The ADU device has built-in textures dedicated to displaying as icons or backgrounds. The user is also able to add their own textures (e.g. backgrounds, icons, etc.).



The textures are managed using the **Texture Manager (Menu > Tools > Texture manager dialogue)**. This window is displayed also when you select a texture for the objects that display it (e.g. Image, Bar graph, etc.) In the **Texture manager** window, it is possible to preview the available textures (1). After selecting a texture in the lower part of the window information about the selected object (3) will be displayed along with its preview (4). In addition, a filter option is available, which allows only textures or only icons to be displayed (5). An icon differs from a texture in that it has to have a fixed size: 64x64 pixels. To add your own icon or texture, select **Add custom texture** or **Add custom icon**. After selecting a file with a graphics (.png .jpg and .bmp formats are supported) a dialogue window will be displayed that will enable you configure the texture.



| Parameter | Description |
|------------------------------|--|
| Image path | File name on the disc |
| Format | The texture target format. The format determines the quality of the texture and the amount of the device memory it takes up. Detailed information about the formats can be found on the next page. |
| Width, height | Information about the size of the source bit map |
| Has alpha | Information whether the bit map has a separate alpha channel. |
| Resize enabled | Allows scaling of the source texture. |
| Constrain proportions | When activated, this option automatically keeps the proportion of the texture during scaling. |
| New width, height | The new width and the new height of the texture |
| Quality | The filtering that will be used during scaling. Auto filtering is recommended. |
| Texture bytes | The size of the texture in the memory of the device. |
| Available space | The available free memory for textures |

| Format | Description | Preview |
|---|---|---|
| A1 8kB | 1-bit format. A pixel of the texture can assume only 2 values. It takes up less memory. |  |
| A2 16kB | 2-bit format. A pixel of the texture can assume 4 values. |  |
| A4 32kB | 4-bit format. A pixel of the texture can assume 16 values. This format is recommended for icons. It guarantees a good quality while using little memory. |  |
| A8 64kB | 8-bit format. A pixel of the texture can assume 256 values. |  |
| Indexed RGBA8 64kB | An 8-bit color format with an alpha channel. The texture is quantized to 256 unique colors. This format is suitable for colour textures (e.g. a company logo). The advantage is low memory consumption, while the disadvantage is low rendering efficiency and visible deterioration in smooth tonal transitions. |  |
| RGB 565 128 kB | A 16-bit color format without an alpha channel. It has a very good quality and rendering efficiency. A flow of this format is that it uses a lot of memory. |  |
| ARGB 1555 128 kB | A 16-bit color format with an alpha channel. It has a very good quality and rendering efficiency. A flow of this format is that it uses a lot of memory. |  |

8. Inputs

The ADU is equipped with 8 analog inputs and 8 digital inputs.

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 | <p>The function that the analog input is to perform:</p> <p>Switch – active low – the analog input will function as a switch (button) activated by a low state.</p> <p>Switch – active high – the analog input will function as a switch (button) activated by a high state.</p> <p>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.</p> <p>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).</p> <p>Calibrated analog sensor – the input is used for measuring the values from sensors with a non-linear scale (e.g. temperature sensors NTC / PTC). A 2D map is used to define the values.</p> |
| Pull-up / Pull-down | 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 to ground. For analog sensors or for measuring voltage, choose the 1M Pulldown option. |

| Parameter | Description |
|------------------------------|--|
| 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 <i>Voltage</i> . |
| Max value for voltage | For Linear analog sensor inputs this value defines the maximum sensor value for the defined <i>Voltage</i> . |

Example configurations:

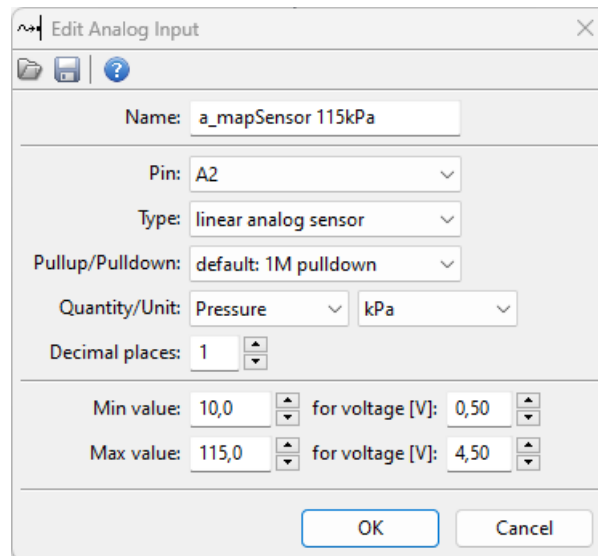
A configuration for a button connected to ground and to the analog input 1. The *a_sampleButton* value will be 0 when the button is not pressed and 1 when it is.

The screenshot shows the 'Edit Analog Input' dialog box with the following settings:

- Name: *a_sampleButton*
- Pin: A1
- Type: switch - active low
- Pullup/Pulldown: default: 10K pullup
- 0 if voltage > [V]: 3,5 for [s]: 0,01
- 1 if voltage < [V]: 1,5 for [s]: 0,01

Buttons: OK, Cancel

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

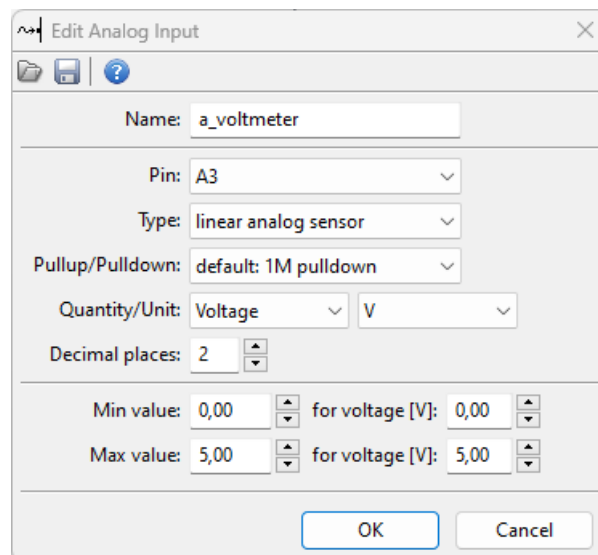


The 'Edit Analog Input' dialog box is shown with the following settings:

- Name: a_mapSensor 115kPa
- Pin: A2
- Type: linear analog sensor
- Pullup/Pulldown: default: 1M pulldown
- Quantity/Unit: Pressure, kPa
- Decimal places: 1
- Min value: 10,0 for voltage [V]: 0,50
- Max value: 115,0 for voltage [V]: 4,50

Buttons: OK, Cancel

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.

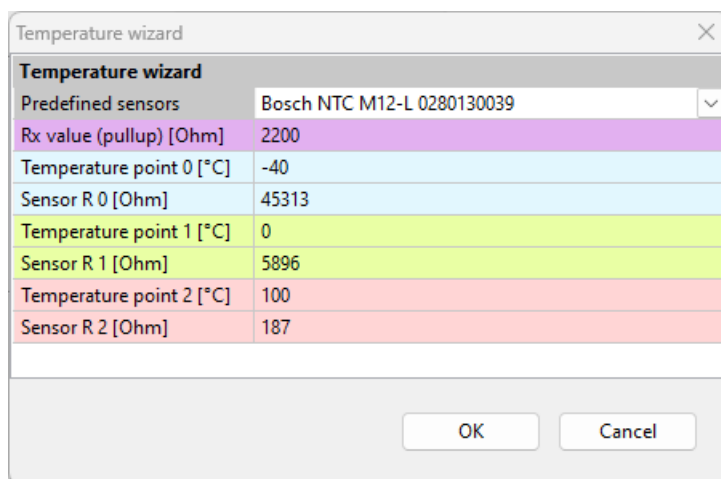


The 'Edit Analog Input' dialog box is shown with the following settings:

- Name: a_voltmeter
- Pin: A3
- Type: linear analog sensor
- Pullup/Pulldown: default: 1M pulldown
- Quantity/Unit: Voltage, V
- Decimal places: 2
- Min value: 0,00 for voltage [V]: 0,00
- Max value: 5,00 for voltage [V]: 5,00

Buttons: OK, Cancel

The easiest way to calibrate non-linear temperature sensors is to use the *Wizard*. After pressing the *Wizzard* 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.

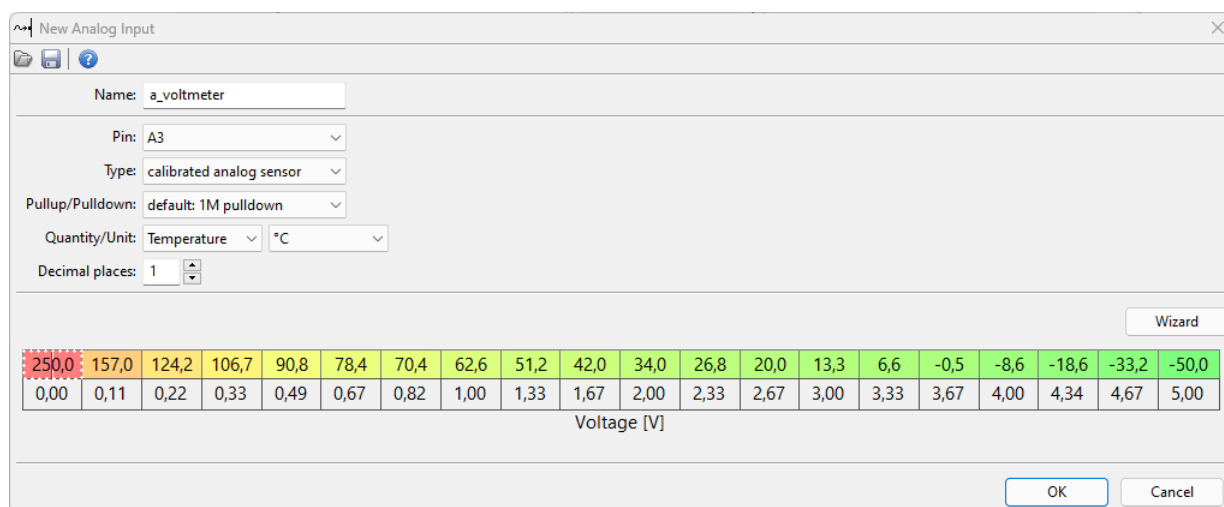


Temperature wizard

| | |
|--------------------------|----------------------------|
| Predefined sensors | Bosch NTC M12-L 0280130039 |
| Rx value (pullup) [Ohm] | 2200 |
| Temperature point 0 [°C] | -40 |
| Sensor R 0 [Ohm] | 45313 |
| Temperature point 1 [°C] | 0 |
| Sensor R 1 [Ohm] | 5896 |
| Temperature point 2 [°C] | 100 |
| Sensor R 2 [Ohm] | 187 |

OK Cancel

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:



New Analog Input

Name: a_voltmeter

Pin: A3

Type: calibrated analog sensor

Pullup/Pulldown: default: 1M pullup

Quantity/Unit: Temperature °C

Decimal places: 1

Wizard

| | | | | | | | | | | | | | | | | | | | |
|-------|-------|-------|-------|------|------|------|------|------|------|------|------|------|------|------|------|------|-------|-------|-------|
| 250,0 | 157,0 | 124,2 | 106,7 | 90,8 | 78,4 | 70,4 | 62,6 | 51,2 | 42,0 | 34,0 | 26,8 | 20,0 | 13,3 | 6,6 | -0,5 | -8,6 | -18,6 | -33,2 | -50,0 |
| 0,00 | 0,11 | 0,22 | 0,33 | 0,49 | 0,67 | 0,82 | 1,00 | 1,33 | 1,67 | 2,00 | 2,33 | 2,67 | 3,00 | 3,33 | 3,67 | 4,00 | 4,34 | 4,67 | 5,00 |

Voltage [V]

OK Cancel

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.

8.2. Digital Inputs

Digital Inputs are used for transforming digital signals such as signals from the crankshaft position sensor, wheel speed sensor or ethanol sensor (*FlexFuel*). These inputs can also be used as inputs for buttons connected to ground.

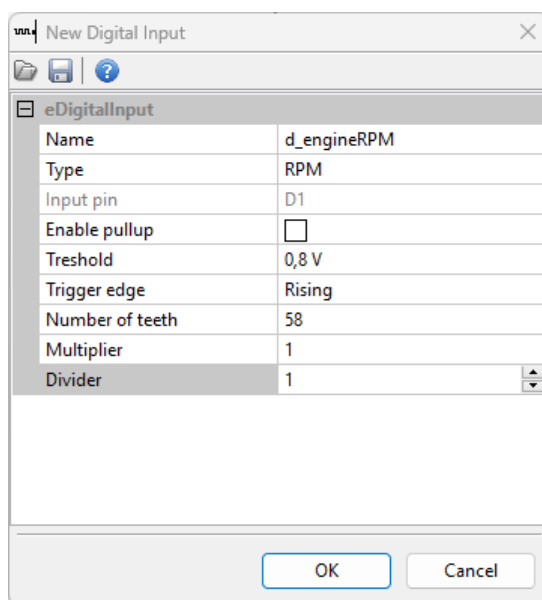
To add an digital input, in **Project Tree** add the **Digital Input** element. The configuration window includes the following options:

| Parameter | Description |
|-----------------------|---|
| Name | The name of the digital input that will be used as the channel name in the project |
| Input pin | Number of the digital input to which the configuration relates |
| Type | <p>The function that the digital input is to perform:</p> <p>Switch – active low – the digital input will function as a switch (button) activated by a low state.</p> <p>Switch – active high – the digital input will function as a switch (button) activated by a high state.</p> <p>Frequency – the digital input will measure the signal frequency.</p> <p>RPM – the digital input will decode the signal from the crankshaft / camshaft position sensor in order to measure the vehicle engine speed. Only the D1 input can be used to measure engine speed.</p> <p>Flex Fuel – a digital input used for reading the content of ethanol in fuel and the fuel temperature from the FlexFuel sensor. Only the D2 input is compatible with the FlexFuel sensor. Values read from the sensor are stored in the following channels: adu.ff.ethanolContent, adu.ff.fuelTemperature, adu.ff.sensorStatus</p> <p>Beacon - digital input used to decode the signal from an AIM's beacon. The D2 input is compatible with the AIM beacon.</p> <p>PULS oil sensor - digital input used to read out the digital signal from the PULS type oil level and temperature sensor. This sensor can only be connected to the D4 input. Values read from the sensor are stored in the following channels: adu.puls.level, adu.puls.temperature, adu.puls.status.</p> <p>An example of a PULS sensor working with an ADU:</p> <p>6PR 010 497-05</p> <p>Range: 18-118.8 mm</p> <p>Connection: 1: +12 V, 2: ground, 3: signal</p> <p>The sensor requires the "Enable pull-up" and the "Threshold" to 2.5 V options to be enabled.</p> <p>The sensor's measuring range is configured in the "PULS oil temperature" > "level sensor" window.</p> |
| Enable pull-up | Allows to activate an internal pull-up resistor 4k7 for a given input. |

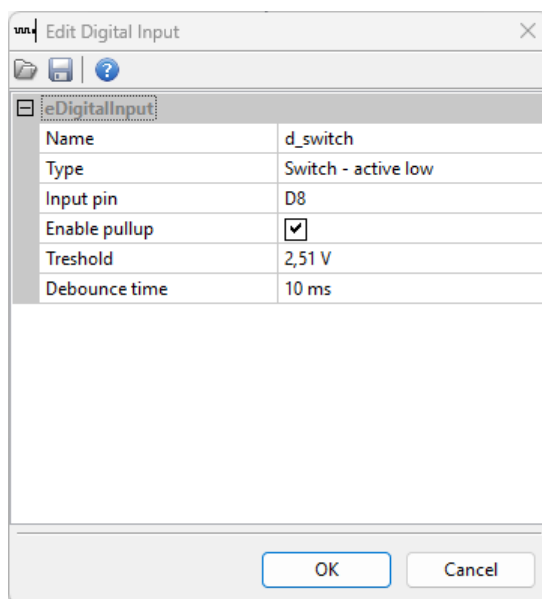
| Parameter | Description |
|------------------------|--|
| Threshold | A reference voltage which, when exceeded, results in a change of the input state from 0 to 1 (and the other way around). For inductive sensors, this will be less than 1 V. For Hall / optical sensors it will be 2.5 V. |
| Debounce time | For a Switch type input, this parameter determines the time needed to stabilize the switch contacts. |
| Trigger edge | The signal edge to be used when transforming a signal. |
| Number of teeth | For an RMP signal - a physical number of teeth on a toothed wheel used by the crankshaft / camshaft position sensor. For a 60-2 toothed wheel the number will be 58, for a 12+1 wheel - 13, etc. |
| Multiplier | The value by which input frequency (Frequency) or engine speed (RPM) will be multiplied. It allows calibration of values such as turbocharger speed or vehicle speed. |
| Divider | The value by which input frequency (Frequency) or engine speed (RPM) will be divided. It allows calibration of values such as turbocharger speed or vehicle speed. |

Example configurations:

Engine speed (RPM) read-out configuration from the inductive camshaft and 60-2 toothed wheel position sensor. In the case of the engine speed read-out, the sensor must be connected to *Digital Input 1*.



Configuration of the status read-out of the button connected to *Digital Input 8*. The button must connect the signal to ground. The *d_switch* variable assumes the value of 0 when the button is not pressed and 1 when it is.

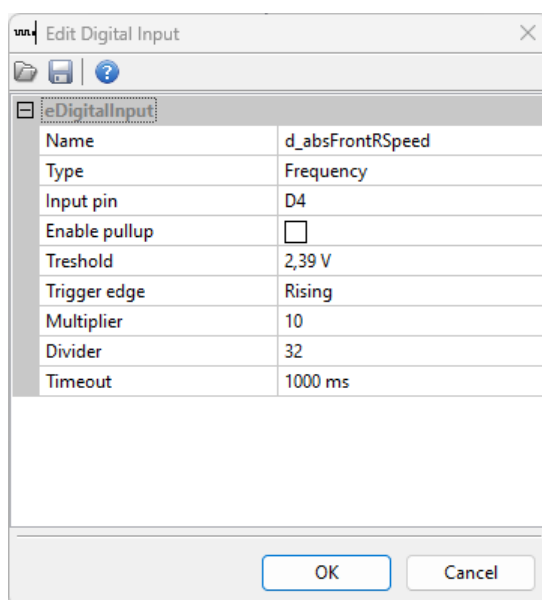


The screenshot shows the 'Edit Digital Input' dialog box with the following configuration:

| eDigitalInput | |
|---------------|-------------------------------------|
| Name | d_switch |
| Type | Switch - active low |
| Input pin | D8 |
| Enable pullup | <input checked="" type="checkbox"/> |
| Threshold | 2,51 V |
| Debounce time | 10 ms |

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Configuration of the wheel speed read-out from the ABS sensor connected to *Digital Input 4*. The *d_absFrontRSpeed* variable value equals the input frequency multiplied by 10 and divided by 32 (multiplier, divider).



The screenshot shows the 'Edit Digital Input' dialog box with the following configuration:

| eDigitalInput | |
|---------------|--------------------------|
| Name | d_absFrontRSpeed |
| Type | Frequency |
| Input pin | D4 |
| Enable pullup | <input type="checkbox"/> |
| Threshold | 2,39 V |
| Trigger edge | Rising |
| Multiplier | 10 |
| Divider | 32 |
| Timeout | 1000 ms |

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

9. Outputs

The ADU device is equipped with two Low-side type outputs (connected to ground) with a load capacity of up to 2 amps. Additionally, an analog output with a signal ranging from 0 to 5 V is available, which can be used to send the voltage signal to another device.

9.1. Low side outputs

Configuration of low side outputs is available in the **Outputs** panel. The user can choose between two output channels **Aux1.channel** and **Aux2.channel**, where it is possible to define the variables / functions that will control the output. When their value is 0, the output is inactive (open). When the value is different from 0, the output is active (shorted to ground).

9.2. Analog output

The analog output control channel is available in the **Outputs** panel as **AOout.channel**. It should be assigned a variable representing a value in V with a 0.001 V accuracy, resulting in the actual value ranging from 0 to 5000 (0-5 V).

The following example will show how to change a signal from the 0-5 V analog input to a 5-0 V signal and how to set the current voltage at the **Analog output**.

We will use a 2D table for this purpose. To create it, select the **Add** button and then **Table**. The configuration dialog should look like the following:

New Table

Name:

Quantity/Unit:

Decimal places:

Axis X: channel: ...

min: ...

max: ...

step: ...

columns:

Axis Y: channel: ... (leave blank for 2D table)

min: ...

max: ...

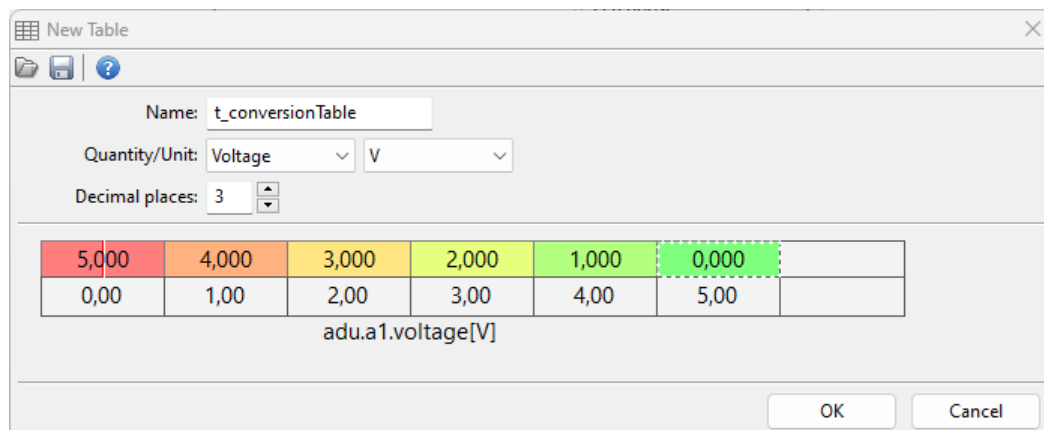
step: ...

rows:

Create

OK Cancel

A 2D Table will be created after pressing the **Create** button. The table should be modified so that for 0 V the value read from the table is 5 V and for 5 V it is 0 V.

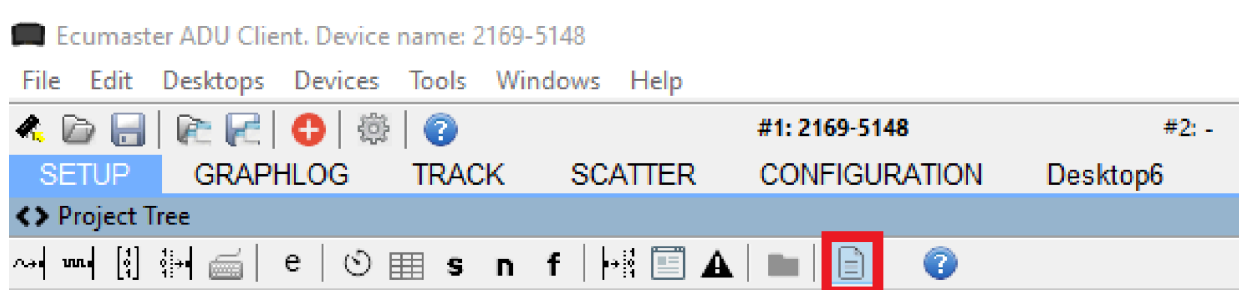


The *t_conversionTable* table value should be assigned to the **AOut.channel** field in the Outputs panel.

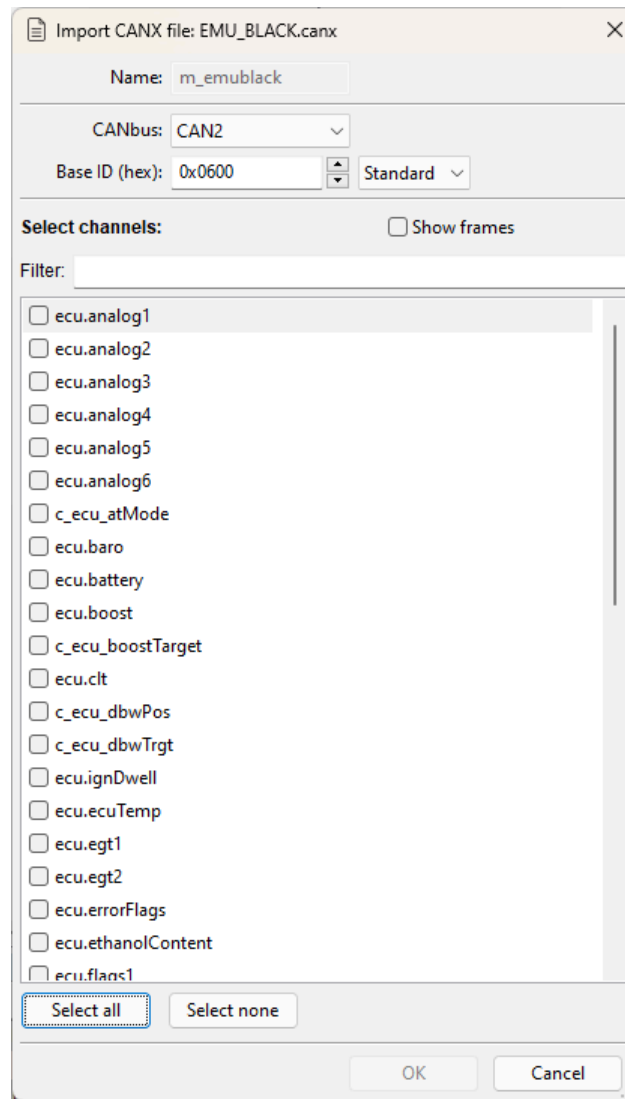
10. Working with CAN buses in ADU

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 ADU Client software.



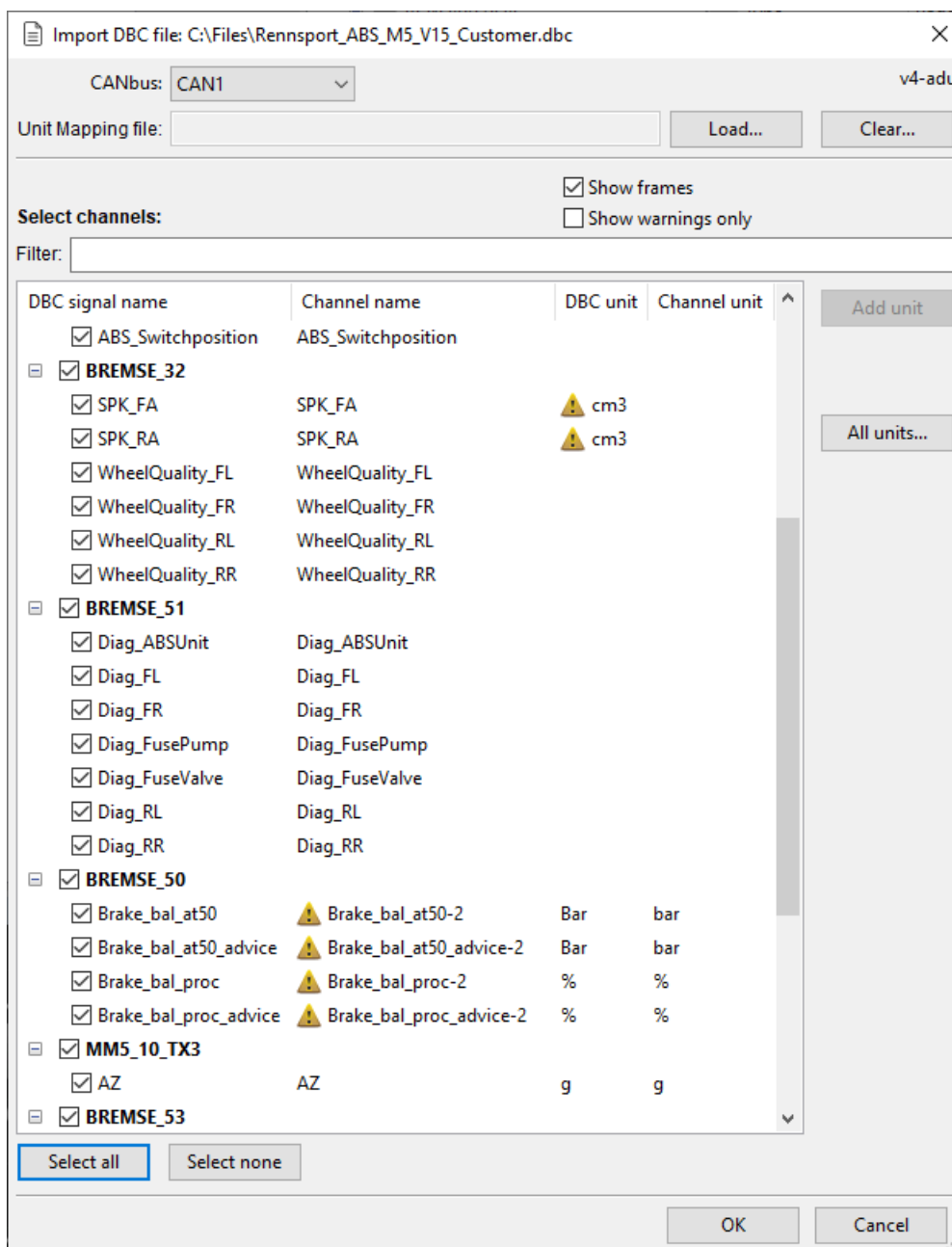
Having selected an icon from the taskbar or choosing the **Project Tree > Add > Import .CANX / .DBC file** option and pointing to the **.CANX** or **.DBC** file, a window with the import settings will open.



First, choose the CAN bus from which data will be received. The ADU 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 ADU 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 ADU;
- about channels already existing under a particular name.

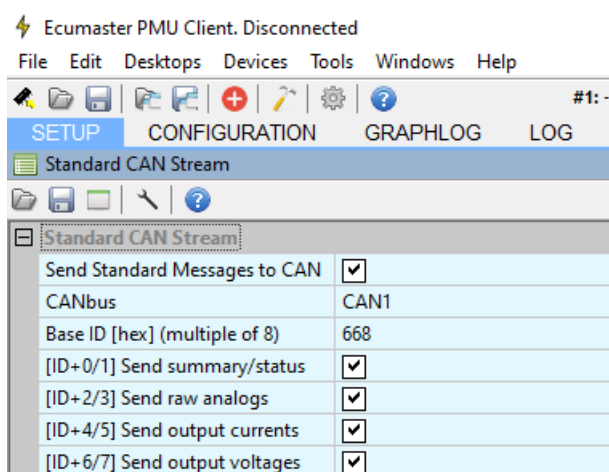


After confirming with the OK button, the selected channels will be added to the **Project Tree**. In addition, one or more **CANbus Message Object** responsible for receiving frame groups will be created.

10.2. Built-in support for PMU

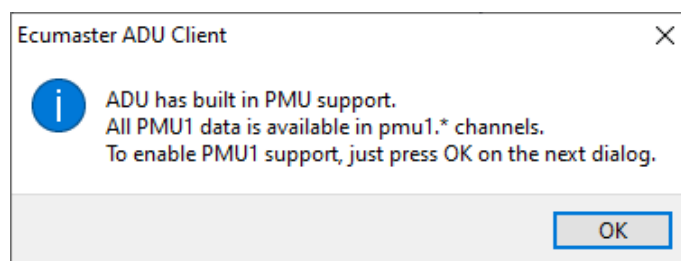
The PMU is the vehicle's intelligent energy management unit. All information about the status of the outputs, voltage and current and analog inputs can be recorded and transmitted via the CAN bus to the display. Several PMUs can be connected on the CAN bus. The ADU is equipped with default inputs for the three PMUs, so that the user does not need to use any of the 150 *CANbus inputs* to monitor channels from the PMUs. Each PMU comprises of 87 channels.

In order for the ADU to receive data correctly, the dedicated CAN Stream in the **Ecumaster PMU Client** software must be configured accordingly for each connected PMU. Use the **F9** key to open the **Select panel** window and then select **Standard CAN Stream**. In the window that opens, tick the **Send Standard Message to CAN** box, select the appropriate CAN bus and define the corresponding CAN ID. The dedicated CAN IDs are as follows: for PMU16 #1 - 0x668, for PMU16 #2 - 0x670, for PMU16 #3 - 0x678, PMU24 #1 - 0x668, for PMU24 #2 - 0x678, for PMU24 #3 - 0x688. For more details on the PMU CAN stream, check: [Appendix C - How to Configure PMU CAN Stream \(on page 209\)](#).

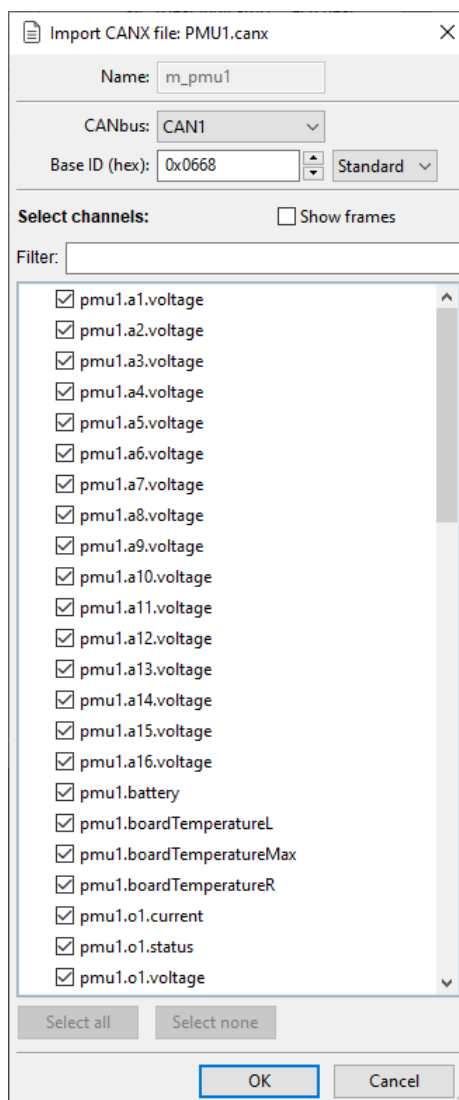


Receiving information

In **Ecumaster ADU Client**, select the appropriate PMU#.canx file from the list in the Open CANX / DBC file window. Upon confirmation of the selection, the following message will be displayed: "The ADU has built-in PMU support. All PMU# data is available in pmu#.* channels. To enable PMU# support, press OK in the next dialog box."



This window will show the list of channels and the pre-defined default CAN bus and CAN ID settings in the PMU, which can be edited.



Once the settings have been accepted, a new Message Object for the selected PMU will appear in the Project Tree.

To keep track of pmu channel status information, you can open a separate Log panel for each PMU.

| PMU1 | | |
|--------------|---------|------|
| Name | Value | Unit |
| totalCurrent | 14 | A |
| battery | 14,04 | V |
| boardTempL | 31 | °C |
| boardTempR | 33 | °C |
| boardTempMax | 33 | °C |
| status | ON (1) | |
| userError | 0 | |
| o1.status | ON (1) | |
| o2.status | ON (1) | |
| o3.status | OFF (0) | |
| o4.status | OFF (0) | |
| o5.status | OFF (0) | |
| o6.status | ON (1) | |
| o7.status | ON (1) | |
| o8.status | OFF (0) | |
| o9.status | OFF (0) | |

10.3. Own CAN streams – CANbus Message Object

Access to a CAN bus in the ADU 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 (**Base ID**), as well as the type (**Type**) and the number of received frames – the **Size** parameter. If the device is 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 ADU 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

| Length: | Data: (hex) | Live Capture | Freq. [Hz] | | | | | | | |
|---------|-------------|--------------|------------|----|----|----|----|----|----|------|
| 0x123: | 8 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 53,7 |

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

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

| | Length: | Data: (hex) | <input checked="" type="checkbox"/> Live Capture | Freq. [Hz]: |
|--------|---------|-------------------------|--|-------------|
| 0x600: | 8 | 00 20 8C 40 80 00 8B 01 | | 24,7 |
| 0x601: | 8 | 70 00 00 00 FF 03 FF 03 | | 24,8 |
| 0x602: | 8 | 24 04 64 00 80 30 80 00 | | 24,8 |
| 0x603: | 8 | 14 43 80 C8 00 00 00 00 | | 24,9 |
| 0x604: | 8 | 01 26 EF 01 00 00 00 00 | | 24,8 |
| 0x605: | 8 | 00 00 00 00 00 00 00 00 | | 24,7 |
| 0x606: | 8 | 00 00 00 00 00 00 00 01 | | 24,8 |
| 0x607: | 8 | 00 00 00 00 00 00 00 00 | | 24,8 |

- **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 MOB's 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

Dialog: Edit CANbus Message Object

Name: m_mob1

CANbus: CAN1

Base ID (hex): 0x0111 Standard

Type: Compound 8 bit (+0)

Size: 4 frames

Timeout [s]: 1,0

Test Data

| | Length: | Data: (hex) | <input checked="" type="checkbox"/> Live Capture | Freq. [Hz]: |
|--------|---------|-------------------------|--|-------------|
| 0x111: | 8 | 00 12 34 12 34 12 34 | | 8,2 |
| 0x111: | 8 | 01 00 56 78 9A BC DE F0 | | 6,2 |
| 0x111: | 8 | 02 00 11 22 33 44 55 66 | | 4,4 |
| 0x111: | - | | | |

OK Cancel

- **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.

10.4. Own CAN streams – CANbus Input

After a **CAN Message Object** has been created, you can start defining **CANbus Input** channels.

The screenshot shows the 'Edit CANbus Input' dialog box with the following settings and annotations:

- 1**: Radio buttons for 'Create new channel' (selected) and 'Override existing'.
- 2**: 'Message object' dropdown set to 'm_600', a '+' button, and 'ID: 0x600'.
- 3**: 'Data format' dropdown set to '16bit unsigned', 'Endian' dropdown set to 'little endian', and 'Byte offset' set to 4.
- 4**: 'Multiplier' set to 1, 'Divider' set to 1, and 'Offset' set to 0.
- 5**: 'Quantity' dropdown set to 'Pressure' and 'Unit' dropdown set to 'kPa'.
- 6**: 'Default value' set to 0.
- 7**: 'If message times out' section with 'use the previous value' selected and 'set value' set to 0.
- 8**: 'Test data' section with 'Length' set to 8, 'Data: (hex)' showing '00 04 28 30 80 00 00 01', 'Live Capture' checked, and 'Result: 128 kPa'.

First, decide whether you will be creating a new channel (option: **Create new channel**) or whether an existing channel shall be overridden (option: **Override existing**) (1).

To create a new channel, choose a unique name so that it can be identified.

To use a channel name already in use, select an existing channel from the “ecu.” group. (e.g. **ecu.rpm**). In addition, the decimal places and unit must be matched with the selected channel.

Next, 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.

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

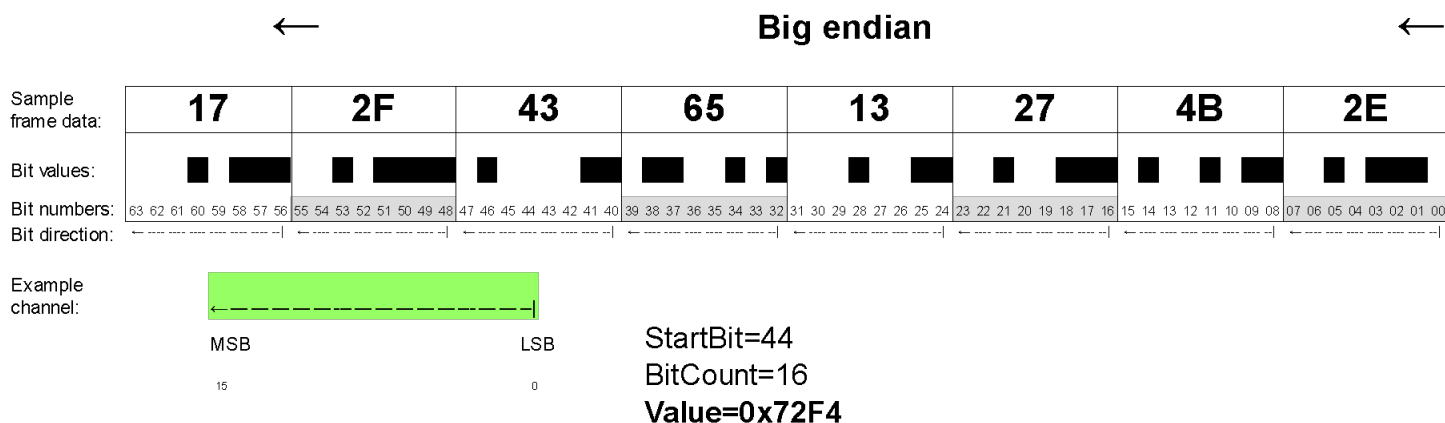
The “raw” value interpreted in Format (3) field can be scaled.

For example, Lambda in the EMU stream is saved as the value 0..255, where:

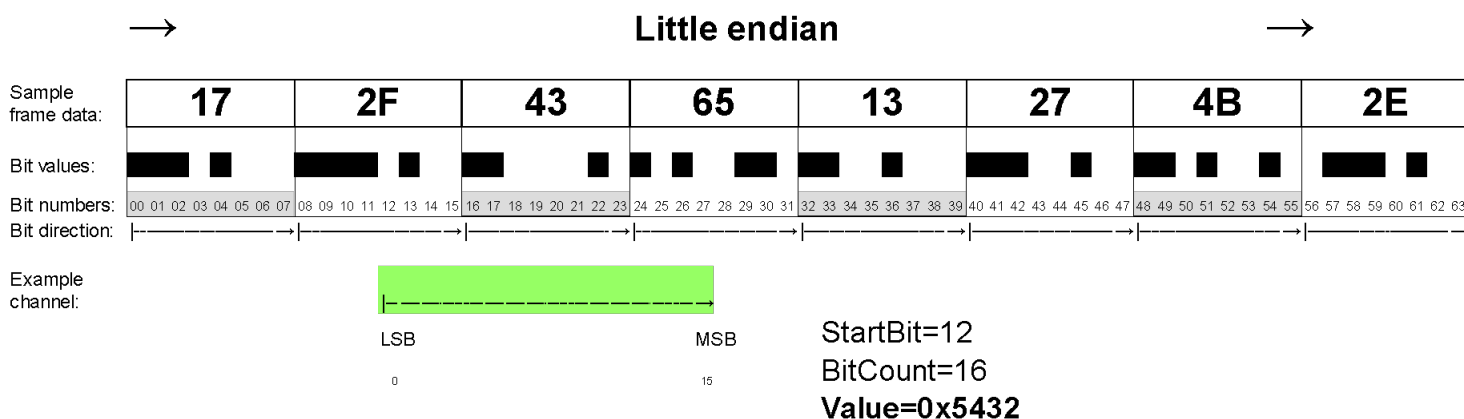
- raw value 0 means Lambda = 0.0,
- raw value 128 means Lambda = 1.0,
- raw value 255 means Lambda= approx. 2.0,

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.

Example for Big Endian, custom Data format:



Example for Little Endian, custom Data format:



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).

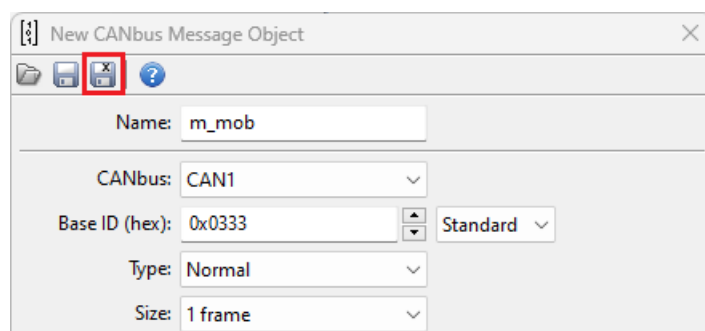
If a given frame cannot be received for a time longer than that defined in **Message Object** configuration (**Timeout** parameter in seconds), two options are available:

- (a) the last value (possibly the default value if a frame was never received) may remain (**Use previous value**);
- (b) 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.

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

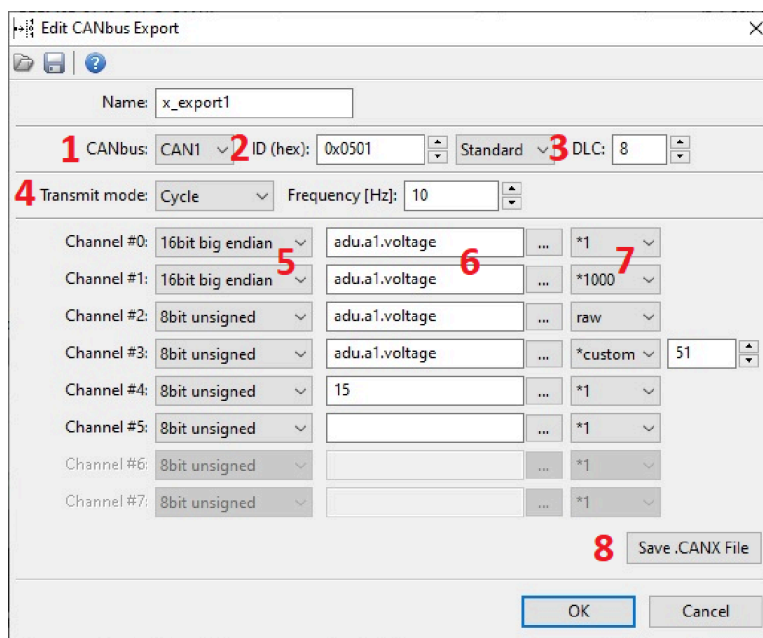
A configured **Message Object** together with all **CANbus Input** channels can be saved into a .CANX file using a toolbar button.



10.6. Sending frames using the CAN bus (CANbus Export)

Access to the CAN bus in the ADU device is completely open, which allows sending any available channel of the device. Frames with any CAN ID can be transmitted on one of two CAN buses.

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



CAN bus selection (1)

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 respect, 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)

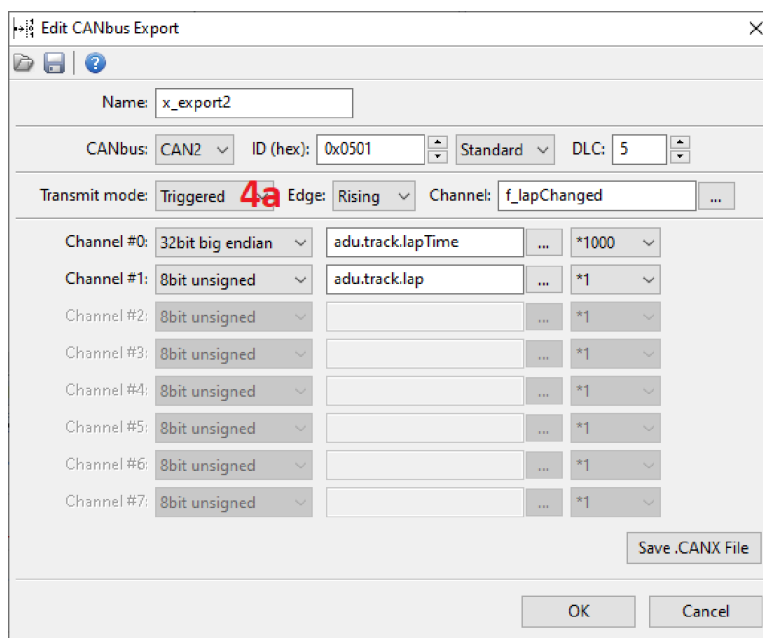
The *DLC* determines the length of a frame: from 0 to 8 bytes.

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 can be sent per second on CAN1 bus and 500 frames on CAN2 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 the 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 sent as the most significant byte, the least significant byte (for example the value 0x1234 will be sent as two consecutive bytes 0x12, 0x34).
- **16bit little endian** – the value is sent as the least significant byte, the most significant byte (for example the value 0x1234 will be sent as two consecutive bytes 0x34, 0x12).
- **32bit big endian** – 4-byte value, sent from the most significant byte to the least significant byte.
- **32bit little endian** – 4-byte value, sent from the least significant byte to the most significant byte.

4-byte values (**32bit**) are used, for example, to transmit the longitude and latitude for the **gps.longitude** and **gps.latitude** channels. In order to maintain adequate accuracy of the transmitted data, the **raw** value may be used, which, for example for gps coordinates corresponds to the actual value (in degrees [**deg**]) multiplied by $\ast 10^7$.

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 a 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 milivolts).
- **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:

| ID | DLC | Bytes | Freq | Count |
|------|-----|-------------------------|---------|-------|
| 501h | 8 | 00 05 13 88 03 FF FF 0F | 10,0 Hz | 1181 |
| | | #0 #1 #2 #3 #4 | | |

- **Channel #0** – 0x0005, i.e. 5 [V]
- **Channel #1** – 0x1388, i.e. 5000 [mV]
- **Channel #2** – 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)

10.7. 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 ADU Client |
| 0x032-0x035 | CAN1 | No, only CAN1 | No, ID is determined | Communication with Light Client |
| 0x400-0x407 | CAN1 | Yes, CAN1 or CAN2 | Yes, ID must be a multiple of 8 | ECUMASTER GPS |
| 0x420-0x42F | CAN1 | Yes, CAN1 or CAN2 | Yes, ID must be a multiple of 16 (0x10) | ECUMASTER Tire Temp Camera (FL, FR, RL, RR) ECUMASTER Disk Temp Camera (FL, FR, RL, RR) |

11. CAN bus keypad support

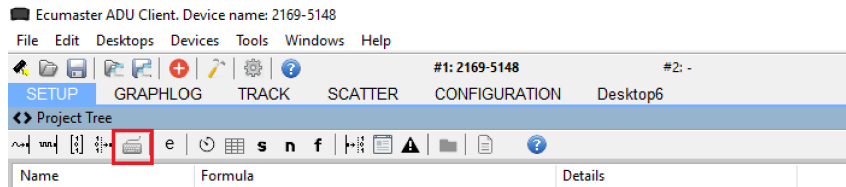


Software version 84.0 and later versions offer the option of connecting and operating a CAN bus keypad.

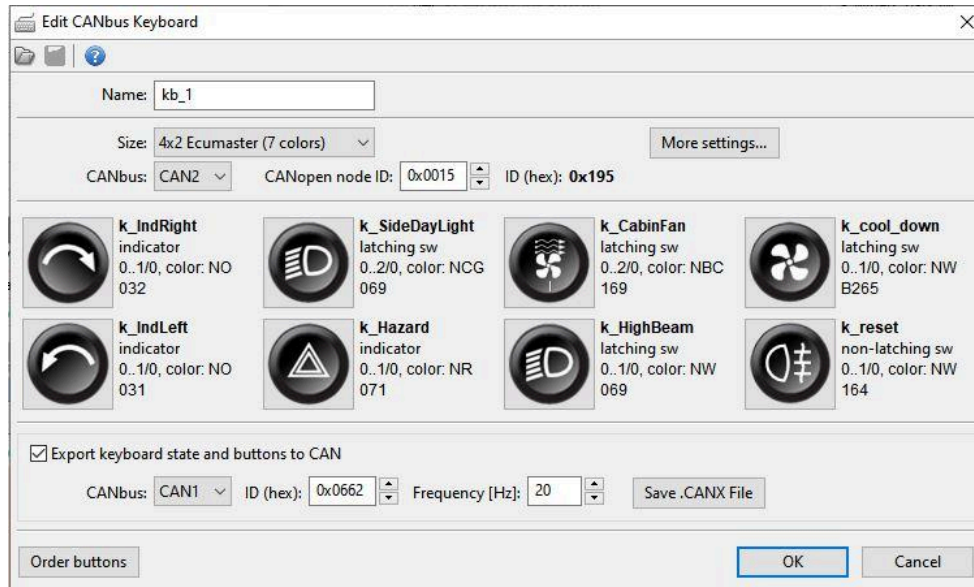
Keyboards from Ecumaster, Grayhill (using the CANopen protocol), MoTeC/RaceGrade, and Emtron are supported. If you're using the 5x3MT keyboard with two encoders, see [Appendix E - How-to Configure 5x3MT Encoders \(on page 217\)](#) for more details.

One keyboard can be connected to the ADU.

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.

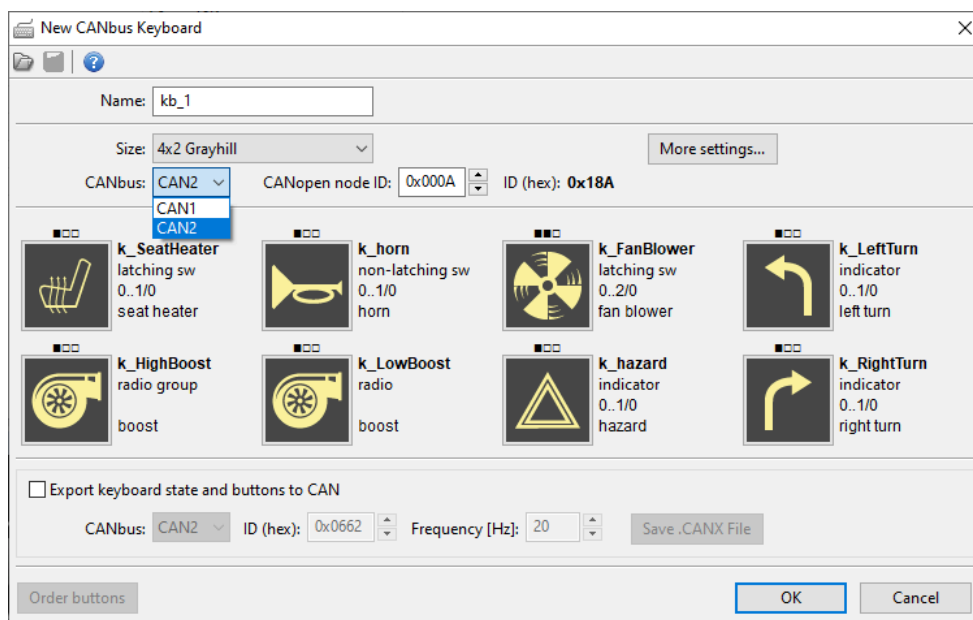


After opening the **CANbus Keyboard** window, individual parameters must be set:



- **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) |

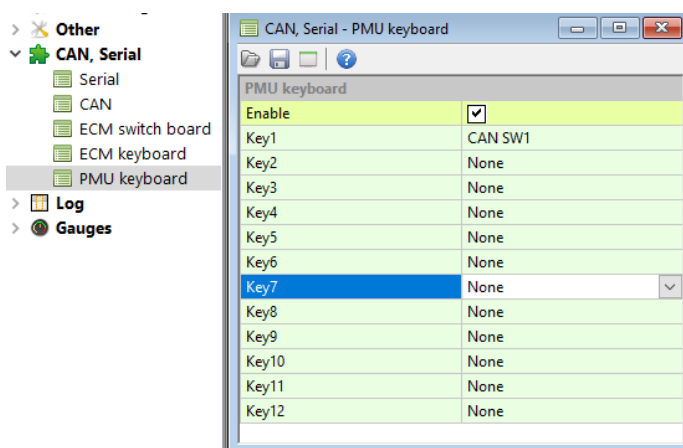
- **CANopen node ID**- the factory setting for Ecumaster keyboards is 0x0015 (this item needs to be changed if there is a **CANopen node ID** conflict with another keyboard connected to the CAN bus).

A detailed description of **CANopen node ID** changes can be found at:

www.ecumaster.com/files/LightClient/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 **ADU** will also work on the **PMUs** and **EMU BLACK**.

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**.

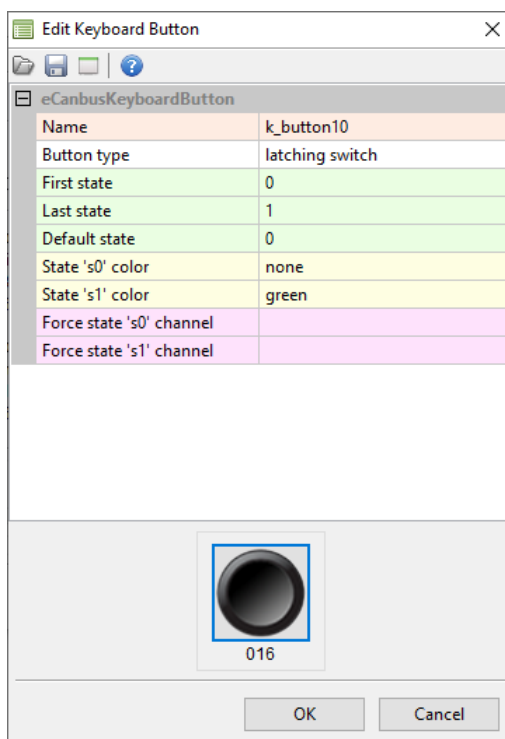


For the **PMU**, simply save the keyboard configuration settings to a **.CANX** file and then load the saved file (Import **.CANX / .DBC** file), which automatically configures the **CANbus Message Objects** and **CANbus Inputs** in the software.

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

| Parameter | Description |
|---------------------|--|
| Name | Button name |
| Button type: | Button mode of operation: <ul style="list-style-type: none"> ◦ 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 |

| Parameter | Description |
|---------------------------------|--|
| | <ul style="list-style-type: none"> ◦ 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. ◦ 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 |

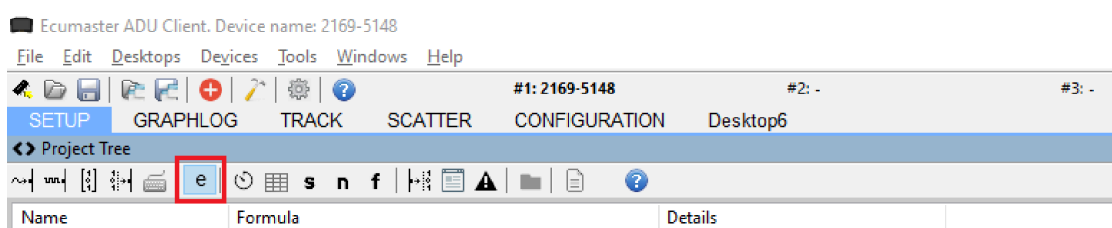


- **Order buttons** - prepares copy-ready information on the numbers of plastic key inserts, with selected laser-engraved icons, for order preparation.

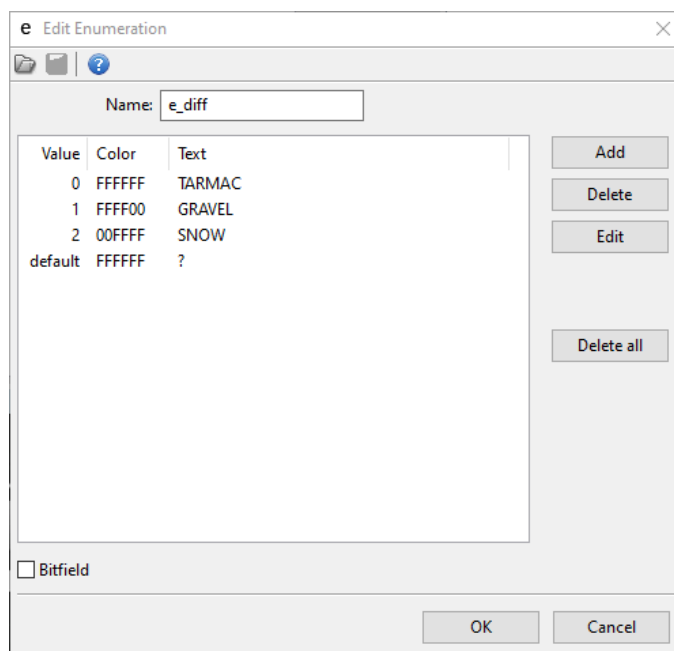
12. Enumeration

Enumeration allows assigning a selected text or color to a specific numerical value. This allows information from the channels to be displayed on a screen in the form of a text message.

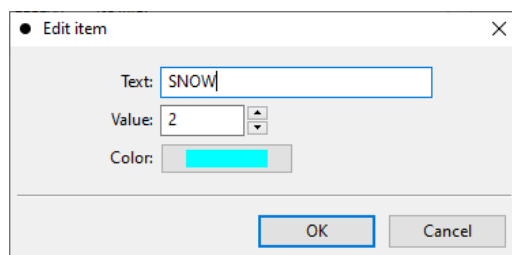
To define an enumeration, click the **e** icon in the **Project Tree** window on the toolbar or select **Enumeration** from the list via **Add**.



After opening the **Edit Enumeration** window, individual attributes must be defined:

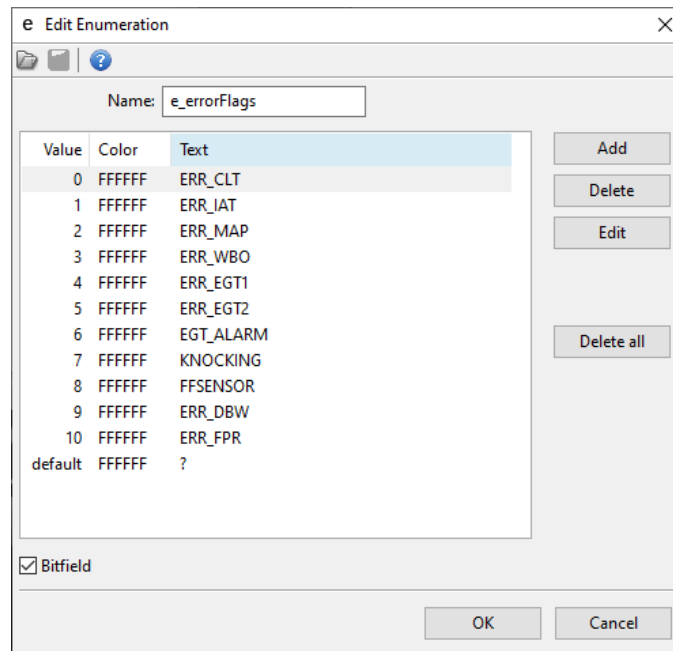


- **Name** - enumeration name
- **Add** - opens the New item window, in which a numerical value (in the decimal system) must be entered, to which the selected text and color are assigned.



- **Delete** - removes the selected assignment
- **Edit** - edits the selected assignment
- **Delete all** - deletes all defined assignments

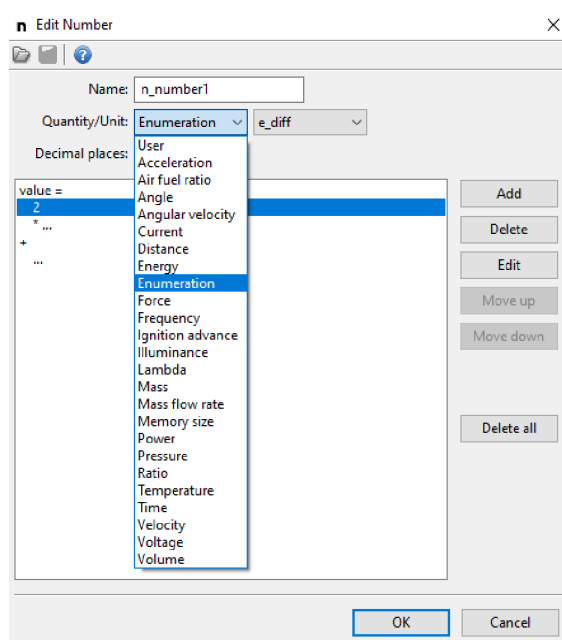
- **Bitfield** - an option that allows up to sixteen messages to be displayed simultaneously by assigning individual messages to the corresponding bit numbers (0 to 15).



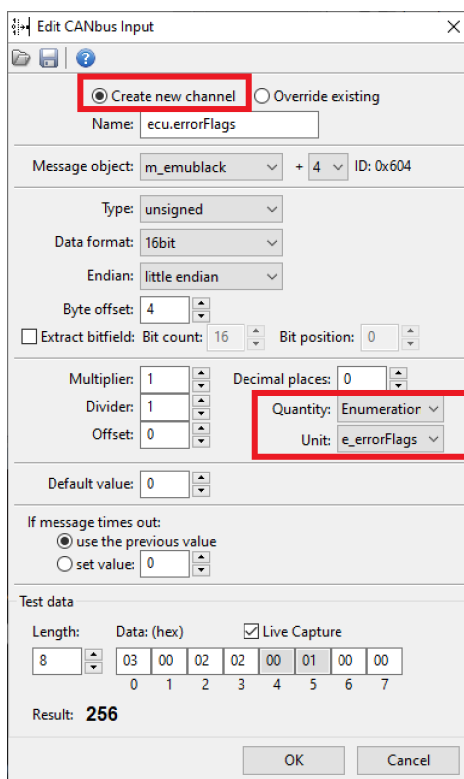
To use an enumeration and display a number in the form of an assigned text or color, it is not enough to simply define an assignment. The created enumeration should also be linked to the channel from which it will be taken. The channels in which enumeration can be used are as follows: **Number**, **Table** and **CANbus Input**.

Enumeration in the mathematical Number channel

In the **Number** channel, the result is a numerical value to which text or color is assigned by enumeration. For this purpose, in the **Quantity / Unit** field, select: **Enumeration** from the first list and the specific enumeration name previously defined by the user from the second list. In this way, enumeration will be associated with a channel. The association will be used when numerical information is displayed in the form of assigned text on the screen.



Enumeration in the *CANbus Input* channel



The assignment of Enumeration to **CANbus Input** is only possible with the **Create new channel** option selected. In the **Quantity:** field, set **Enumeration** and select the name of the previously created enumeration as the unit (**Unit:**).

The value to which the text or color is assigned is **Result**. This is the number resulting from scaling values from the indicated CAN frame by the selected coefficients: *Multiplier*, *Divider*, *Offset* or *Decimal places*.

Enumeration in the *Table* channel.

The enumeration is assigned to the **Table** channel under **Quantity / Unit** item: Select **Enumeration** from the first list and the appropriate name of the previously created enumeration from the second list.

New Table

Name: t_temp

Quantity/Unit: Enumeration e_temperatura

Decimal places: 0

Axis X: channel: ecu.clt

min: 0,0

max: 100,0

step: 10,0

columns: 11

Axis Y: channel: (leave blank for 2D table)

min: 0

max: 100

step: 10

rows: 10

Create

OK Cancel

The individual arguments in the table are assigned values for which text / colour assignments are defined.

Edit Table

Name: t_table2

Quantity/Unit: Enumeration e_temperatura

Decimal places: 0

| | | | | | | | | | | | |
|-----|------|------|------|------|------|------|------|------|------|-------|--|
| 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | |
| 0,0 | 10,0 | 20,0 | 30,0 | 40,0 | 50,0 | 60,0 | 70,0 | 80,0 | 90,0 | 100,0 | |

ecu.clt[°C]

OK Cancel

Display of enumeration on screen

Once an enumeration has been defined and associated with the relevant channel, its result can be displayed on screen in the form of an assigned text / colour. Some page elements in the **Page editor**, such as: **Image**, **Text** and **Grid**.

Image allows for the display of graphics, its color may be defined by a channel under the **Color channel** item.

| | |
|--------------------|---|
| Position | |
| Position X | 97 |
| Position Y | 97 |
| Image | |
| Default color | <input type="checkbox"/> (255,255,255,255) |
| Function color | <input checked="" type="checkbox"/> (255,0,0,255) |
| Texture | tractionControl.png |
| Repeat mode | Stretch |
| Scale | 100 % |
| Mirror X | <input type="checkbox"/> |
| Mirror Y | <input type="checkbox"/> |
| Mode | Normal |
| Color channel | c_diffMode |
| Visibility channel | one |

Text allows the following values to be displayed:

- using text - by selecting the channel with the assigned enumeration under **Channel** item;
- in an appropriate color - by selecting an enumerated channel in the **Color channel** item.

| | |
|--------------------|--|
| Position | |
| Position X | 165 |
| Position Y | 98 |
| Text | |
| Color | <input type="checkbox"/> (255,255,255,255) |
| Font | 4 |
| Italic | <input type="checkbox"/> |
| Two lines | <input type="checkbox"/> |
| Text | |
| Text width | 0 px |
| Text align | Left |
| Channel | c_diffMode |
| Decimal places | 2 |
| Value width | 0 px |
| Value align | Left |
| Unit | e_div |
| Display unit | <input checked="" type="checkbox"/> |
| Custom unit | |
| Color channel | c_diffMode |
| Visibility channel | one |
| Update frequency | 50 Hz |

An example of displaying an enumeration on the screen (*e_diff* connected to the channel *c_diffMode*) using *Image* and *Text*:



Example of on-screen display of enumeration with option *bitfield* (*e_errorFlags*). Depending on the value of the channel to which the sample enumeration is attached, between 0 and 10 messages can appear simultaneously.

ERR_EGT2 EGT_ALARM KNOCKING ERR_DBW ERR_FPR

Grid displays a table, which, in each of its cells, displays values defined by separate channels. The channel determining the colors of the cell values is defined separately for each row of the table.

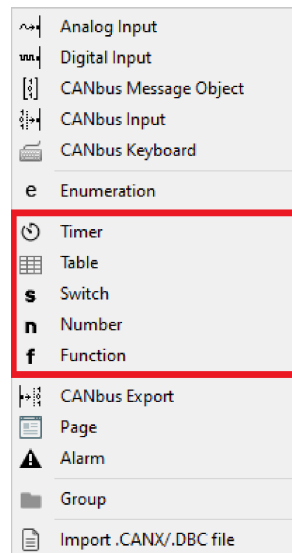
| | |
|---|--|
| + | Position |
| + | General |
| + | Index column |
| + | Description column |
| - | Custom column A |
| | Active <input checked="" type="checkbox"/> |
| | Column width 100 |
| | Decimal places 1 |
| | Text align Left |
| | Channel 1 |
| | Channel 2 n_number4 |
| | Channel 3 |
| | Channel 4 |
| | Channel 5 n_number5 |
| - | Custom column B |
| | Active <input checked="" type="checkbox"/> |
| | Column width 200 |
| | Decimal places 1 |
| | Text align Left |
| | Channel 1 |
| | Channel 2 n_number5 |
| | Channel 3 n_number3 |
| | Channel 4 |
| | Channel 5 |
| + | Custom column C |
| - | Color control |
| | Default text color <input type="color"/> (255,255,255) |
| | Fill color <input type="color"/> (0,37,74,255) |
| | Alternating fill color <input checked="" type="checkbox"/> |
| | Fill color #2 <input type="color"/> (0,27,53,255) |
| | Channel 1 |
| | Channel 2 n_number4 |
| | Channel 3 n_number5 |
| | Channel 4 |
| | Channel 5 n_number3 |

The color of the values on the screen can be assigned in **Color Channel** or defined by a primary color for a channel / condition value of 0 (**Default color** - Image, **Default text color** - Grid, **Color** - Text) and an alternative color for a value different from zero (non-zero) (**function color** - Image, red color - Text and Grid). When selecting an enumeration-related channel (without the bitfield option), the color assigned in the enumeration overrides the primary color. With bitfield enumeration, the color displayed on the screen is the primary or alternative color, depending on the condition met.

| | Image | Text | Grid |
|--|--|-------------------------------|--|
| Color defined directly | 0 - Default color non-zero - function color | 0 - Color non-zero - red | 0 – default text color non-zero - red |
| Color defined by channel with enumeration | Color assigned by enumeration | Color assigned by enumeration | Color assigned by enumeration |
| Channel-defined color with bitfield enumeration | 0 - Default color non-zero - function color | 0 - Color non-zero - red | 0 – default text color non-zero - red |

13. Processing information in the ADU

The ADU has 5 information processing elements:

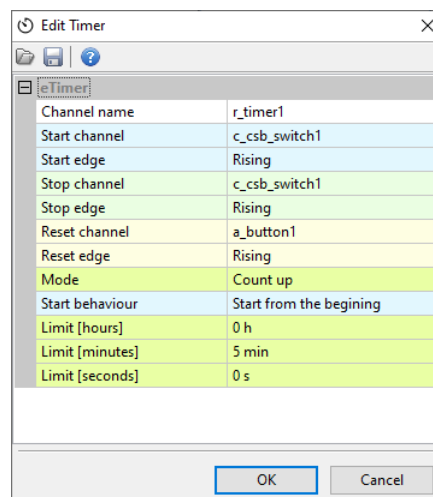


1. **Timers** – counting time
2. **Tables** – lookup tables
3. **Switches** – virtual switches, counters
4. **Numbers** – mathematical channels
5. **Functions** – logical functions

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

13.1. 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 can store times up to 200 hours with an accuracy of up to 0.01 seconds.

Timers are started using a channel specified by the **Start Channel** and an edge defined by the **Start Edge** (*Rising* or *Falling*). Similarly, they can be stopped using the **Stop Channel** and a corresponding **Stop Edge**.

The **Reset Channel** resets the timer to its initial value, whether it's running or stopped. Depending on whether the timer is in *Count Up* or *Count Down* mode, it resets to either zero or the set *Limit*.

When a timer is stopped, a starting edge detected in the Start Channel will trigger behavior defined by **Start Behavior**:

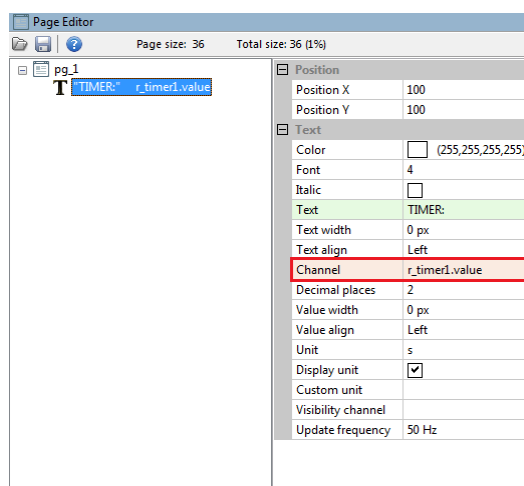
- **Start from Beginning** restarts the timer from the initial value.
- **Resume if Not Yet Elapsed** continues the timer if the limit was not reached before.

If the timer is already running, any further starting edges will be ignored. The timer will only react to a starting edge once it has been stopped or has finished counting. This allows you to start and stop the timer 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.

A timer can be used on a page in the **Text** indicator.



Use the **.value** subchannel for this purpose.

Time can be formatted using the Unit option e.g. using the hh:mm:ss format. Accuracy can be changed using the **Decimal places** parameter.

13.2. Tables – 2D / 3D lookup tables

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.

The 'New Table' dialog box shows the following configuration:

- Name: t_table1
- Quantity/Unit: User (dropdown), user (dropdown)
- Decimal places: 1
- Axis X: channel: adu.a1.voltage (dropdown), min: 0,000, max: 5,000, step: 1,000, columns: 6
- Axis Y: channel: adu.a2.voltage (dropdown), min: 0,000, max: 5,000, step: 1,000, rows: 6 (leave blank for 2D table)
- Create button
- OK and Cancel buttons

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.

The 'New Table' dialog box shows the following configuration and data:

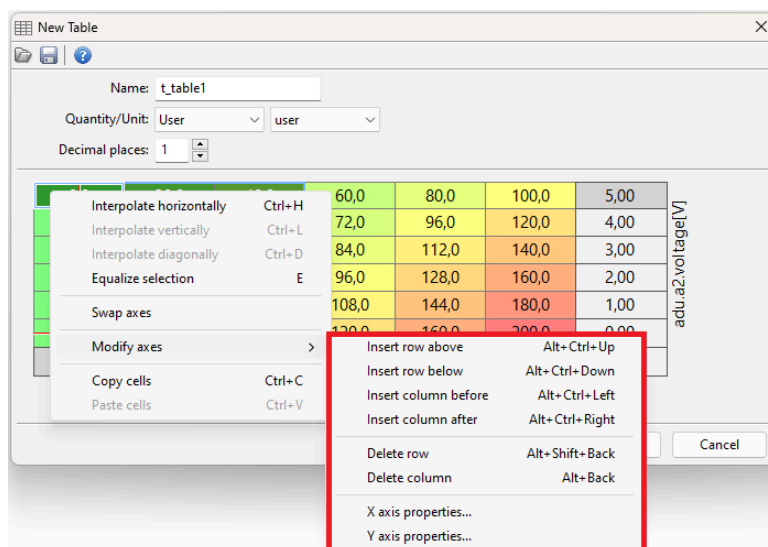
- Name: t_table1
- Quantity/Unit: User (dropdown), user (dropdown)
- Decimal places: 1
- Axis X: channel: adu.a1.voltage (dropdown), min: 0,000, max: 5,000, step: 1,000, columns: 6
- Axis Y: channel: adu.a2.voltage (dropdown), min: 0,000, max: 5,000, step: 1,000, rows: 6 (leave blank for 2D table)
- Create button
- OK and Cancel buttons

The table data is as follows:

| adu.a1.voltage[V] | adu.a2.voltage[V] | 0,0 | 20,0 | 40,0 | 60,0 | 80,0 | 100,0 | 5,00 |
|-------------------|-------------------|------|------|-------|-------|-------|-------|------|
| 0,0 | 0,0 | 24,0 | 48,0 | 72,0 | 96,0 | 120,0 | 140,0 | 4,00 |
| 0,0 | 0,0 | 28,0 | 56,0 | 84,0 | 112,0 | 140,0 | 160,0 | 3,00 |
| 0,0 | 0,0 | 32,0 | 64,0 | 96,0 | 128,0 | 160,0 | 200,0 | 2,00 |
| 0,0 | 0,0 | 36,0 | 72,0 | 108,0 | 144,0 | 180,0 | 200,0 | 1,00 |
| 0,0 | 0,0 | 40,0 | 80,0 | 120,0 | 160,0 | 200,0 | 200,0 | 0,00 |
| 0,00 | 1,00 | 1,00 | 2,00 | 3,00 | 4,00 | 5,00 | 5,00 | |

You can select several cells using 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 |
|--|----------------|---|
| <i>Interpolate horizontally</i> | 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. |
| <i>Interpolate vertically</i> | 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. |
| <i>Interpolate diagonally</i> | 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. |
| <i>Equalize selection</i> | E | Smooths out selected cells |
| <i>Insert row above</i> | Alt+Ctrl+Up | Inserts a row above the selected cell |
| <i>Insert row below</i> | Alt+Ctrl+Down | Inserts a row below the selected cell |
| <i>Insert column before</i> | Alt+Ctrl+Left | Inserting a column to the left of the selected cell |
| <i>Insert column after</i> | Alt+Ctrl+Right | Inserts a column to the right of the selected cell |

| Command | Key shortcut | Description |
|---------------------------|----------------|--|
| 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 |

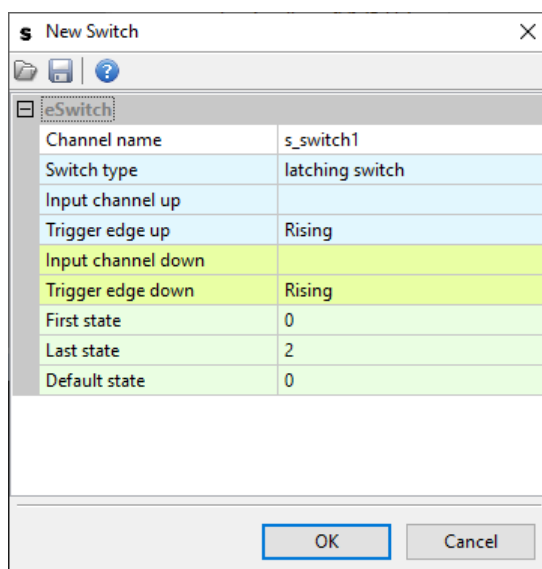
13.3. Switches virtual switches, counters

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 using 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**.

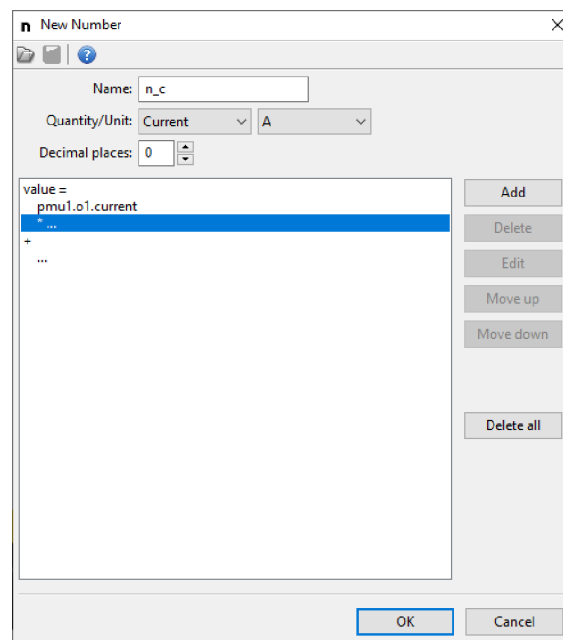
After each appearance of a **Trigger edge down** in the **Input channel down** the element value will increase by 1. If the element reaches a value equal to **First state** and it appears **Trigger edge down**, it will change to the value of **Last state**.



13.4. 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**

*** ...**

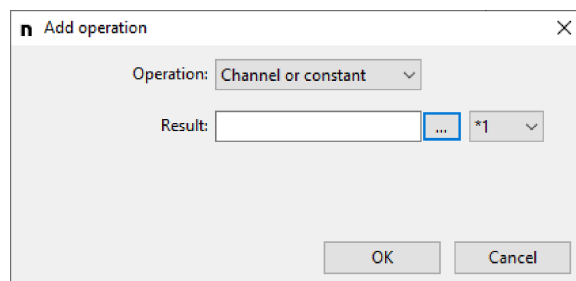
+

...

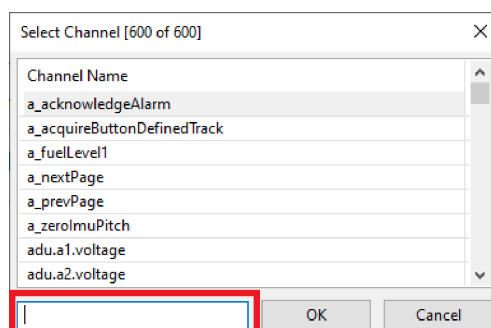
where C1, C2, C3... - is the selected channel or constant value (*Channel or Constant*)

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 (*Delete*), edit (*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 = <i>Result</i> |
| Constant | Result ² | FACTOR = <i>Result</i> |
| Choose | Condition channel Result if true Result if false | if <i>Condition_channel</i> ≠ 0 then FACTOR = <i>Result_if_true</i> else FACTOR = <i>Result_if_false</i> |
| Divide | Value | RESULT := RESULT DIV <i>Value</i> (DIV - integer division; e.g.: 9 DIV 2 = 4) |
| Modulo | Value | RESULT := RESULT MOD <i>Value</i> (MOD - division reminder; e.g.: 9 MOD 5 = 4) |
| Addition | Value 1 Value 2 | FACTOR = <i>Value_1</i> + <i>Value_2</i> |
| Subtraction | Value 1 Value 2 | FACTOR = <i>Value_1</i> - <i>Value_2</i> |
| Min | Value 1 Value 2 | if <i>Value_1</i> < <i>Value_2</i> then FACTOR = <i>Value_1</i> else FACTOR = <i>Value_2</i> |
| Max | Value 1 Value 2 | if <i>Value_1</i> > <i>Value_2</i> then FACTOR = <i>Value_1</i> else FACTOR = <i>Value_2</i> |
| Clamp | Input | if <i>Input</i> < <i>Min</i> then FACTOR = <i>Min</i> |

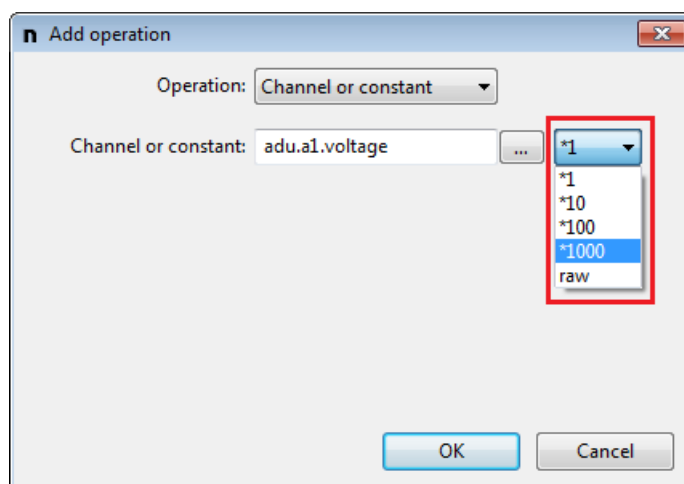
| Operation | Parameter | Pseudocode |
|--------------------------------|--|---|
| | Min Max | else if <i>Input</i> > <i>Max</i> then FACTOR = <i>Max</i> else FACTOR = <i>Input</i> |
| Lookup2 | Channel First index Value[0], Value[1] | if <i>Channel</i> ≤ <i>First_index</i> then FACTOR = <i>Value[0]</i> else FACTOR = <i>Value[1]</i> |
| Lookup3 | Channel First index Value[0], Value[1] Value[2] | if <i>Channel</i> ≤ <i>First_index</i> then FACTOR = <i>Value[0]</i> else if <i>Channel</i> = <i>First_index</i> +1 then FACTOR = <i>Value[1]</i> else FACTOR = <i>Value[2]</i> |
| Lookup4 | Channel First index Value[0], Value[1] Value[2], Value[3] | if <i>Channel</i> ≤ <i>First_index</i> then FACTOR = <i>Value[0]</i> else if <i>Channel</i> = <i>First_index</i> +1 then FACTOR = <i>Value[1]</i> else if <i>Channel</i> = <i>First_index</i> +2 then FACTOR = <i>Value[2]</i> else FACTOR = <i>Value[3]</i> |
| Get temperature element | Temperature x16 channel Element index | Channels: tire temperature camera <i>ttc.tireTemp</i> and brake disc temperature camera <i>btc.brakeTemp</i> Element index: 1-16 (temperature measurement point for the selected sensor) |

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

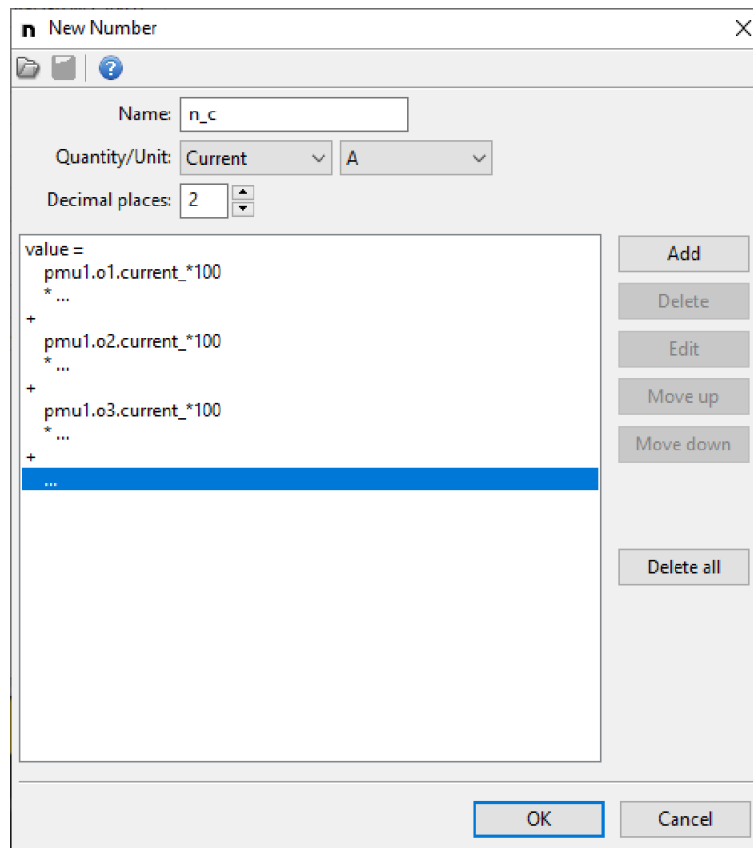
Each mathematical channel can store raw values within the range of [-32768, +32767].

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. $\pm 2 \times 10^9$). For example, calculations can be performed for the following values $1000 \times 1000 / 123$. 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:

pmu_o1.current, *pmu_o2.current*, *pmu_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

n Edit Number

Name:

Quantity/Unit:

Decimal places:

value =

(c_wheelSpeedRL*10 + c_wheelSpeedRR*10)

/ 2

* ...

+ ...

Buttons: Add, Delete, Edit, Move up, Move down, Delete all

Buttons: OK, Cancel

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:

$$\mathbf{n_averageRearSpeed = (c_speedRL*10 + c_speedRR*10) / 2}$$

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. $n_1 = C1 + C2 + C3$

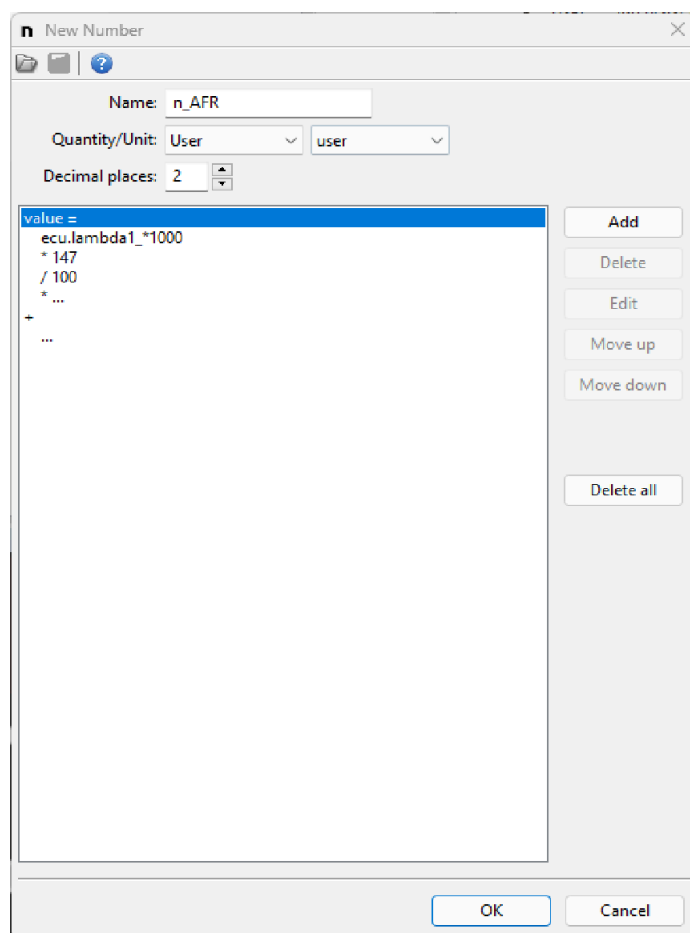
$$n_2 = C4 + C5 + C6$$

$$\mathbf{n_3 = n_1 * n_2}$$

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:

$$n_AFR = (ecu.lambda*1000) * 147 / 100$$

13.5. Functions – logical functions

Logical functions are used to define an extended behaviour of the display depending on the channel input values.

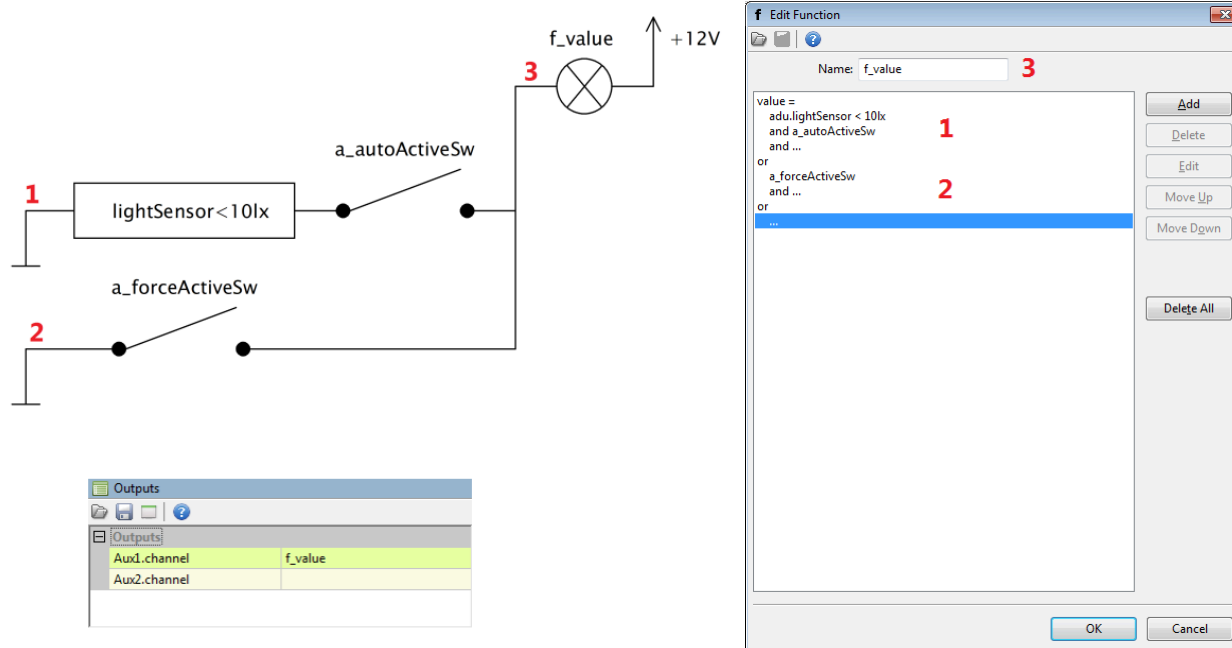
Controlling the bulb using a light sensor

The bulb lights up when at least one of the following conditions is met:

- when it is dark and the automatic mode is active (the *a_autoActiveSw* switch is on);
- when the manual lighting mode is active (the *a_forceActiveSw* switch is on).

Both switches are connected to the analog inputs of the device.

Below an analogy between a logical function and an electrical diagram is presented. A logical function will be real if at least one of the branches is true. A branch, in turn, is true when all operations in such branch are true.



Additionally, you should also set the Aux1 output, so that it can be controlled by the *f_value*:

List of operations available for logical functions.

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

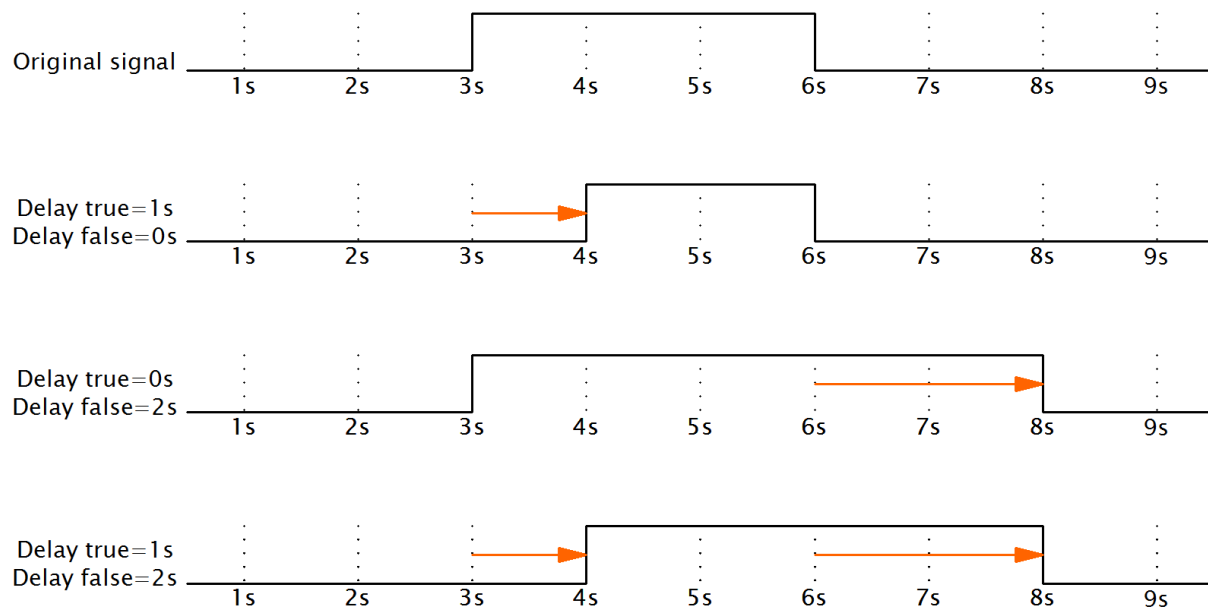
Simple operations are those whose result depends on the input state (alternatively a delay can be used for this result). Simple operations include: testing (*Is False*, *Is True*), (*=*, *≠*, *<*, *≤*, *>*, *≥*) comparisons and logic operations (*And*, *Or*, *Xor*).

**Important:**

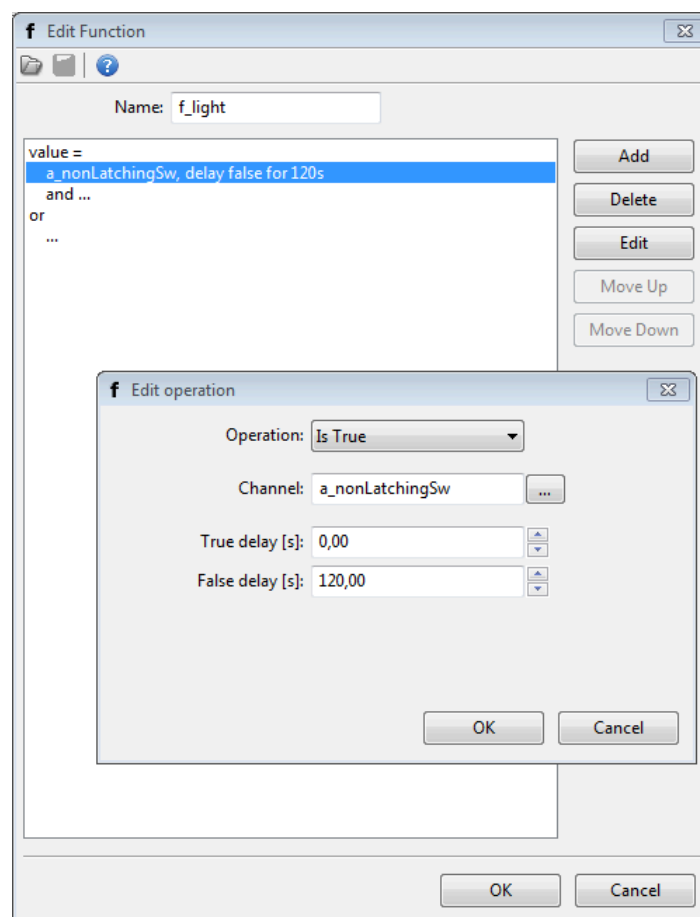
The following description contains **false** and **true** notions. **False** means a value of '0' (zero), and **true** means any value **other** than zero (e.g. '1').

| 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 \neq Constant; returns 0 otherwise. |
| Less | Returns 1 when the Channel value $<$ Constant; returns 0 otherwise. |
| Less or Equal | Returns 1 when the Channel value \leq Constant; returns 0 otherwise. |
| Greater | Returns 1 when the Channel value $>$ Constant; returns 0 otherwise. |
| Greater or Equal | Returns 1 when the Channel value \geq 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

| Signal generating | | | | | | | | | | | | | | | | | |
|--------------------------|--|------------------------|--------------------------|----------------------------|------------------------|-----------------|-----------|---|-----------|-----------------|---|-----------------|-----------------|---|-----------|-----------|----------------|
| Flash | <p>This operation generates impulses so long as the Channel is true (non-zero).</p> <p>When the Channel value assumes false (zero), the operation returns the value 0.</p> <p>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.</p> | | | | | | | | | | | | | | | | |
| Pulse | <p>This operation generates N impulses following appearance of a trigger edge.</p> <p>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).</p> <p>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.</p> | | | | | | | | | | | | | | | | |
| State-storing operations | | | | | | | | | | | | | | | | | |
| Set-Reset Latch | <p>The operation sets a new or returns the previous one according to the settings of the two input channels: Set Channel and Reset Channel.</p> <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> <p>An analogous operation is performed in the electronic SR latch. SR latch): https://en.wikipedia.org/wiki/Flip-flop_(electronics)</p> <p>The initial value of this operation following starting the device can be defined using the Default State parameter.</p> | | Set channel value | Reset channel value | Operation value | true (non-zero) | false (0) | 1 | false (0) | true (non-zero) | 0 | true (non-zero) | true (non-zero) | 0 | false (0) | false (0) | previous value |
| Set channel value | Reset channel value | Operation value | | | | | | | | | | | | | | | |
| true (non-zero) | false (0) | 1 | | | | | | | | | | | | | | | |
| false (0) | true (non-zero) | 0 | | | | | | | | | | | | | | | |
| true (non-zero) | true (non-zero) | 0 | | | | | | | | | | | | | | | |
| false (0) | false (0) | previous value | | | | | | | | | | | | | | | |

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 channelvalue | Reset channelvalue | 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.



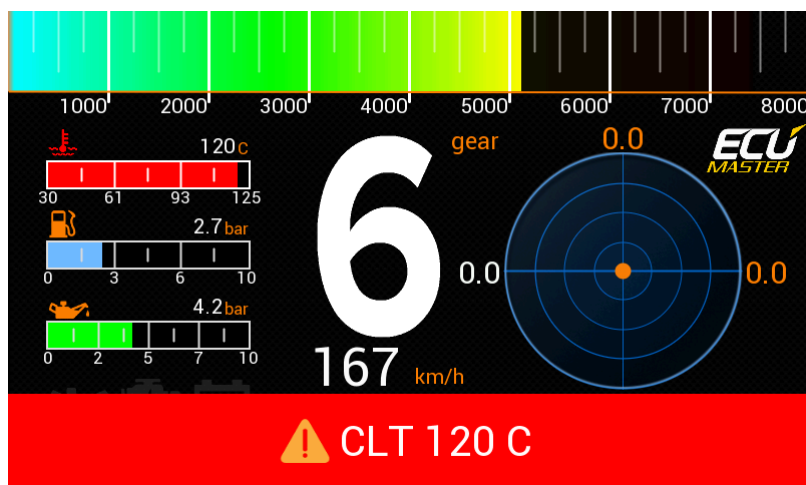
Important:

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

14. Alarms

The role of alarms is to display information about emergency states detected by the device.

A screen displaying an example alarm



1. Basic alarm configuration

| New Alarm | |
|-------------------------------|-------------------------------------|
| eDashboardAlarm | |
| Name | al_clt |
| General | |
| Type | Error |
| Text (use # to display value) | CLT # |
| Position | Bottom |
| Channel | |
| Channel | ecu.clt |
| Decimal places | 0 |
| Unit | °C |
| Display unit | <input checked="" type="checkbox"/> |
| Condition | |
| Condition | Greater or Equal |
| Value | 110 °C |
| Qualifier | |
| Use qualifier | <input type="checkbox"/> |
| Delay | |
| Guard time | 0 s |
| Acknowledge | |
| Allow acknowledge | <input checked="" type="checkbox"/> |
| Step | 5 °C |
| Presentation | |
| Min show time | 3 s |
| Retrigger time | 3 s |
| OK Cancel | |

Configuration consist in defining the **Channel** and the **Condition / Value** that will activate an alarm. You should also define the text (**Text** parameter).

The alarm text can show the current channel value. To do this, you should put the special symbol # where the channel value with a unit is to appear.

If you need to display the # character alone, enter two # characters (##).

2. Qualifier

A precondition, the so-called **Qualifier**, can additionally be entered into the alarm configuration. For example, when an alarm is to be checked only at a high speed, you can use the following precondition: **ecu.rpm > 4000**.

3. Guard time

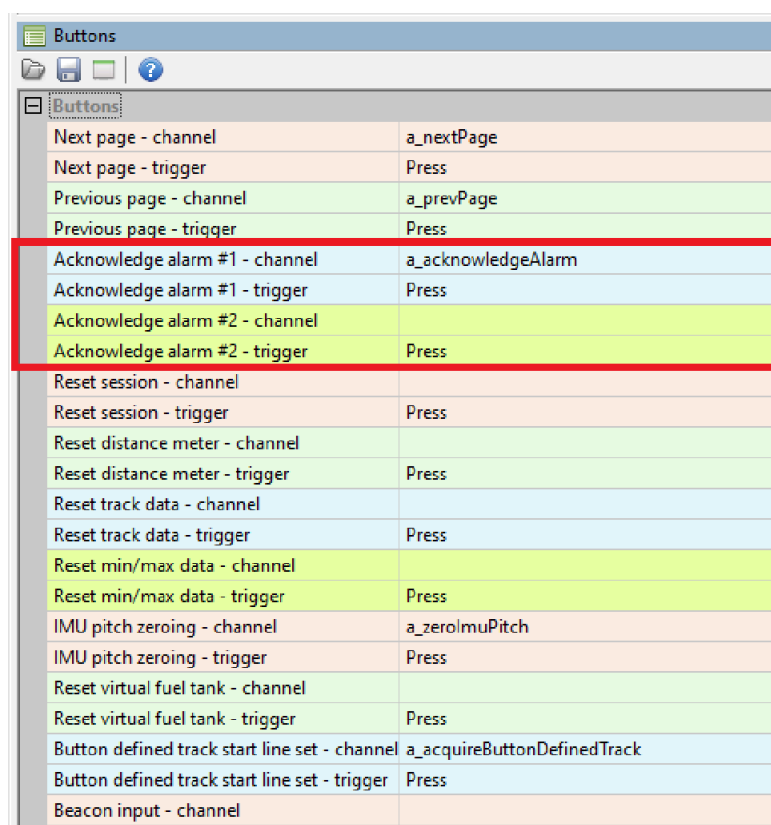
Specifies the minimum time for which the condition must be met for the alarm to be displayed.

4. Presentation

To prevent an alarm from appearing and disappearing and thus causing confusion, it is worth defining two additional parameters:

- **Min show time** - defines the minimum time during which an alarm will be visible. An alarm will be visible during this time even if the alarm condition is no longer met.
- **Retrigger time** - defines the minimum time during which an alarm will not reappear after it disappears.

5. Acknowledge



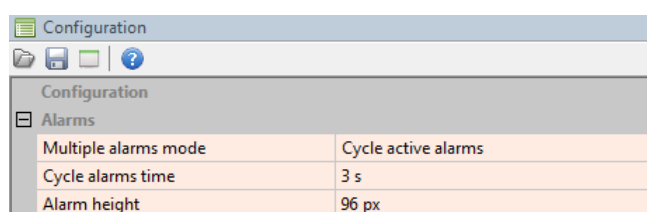
| Buttons | |
|---|-----------------------------|
| Next page - channel | a_nextPage |
| Next page - trigger | Press |
| Previous page - channel | a_prevPage |
| Previous page - trigger | Press |
| Acknowledge alarm #1 - channel | a_acknowledgeAlarm |
| Acknowledge alarm #1 - trigger | Press |
| Acknowledge alarm #2 - channel | |
| Acknowledge alarm #2 - trigger | Press |
| Reset session - channel | |
| Reset session - trigger | Press |
| Reset distance meter - channel | |
| Reset distance meter - trigger | Press |
| Reset track data - channel | |
| Reset track data - trigger | Press |
| Reset min/max data - channel | |
| Reset min/max data - trigger | Press |
| IMU pitch zeroing - channel | a_zeroImuPitch |
| IMU pitch zeroing - trigger | Press |
| Reset virtual fuel tank - channel | |
| Reset virtual fuel tank - trigger | Press |
| Button defined track start line set - channel | a_acquireButtonDefinedTrack |
| Button defined track start line set - trigger | Press |
| Beacon input - channel | |

After reading the alarm message, the driver can confirm it (the **Allow acknowledge** option). The alarm will disappear and the threshold value will move by a predefined **Step**.

In the **Buttons** section of the **Configuration** panel one can define one or two buttons for acknowledging an alarm.

6. Global settings

It is possible for two alarms to occur simultaneously.



| Configuration | |
|----------------------|---------------------|
| Alarms | |
| Multiple alarms mode | Cycle active alarms |
| Cycle alarms time | 3 s |
| Alarm height | 96 px |

There are two ways to solve this problem (depending on the settings of the **Multiple alarms mode**):

- **Cycle active alarms** causes the active alarms to be displayed consecutively, each for a specified period of time: **Cycle alarms time**.
- **Only one active with highest priority (last in Project Tree)** displays only the alarm with the highest priority. The highest priority is understood to be in the **Project Tree** - alarm, which is at the end, has the highest priority.

In addition, there is a channel called **adu.anyAlarmActive**, which indicates whether any defined alarms are active. A list of active alarms can be displayed in a separate logging panel: **Alarms / active**, where you can also check their status.

15. Logging channels

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

| Name | 5 [Base] | Cond2 | Cond3 | Cond4 |
|-----------------------|----------|-------|-------|-------|
| Acc/Gyro | | | | |
| Gyro X | 25 | 25 | 25 | 25 |
| Gyro Y | 25 | 25 | 25 | 25 |
| Gyro Z | 25 | 25 | 25 | 25 |
| Acc X | 125 | 125 | 125 | 125 |
| Acc Y | 125 | 125 | 125 | 125 |
| Acc Z | 125 | 125 | 125 | 125 |
| ADU | | | | |
| Brightness | 25 | 25 | 25 | 25 |
| Reset detector | 25 | 25 | 25 | 25 |
| Status | 25 | 25 | 25 | 25 |
| User error | 25 | 25 | 25 | 25 |
| Board temperature | 25 | 25 | 25 | 25 |
| Battery voltage | 25 | 25 | 25 | 25 |
| 5V output | 25 | 25 | 25 | 25 |
| Led driver voltage | 25 | 25 | 25 | 25 |
| Light sensor | 25 | 25 | 25 | 25 |
| Analog inputs | (25) | (25) | (25) | (25) |
| a_nextPage | 25 | 25 | 25 | 25 |
| a_prevPage | 25 | 25 | 25 | 25 |
| CANbus inputs | (25) | (25) | (25) | (25) |
| CANbus Message Object | (25) | (25) | (25) | (25) |
| CANbus State | | | | |
| Digital inputs | (25) | (25) | (25) | (25) |
| ECU | | | | |
| ecu.rpm | 25 | 25 | 25 | 25 |
| ecu.gear | 25 | 25 | 25 | 25 |
| ecu.tps | 25 | 25 | 25 | 25 |
| ecu.map | 25 | 25 | 25 | 25 |
| ecu.mgp | 25 | 25 | 25 | 25 |
| ecu.boost | 25 | 25 | 25 | 25 |
| ecu.baro | 25 | 25 | 25 | 25 |
| ecu.clt | 25 | 25 | 25 | 25 |
| ecu.iat | 25 | 25 | 25 | 25 |
| ecu.speed | 25 | 25 | 25 | 25 |
| ecu.battery | 25 | 25 | 25 | 25 |
| ecu.egt1 | 25 | 25 | 25 | 25 |
| eri.ent2 | 25 | 25 | 25 | 25 |
| Usage | 6 41% | 41% | 41% | 41% |

25 logged part(s), 408 B.

It is worth noting that the same frequencies are used for both logging to the USB storage device and for logging directly to the ADU Client programme on a PC.

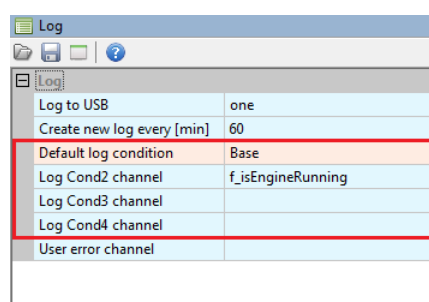
In the configuration window, we can distinguish the following elements:

- 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).

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.

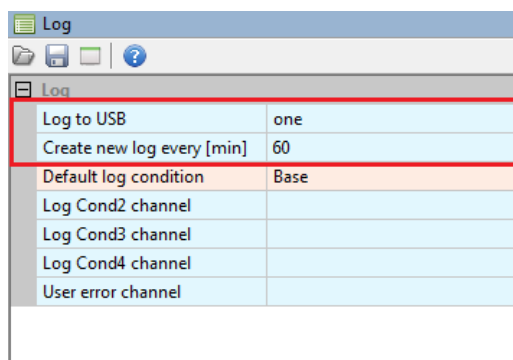
| Key: | Logging frequency: |
|-------|---|
| Alt+~ | Deactivation of channel / group logging |
| Alt+1 | 1 Hz |
| Alt+2 | 5 Hz |
| Alt+3 | 25 Hz |
| Alt+4 | 50 Hz |
| Alt+5 | 125 Hz |
| Alt+6 | 250 Hz |
| Alt+7 | 500 Hz |

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.

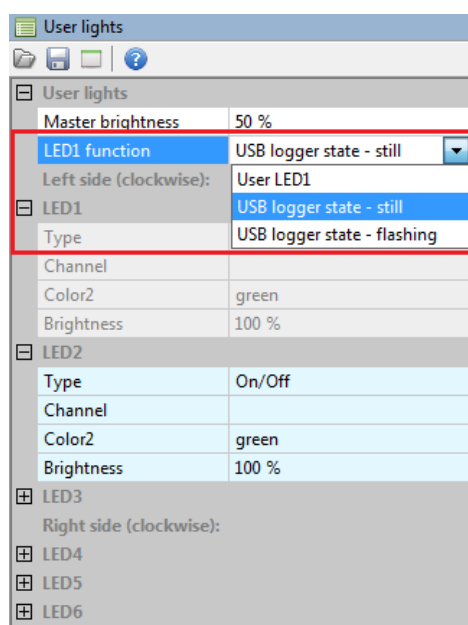


16. Logging to a USB storage device

Logging to a USB storage device is activated by default. Immediately after detecting a USB storage device connected to the device, ADU starts to save the log file in the .ADULOG format. It saves the ADU client logs in the same format. Files are created cyclically at defined time intervals (default 1 h, maximum 5 h). The shorter the intervals, the smaller the files and the easier their subsequent analysis.



The logging state can be shown using the LED1 diode.

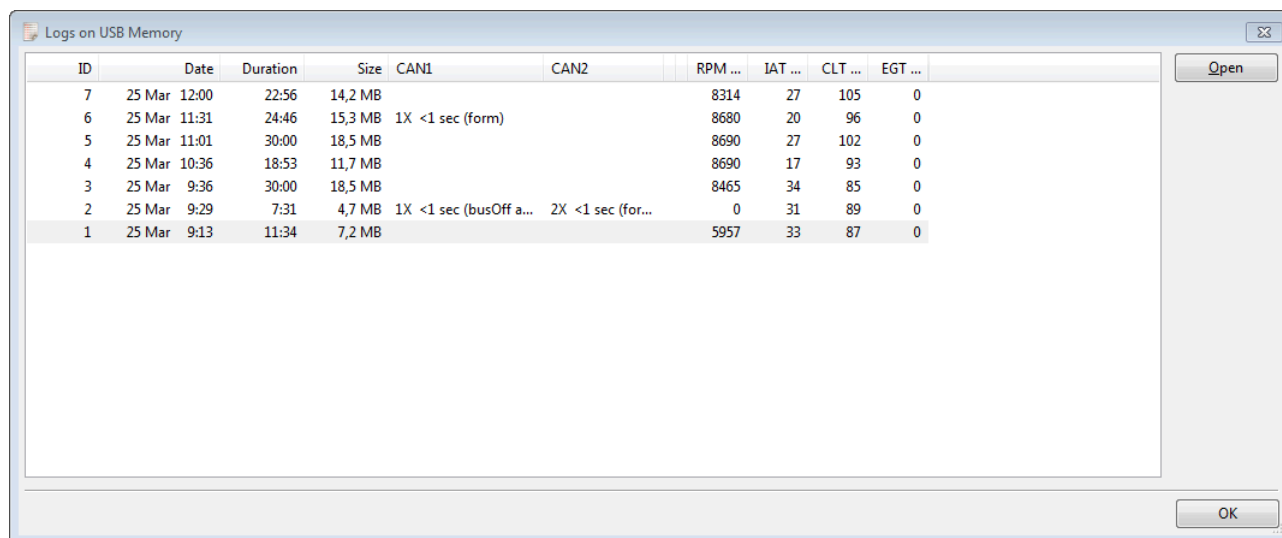


This is done using the **LED1 function** parameter, which has two options available:

- **User LED1** (the diode works just like the other **User light LED**)
- **USB logger state - still** (the green diode stays on throughout the entire log saving time)
- **USB logger state - flashing** (the green diode flashes each time a log is saved - option useful for a diagnosis)

| Color: | Description: |
|------------------------------------|---|
| ● Blue | USB storage device is not connected to the ADU device. |
| ● Orange | The process of recognising the USB storage device stick and the process of loading the file system information is in progress. A state usually lasts a few seconds. |
| ● Green | Log file saving in progress. |
| ● Red | Error: the USB storage device is not compatible with the ADU device. The reason could be, for example, that the ADU does not support the exFAT file system on the USB stick. |
| ●● Red and Green interchangeably. | USB storage device is full. |
| ●● Orange and blue interchangeably | The USB stick is of poor quality - it saves logging data slower than the ADU generates it. |

IMPORTANT: The USB logger state - flashing option can be used to diagnose problems with poor-quality USB storage devices. After starting recording a new log, the green diode will be lit continuously for a few seconds, but later, during operation, green should not be visible longer than 1 second. Otherwise, it means that the memory does not guarantee the correct write speed. Once the data has been saved, the USB storage device should be removed from the port connected to the ADU and connected to a PC. Then, in the ADU Client, from the **Devices** menu, select the **Receive logs** command (Shift+F4 key) to view the logs stored in memory.



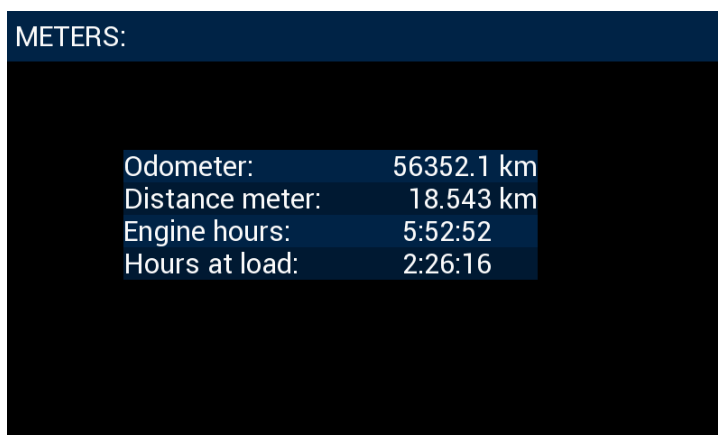
| ID | Date | Duration | Size | CAN1 | CAN2 | RPM ... | IAT ... | CLT ... | EGT ... |
|----|--------------|----------|---------|------------------------|-------------------|---------|---------|---------|---------|
| 7 | 25 Mar 12:00 | 22:56 | 14,2 MB | | | 8314 | 27 | 105 | 0 |
| 6 | 25 Mar 11:31 | 24:46 | 15,3 MB | 1X <1 sec (form) | | 8680 | 20 | 96 | 0 |
| 5 | 25 Mar 11:01 | 30:00 | 18,5 MB | | | 8690 | 27 | 102 | 0 |
| 4 | 25 Mar 10:36 | 18:53 | 11,7 MB | | | 8690 | 17 | 93 | 0 |
| 3 | 25 Mar 9:36 | 30:00 | 18,5 MB | | | 8465 | 34 | 85 | 0 |
| 2 | 25 Mar 9:29 | 7:31 | 4,7 MB | 1X <1 sec (busOff a... | 2X <1 sec (for... | 0 | 31 | 89 | 0 |
| 1 | 25 Mar 9:13 | 11:34 | 7,2 MB | | | 5957 | 33 | 87 | 0 |

17. Permanent meters

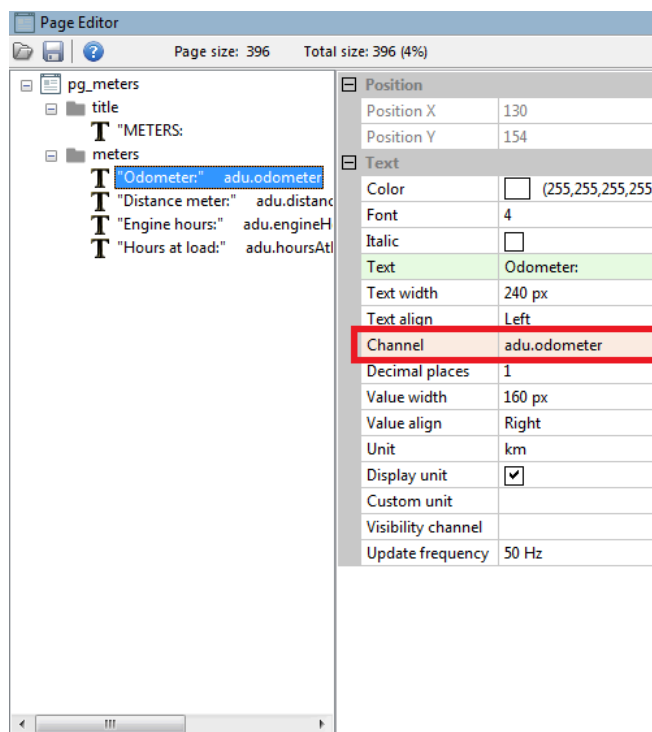
The ADU is equipped with 4 permanent meters, i.e. meters the value of which is kept in memory even after switching the device off.

| Meter name | Channel | Description |
|----------------------------|--------------------------|---|
| Odometer | <i>adu.odometer</i> | An odometer, a travel distance meter |
| Distance meter | <i>adu.distanceMeter</i> | An additional, resettable odometer. Reset is performed via the button defined in the <i>Buttons > Reset distance meter - channel > trigger</i> panel |
| Engine hour meter | <i>adu.engineHours</i> | It measures the number of hours the engine has run. (<i>ecu.rpm > 0</i> channel) |
| Hours at load meter | <i>adu.hoursAtLoad</i> | It measures the number of hours the engine ran under a big load. The parameters for a big engine load can be defined in the Configuration / Hours at load meter: <ul style="list-style-type: none">• <i>Minimal ecu.rpm</i>• <i>Minimal ecu.tps</i>• <i>Minimal ecu.map</i> |

A screen presenting meters with example values

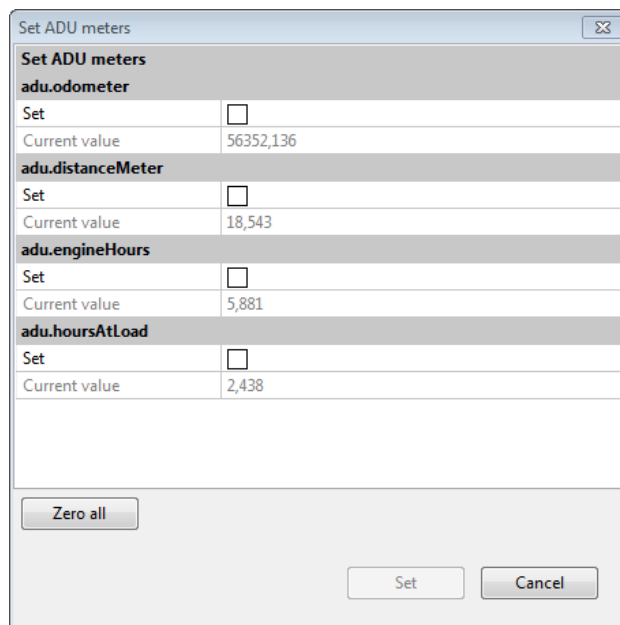


To add a meter to a page, insert the *Text* object and select a channel from the above table.



17.1. Resetting / changing meter status

Changes to the status of the meters can be carried out from the ADU Client. It is worth setting the actual vehicle mileage as the initial odometer mileage.



This can be done using the menu command **Tools > Set meters...**

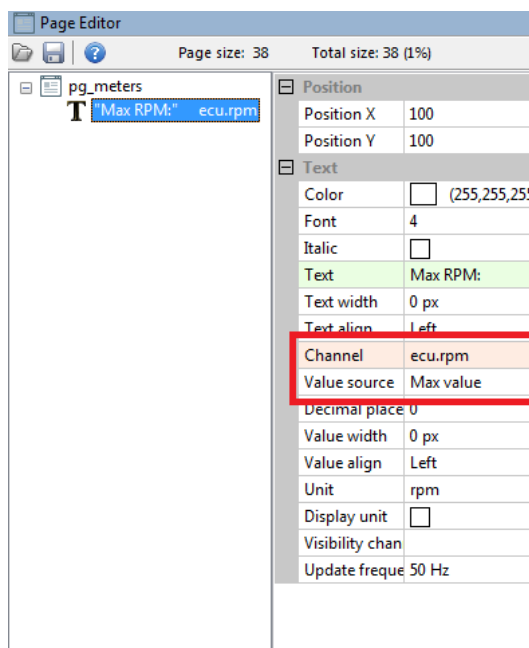
To set a given meter value, check the Set selection field for the meter and enter the new value.

To set all values at zero, you can use the *Zero all* button.

After entering the desired values accept them by using the **Set** button at the bottom of the window.

18. Min / max values for ECU channels

Each channel from the ECU group (e.g. *ecu.rpm*) records its minimum and maximum value. These values can be displayed on a page using the **Text** object.



Choose the correct channel from the ECU group and set the **Value source** parameter to **Min value** or **Max value**.

The **Value source** parameter appears only for ECU group channels.

If a channel has not recorded any value yet (e.g. there is no CANbus Input element recording data in this channel), the **Text** object will display a “?” (question mark).

Just like for permanent meters, the min / max values of ECU channels are recorded in non-volatile memory of the device and remain stored even after the device has been switched off.

However, the process of registering the min / max values can be started again (by deleting the previous values). For this purpose, use the **Reset min / max mode** option in the **Min / Max reset** section of the **Configuration** menu. The option offers two options to choose from:

- **Every firmware upgrade** – resetting the min / max values with each replacement of the device software
- **Every power off** – resetting the min/max values each time the device is turned on

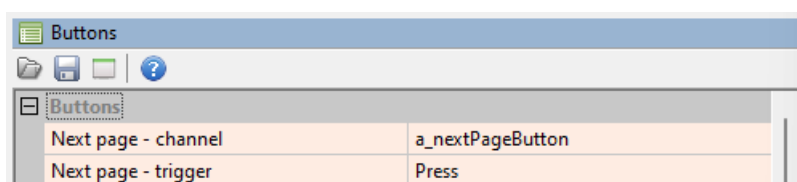
Independently of the above options, a manual delete button can also be defined and made available to the driver. The button can be defined in the **Buttons > Reset min / max data channel > triggerpanel**

19. Panels

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

19.1. Buttons

The **Buttons** panel makes it possible to assign buttons to different internal functions of the device, such as toggling between the pages, cancelling alarms, cancelling times for a given track, etc. A given function (e.g. opening the next page) is configured by selecting a channel / function representing a button and the button interpretation methods (trigger).



There are 4 channel interpretation types (triggers):

Press – function activated by pressing a button

Release – function activated by releasing a button

Click – function activated by quickly pressing and releasing a button

Hold – function activated by pressing and holding a button

| Position | Description |
|-------------------------------|--|
| Next page | Toggling to the next page |
| Previous page | Toggling to the previous page |
| Acknowledge alarm#1 | Cancelling the alarm displayed on the screen |
| Acknowledge alarm#2 | Cancelling the alarm displayed on the screen |
| Reset session | Deletes: internal clock Session time , current lap time CurrentLapTime , last lap time LastLapTime , predicted lap time PredictiveLapTime , the channel recording lap number adu.track.lap , the channel recording the time difference between the best lap and the current lap adu.track.gainLoss , the channel recording the current lap time adu.track.lapTime and the channel recording the distance travelled from the start line adu.track.lapDistance |
| Reset distance meter | Cancels the distance meter (the adu.distanceMeter channel) |
| Reset distance meter 2 | Cancels the distance meter (the adu.distanceMeter2 channel) |

| Position | Description |
|--|--|
| Freeze / unfreeze distance meter | Temporarily stops updating the value of the adu.distanceMeter channel. When you freeze the meter, the channel shows the value from the moment the button was pressed, while the internal distance measurement continues. When unfreezing, the channel value updates to account for the distance covered while the display was frozen. |
| Freeze / unfreeze distance meter 2 | Temporarily stops updating the value of the adu.distanceMeter2 channel. When you freeze the meter, the channel shows the value from the moment the button was pressed, while the internal distance measurement continues. When unfreezing, the channel value updates to account for the distance covered while the display was frozen. |
| Reset track data | Cancels the list of the best times and the reference lap for the current racing track |
| Reset min / max data | Deletes the min / max values for the ecu.* channels. |
| IMU pitch zeroing | Calibrates the overload sensor (accelerometer) |
| Reset virtual fuel tank | Resets the virtual fuel tank, a function used at the end of refuelling |
| Button defined track start line set | Determines the start / finish line when defining a new track |
| Beacon input | Channel input from an external beacon. It informs the ADU about the completion of another lap on a track. |

19.2. Shift light

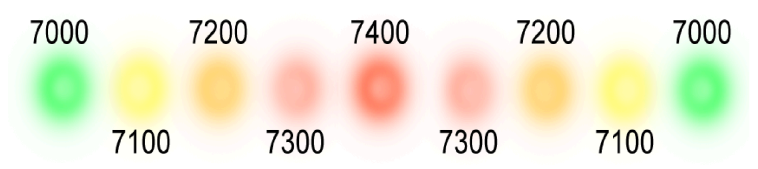
Shift light uses the upper LEDs of the device to indicate the optimal gear change time. Three methods of LEDs control are available. The first of them is a parametric method where for each gear you can define the moment when the first LED lights up. The second method involves a 3D table where for each LED and a given gear it is possible to input the engine speed at which the LED will light up. The third method consists in sending information about the shift light status from the ECU, which shall result in the diodes lighting up.

Parametric control

To activate the parametric shift light, select **Parametric** in the **Control type** field.

| Parameter | Description |
|----------------------------|--|
| Brightness | LED brightness |
| Maximum RPM | The maximum RMP at which the upper LEDs begin to flash red |
| Neutral and reverse | The RPM at which the first LED for the reverse and neutral gear lights up |
| 1 to 8 | The RPM at which the first LED for gears 1 to 8 lights up |
| RPM channel | Channel with the current engine speed. |
| Gear channel | Channel with the current gear. If not defined, neutral gear will be selected by default. |

Below you can find the RMP at which the individual diodes will light up when the Maximum RPM setting is defined at 7400, and the initial RPM for a given gear is at 7000. Above 7400 RMP all diodes begin to flash red.



Control using a 3D table

To activate control using a 3D table, select *Table 3D* from the *Control type* field.

| Parameter | Description |
|-------------------------------|--|
| Brightness | LED brightness |
| RPM channel | Channel with the current engine speed. |
| Gear channel | Channel with the current gear. If not defined, the neutral gear will be selected by default. |
| Shift light color #n | Color assigned to a given diode |
| Flash when all leds on | Activation of the LED flash mode. When active, the LEDs flash instead of being on continuously. |
| Flash color | Selects flashing LED color. If you choose <i>Use default colors</i> , the color of the flashing diodes will correspond to the colors assigned in the <i>Shift light color</i> fields. Otherwise, the diodes will change color to the one selected in this field. |

The following map lights up individual LEDs between 6100 and 6400 RMP with at 50 step.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|------|------|------|------|------|------|------|------|
| 9 | 6100 | 6150 | 6200 | 6250 | 6300 | 6350 | 6400 | Gear |
| 8 | 6100 | 6150 | 6200 | 6250 | 6300 | 6350 | 6400 | |
| 7 | 6100 | 6150 | 6200 | 6250 | 6300 | 6350 | 6400 | |
| 6 | 6100 | 6150 | 6200 | 6250 | 6300 | 6350 | 6400 | |
| 5 | 6100 | 6150 | 6200 | 6250 | 6300 | 6350 | 6400 | |
| 4 | 6100 | 6150 | 6200 | 6250 | 6300 | 6350 | 6400 | |
| 3 | 6100 | 6150 | 6200 | 6250 | 6300 | 6350 | 6400 | |
| 2 | 6100 | 6150 | 6200 | 6250 | 6300 | 6350 | 6400 | |
| 1 | 6100 | 6150 | 6200 | 6250 | 6300 | 6350 | 6400 | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |

Shift light LED

First gear values re used for reverse and neutral gear.

Control using an external channel (CAN bus)

To activate control using a channel / variable, select **CAN bus** in the **Control type** field.

| Parameter | Description |
|----------------------------|--|
| Brightness | LED brightness |
| Num states | Number of the shift light (4-6) states |
| Shift light channel | Channel with information about the number of diodes currently on |

This control lights up respective diodes depending on the value of the variable assigned to the **Shift light channel** field.

Pit limiter indicator

The following settings allow you to configure the behavior and appearance of LED indicators for the pit limiter. These options include channel activation, color customization for odd and even LEDs, and the cycle time for alternating between colors.

| Parameter | Description |
|---------------------------|--|
| Activation channel | The pit limiter indicator remains on while this channel has a value of 1. |
| Odd LEDs colour 1 | Primary color for odd-numbered LEDs (1, 3, 5, ...). |
| Even LEDs colour 1 | Primary color for even-numbered LEDs (2, 4, 6, ...). |
| Odd LEDs colour 2 | Alternate color for odd-numbered LEDs (1, 3, 5, ...). |
| Even LEDs colour 2 | Alternate color for even-numbered LEDs (2, 4, 6, ...). |
| Cycle time | Duration for switching between the primary and alternate colors of the LEDs. |

If the pit limiter is active, the LEDs change colour according to the parameters set.

19.3. User lights

The **User lights** panel allows you to control the color of the LEDs located on the left and the right hand side of the device. The LEDs are numbered clockwise LED1 to LED6.

The screenshot shows the 'User lights' configuration window. It contains a tree view on the left with expandable sections for 'User lights', 'LED1', 'LED2', 'LED3', 'LED4', 'LED5', and 'LED6'. The main area displays the settings for the selected LED. The 'User lights' section includes 'Master brightness' (50 %) and 'LED1 function' (USB logger state - still). Below this, the 'Left side (clockwise):' section contains settings for LED1, LED2, and LED3. The 'Right side (clockwise):' section contains settings for LED4, LED5, and LED6. The settings for each LED include 'Type', 'Channel', 'Color1', 'Color2', 'Color3' (for LED4), and 'Brightness'.

| Section | Parameter | Value |
|-------------------------|-------------------|--------------------------|
| User lights | Master brightness | 50 % |
| | LED1 function | USB logger state - still |
| Left side (clockwise): | LED1 | |
| | Type | On/Off |
| | Channel | |
| | Color2 | green |
| | Brightness | 100 % |
| LED2 | Type | On/Off |
| | Channel | f_egtAlarm |
| | Color2 | red |
| | Brightness | 100 % |
| | LED3 | |
| LED3 | Type | Choose |
| | Channel | f_oilAlarm |
| | Color1 | blue |
| | Color2 | red - blinking |
| | Brightness | 100 % |
| Right side (clockwise): | LED4 | |
| | Type | Choose + Override |
| | Channel | f_cltOver70C |
| | Color1 | blue |
| | Color2 | green |
| | Channel3 | f_cltOver110C |
| | Color3 | red |
| | Brightness | 100 % |
| | LED5 | |
| | LED6 | |

The LEDs are used to inform about the condition of the vehicle or emergency situations. Colors and channels can be assigned for each LED that will affect them.

The LED1, depending on the selection of **LED1 function**, can be assigned a user control method (**User LED1**) or used as a USB log state indicator (**USB logger state**). For more information, see *Logging on to a USB stick*.

The **Master brightness** parameter allows you to set the brightness jointly for all LEDs on a scale of 0-100%.

The LED#/**Brightness** parameter sets the brightness of an individual LED.

We can select the control method for each LED:

| Type | Parameters | Description |
|--------------------------|--|--|
| On / Off | Channel Color2 | When Channel value is true (non-zero), the LED color is as defined by Color2 ; otherwise it is off. |
| Choose | Channel Color1 Color2 | When Channel value is true (non-zero), the LED color is as defined by Color2 ; otherwise it has the Color1 . |
| Choose + Override | Channel Color1 Color2 Channel3 Color3 | When Channel3 value is true (non-zero), the LED color is as defined by Color3 (irrespective of the value of the variable in the Channel field). Otherwise, the color is defined by the Channel value. When Channel value is true (non-zero), the LED color is as defined by Color2 ; otherwise it is Color1 . |

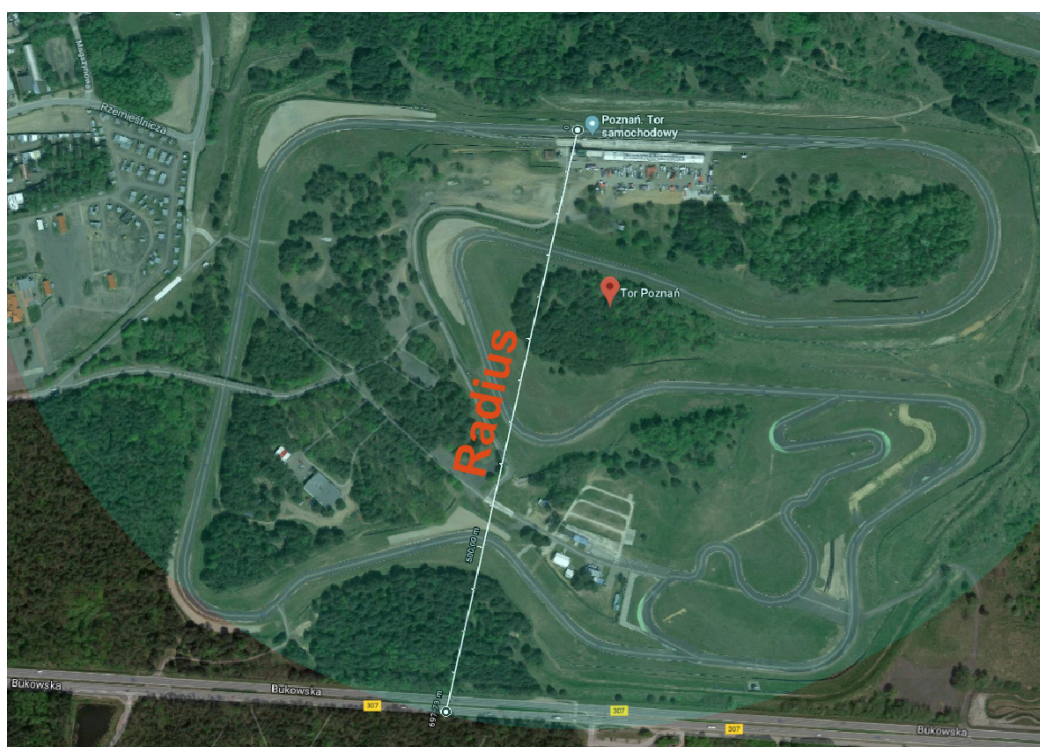
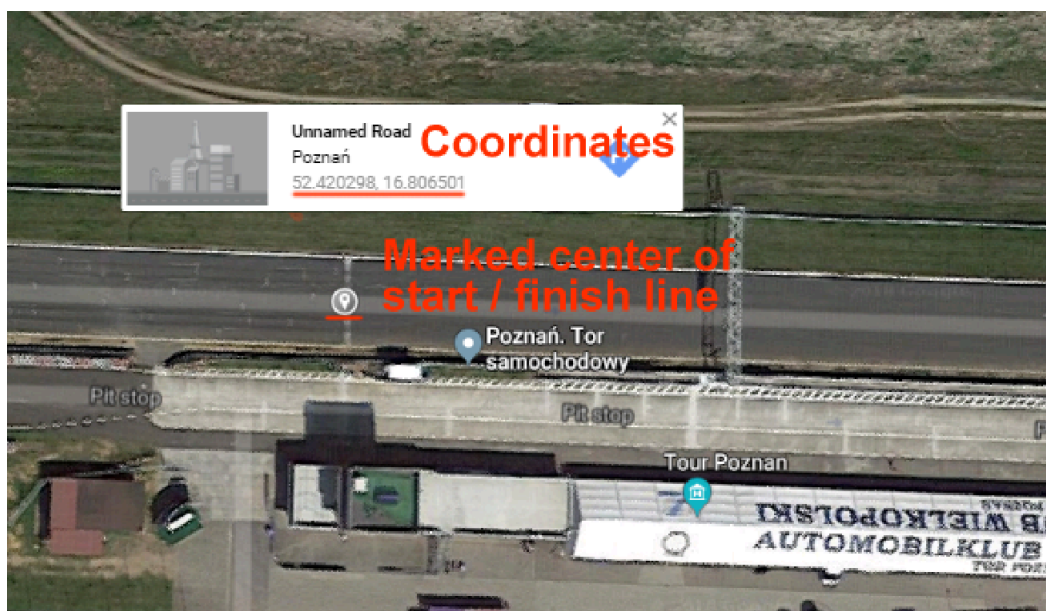
19.4. User tracks

The ADU device includes a built-in database of car racing tracks popular all over the world. This database continues growing with each new version of the software. The currently available tracks can be found in Appendix no.1. If a track is not available or if a track configuration differs from that assumed in the device, you can define a track of your own. You should enter the coordinates (latitude and longitude) of the centre of the start / finish line.

You can define up to 5 tracks of your own. If the track area coincides with a track area defined in the device, the one defined by the user will have priority.

| Parameter | Description |
|---------------------|---|
| Enable | Enables a given track. |
| Name | Track name |
| Latitude | Latitude of the centre of the start / finish line |
| Longitude | Longitude of the centre of the start / finish line |
| Track length | Track length in meters |
| Track width | Track width |
| Track radius | Radius of the circle defining the track area. The centre of the circle is defined as the centre of the start / finish line. |

The following drawings are a graphical representation of the track in Poznań.



19.5. User track defined by the button

It is possible to define an own race track using an external button connected to the ADU.

| Buttons | |
|--|---------|
| Next page - channel | |
| Next page - trigger | Press |
| Previous page - channel | |
| Previous page - trigger | Press |
| Acknowledge alarm #1 - channel | |
| Acknowledge alarm #1 - trigger | Press |
| Acknowledge alarm #2 - channel | |
| Acknowledge alarm #2 - trigger | Press |
| Reset session - channel | |
| Reset session - trigger | Hold |
| Reset distance meter - channel | |
| Reset distance meter - trigger | Press |
| Reset distance meter 2 - channel | |
| Reset distance meter 2 - trigger | Press |
| Freeze/unfreeze distance meter - channel | |
| Freeze/unfreeze distance meter - trigger | Press |
| Freeze/unfreeze distance meter 2 - channel | |
| Freeze/unfreeze distance meter 2 - trigger | Press |
| Reset track data - channel | |
| Reset track data - trigger | Click |
| Reset min/max data - channel | |
| Reset min/max data - trigger | Release |
| IMU pitch zeroing - channel | 0 |
| IMU pitch zeroing - trigger | Hold |
| Reset virtual fuel tank - channel | 0 |
| Reset virtual fuel tank - trigger | Press |
| Button defined track start/finish line set - channel | |
| Button defined track start/finish line set - trigger | Press |
| Beacon input - channel | 0 |

The button should be defined in the Buttons panel. When the driver presses the button for the first time, the GPS coordinates of the vehicle are downloaded. For a track to be defined, the driver must complete a full lap. If the driver presses the button a second time during a lap, the track definition process will be interrupted. Information about the process of defining a track can be displayed on a specially prepared Overlay page. To load its template, create a new page, press the Open file icon in the window defining the page layout and load the *ov_buttonDefinedTrackOverlay.aduepg* file. When the button is first pressed, the following page will be displayed.



It will disappear automatically at the end of the definition process - when the driver has completed a full lap. The track defined in this way will be given the name "Button defined track" and can be displayed using the \$(TRACK) command in the Text control. The overlay (shown above) uses the

adu.track.userButtonDefinedTrackAcquiring channel as a display condition (the channel assumes value 1 when defining a new track). The ADU supports 3 types of track definition. The first type is the tracks defined in the device's internal software (built-in), the second type is user-defined tracks in the ADU software (5 tracks in the User tracks panel) and the third type is button-defined tracks. The priority of these tracks is as follows:

1. Button-defined tracks
2. User tracks
3. Embedded tracks

19.6. Fuel level filter

The **Fuel level filter** function is used for filtering the signal from the tank fuel level sensor.

The result of operation of this function is saved in the **ecu.fuelLevel** channel.

In **two fuel level sensor mode (average)** mode the average of both sensors is stored on the **ecu.fuelLevel1** channel and the **ecu.fuelLevel2** channel assumes the 0.0 value. In **two fuel level sensors (separate level)** mode the signal from each sensor is recorded in a separate channel - the first one in **ecu.fuelLevel1** and the second one in **ecu.fuelLevel2**.

| Parameter | Description |
|--------------------------------------|---|
| Fuel level mode | Fuel level mode: one fuel level sensor two fuel level sensor (average) - average of two fuel level sensors stored in the channel ecu.fuelLevel1 two fuel level sensors (separate level) |
| Fuel level 1 – source channel | Channel / variable that determines the amount of fuel in the tank based on the signal from the first sensor (for one fuel level sensor mode) or from the average of two sensors (for two fuel level sensor (average) mode). A 2D table with calibration is most often used to indicate the fuel level. |
| Fuel level 2 – source channel | Channel / variable indicating the amount of fuel in the tank on the basis of the signal from the second sensor (for two fuel level sensor mode (separate level)) |
| Maximum change step | The maximum allowed fuel level change during one second. |
| Filter time | The time when samples are to be collected and averaged. The longer the time, the more robust the filtering. |

| Parameter | Description |
|-----------------------------------|---|
| Disable filter when no RPM | Switching off the fuel level filter when the engine is not revving (useful for refuelling); window checked by default |
| RPM channel | Engine speed information channel |

In a typical car the fuel level sensor is a potentiometer that requires an additional pull-up resistor (these are usually small values ranging between 100–200 ohm). Connect the signal to an analog input and select Calibrated analog sensor. In this way, a calibration of the fuel quantity for a given voltage can be carried out in the table.

| 0 | 10 | 30 | 40 | 70 | 100 |
|-------------|------|------|------|------|------|
| 0,50 | 1,00 | 2,00 | 3,00 | 4,00 | 4,50 |
| Voltage [V] | | | | | |

If the size of the table needs to be changed, right-click the table cells and select Modify Bins / Insert cell (to add a cell) or Modify Bins / Delete cell (to delete it).

Then enter the name of the (**a_fuelLevelInput**) channel in the Source channel field. After filtering the value is available in the `ecu.fuelLevel` channel.

19.7. OBD 2

The OBD 2 panel is used for configuring the communication using the OBD 2 communication protocol. The read channels are divided into two groups. Fast channels (such as RPM, TPS) are marked green, while slow channels (such as CLT, Fuel Level) are marked pink. The more channels that are selected from a given group, the lower their log-in frequency will be. The read channels are mapped to `ecu.*` channels. It should be pointed out that some engine management do not allows all of the specified channels to be read.

| Parameter | Description |
|---------------|---|
| Enable | Activates support for the OBD 2 protocol on CAN bus 2. |
| Mode | Off - the device is inactive and not engaged with the vehicle's systems. |

| Parameter | Description |
|--|--|
| | <p>Passive - the device reads data only when another active device requests it, preventing communication disruptions when multiple devices are connected. Ideal for monitoring without initiating data queries.</p> <p>Active - the device actively requests real-time data from the vehicle's systems, providing continuous and direct access to selected parameters. Only one device should operate in this mode at a time to maintain efficient and reliable communication.</p> |
| RPM | Engine speed readout. Parameter shown in the log as <i>ecu.rpm</i> |
| Intake manifold absolute pressure | The absolute pressure in the intake manifold. Parameter shown in the log as <i>ecu.map</i> |
| Vehicle speed | Vehicle speed. Parameter shown in the log as <i>ecu.speed</i> |
| Throttle position | Throttle position. Parameter shown in the log as <i>ecu.tps</i> |
| Timing advance | Ignition timing angle. Parameter shown in the log as <i>ecu.ignAngle</i> |
| Fuel pressure | Fuel pressure. Parameter shown in the log as <i>ecu.fuelPress</i> |
| Engine coolant temperature | Engine coolant temperature. Parameter shown in the log as <i>ecu.clt</i> |
| Intake air temperature | Temperature in the intake manifold. Parameter shown in the log as <i>ecu.iat</i> |
| Fuel level | Fuel level. Parameter shown in the log as <i>ecu.fuelLevel</i> |
| Barometric pressure | Barometric pressure. Parameter shown in the log as <i>ecu.baro</i> |
| Ambient air temperature | Temperature outside the vehicle. Parameter shown in the log as <i>ecu.ambientAirTemp</i> |
| Ethanol fuel % | Content of ethanol in fuel. Parameter shown in the log as <i>ecu.ethanolContent</i> |
| Engine oil temperature | Engine oil temperature. Parameter shown in the log as <i>ecu.oilTemp</i> |
| Update rate | A parameter determining the frequency of sending inquiries to the engine control unit. The higher the frequency of sending requests, the higher the frequency of logging can be obtained. Unfortunately, not all control units are capable of responding with a frequency of 100 Hz and may require a lower frequency. The lowest possible frequency is 1 Hz. |

19.8. Outputs

The Outputs panel is used for configuring the ADU outputs control (AUX1, AUX2, Analog out).

| Parameter | Description |
|---------------------|---|
| Aux1.channel | A channel / variable controlling the AUX 1 output. 0 means an inactive output, 1 means an output is switched to ground. |
| Aux2.channel | A channel / variable controlling the AUX 2 output. 0 means an inactive output, 1 means an output is switched to ground. |
| AOut.channel | A channel / variable controlling the AUX2 output. 0 corresponds to 0 V voltage, and 5000 corresponds to 5 V (mV scale). |

19.9. Autobrightness

The *Autobrightness* table defines the brightness of the screen and the leds as a function of the external light intensity.

The method for calculating the brightness of the screen and the leds (all values are comprised within the range 0% and 100%) is shown below.

LCD brightness = Autobrightness

User led brightness = Autobrightness * User led master brightness * Led brightness

Shift light brightness = Autobrightness * Shift light master brightness

The automatic brightness is additionally controlled by a *Configuration* panel parameter named *Adaptation rate*. It determines the maximum permitted brightness changes per second.

19.10. Virtual fuel tank

The Virtual fuel tank function allows the user to calculate the amount of fuel remaining in the tank. Usually, this method of calculation is much more accurate than a calculation using a fuel level sensor. The function is based on the fuel consumption information from the ECU (ecu.usedFuel channel). It calculates not only the amount of fuel remaining in the tank, but also the number of laps the vehicle can run on that fuel (based on the fuel consumption information from the previous lap). This information allows adapting the race strategy to the amount of fuel remaining in the tank.

Parameters

| Parameter | Description |
|----------------------------------|--|
| Fuel tank size definition | <p>Fixed volume - tank is always filled to its maximum volume (defined by Fuel tank size (fixed))</p> <p>Defined by channel - volume of fuel in the tank after the refill is defined with a channel (Fuel tank size channel [L])</p> |
| Fuel tank size (fixed) | Capacity of fuel tank in litres. This parameter is active when the Fuel tank size definition is set to Fixed Volume |
| Defined by channel | Channel defines total volume of fuel in the tank after refill (what was left + what was filled). This parameter is active when the Fuel tank size definition is set to Defined by channel |
| Used fuel channel | The channel contains used fuel information sent by the ECU By default, this is the ecu.usedFuel channel. |
| Max used fuel change | Maximum permissible difference between two consecutive readings of fuel consumption information. If the change is greater than this parameter, it will be ignored. The parameter also enables the function to work correctly if the fuel consumption channel in the ECU is reset. |
| Fuel usage correction | Correction factor when there is a difference between the fuel consumption information provided by the ECU and the actual fuel consumption. This parameter is selected empirically based on the measured actual fuel consumption. |
| Stage mode channel | <p>The Virtual Fuel Tank can monitor fuel consumption and distance traveled simultaneously in two modes: stage and road. Each mode has two channels for distance and used fuel.</p> <p>The Stage mode channel determines which mode is active.</p> <p>If the value of the channel assigned in this field is 0 - used fuel and covered distance is assigned to the road channel.</p> <p>If the value of the channel assigned in this field is 1 - used fuel and covered distance is assigned to the stage channel.</p> |

In addition, in the Buttons panel a **Reset virtual fuel tank** button has been defined. This button is crucial to the whole method, as the driver has to reset the function during refuelling. The value of the fuel remaining in the tank is then set to the value of the tank capacity, and all data is

counted anew. All parameters are stored in the Flash memory of the device, so that they will also be remembered by the ADU when the ignition is switched off. It should be emphasised that the correct connection of the power supply to the device is required for this function to work correctly, i.e. a separate connection of the *battery* and a separate *ignition switch*. In this way, when the ignition is switched off, the parameters are saved in the Flash memory.

Logging channels

| Channel | Description |
|---|---|
| <i>adu.vft.fuelTankSize</i> | Currently used fuel tank size |
| <i>adu.vft.remainingFuel</i> | Amount of fuel remaining in the tank |
| <i>adu.vft.numLapsOnRemainingFuelAvg</i> | Number of laps that can be completed using the fuel remaining in the tank, based on the average fuel consumption (Fuel used per lap avg) |
| <i>adu.vft.numLapsOnRemainingFuelLast</i> | Number of laps that can be completed using the fuel remaining in the tank, based on the fuel consumption of the last lap (Fuel used per last lap) |
| <i>adu.vft.usedFuelPerLap</i> | Average fuel consumption per lap (since resetting the fuel tank) |
| <i>adu.vft.usedFuelPerLastLap</i> | Fuel consumption on the last lap |
| <i>adu.vft.usedFuelFromLastReset</i> | Fuel used since last virtual fuel tank reset |
| <i>adu.vft.isStageMode</i> | Shows whether fuel is assigned to stage or road |
| <i>adu.vft.stage.distanceFromLastReset</i> <i>adu.vft.road.distanceFromLastReset</i> | Distance traveled in stage/road mode since the last refueling |
| <i>adu.vft.stage.usedFuelFromLastReset</i> <i>adu.vft.road.usedFuelFromLastReset</i> | Fuel used in stage/road mode since the last refueling |

For more details and an example configuration, see [Appendix F - How-to Use Virtual Fuel Tank \(on page 223\)](#).

19.11. Configuration

The *Configuration* panel contains the configuration parameters of elements such as the startup screen, tire/ brake temperature cameras, as well as internal accelerometer calibration, screen rotation, etc.

| Parameter | Description |
|-----------------------------------|--|
| Alarms | |
| Multiple alarm mode | <p>Alarm system operation mode. If a number of alarms are activated simultaneously, the parameter determines their behaviour.</p> <p>Only one active with highest priority – only the highest-priority alarm will be displayed. Priority is determined based on the order of the alarms in the project. The lower the position in the list, the higher the priority of the alarm.</p> <p>Cycle active alarms – toggling between all active alarms will be activated.</p> |
| Cycle alarm time | If the Cycle active alarms option is set in the Multiple alarm mode parameter, this parameter defines the length of time the alarm will be displayed. (in seconds). |
| Alarm height | The height of the rectangle displayed as an alarm background |
| Screen | |
| Rotate screen | It allows to rotate the screen 180 degrees. Option used if the screen is installed turned 180 degrees (with the shift light leds at the bottom) |
| Brightness | |
| Adaptation rate | A parameter handling the automatic brightness of the screen and the LEDs determining the maximum allowed brightness change during one second |
| IMU | |
| Pitch | The IMU pitch calibration |
| Speed source for ecu.speed | |
| Source | <p>The speed source for the ecu.speed channel can be obtained from:</p> <ul style="list-style-type: none"> • CAN bus (requires CANbus Input with override); • GPS; • digital Inputs (Wheels). |
| Wheel #1 [km/h] | When selecting the Wheels speed source, select the Digital Input channel in which the sensor is defined. |
| Wheel #2 [km/h] | For Wheels source channel for the second digital input with a defined sensor |
| Odometer / Distance meter | |

| Parameter | Description |
|----------------------------------|---|
| Speed source | The source of vehicle speed information for the odometers can be obtained from the ecu.speed or gps.speed channels. |
| Hours at load meter | |
| Minimal ecu.rpm | Minimum rpm defining engine load |
| Minimal ecu.tps | Minimum throttle angle defining the engine load |
| Minimal ecu.map | Minimum intake manifold pressure defining the engine load |
| Tire temperature cameras | |
| Tire temp. range min | Minimum tire temperature displayed on the graph in blue |
| Tire temp. range max | Maximum tire temperature displayed on the graph in red |
| Brake temperature cameras | |
| Brake temp. range min | Minimum brake disk temperature displayed on the graph in blue |
| Brake temp. range max | Maximum brake disk temperature displayed on the graph in red |
| Min / Max reset | |
| Reset min / max mode | <p>This parameter defines the behaviour of the min / max values for the ecu channels*.</p> <p>Every power off – the min / max values are deleted each time the device is activated.</p> <p>Every firmware upgrade – the min / max values are deleted when the internal device software is replaced.</p> |
| Startup screen | |
| Enable | It activates the startup screen. |
| Texture | Texture that will be displayed on the startup screen (centred) |
| Scale | Scale of the displayed texture |
| Duration | Start screen display time |
| Color | Color of the displayed texture |
| Background color | Color of the background |
| Lap timing | |
| Lap timing mode | Lap time countdown: |

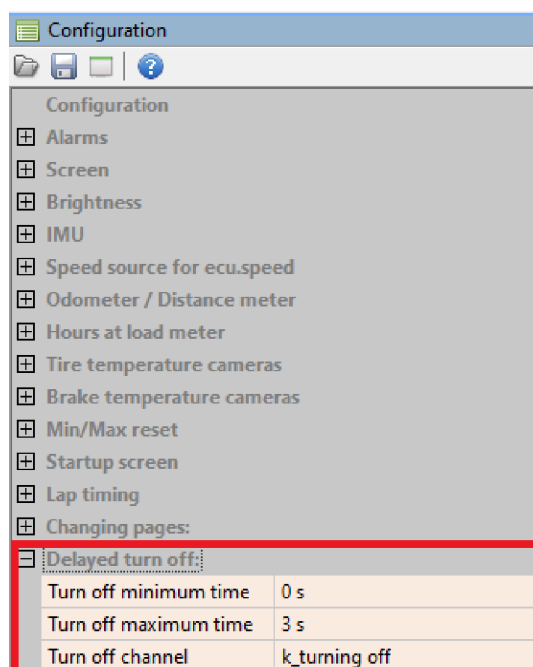
| Parameter | Description |
|----------------------------------|---|
| | Circuit - for lap timing with a single start/finish line Bracket racing - from the time set on the first lap down to zero Hill climb - for lap timing with separate start and finish points |
| Start speed threshold | When in Hill Climb mode, this parameter defines the maximum speed at which the Start point can be recognized. |
| Changing pages | |
| Page selector channel | Pages can be changed directly from the CAN input or via a rotary switch connected to the analog input. Page selector takes control of the Next page and Previous page keys defined in the Buttons panel. |
| Page selector first value | Setting the first value of the page selector |
| Delayed turn off | |
| Turn off minimum time | Minimum time after turning off the ignition that must elapse for the Turn off channel to turn off the ADU |
| Turn off maximum time | Maximum time after turning off the ignition that the ADU will remain on (provided that the Turn off channel has not been triggered beforehand) |
| Turn off channel | ADU shutdown control channel. When the defined channel has a non-zero value between minimum time and maximum time , the ADU will switch off. |

19.11.1. Delayed turning off

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

The ADU 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 **Configuration** from the list.



Define the following parameters in the **Configuration** panel under **Delayed turning off**:

- **Turn off minimum time**: the minimum time after the ignition is switched off that must elapse for the additional factor (**Turn off channel**) to take effect, causing the ADU to switch off.
- **Turn off maximum time**: maximum time after turning off the ignition for which the ADU will remain switched on (provided that the turn-off channel has not been triggered beforehand).
- **Turn off channel**: when the defined channel has a non-zero value between minimum time and maximum time, the ADU will turn off.

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

19.12. 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. |
| Copyrights | Information displayed when attempting to connect to the device if protected |

| Parameter | Description |
|---------------------|---|
| Contact info | Information including contact details (e-mail, telephone) displayed when attempting to connect to the device if protected |

19.13. Log

The *Log* panel contains the logging configuration.

| Parameter | Description |
|-----------------------------------|--|
| Log to USB | A channel / variable determining if the device is to log to a USB. The value <i>one</i> 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 ADU device allows to create 4 logging profiles. These profiles are defined in <i>Menu > Logged channels</i> . They allow to record to a file different channels with different frequencies depending on the current condition. The <i>Default log condition</i> 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 <i>Cond 2</i> profile. Log profile may change every 40ms (25 times per second). |
| Log Cond3 channel | A channel / function activating logging of the <i>Cond 3</i> profile |
| Log Cond4 channel | A channel / function activating logging of the <i>Cond 4</i> profile |

19.14. CANbus / Serial Setup

The CANbus / Serial Setup panel is used for configuring the CAN bus and for RS232 serial communication.

| Parameter | Description |
|-----------------|-------------|
| General: | |

| Parameter | Description |
|---|--|
| CAN2 terminator | Activation of the terminator on the CAN2 bus. |
| CAN2 speed | The speed of the CAN2 bus |
| <u>GPS:</u> | |
| CANbus | Selection of the CAN bus to which the GPS module is connected. |
| Base ID | The base CAN ID for the GPS module is set to 0x400 by default. This value can be changed, but it must be a multiple of 8. |
| Static hold | GPS position freeze (when the channel with vehicle speed information, e.g. ecu.speed, has a value of zero) |
| <u>Tire / Brake temperature cameras:</u> | |
| CANbus | Selection of the CAN bus to which the thermal cameras measuring the tire / brake disks temperature are connected |
| Base ID | The base ID of the first thermal imaging camera (in the hexadecimal notation). |
| <u>Serial:</u> | |
| Serial protocol | <p>Selection of a serial protocol:</p> <p>Ecumaster serial protocol - serial protocol supported by EMU Classic and EMU Black</p> <p>Ecumaster Classic EDL protocol - serial protocol sent by the Ecumaster EMU Classic to the Ecumaster EDL1 device</p> <p>AIM serial – a serial protocol consistent with the AIM protocol</p> <p>Hondata 9600 - a serial protocol consistent with the Hondata PRO protocol (speed of 9600 bps). A signal inverter should be used between the Hondata and the ADU.</p> <p>Hondata 115200 – a serial protocol consistent with the Hondata PRO protocol (speed of 115 kbps). A signal inverter should be used between the Hondata and the ADU.</p> <p>Autronic SM 4 – a serial protocol consistent with the Autronic SM 4 protocol. This version refers to the SM 4 software version.</p> <p>AEM – a serial protocol compatible with AEM protocol</p> <p>GEMS – a serial protocol compatible with GEMS protocol</p> <p>Athena GET – a serial protocol compatible with Athena GET protocol</p> |

19.15. 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 G - How-to Configure Autosaved Channels \(on page 233\)](#).

19.16. Channel Simulator

Channel Simulator is used to simulate channel values in order to check the correct operation of established strategies or functions. Any existing channel or multiple channels can be simulated simultaneously.

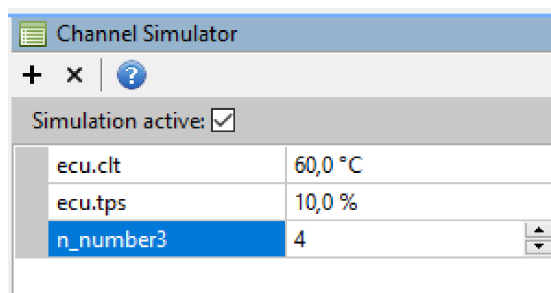
Information flowing from the simulator has the highest priority, with the result that data flowing over the CAN bus is bypassed.

To open **Channel Simulator**, select **Desktop > Add new panel** from the menu bar or use the **F9** key and then select **Channel Simulator** from the list in the **Select panel** window.

Adding channels to the simulator is done by clicking the **+** icon in the top left corner of the window or using the **Insert** shortcut, and then in the **Select Channel** window selecting the appropriate channel from the list.

The channel value to be simulated is shown in the cell next to the channel name.

The simulator is active when the **Simulation active** box is checked.



To delete a selected channel, select it and then click the **x** icon in the top left corner of the window or use the **Delete** key.

19.17. Channel report

Channel report is a panel displaying the minimum, maximum and average values for the selected engine channels. The data is continuously updated.

| Channel report | | | |
|----------------|-------|-------|-------|
| Channel | Min | Max | Avg |
| ecu.rpm | 0 | 8551 | 4881 |
| ecu.oilPress | 0,00 | 9,00 | 3,13 |
| ecu.oilTemp | 0,0 | 0,0 | 0,0 |
| ecu.clt | 86,0 | 109,0 | 93,2 |
| ecu.battery | 9,96 | 14,28 | 13,99 |
| ecu.lambda1 | 0,562 | 1,992 | 1,037 |
| ecu.lambda2 | 0,000 | 0,000 | 0,000 |
| ecu.egt1 | 0 | 0 | 0 |
| ecu.egt2 | 0 | 21760 | 0 |
| ecu.ignAngle | -4,5 | 37,0 | 20,1 |

20. Timing

The ADU may operate as a Lap timer, i.e. a device for measuring time on a racing track. The time can be measured using a GPS or a beacon.



Note:

The ADU only supports the **Ecumaster GPS to CAN** module and does not support other GPS modules or interfaces.

The beacon is placed near the start / finish line and then it sends an infra-red beam, which is registered by a receiver inside a car. The receiver “informs” the ADU when the start / finish line has been crossed.

The lap timing system supports both closed circuits (single start/finish line) and **Hill Climb** mode (separate start and finish points). Detailed information about Hill Climb mode can be found in: [Appendix B - How-to Configure the Hill Climb \(on page 204\)](#). Circuit mode is described in the following topics.

In addition to measuring time, the GPS module allows to display the anticipated lap time (Predictive timing) and a change in the anticipated lap time (gain / loss) in the form of a graph. Once the data from the GPS module or other data, such as sensor values or engine parameters, have been logged onto an external USB storage device stick, the user is able to carry out further data analysis using Ecumaster Data Master software package: <https://www.ecumaster.com/products/data-master/>.

20.1. Configuration of time measurement with a beacon

Depending on the beacon system used, the output of the receiver should be connected to an analog or digital input of the ADU and an appropriate channel should be created.

The channel should be assigned in the **Buttons** panel in the **Beacon input - channel** field. When the channel value changes from 0 to 1, the last lap time counter (*last lap*) will assume the value of the lap time counter (*lap time*), which will be reset. If a current lap time is better than the previous best lap time (*best lap*), it will be saved as the best. Additionally, the **adu.track.lap** channel will be increased by 1. To display the times on the screen, use the Time object. To display the current lap number, use the Text object and enter the **adu.track.lap** channel in the **Channel** field. If a *beacon* system is used, the *predictive timing* function is not available.

20.2. Configuration of time measurement with a GPS

A GPS module offers more possibilities than a beacon system. Based on GPS coordinates, the ADU automatically detects the track the vehicle is currently on. A list of automatically detected tracks can be found in attachment 1. If a track is not listed, it can be added as a user track (see *Panels / User tracks* for more information).

The ADU is capable of storing information from 20 tracks in its memory. 8 best times are recorded, including their date, time and maximum speed. This data can be displayed using the *Track record table* object.

The best time from a given track is used as a reference time for the algorithm determining the predicted lap time. This algorithm assumes the current time of the run for a given place on the track and the time from the best run for the rest of the track.

By displaying the Predictive time graph, it is possible to check in real time if a given part of the track has been completed faster or slower than during the best lap.

To reset a given track data you can use the *Tools> Reset track* data menu or connect a button and assign it to the Reset track data channel function in the Buttons panel. It is recommended to use a *Hold* type trigger to prevent accidental data deletion.

To display the name of the track the car is currently on, use the **Text** object and enter \$(TRACK) in the text field.



Important:

The GPS module should be installed using anti-vibration rubber pads included in the module kit.

21. Data analysis

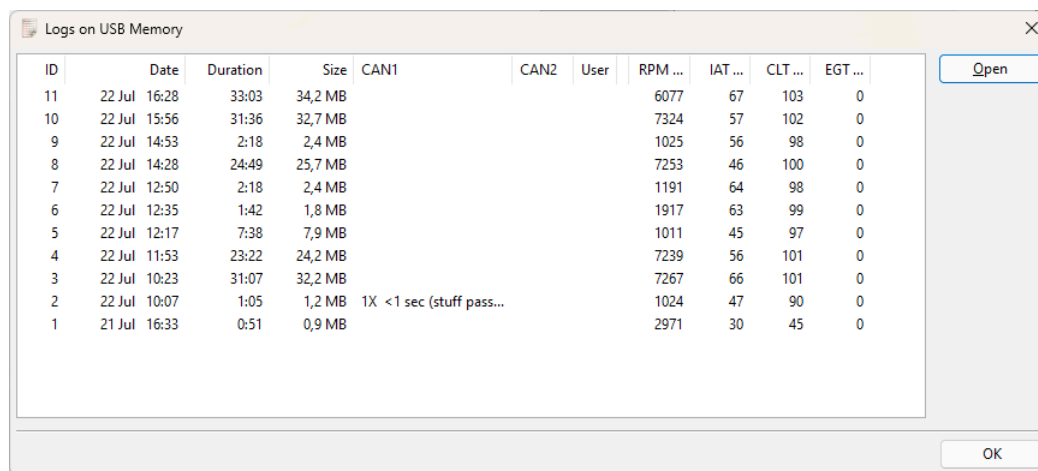
By saving racetrack data on a memory stick, users can later perform more advanced analysis using Windows software.

ADU software provides basic data analysis capabilities, allowing users to check the accuracy of custom functions, assess GPS data quality, and validate lap timing precision. This software equips users with key panels: **Graph Log**, **Scatter Plot**, **Track Preview**, **Section Times**, **Histogram**, and **Lap Time Plot**.

For more comprehensive analysis, ECUMaster offers a free, fully-capable data analysis tool called Data Master, which provides additional advanced features and functionalities.

The log is loaded via the **Open log** icon in the **Graph Log** or **Scatter Plot** panel.

Another way to read data from a memory stick is to use the **log file** option in the **Devices / Receive** menu or use the **SHIFT+F4** keyboard shortcut. A window with a selection of log files will appear.

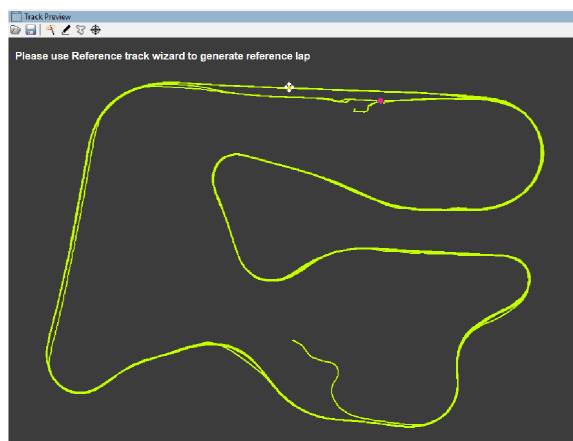


| ID | Date | Duration | Size | CAN1 | CAN2 | User | RPM ... | IAT ... | CLT ... | EGT ... | |
|----|--------------|----------|---------|--------------------------|------|------|---------|---------|---------|---------|------|
| 11 | 22 Jul 16:28 | 33:03 | 34,2 MB | | | | 6077 | 67 | 103 | 0 | Open |
| 10 | 22 Jul 15:56 | 31:36 | 32,7 MB | | | | 7324 | 57 | 102 | 0 | |
| 9 | 22 Jul 14:53 | 2:18 | 2,4 MB | | | | 1025 | 56 | 98 | 0 | |
| 8 | 22 Jul 14:28 | 24:49 | 25,7 MB | | | | 7253 | 46 | 100 | 0 | |
| 7 | 22 Jul 12:50 | 2:18 | 2,4 MB | | | | 1191 | 64 | 98 | 0 | |
| 6 | 22 Jul 12:35 | 1:42 | 1,8 MB | | | | 1917 | 63 | 99 | 0 | |
| 5 | 22 Jul 12:17 | 7:38 | 7,9 MB | | | | 1011 | 45 | 97 | 0 | |
| 4 | 22 Jul 11:53 | 23:22 | 24,2 MB | | | | 7239 | 56 | 101 | 0 | |
| 3 | 22 Jul 10:23 | 31:07 | 32,2 MB | | | | 7267 | 66 | 101 | 0 | |
| 2 | 22 Jul 10:07 | 1:05 | 1,2 MB | 1X <1 sec (stuff pass... | | | 1024 | 47 | 90 | 0 | |
| 1 | 21 Jul 16:33 | 0:51 | 0,9 MB | | | | 2971 | 30 | 45 | 0 | |

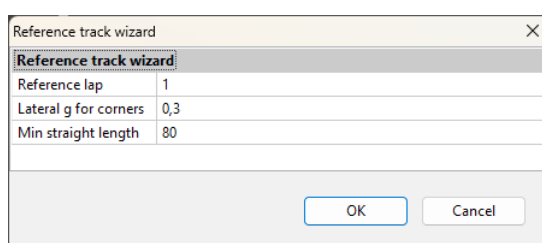
OK

To upload the respective log, double click on it with the left button.

The next step is to go to the **Track** tab. The **Track Preview** window should display the entire track covered by the car and saved in the file. The figure shows the saved route from the example log. Depending on the quality of the signal, the raw GPS data points are shown in colors ranging from red (poor signal quality) to green (for the best signal quality). The channel indicating the quality of the signal is **gps.status**, where status with a value of 0 means no signal, a value of 1 a very weak signal and successively up to values 4 and 5, which indicate the best signal quality. The start / finish line of the track is marked with a white circular marker.

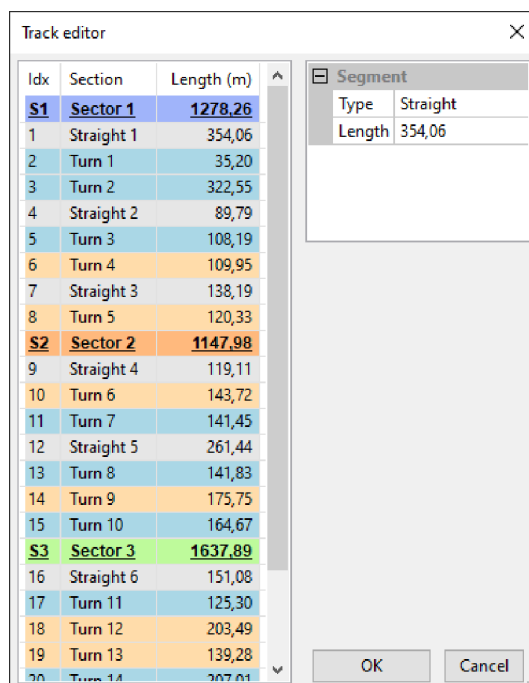


In order to be able to analyse a run, a track outline must be generated from the collected data. To do this, press the magic wand icon (*create reference track*) in the tool bar of the **Track Preview** window. The **Reference track wizard** window should appear. From the **Reference lap** field select the lap to be used for creating your reference track. When switching laps, the track drawing changes in real time. The **Lateral g for corners** parameter defines the lateral g-force above which a given track part is interpreted as a turn. The **Min straight length** parameter defines the value of the minimum section of a track where the g-force was lower than Parameters **Lateral g for corners** for the distance to be considered a straight section.



After pressing OK, a reference track divided into segments (left turn, right turn, straight line) and their constituent sectors will be generated. By default the track is divided into three sectors. The start / finish line is marked as a chequered flag with an arrow indicating the direction of travel on the track. This line is also a fixed boundary of the first sector –, i.e. it cannot be removed or moved. Other sector boundaries can be created at any given point. These boundaries are marked with cross lines and described by the number of the sector they begin, e.g. S2.

To edit segments and their constituent track sectors, select the pencil icon in the toolbar (*Edit reference track*). The **Track editor** window will appear.



In this window, segments and sectors can be deleted or divided, and the type and length of segments can be changed. When a segment or sector is selected, it is automatically highlighted on the preview screen. To edit the division of the reference track, click on the corresponding segment with the right mouse button.

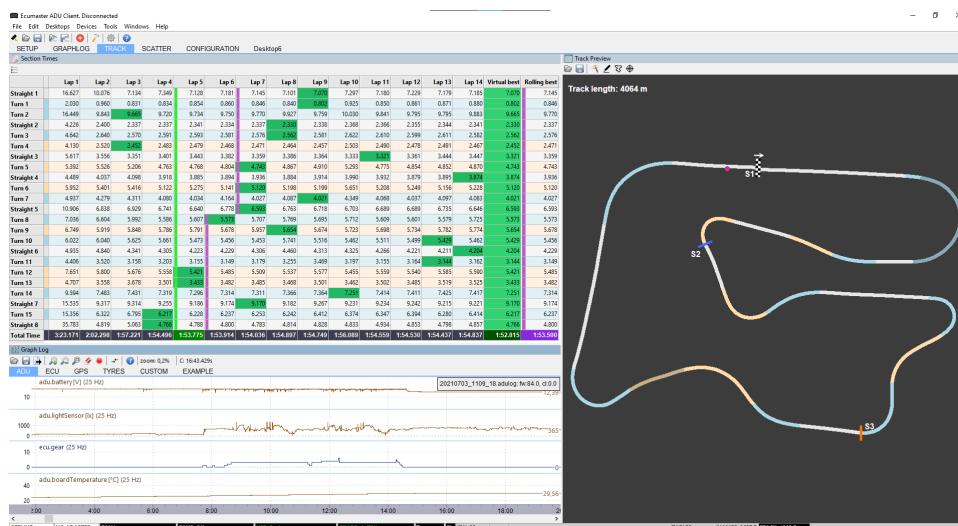
A window will appear with the following options:

| Option | Description |
|----------------------------------|---|
| Remove segment | Deletes the selected segment by attaching it to the segment immediately following it. |
| Split segment | Divides the marked segment in the middle of its length. |
| Split sector | Adds a sector border at the end of the selected segment. |
| Remove sector | Deletes a sector in which the selected segment is located by adding it to the next sector. The exception is the last sector (whose end is the start / finish line)– deleting the last sector adds it to the previous sector. |
| Move current sector end | Moves the end border of the sector in which the selected segment is located to the end of the selected segment. If the end of the selected sector is the start / finish line, an additional sector border will be added at the end of the selected segment (additional sector will be created). |
| Move current sector start | Moves the initial boundary of the sector in which the selected segment is located to the beginning of the selected segment. If the beginning of the selected sector is the start / finish line, an additional sector border will be |

| Option | Description |
|--------|---|
| | added at the beginning of the selected segment (additional sector will be created). |

Press OK after you finish editing. The created track can be saved by pressing the floppy disk icon. The next time the programme is started, the reference track that was last saved or read will be loaded.

Once the reference track has been created, the times in the sections should recalculate automatically. A sample application screen is presented below.



The purple point on the reference track determines the position of the vehicle for the position selected in the logging window. This way, by moving along the log window, you can preview the set of parameters for particular vehicle positions.

The segments into which a track is divided are colour-coded (white, light orange and light blue), depending on the type of segment (straight, left turn, right turn). To make the data easier to read, the same colors were used to mark the corresponding segments in the table in the *Section Times*. The **Section Times** panel displays times in defined segments for all laps. The best times are marked green. The green color of the individual cells shows the best times achieved for all runs in each section of the track, while the vertical green line indicates the best time for the entire lap. Additionally, this window comprises of two columns: **Virtual best** and **Rolling best**. The **Virtual best** is the sum of the best times in individual sections, while **Rolling best** shows the best continuous time of a single lap. It is marked with a purple vertical line and, in the example in question, was obtained in the 5th and 6th laps. **Rolling best** shows a realistic time that could have been achieved.

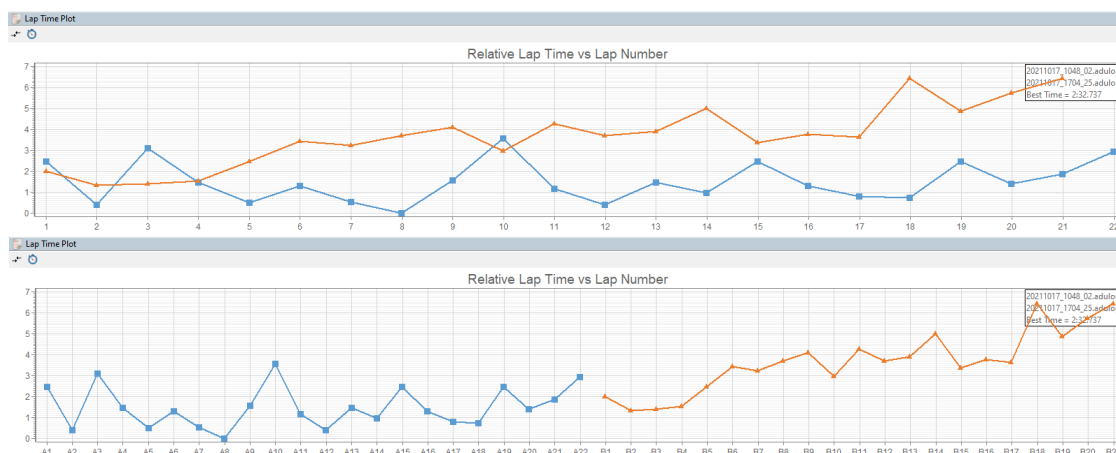
The **Section Times** window can also be used to display the run times of all laps divided into sectors. This form of data presentation can be clearer and easier to analyse. Use the **Sectors / Segments** icon at the top left of the **Section Times** panel to switch the view between tables.

You can use the graph of lap times displayed in the **Lap Time Plot** panel to analyse the runs. For correct operation of the **Lap Time Plot** visualisation it is necessary to generate a reference lap in the **Track Preview**.

In the graph, the X axis represents the lap numbers and the Y axis represents the lap times. In the top right corner, a box shows which logs are currently being analysed and what the best lap time was. Several logs can be analysed simultaneously on a single chart.

When analysing two logs simultaneously, by default the graphs from both logs are plotted side by side, so that one is a continuation of the other. Using the arrow icon **Compare / Append Mode** they can be overlaid on top of each other. By clicking again on the **Compare / Append Mode** icon, you return to the previous view. When the graphs of the two logs are lined up side by side, the laps are described on the X-axis by a letter denoting the log in question and the lap number, e.g. A1, A2 etc. for the first log and B1, B2 etc. for the second log.

In addition, for better clarity of the graph, the Y axis can be scaled using the icon with the stop watch symbol **Absolute / Relative Mode**. By default, the times achieved on the individual laps are marked on the Y axis. Clicking on the aforementioned icon will cause the Y-axis to show the time difference to the best lap, i.e. the best lap will be marked as zero on the axis and any other lap will show the loss to that best race.



The channels that may be helpful in analysing data:

| Channel | Description |
|----------------------|---------------------------|
| adu.track.lap | Displays the current lap. |

| Channel | Description |
|----------------------------------|---|
| <i>adu.track.lapTime</i> | Displays the current lap time. |
| <i>adu.track.gainLoss</i> | Displays the gain / loss relative to the best recorded lap. |
| <i>adu.latG</i> | Lateral overload acting on the vehicle (negative values for a right turn, positive values for a left turn) |
| <i>adu.longG</i> | Longitudinal overload acting on the vehicle (negative values when accelerating, positive values when braking) |
| <i>gps.speed</i> | Vehicle speed read from the GPS |
| <i>gps.status</i> | GPS module status If signal problems occur, the analysed data may be inaccurate. |
| <i>ecu.rpm</i> | Engine speed |
| <i>ecu.tps</i> | Throttle position. |
| <i>ecu.clt</i> | Engine coolant temperature |
| <i>ecu.map</i> | Pressure in the intake manifold |
| <i>ecu.oilPress</i> | Engine oil pressure |
| <i>ecu.oilTemp</i> | Engine oil temperature |

Incidence of a given phenomenon can be analysed using the **Histogram** panel. To define a suitable channel, click on the pencil icon which opens the **Define channels** window.

| Option | Description |
|---|---|
| <i>Channel X</i> | The channel defining the X axis |
| <i>Filter channel</i> | The filter channel introducing an additional condition on a separate channel, the fulfilment of which determines whether (at a particular point in time) data from channel X will be collected for analysis or rejected |
| <i>Discard samples above</i> | Rejects samples above the set value for the filter channel |
| <i>above value</i> | Upper limit value for the filter channel |
| <i>Discard samples below</i> | Rejects samples below the set value for the filter channel |
| <i>below value</i> | Lower limit value for the filter channel |
| <i>Autoscale X axis and bins</i> | Automatic scaling of the X axis range and the number of bins |

| Option | Description |
|-------------------------|---|
| Autoscale X axis | Automatic scaling of the X axis range |
| Min | Initial value of the X axis |
| Max | Final value of the X axis |
| Set bin width | Sets the bin width |
| Bin width | Width of a bin |
| Bin count | Number of bins |
| Exclude outliers | Rejects outliers (deviating from normal distribution) |
| Y axis unit | Y-axis unit defining the frequency of occurrence of the defined phenomenon Percent - percentage Time - time frame |
| Entered bins | Sets the description of the X axis values at the centre of the bins. |

The **Histogram** panel allows data analysis from several logs at the same time. They can be analysed merged into one long log on one histogram (default setting) or separately on separate axes (one below the other). Use the arrow icon to switch between these views **Compare / Append** mode. In the split log analysis mode we can plot the logs onto a single histogram, where each log will be marked with a different color. The **Tiled / Overlay** icon is used for this purpose.

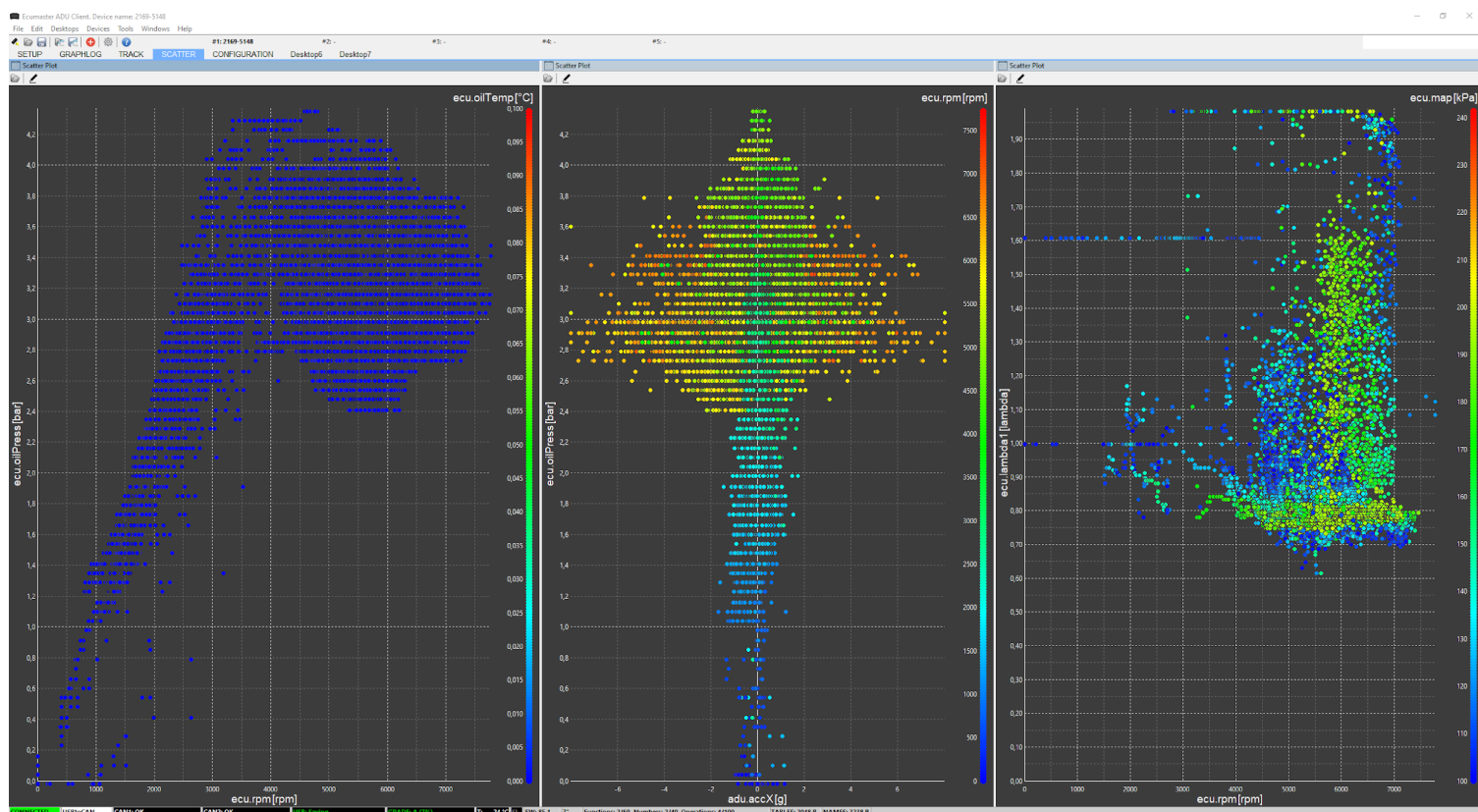


Another way of visualising data is A scattered plot (with three axes: X, Y and color axes). This chart can be configured independently, while several pre-defined configurations can be used once the **Scatter Plot** panel has been added:

- *Scatter Plot: oilPress vs rpm (color oil Temp)*
- *Scatter Plot: oilPress vs adu.latG (color rpm)*
- *Scatter Plot: oilPress vs gps.latG (color rpm)*
- *Scatter Plot: lambda vs rpm (color map)*

To define axes independently, click the pencil icon, which opens the **Define channels** window.

| Option | Description |
|------------------------------------|---|
| Channel X | Channel defining the X axis |
| Channel Y | Channel defining the Y axis |
| Color channel | Channel defining the color axis |
| Discard color samples above | Rejects samples above the set value for the Color channel |
| above value | Upper limit value for the Color channel |
| Discard color samples below | Rejects samples below the set value for the Color channel |
| below value | Lower limit value for the Color channel |



22. Appendix A - List of built-in tracks

Tracks stored in the device's memory (automatically detected):

Belgium

Circuit de Spa-Francorchamps, Circuit Zolder

Croatia

Rijeka Grobnik

Czech Republic

Automotodrom Brno

France

Circuit de Nevers Magny-Cours

Germany

Eurospeedway Lausitzring, Hockenheim, Motorsport Arena Oschersleben, Nurburgring, Schleiz

Hungary

Pannonia Ring

Italy

ACI Vallelunga Circuit, Autodromo Nazionale Monza, Fanciacorta International, Imola, Misano World Circuit, Mugello Circuit, Racalmuto

Latvia

Bikernieki

Netherlands

Circuit Zandvoort, TT Circuit Assen

Poland

Tor Poznań, Tor Słomczyn

Portugal

Autódromo Internacional do Algarve

Scotland

Knockhill Racing Circuit

Slovakia

Slovakia ring

Spain

Circuit de Barcelona-Catalunya, Circuit de la Comunitat Valenciana Ricardo Tormo, Circuito de Albacete, Circuito de Navarra, Circuito Permanente de Jerez, Ciudad del Motor de Aragón

Sweden

Anderstorp Raceway, Falkenbergs Motorbana, Gelleråsen Arena, Gotland Ring, Kinnekulle Ring, Linköpings Motorstadion, Mantorp Park, Mittsverigebanan, Ring Knutstorp, Sturup Raceway, Tierp Arena

UK

Anglesey International, Brands Hatch GP, Cadwell Park, Donington Park, Oulton Park, Silverstone Natl., Snetterton Circuit, Thruxton Motorsport Centre.

USA

APEX Motor Club, Arizona Park, Barber Motorsport Park, Buttonwillow, Chuckwalla, Circuit of the Americas, High Plains, Laguna Seca Raceway, Michelin Raceway Road Atlanta, Miller Motorsports Park, New Jersey Motorsport Park, Pittsburgh International Race Complex, Pueblo Park, Road America, Sears Point Raceway, Thermal Blue, Thunderhill, Virginia International Raceway, Willow Springs

Qatar

Lusail Circuit

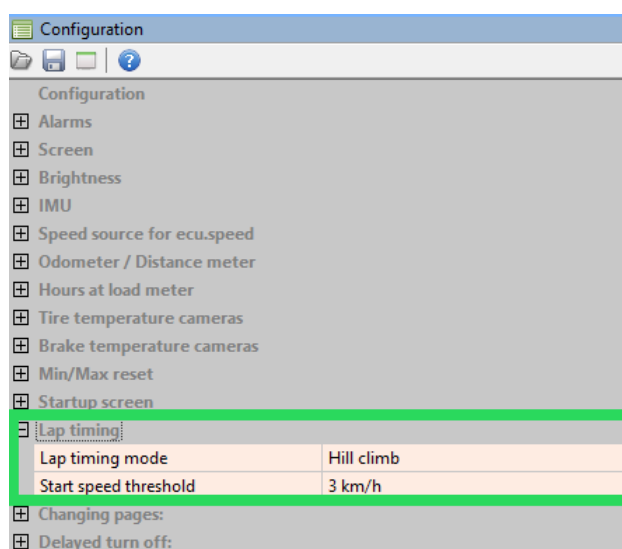
23. Appendix B - How-to Configure the Hill Climb

23.1. Description

Details about version 105.0 or later are provided at the end of the document.

Upgrade the firmware of your ADU to version 94.0 or later.

In the *Configuration* panel, change the *Lap timing mode* to the 'Hill climb'.



A start of an event is triggered when the vehicle speed ('ecu.speed') exceeds the *Start speed threshold*. The vehicle speed channel ('ecu.speed') is required.

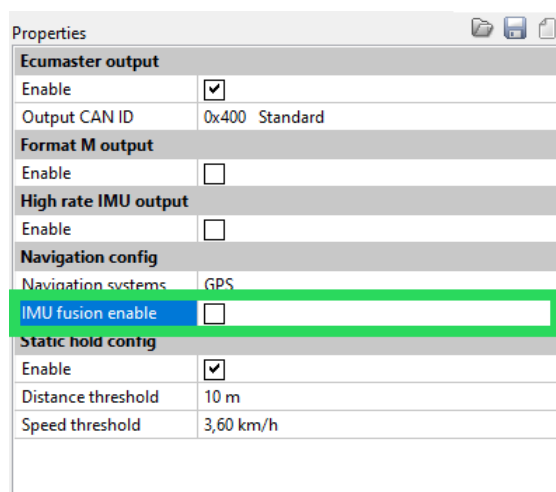


Note:

We are recommending to use vehicle speed sensor (VSS) for this channel. You can still use a speed from the GPStoCAN module, but you may need to increase the *Start speed threshold* (*Lap timing* section of the *Configuration* panel).

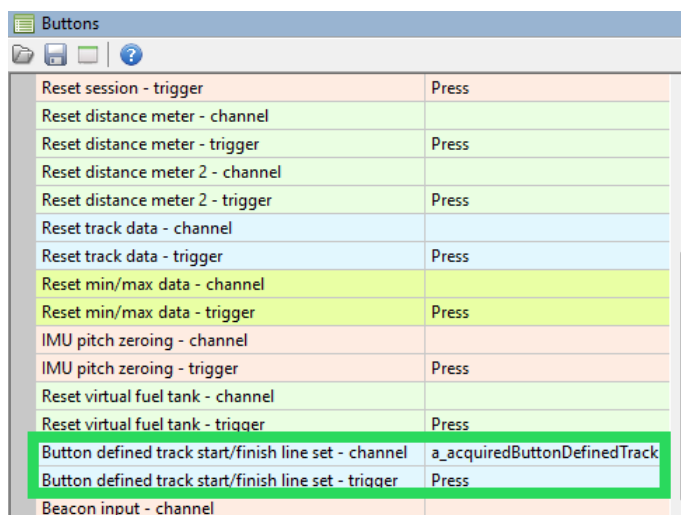
GPStoCAN V1 users:

If you are using GPStoCAN V1 as a speed source, you can consider to untick the *IMU fusion enable* option in the *Light Client* app with your GPS connected. In this way, the GPStoCAN module will use only the GPS signal (and NOT a GPS+IMU). It decrease an update frequency of your GPStoCAN V1 module to 10 Hz, but it will help to prevent jitter on the GPS speed when your car is standing at the start line.



How to configure the track in ADU memory:

1. The Button to define a track is wired to analog input A5 in the default project. In the Buttons panel, you need to define the *Button defined track start/finish line set* parameter (referred to as 'Button' in the remainder of this document).



2. Make sure your GPS is working correctly (has a fix).
3. You need to stop at the start line and press the Button
4. Drive to the finish line and press the Button again. And now your track will be defined.

How to start the lap timing:

1. Make sure your GPS is working correctly (has a fix).
2. You need to stop at the start line (the radius around the start point is 15 m).
3. The timer will start when the speed will be higher than *Start speed threshold* (default: 3 km/h).
4. Drive to the finish line and when you cross the finish line the timer will stop automatically.

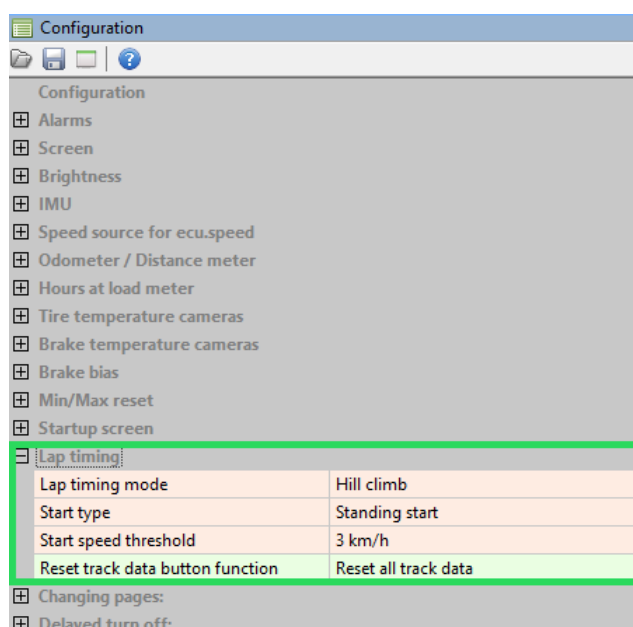
There are two diagnostics channel available: enumeration 'adu.track.lapTimingState' and 'adu.track.distanceToStart' in meters.

All racing features in the ADU are functional in Hill Climb - predicted time, gain/loss, gain/loss graph, record table, and session times.

In version 105.0 or later:

Upgrade the firmware of your ADU to version 105.0 or later.

In the *Configuration* panel, change the *Lap timing mode* to the 'Hill climb'.



Select the appropriate *Start type* mode: 'Standing start' or 'Flying start'

The vehicle speed channel ('ecu.speed') is required.

Start type: Standing start:

A start of an event is triggered when the vehicle speed ('ecu.speed') exceeds the *Start speed threshold*.

Start type: Flying start:

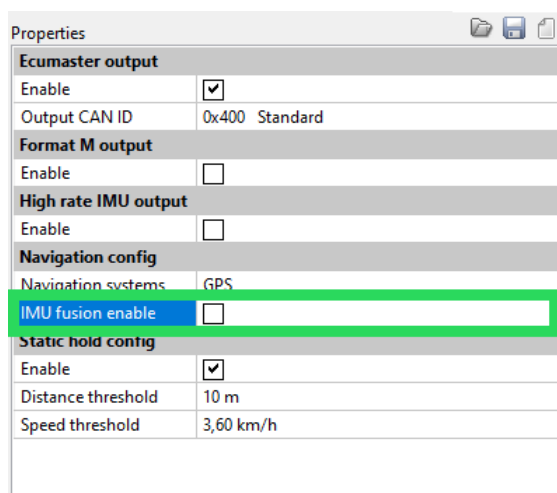
A start of an event is triggered when the vehicle crosses the start line.

**Note:**

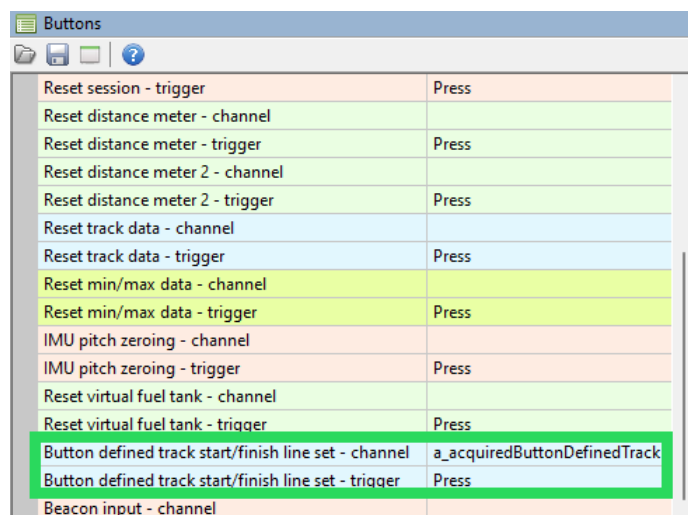
We are recommending to use vehicle speed sensor (VSS) for this channel. You can still use a speed from the GPStoCAN module, but you may need to increase the *Start speed threshold* (Lap timing section of the *Configuration* panel).

GPStoCAN V1 users:

If you are using GPStoCAN V1 as a speed source, you can consider to untick the *IMU fusion enable* option in the *Light Client* app with your GPS connected. In this way, the GPStoCAN module will use only the GPS signal (and NOT a GPS+IMU). It decrease an update frequency of your GPStoCAN V1 module to 10 Hz, but it will help to prevent jitter on the GPS speed when your car is standing at the start line.



The Button to define a track is wired to analog input A5 in the default project. In the Buttons panel, you need to define the *Button defined track start/finish line set* parameter (referred to as 'Button' in the remainder of this document).



Standing start

How to configure the track in ADU memory:

1. Make sure your GPS is working correctly (has a fix).
2. You need to stop at the start line and press the Button.
3. Drive to the finish line and press the Button again. And now your track will be defined.

How to start the lap timing:

1. Make sure your GPS is working correctly (has a fix).
2. You need to stop at the start line (the radius around the start point is 15 m).
3. The timer will start when the speed will be higher than *Start speed threshold* (default: 3 km/h).
4. Drive to the finish line and when you cross the finish line the timer will stop automatically.

Flying start

How to configure the track in ADU memory:

1. Make sure your GPS is working correctly (has a fix).
2. You need to press the button as you cross the start line.
3. Drive to the finish line and press the Button again. And now your track will be defined.

How to start the lap timing:

1. Make sure your GPS is working correctly (has a fix).
2. The timer will start when you cross the start line.
3. Drive to the finish line and when you cross the finish line the timer will stop automatically.

There are two diagnostics channel available: enumeration 'adu.track.lapTimingState' and 'adu.track.distanceToStart' in meters.

All racing features in the ADU are functional in Hill Climb - predicted time, gain/loss, gain/loss graph, record table, and session times.

23.2. Document history

| Version | Date | Changes |
|---------|------------|--|
| 1.0 | 2023.10.10 | Initial release |
| 1.1 | 2024.01.19 | Screenshots changed |
| 1.2 | 2024.09.03 | Additions in version 105.0 - 'Standing' or 'Flying' Start option |

24. Appendix C - How to Configure PMU CAN Stream

24.1. Description

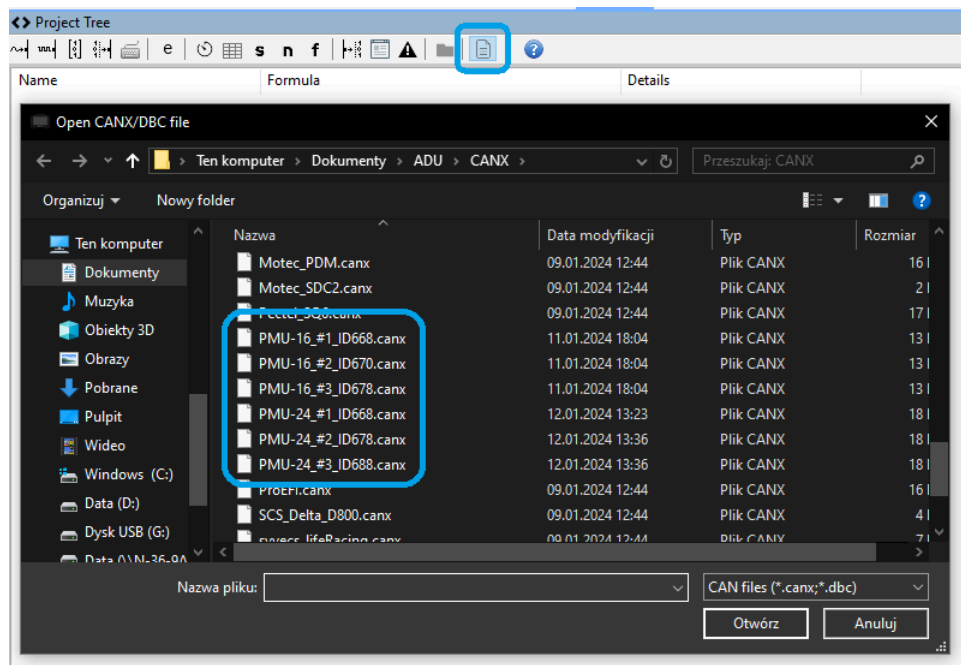
ADU firmware 100.0 adds the native support of the PMU-24 to the previously supported PMU-16. The ADU supports up to 3 PMUs, each of them being 16 or 24 version.

To add the PMU CAN stream, import one of the .CANX files. There are three 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

Additionally, the names of the CANX files for PMU-16 have changed. These are three files:

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



It's important to notice that importing the PMU .CANX file creates only CANbus Message objects in the Project Tree. In the case of PMU-24, for PMU number #1, these are two CANbus Message Objects of the following types:

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

In the case of PMU-16 for PMU number #1 there is only one CANbus Message Object of type:

- PMU1 [1-16]

The CANbus Inputs are not necessary, because the channels decode automatically. This saves the 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.

| ID | Default CAN stream locations for PMU-16 | Default CAN stream locations for PMU-24 |
|-------|---|---|
| 0x668 | PMU-16 #1 at 0x668 | PMU-24 #1 at 0x668 |
| 0x669 | | |
| 0x66A | | |
| 0x66B | | |
| 0x66C | | |
| 0x66D | | |
| 0x66E | | |
| 0x66F | | |
| 0x670 | PMU-16 #2 at 0x670 | |
| 0x671 | | |
| 0x672 | | |
| 0x673 | | |
| 0x674 | | |
| 0x675 | | |
| 0x676 | | |
| 0x677 | | |
| 0x678 | PMU-16 #3 at 0x678 | PMU-24 #2 at 0x678 |
| 0x679 | | |
| 0x67A | | |
| 0x67B | | |
| 0x67C | | |
| 0x67D | | |
| 0x67E | | |
| 0x67F | | |
| 0x680 | | |
| 0x681 | | |
| 0x682 | | |
| 0x683 | | |
| 0x684 | | |
| 0x685 | | |
| 0x686 | | |
| 0x687 | | |
| 0x688 | | PMU-24 #3 at 0x688 |
| 0x689 | | |
| 0x68A | | |
| 0x68B | | |
| 0x68C | | |
| 0x68D | | |
| 0x68E | | |
| 0x68F | | |
| 0x690 | | |
| 0x691 | | |
| 0x692 | | |
| 0x693 | | |
| 0x694 | | |
| 0x695 | | |
| 0x696 | | |
| 0x697 | | |

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 01-016, the current value resolution is 0.25A, and for outputs 017-024, it is 0.1A.

(**) – For outputs 01-016, the measurement range is 0-16V, and for outputs 017-024, it is 0-20V.

Inputs A9-A16 have the capability to 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 instance, the voltage value on input A9 is also available on `pmuX.o17.voltage`. Similarly, the voltage value on input A16 is also available on `pmuX.o24.voltage`.

24.2. Document history

| Version | Date | Changes |
|---------|------------|-----------------|
| 1.0 | 2024.01.19 | Initial release |

25. Appendix D - How-to Merge Projects

25.1. Description

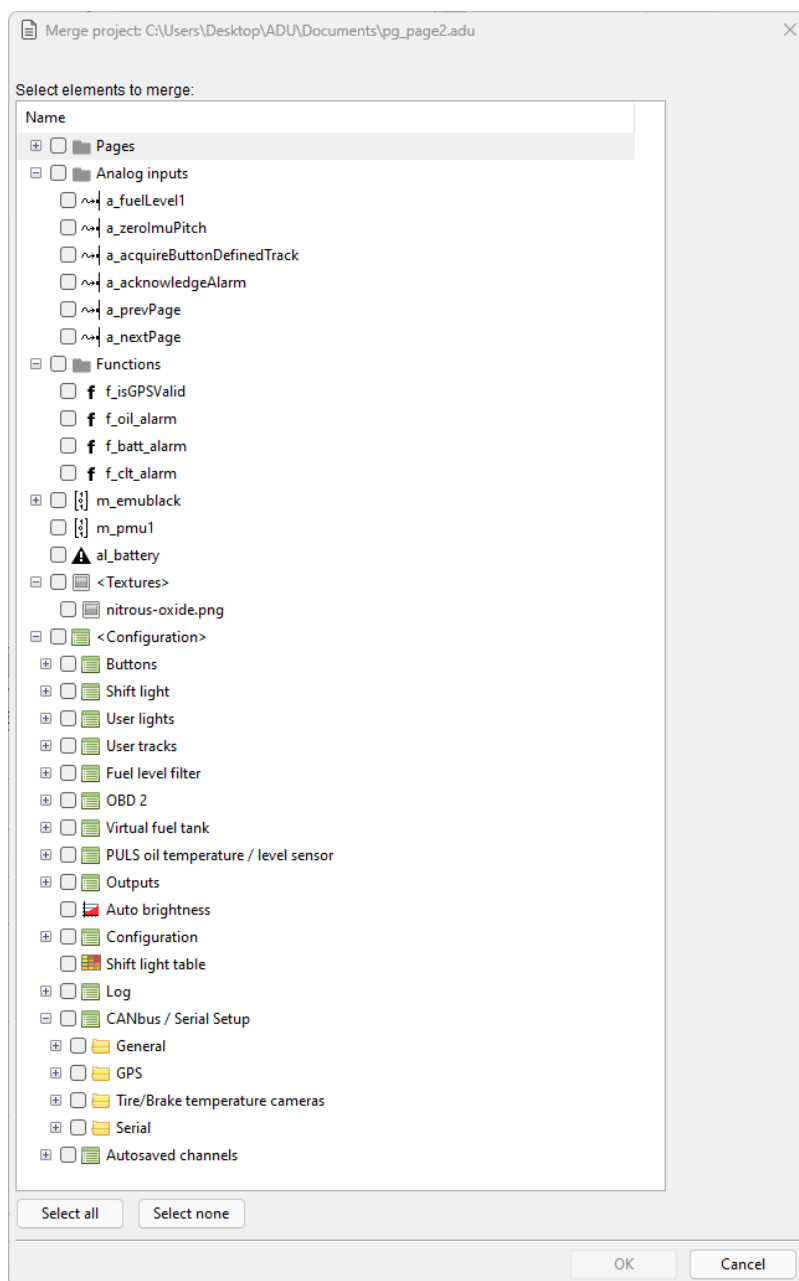
Merging projects enables you to pick specific elements from one project and integrate them into another.

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
- Timers
- Tables
- Switches
- Numbers
- Functions
- CANbus exports
- Pages
- Alarms
- Textures
- 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.

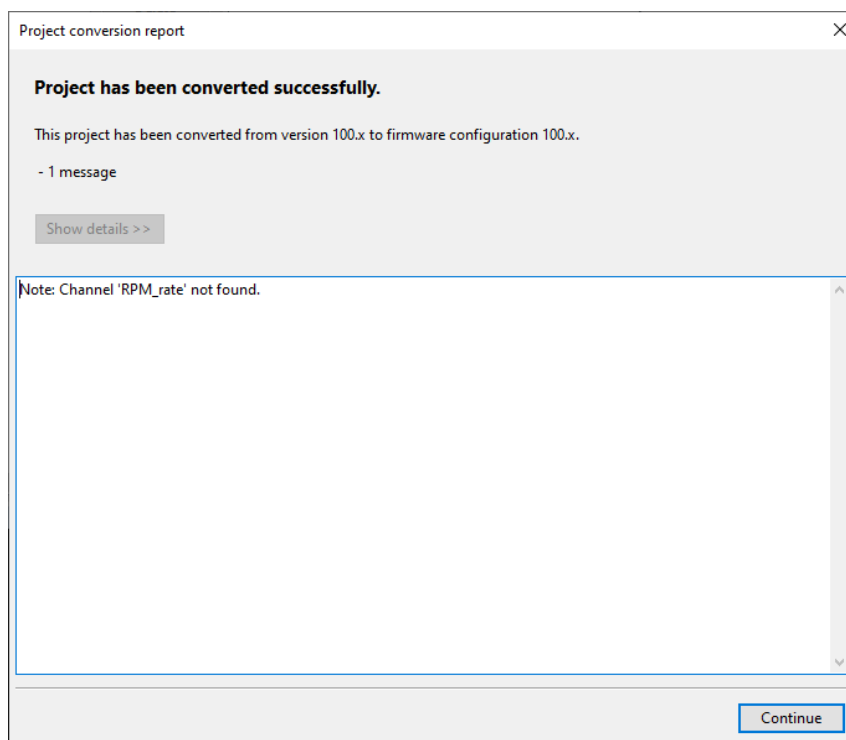
25.2. Resolving potential errors

Name conflict

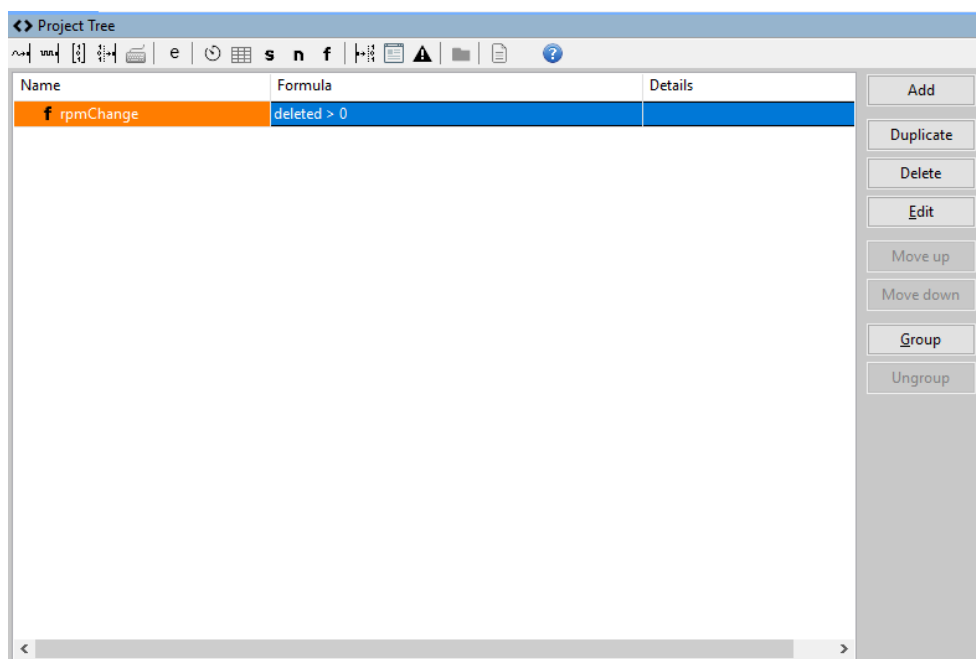
Name conflict is a situation where an incoming element has the same name as any element from the base project. In this case, the incoming element will be assigned a name with a '2' appended at the end.

Missing dependency

During the project merging process, you may encounter a missing dependency error. This occurs when 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.

25.3. Document history

| Version | Date | Changes |
|---------|------------|---|
| 1.0 | 2024.01.19 | Initial release |
| 1.1 | 2024.02.29 | Add the list of mergeable elements Separated a section on potential errors |

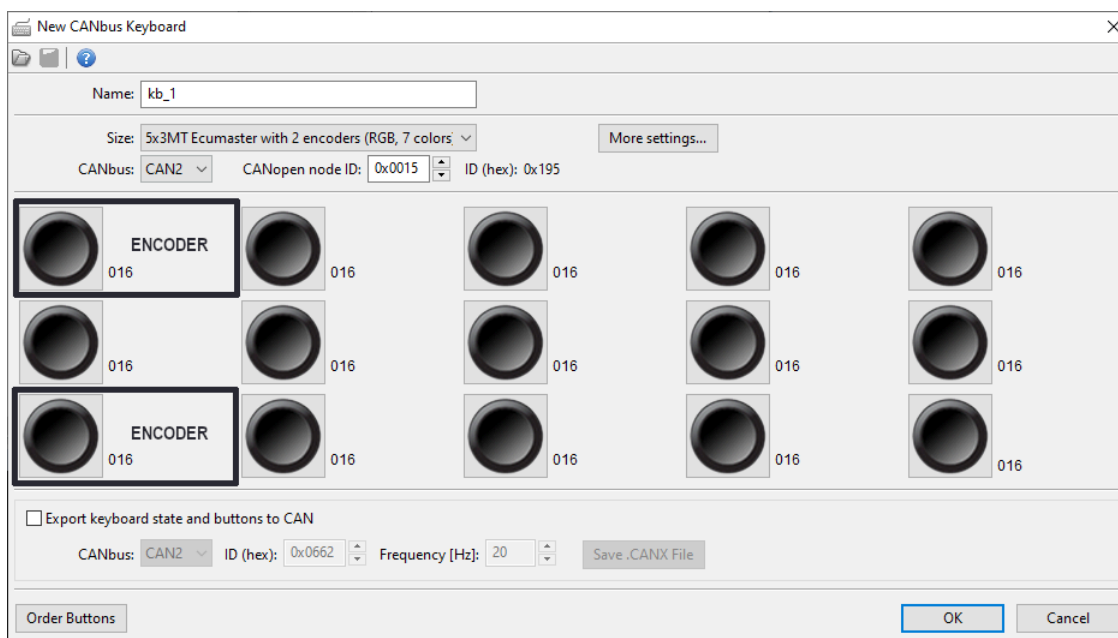
26. Appendix E - How-to Configure 5x3MT Encoders

26.1. 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 ADU 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:



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.

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 unless the *Autosaved channels* feature is used. (Refer to the how-to guide document for detailed instructions on using *Autosaved channels* https://www.ecumaster.com/files/ADU/HowTo/How_to_Configure_Autosaved_Channels_in_ADU.pdf)

2. **page changer**

Turning the encoder clockwise moves to the next page, and turning counter-clockwise goes back to the previous page. In this mode, the LEDs illuminate in a fan-like pattern, offering a visual indication of the operating mode.

- *Wrap pages* decide if the encoder should jump from the last page to the first and vice versa.

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.

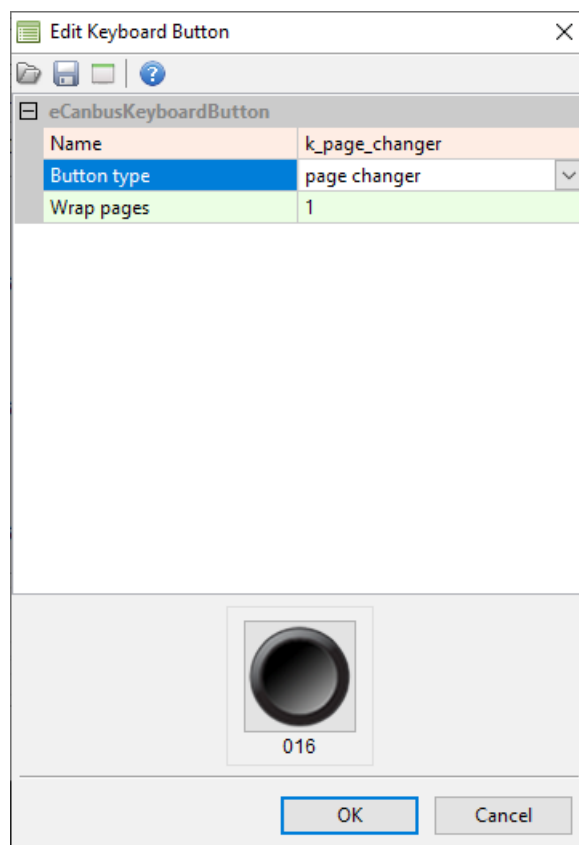
Each *parameter selector* comes with its own range and state, updating the *parameter controller* when pressed.

- 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*.)

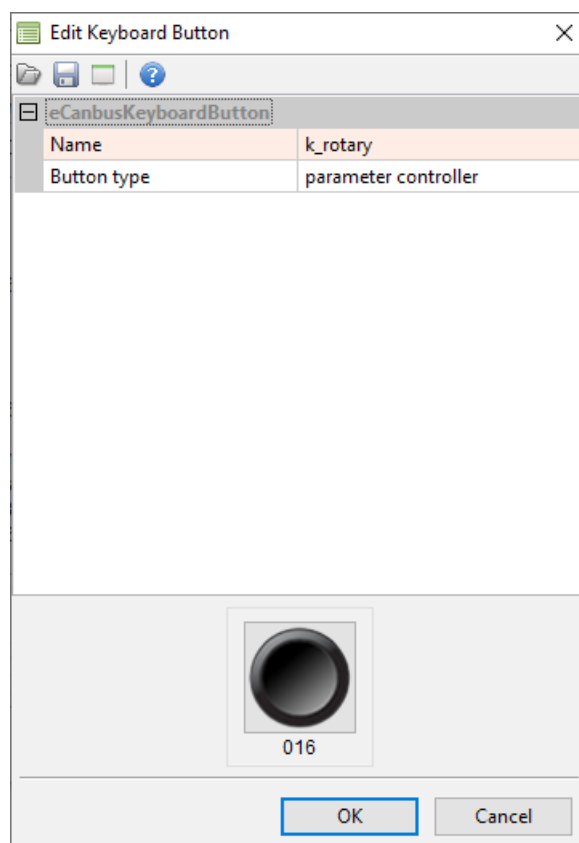
26.2. Example

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

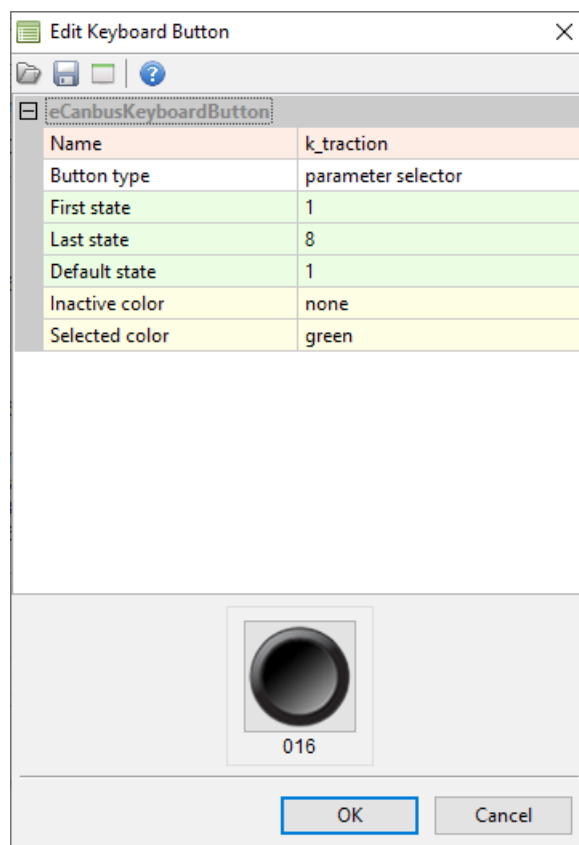
Let's start with the pages. In the keyboard settings, select the top-left box for the rotary encoder. Set it as a '*page changer*' and determine whether pages should wrap around.



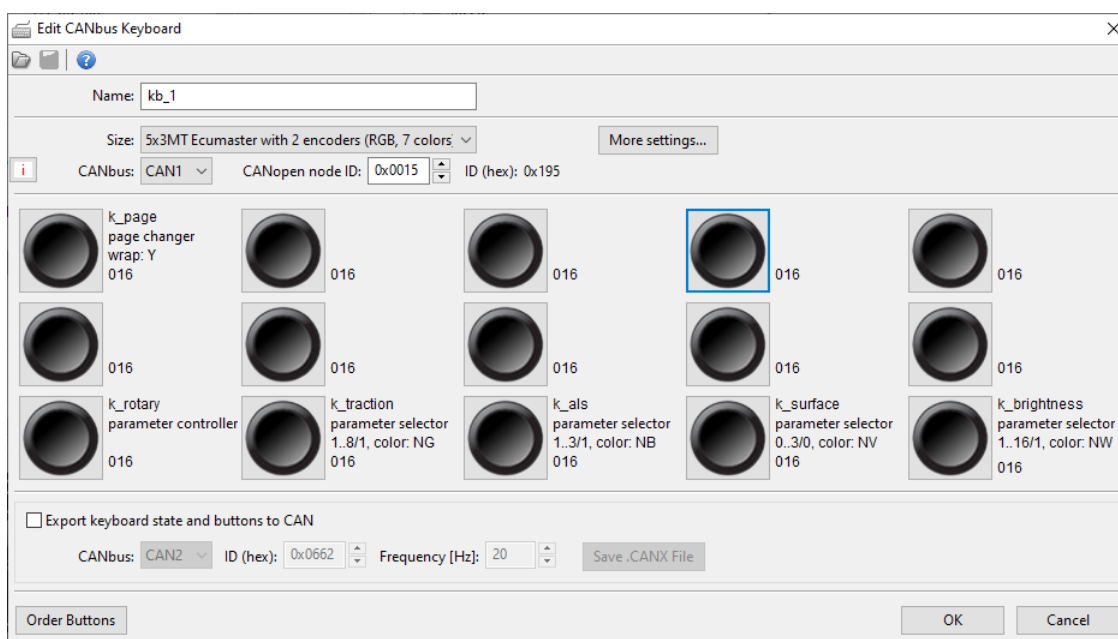
Moving on to our second goal, we aim to use the other encoder for controlling four parameters. Begin by selecting the bottom-left box and designate it as a '*parameter controller*'.



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.

26.3. Document history

| Version | Data | Changes |
|---------|------------|-----------------|
| 1.0 | 2024.02.27 | Initial release |

27. Appendix F - How-to Use Virtual Fuel Tank

27.1. Description

The Virtual Fuel Tank feature helps you know how much fuel is left in your tank. By analyzing fuel usage data from the ECU (*ecu.usedFuel* channel), it not only tells you the remaining fuel but also estimates how many laps you can do with that fuel. This helps you plan your race strategy based on your fuel levels.

| Parameter | Description |
|-----------------------------------|--|
| Fuel tank size definition | <ul style="list-style-type: none">- Fixed volume – tank is always filled to its maximum volume (defined by <i>Fuel tank size (fixed)</i>)- Defined by channel – volume of fuel in the tank after the refill is defined with a channel (<i>Fuel tank size channel [L]</i>) |
| Fuel tank size (fixed) | Capacity of fuel tank in litres. This parameter is active when the <i>Fuel tank size definition</i> is set to " <u>Fixed Volume</u> " |
| Fuel tank size channel [L] | Channel defines total volume of fuel in the tank after refill (what was left + what was filled). This parameter is active when the <i>Fuel tank size definition</i> is set to " <u>Defined by channel</u> " |
| Used fuel channel | The channel contains used fuel information sent by the ECU. By default, this is the <i>ecu.usedFuel</i> channel |
| Max used fuel change | Maximum acceptable difference between two consecutive readings of fuel consumption information. If the change is greater than this parameter, it will be ignored. The parameter also enables the function to work correctly if the fuel consumption channel in the ECU is reset |
| Fuel usage correction | Correction factor when there is a difference between the fuel consumption information provided by the ECU and the actual fuel consumption. This parameter is adjusted empirically based on the measured actual fuel consumption |
| Stage mode channel | The Virtual Fuel Tank can monitor fuel consumption and distance traveled simultaneously in two modes: stage and road. Each mode has two channels for distance and used fuel. The Stage mode channel determines which mode is active. |

| Parameter | Description |
|-----------|--|
| | <p>If the value of the channel assigned in this field is 0 - used fuel and covered distance is assigned to the road channel.</p> <p>If the value of the channel assigned in this field is 1 - used fuel and covered distance is assigned to the stage channel.</p> |

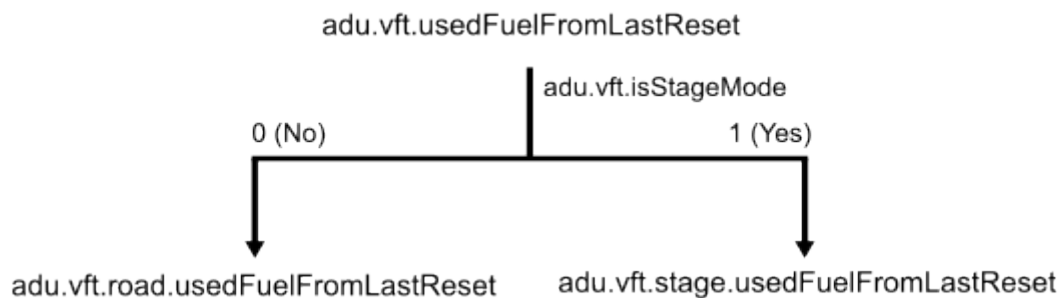
| Channel | Description |
|---|---|
| <i>adu.vft.fuelTankSize</i> | Currently used fuel tank size |
| <i>adu.vft.remainingFuel</i> | Amount of fuel remaining in the tank |
| <i>adu.vft.numLapsOnRemainingFuelAvg</i> | Number of laps that can be completed using the fuel remaining in the tank, based on the average fuel consumption (Fuel used per lap avg) |
| <i>adu.vft.numLapsOnRemainingFuelLast</i> | Number of laps that can be completed using the fuel remaining in the tank, based on the fuel consumption of the last lap (Fuel used per last lap) |
| <i>adu.vft.usedFuelPerLap</i> | Average fuel consumption per lap (since resetting the fuel tank) |
| <i>adu.vft.usedFuelPerLastLap</i> | Fuel consumption on the last lap |
| <i>adu.vft.usedFuelFromLastReset</i> | Fuel used since last virtual fuel tank reset |
| <i>adu.vft.isStageMode</i> | Shows whether fuel is assigned to stage or road |
| <i>adu.vft.stage.distanceFromLastReset</i> <i>adu.vft.road.distanceFromLastReset</i> | Distance traveled in stage/road mode since the last refueling |
| <i>adu.vft.stage.usedFuelFromLastReset</i> <i>adu.vft.road.usedFuelFromLastReset</i> | Fuel used in stage/road mode since the last refueling |

Additionally, it's important to set up a button to reset the Virtual Fuel Tank. In the *Buttons* panel, there's a button defined as "Reset virtual fuel tank." This button is crucial because the driver needs to press it when refueling. When pressed, it resets the fuel level to full and begins a fresh count of all the data.

The Virtual Fuel Tank data is saved in the device's memory, so even when you turn off the ignition, it's retained. Properly connecting the device to power is crucial for this function to work correctly, requiring separate connections for the battery and ignition switch. This ensures that even when the ignition is off, the settings remain saved.

Using Stage mode channel If there's a significant difference in fuel use, like when driving to a special stage compared to driving the stage itself, you can measure fuel use for two separate situations: on the road and during the stage. With this setup, the overall fuel use channel, *adu.vft.usedFuelFromLastReset*, gets additionally divided into two separate channels: *adu.vft.stage.usedFuelFromLastReset* and *adu.vft.road.usedFuelFromLastReset*. This lets you

analyze and predict fuel use in more detail. The channel determining which of the two channels to assign the data is the one defined in the setting: *Stage mode channel*.



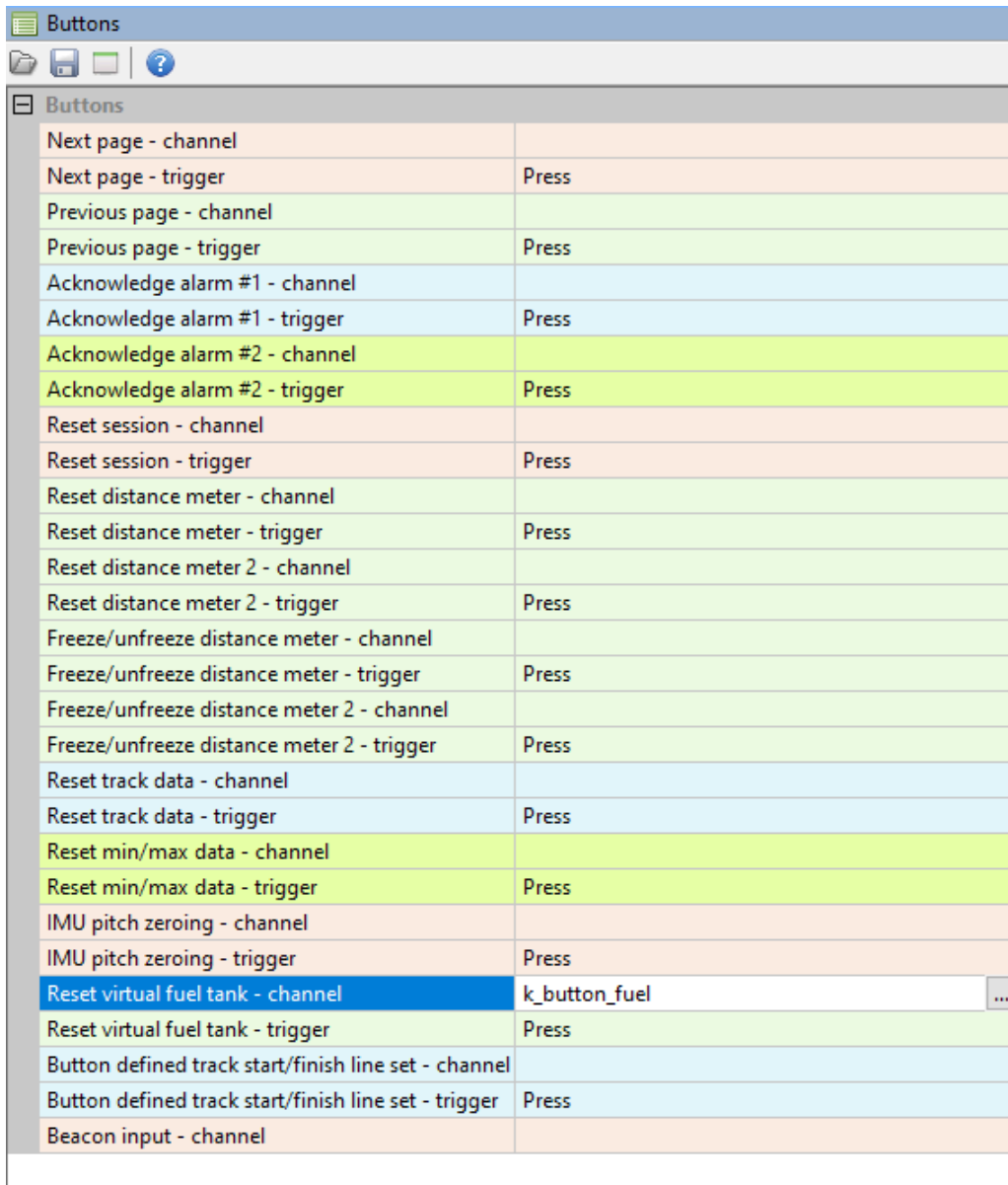
Similarly, with the distance channels: *adu.vft.stage.usedFuelFromLastReset* and *adu.vft.road.usedFuelFromLastReset*.

Using Fuel tank size channel [L] You can adjust what is the value of the Virtual Fuel Tank when you reset it. This is possible when the *Fuel tank size definition* is set to "Defined by channel". Set the *Fuel tank size channel [L]* to the channel indicating the amount of fuel in the tank after refill, which includes the remaining amount plus what was added. For example, this can be adjusted using a knob during refill. Then, pressing the "Reset virtual fuel tank" button will set the remaining fuel value to the current channel's value.

27.2. Example

In this example we will cover a step-by-step basic configuration and use the Virtual Fuel Tank (VFT) for a real-time analysis of the consumption data.

To properly configure the VFT, it's very important to set up the button *Reset virtual fuel tank* in the *Buttons* panel:



| Buttons | |
|--|---------------|
| Next page - channel | |
| Next page - trigger | Press |
| Previous page - channel | |
| Previous page - trigger | Press |
| Acknowledge alarm #1 - channel | |
| Acknowledge alarm #1 - trigger | Press |
| Acknowledge alarm #2 - channel | |
| Acknowledge alarm #2 - trigger | Press |
| Reset session - channel | |
| Reset session - trigger | Press |
| Reset distance meter - channel | |
| Reset distance meter - trigger | Press |
| Reset distance meter 2 - channel | |
| Reset distance meter 2 - trigger | Press |
| Freeze/unfreeze distance meter - channel | |
| Freeze/unfreeze distance meter - trigger | Press |
| Freeze/unfreeze distance meter 2 - channel | |
| Freeze/unfreeze distance meter 2 - trigger | Press |
| Reset track data - channel | |
| Reset track data - trigger | Press |
| Reset min/max data - channel | |
| Reset min/max data - trigger | Press |
| IMU pitch zeroing - channel | |
| IMU pitch zeroing - trigger | Press |
| Reset virtual fuel tank - channel | k_button_fuel |
| Reset virtual fuel tank - trigger | Press |
| Button defined track start/finish line set - channel | |
| Button defined track start/finish line set - trigger | Press |
| Beacon input - channel | |

Each time the car is refueled, the driver has to press the button to let know the system, the tank is full once again. This action will reset the remaining fuel.

In this example, we will cover the basic mode of operation, with the fuel tank being filled to full every time. Hence, we will leave the *Fuel tank size definition* as "Fixed volume". Next, we have to input the *Fuel tank size*.

To enable the calculation of the fuel, it's necessary to configure the *ecu.usedFuel* channel. Most ECUs are able to export this data based on injectors flow and injection times.

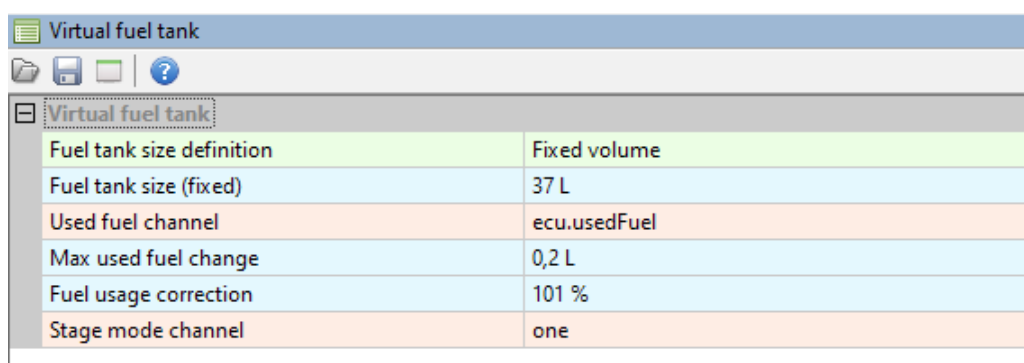
Max used fuel, as defined before, is maximum acceptable difference between two consecutive readings of fuel consumption information. If the change is greater than this parameter, it will be ignored. For most cases, the default parameter 0,2 l should be correct. A typical race car uses about 0,5 l of fuel per minute of race drive.

The last parameter, *Fuel usage correction*, can be used for precise adjustment of the calculations. We recommend testing it before the event. To determine the value of this parameter, measure the actual volume of fuel used during a session and the last value of the *adu.vft.fuelUsedFromLastReset*.

Fuel usage correction = (Actual volume of fuel / *adu.vft.fuelUsedFromLastReset*) * 100%.

If the real value was greater than the VFT value, the *Fuel usage correction* value will be greater than 100%, and vice versa.

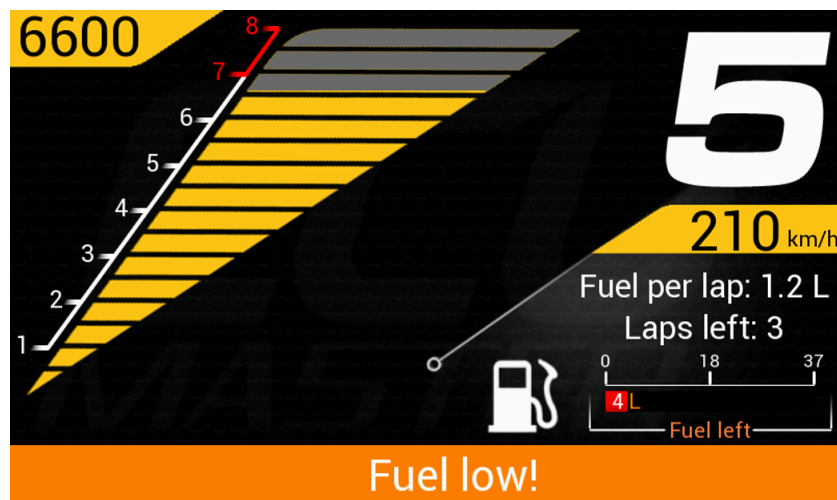
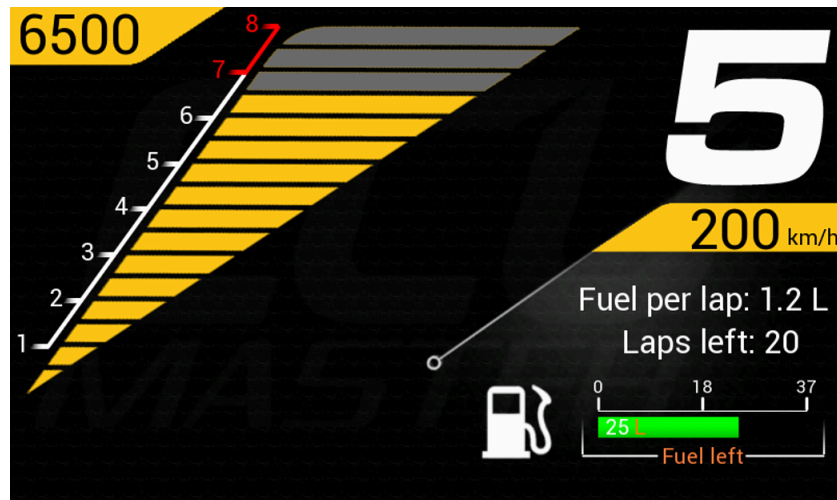
In this case, we will utilize only one used fuel meter, hence the *Stage Mode Channel* will be left to default channel one. For more details on the *Stage Mode Channel*, see: [Description \(on page 223\)](#).



| | |
|---------------------------|--------------|
| Virtual fuel tank | |
| Virtual fuel tank | |
| Fuel tank size definition | Fixed volume |
| Fuel tank size (fixed) | 37 L |
| Used fuel channel | ecu.usedFuel |
| Max used fuel change | 0,2 L |
| Fuel usage correction | 101 % |
| Stage mode channel | one |

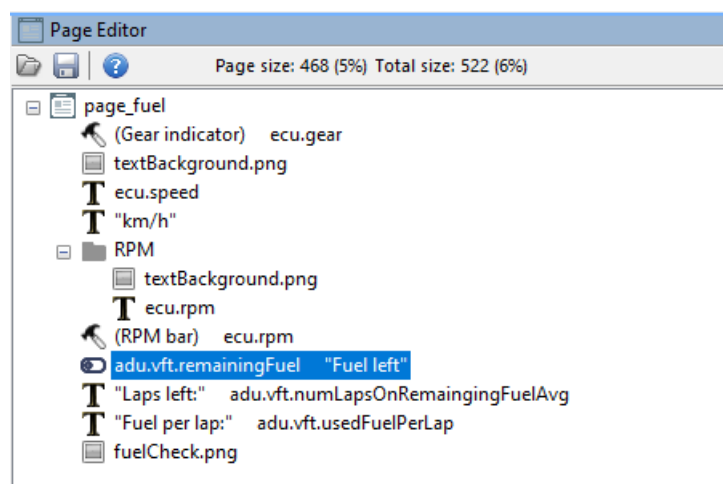
Now, with the VFT configured, let's utilize its abilities.

We'll display the remaining fuel, average fuel usage, and laps left on one page, with a warning if fuel drops below 5 liters. You can easily change an existing page or add a new one using the *Project Tree* panel. We'll use the *Page Editor* and *Dashboard Preview* for simple layout.



The current document covers practical use of the ADU features. For more details on using the *Page Editor*, refer to the ADU Manual.

These are all the elements of our example page:



Firstly add the *Bar graph* to a page.

| | |
|-------------------------|-------------------------------------|
| Position | |
| Position X | 557 |
| Position Y | 331 |
| Channel and unit | |
| Channel | adu.vft.remainingFuel |
| Decimal places | 0 |
| Min | 0 |
| Max | 37 |
| Display value | <input checked="" type="checkbox"/> |
| Update frequency | 50 Hz |
| Display unit | <input checked="" type="checkbox"/> |
| Unit | L |
| Custom unit | |
| Description | |
| Text | Fuel left |
| Icon | |
| Use icon | <input type="checkbox"/> |
| Icon texture | |
| Icon scale | 130 % |
| Alarm | |
| Alarm channel | |
| General | |
| Bar type | Horizontal |
| Bar style | Style 1 |
| Length | 200 |
| Width | 20 |
| Border width | 1 |
| Number of ticks | 3 |
| Decimal places for tick | 0 |
| Colors | |
| Color | <input type="color"/> (255,255,255) |
| Background color | <input type="color"/> (0,0,0) |
| Bar color | <input type="color"/> (0,255,0) |
| Alarm color | <input type="color"/> (255,0,0) |
| Icon color | <input type="color"/> (255,255,255) |
| Value color | <input type="color"/> (255,255,255) |
| Text color | <input type="color"/> (255,128,64) |
| Unit color | <input type="color"/> (255,128,0) |
| Fonts | |
| Tick font | 1 |
| Value font | 2 |
| Text font | 2 |
| Functions | |
| Redline when | When value below |
| Redline start | 5 |
| Visibility channel | one |

The suggested configuration of the **Bar graph**:

- In the *Channel* field select *adu.vft.remainingFuel*.
- Set *Max* to the value previously input as the *Fuel tank size*.
- You can add some description in *Description / Text*, we entered "Fuel left"

For some extra information, you can define when the bar turns red:

- *Redline when* set to "When value below"
- *Redline start* in our case it's 5 liters
- Leave *Visibility channel* with it's default value "one"

Next, we'll add a **Text** element informing the driver about remaining laps on fuel. The configuration is straightforward, requiring only two parameters to be filled.

- Add text description in field *Text*. We set it to "Laps left: "
- In the *Channel* field select *adu.vft.numLapsOnRemainingFuelAvg*.

| | |
|--------------------|--|
| Position | |
| Position X | 588 |
| Position Y | 304 |
| Text | |
| Color | <input type="checkbox"/> (255,255,255,255) |
| Font | 4 |
| Italic | <input type="checkbox"/> |
| Two lines | <input type="checkbox"/> |
| Text | Laps left: |
| Text width | 0 px |
| Text align | Left |
| Channel | adu.vft.numLapsOnRemainingFuelAvg |
| Decimal places | 0 |
| Value width | 0 px |
| Value align | Left |
| Unit | user |
| Display unit | <input checked="" type="checkbox"/> |
| Custom unit | |
| Color channel | |
| Visibility channel | one |
| Update frequency | 50 Hz |

Similarly, for the "Fuel per lap" element, follow the same steps.

The final step is to set up an alarm that triggers when the fuel level drops below 5 liters. To do this, navigate to the *Project Tree* and add a new *Alarm*. For further instructions on using the *Project Tree*, refer to the ADU Manual.

| eDashboardAlarm | |
|-------------------------------|-------------------------------------|
| Name | alarm_fuel |
| General | |
| Type | Warning |
| Text (use # to display value) | Fuel low! |
| Position | Bottom |
| Channel | |
| Channel | adu.vft.remainingFuel |
| Decimal places | 0 |
| Unit | L |
| Display unit | <input checked="" type="checkbox"/> |
| Condition | |
| Condition | Less or Equal |
| Value | 5 L |
| Qualifier | |
| Use qualifier | <input type="checkbox"/> |
| Delay | |
| Guard time | 0 s |
| Acknowledge | |
| Allow acknowledge | <input checked="" type="checkbox"/> |
| Step | 1 L |
| Presentation | |
| Min show time | 3 s |
| Retrigger time | 3 s |

In the example, we defined the following parameters, leaving the rest as default:

- Add a meaningful *Name*
- Add an informative *Text* for the driver, such as "Fuel low!".
- In the *Channel* field select *adu.vft.remainingFuel*.
- Set the *Condition* to "Less or equal"
- Set the *Value* to the trigger point for the alarm. In our case that's 5 L.
- Set the *Step* to 1 L. Even if the driver acknowledges the alarm, it will reappear if the fuel level drops by another liter.

27.3. Document history

| Version | Date | Changes |
|---------|------------|-----------------|
| 1.0 | 2024.04.19 | Initial release |

28. Appendix G - How-to Configure Autosaved Channels

28.1. 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

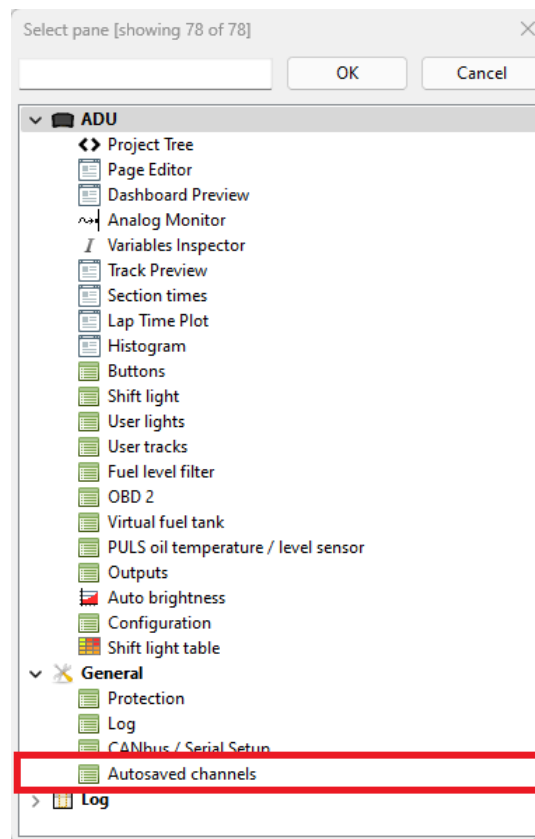
The Autosave Channels prove useful in various scenarios, such as preserving the state of any encoder. By doing so, users can easily recover the encoder's state after the device is turned on.

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

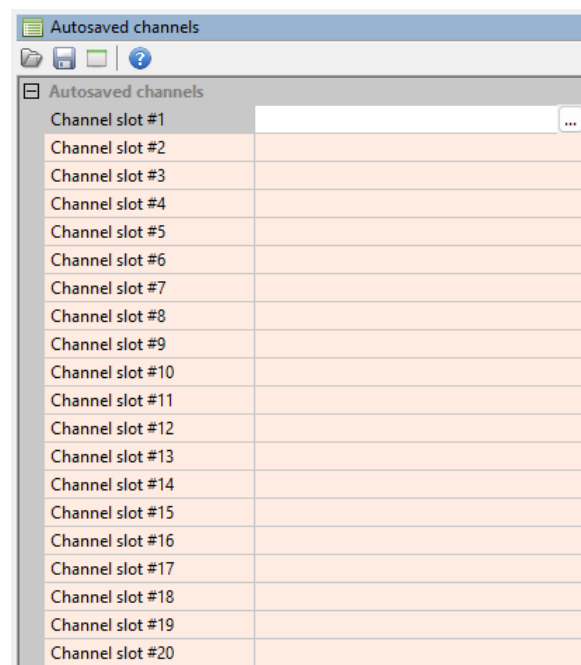
- **Ignition switch** – the save of all channels occurs when the ignition switch +12V is disconnected, but the constant battery +12V remains powered (requires separate wiring for battery and ignition switch). This method provides most reliable results.
- **On change** – after detecting a change in any of the autosaved channels, a save occurs, followed by a 2-minute period without further saving. If none of the selected channels change their values, no save occurs, and the system awaits a channel change.

The 2-minute intervals without saving are implemented to prevent memory wear as it is not designed to handle rapid changes.

To open the *Autosave Channels* panel, press F9 and select it from the *General* category.



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.

28.2. Document history

| Version | Data | Changes |
|---------|------------|----------------------|
| 1.0 | 2024.01.19 | Initial release |
| 1.1 | 2024.05.13 | Improved description |

29. Document history

| Version | Date | Changes |
|---------|------------|--|
| 1.206 | 2020.04.09 | Virtual fuel tank added Button defined track added Updated list of automatically detected race tracks Correction of "Signal (transmit) for RS232 serial bus" |
| 92.2 | 2023.09.12 | Keyboard configuration added Enumerations added Added specification of ADU 5 Rev.2 and ADU 7 Rev.2 CAD technical drawings added USBtoCAN error descriptions added Channel Simulator description added DBC Importer added Updated list of mathematical / logical operations Added description of Delayed Turning Off functionality Added description of built-in support for Ecumaster PMU Added description of default project "pg_page1.adu" Description of new controls added: Session Results, Grid The possibility of changing pages with a variable / rotary switch is described Added description for PULS Oil Level and Temperature Sensor Added description of the "Time Range" parameter Predictive time graph Added description of gradients for the Bargraph control Added channel description for Alarm objects Descriptions of visualisations related to data analysis have been expanded: Channel Report, Histogram, Lap Time Plot View, Scatter Plots, Section Times, Track Preview Added description of Memory Report window Added channel description for Ecumaster Brake Temperature Cameras (btc. channels) Added table of reserved CAN IDs |
| 104.1 | 2024.08.23 | New Ecumaster standard layout applied Added description of <i>Autosaved Channels</i> panel Updated description of <i>Timers</i> |

| Version | Date | Changes |
|---------|------------|--|
| | | Updated description of <i>Virtual fuel tank</i> Updated description of <i>Buttons</i> Updated description of <i>Shift light</i> Added How-to documents as appendices: "How-to Configure the Hill Climb" "How-to Configure PMU CAN Stream" "How-to Merge Projects" "How-to Configure 5x3MT Encoders" "How-to Use Virtual Fuel Tank" |
| 104.1.1 | 2024.09.19 | Corrected description of <i>Freeze / unfreeze distance meter</i> and <i>Freeze / unfreeze distance meter2</i> parameters in the <i>Buttons</i> chapter |
| 104.1.2 | 2024.09.26 | Added note about 'GPS to CAN' module support in the <i>GPS module</i> and <i>Timing</i> chapters. |
| 104.1.3 | 2024.10.03 | Added 11 built-in tracks from Sweden to <i>Appendix A - List of built-in tracks</i> . |
| 104.1.4 | 2024.10.08 | Added information about CAN1 and CAN2 bus termination. |
| 104.1.5 | 2025.02.05 | Detailed information about CAN added to the <i>Characteristics</i> table |