Hardwire electronics

PDM15/25/35 & PDM28

Instruction Manual



www.hardwire-electronics.co.uk

Version 5 - 18/11/26

Change Log

Version	Author	Date	Changes
1.0	George S	28th July 2023	Initial Release
2.0	George S	22nd November 2024	Configuration Software Password Input - Turn on delay, EMA filter Generic Functions Sensor Calibrations 2D Tables Can Output Stream LIN Bus
3.0	George S	3rd June 2025	USB to CAN connection Timers pulse/duration Maths sin cos etc CAN output float CAN Stream overhaul Configuration warning Monitor tab variable viewer graphs
4.0	George S	22nd April 2026	Tripped under voltage Capture view Configuration warnings CAN Stream DBC export CAN J1939 output stream CAN keypad dial configuration CAN Wizard
5.0	George S	18th November 2026	PDM scripts and examples IMU and 2D Tables CAN input bus selection and automatic filters CRC profiles and rolling counters Monitor, logging and configuration tools PDM15/25/35 and PDM28 manuals combined

Contents

Introduction 6
Specification 6
System Overview Diagram 9
Installation 10
Mounting 10
PDM15/25/35 Pinout 13
PDM28 Pinout 13
PDM15/25/35 Device Power 14
PDM28 Device Power 14
PDM28 Status LEDs 14
Inputs 15
Outputs 15
CAN Bus Wiring 15
Configuration Software 16
- Connection 16
- Offline Mode 16
- Software and Firmware Versions 17
- Configuration - Configuration Files 17
- Undo 17
- Configuration Synchronisation 17
- Opening an Older Configuration 17
- General 17
- CAN Bus Settings 18
- Reset Function 18
- Reset Now 18
- Cut Off Function 18
- Password 19
- Notes 19

Inputs 19
- Input Label 19
- Voltage/Frequency 19
- Mode 20

Outputs 21
- Output Label 21
- Thermal Shutdown 23
- Auto-tune 23
- Follow Output 24
- Copy and Paste 24
- Disable Diagnostic Pulse 25

Timers 25
- Timer Range 26

Counters 27
Maths Channels 27
Scripts 29
Generic Functions 36

Contents

Logic Lines 36
Sensor Calibrations 37
IMU 38
2D Tables 38
CAN Keypads 39
Keypad Configuration 40
Remember Button States 40
Keypad Button Configuration 41
Keypad Button Icons 41
CAN Keypad Dial Configuration 42
Pass Key 42
CAN Bus Basics 42
For a full detailed video explanation of the CAN Bus, check out our 43
beginners guide which can be found in the following QR code 43
CAN Inputs 43
CAN Configuration Files 43
DBC File Loader 43
CAN Bus Filter Configuration 44
Optimise CAN Filters 45
CAN Input Configuration 45
Use Timeout 46
Timeout Flag 47
CAN Outputs 48
User CAN Outputs 48
Signals 49
Custom Signed and Unsigned Fields 49
Flag and Float Fields 50
Start Bit and Byte Order 50
Remote Frames 50
DBC Export 50
CAN Output Stream 52
CAN Stream DBC Export 52
CAN J1939 Output Stream 52
LIN Bus (PDM28 only) 53
Signals 54
Frame Timing 58
Data Logging 58
Configuration Warnings 58

Contents

Monitoring 59
Overview 59
View Panels 60
Arrange Panels 60
Custom Variables 60
Overview Graphs 60
Save and Load View 60
Show Disabled Items 60
Graph View 61
Graph Interpolation 61
Variable View 61
History Window 62
Variable Logging 62
Capture View 62
Cancel Capture 62
CAN Bus Analyser 63
DBC Decoding 64
Scheduled Transmit 64
Device Statistics 65
Calibration 66
Zero Calibration 66
Scaler Calibration 67
Logging 67
Log Library 67
Resume Download 67
Import Log 67
Graph Workspace 67
Export CSV 68
Format Log Storage 68
Firmware Updates 68
PDM Firmware Update 68
Firmware Selection 68
Update Progress 69
Configuration Backup 69
Pin Manager 69
Pin Notes 70
Export Pin Report 70
Loom View 70
Examples 71
Script example for cooling demand 74

Introduction

A PDM (Power Distribution Module) is a device used to replace conventional relays and fuses in a vehicle electrical system. The Hardwire Electronics PDM15, PDM25, PDM35 and PDM28 take inputs in the form of physical switches, analogue voltages, or CAN bus messages, and provides power to different electronic devices - such as radiator fans, fuel pumps, ECUs and headlights - using user defined logic functions. The current being drawn from each connected device is continuously monitored. If the measured current is too high due to a fault, the PDM switches off the respective output to prevent further damage to the wiring loom or the connected device. The PDM can then retry the output to see if the fault has cleared, or the user can manually reset the outputs via an input switch. PDMs offer distinct advantages over conventional systems using fuse boards and relays, listed below.

• Greatly simplifies system wiring and reduces system component count, overall saving you money when labour/ time is accounted for.
• Reduces the weight of the electrical system.
• Provides increased reliability over traditional systems using mechanical relays.
• Solid state technology removes the problem of relay contact corrosion and relay bouncing in vibration heavy applications.
• Current draw can be monitored and logged to pre-emptively detect faults.
• CAN bus communication allows your PDM to send/receive data from other CAN bus devices in the vehicle, such as the vehicle ECU. This paves the way for advanced control/ safety strategies such as automatic engine cut offs, fuel pump cut offs, radiator fan control and more.
• Easy integration with popular CAN bus keypads, improving system reliability and functionality.

This manual covers the PDM15, PDM25, PDM35 and PDM28. The PDM15, PDM25 and PDM35 share the same enclosure and connectors, and differ in the number of outputs fitted, so they are referred to together as the PDM15/25/35. Where the hardware or a feature differs between models, the model is named in the text and tables. Everything else applies to all models.

Specification

	PDM15/25/35	PDM28
General		
Size	200x160x54mm	318x124x58mm
Weight	PDM15 - 645g PDM25 - 690g PDM35 - 735g	1640g
Operating Temperature	-55 - 90°C [-67 - 194°F]	
Operating Voltage	4-32V	
Quiescent Current	4-5mA	
Operating Current	210mA at 12V	-
5V Output Current	100mA maximum	Not available
Construction	Glass fibre reinforced plastic	Machined Aluminium Enclosure, Media Blasted and Anodised Black
IP Rating	IP67 Water and Dust Ingress Protection	
Connectors	4x Deutsch DT 12-pin	2x 12-Pin DTM, 2x 12-Pin DT, 1x 4-Pin DTP
Power Connection	M8 battery positive stud	2x Amphenol RADLOK (positive and negative)
Indicator LEDs	-	50x RGB LEDs
Certifications	UNECE Reg10.06 and FCC	-
Inputs		

	PDM15/25/35	PDM28
Type	16 analogue inputs which can also serve as digital inputs. On the PDM35, inputs 7-16 share pins with outputs 26-35.	16 analogue inputs which can also serve as digital inputs.
Measurement Range	Each input can measure voltages between 0-28V, independent of the battery voltage.	Input voltage range depends on hardware and pin. Refer to the input warning in the Configurator.
Measurement Resolution	12-bit or 0.007V	16-bit or 0.001V
Frequency Measurement	Automatic frequency measurement up to 300Hz	
Protection	Fully protected for input voltages between -24V to 40V.	
Pull Up/Pull Down Resistors	40 kΩ pull-down or 200 kΩ pull-up, fixed.	Selectable pull-up or 10 kΩ to ground. See the pin table for pull-up values.
Added Features	Configure as active high or low. Configurable Hysteresis voltage. Latching or momentary operation.	

Specification

	PDM15/25/35	PDM28
Outputs		
Type	4 high/low side outputs 80A peak, 20A continuous, with reverse current protection. (Pin-limited to 13A @125degC) High side outputs 80A peak, 20A continuous. (Pin-limited to 13A @125degC) PDM15 - 11, PDM25 - 21, PDM35 - 31 high side outputs.	4 high/low side outputs 100A peak, 30A continuous, with reverse current protection. (Pin-limited to 25A @125degC) 6 high/low side outputs 80A peak, 20A continuous, with reverse current protection. (Pin-limited to 13A @125degC) 18 high/low side outputs 80A peak, 20A continuous. (Pin-limited to 13A @125degC)
Combined	120A combined output maximum current	200A combined output maximum current
Output current control steps	100mA	
Protection	Each output is overcurrent, overtemperature, load dump and reverse polarity protected.	
Soft Start	PWM soft start can be enabled on outputs to limit inrush current.	
PWM	Each output can be Pulse Width Modulated based on a user defined variable. This can limit the speed of radiator fans, fuel pumps, and dim lights. 1000Hz maximum PWM frequency.	
Added Features	Outputs can be configured to stay on for a specified time after the input is removed, useful for thermo-fans. Outputs can be retried multiple times after an over-current event is detected.	
LIN Bus		
Interface	Not available	1x LIN Bus Port.
CAN Bus		
Interface	1x CAN2.0A/B port	3x CAN2.0A/B ports (CAN 1 and CAN 2 for user devices; N-CAN currently unsupported)
Bus Speed	50, 100, 125, 250, 500, 1000 kbps	
Termination Resistor	The CAN bus port has a software selectable CAN termination resistor.	Both CAN bus ports have software selectable CAN termination resistors.
DBC File Import	Easily import DBC files to configure the CAN inputs of the PDM	
Input Stream	100x inputs, fully customisable variable parsing with bit masking, math functions, and compound CAN message filtering.	
Output Stream	100 CAN outputs, up to 32 signals per frame. Periodic or triggered, with scaling, byte order, rolling counter, CRC and DBC export.	
CAN Keypad		
Supported	Blink Marine, Grayhill, Motec, Haltech and Marlin CAN keypads. Up to four keypads can be used at once.	

	PDM15/25/35	PDM28
Logging		
Storage	128Mb of onboard flash storage for logging PDM data.	
Speed	1 Hz to 1000 Hz	1 Hz to 1000 Hz
Viewing	Data can be downloaded and graphed via the Hardwire Electronics PDM configurator software or exported as a .CSV file and graphed in external software.	
Functions		
Output Functions	Each Output can be switched with a user defined function.	
Logical Operators	AND, OR, NOR, XOR, NAND, NOR, >, >=, <, <=, Equal, Not Equal.	
Timers		
Pulse	On/Off Square wave, useful for indicators, wipers etc.	
Duration	Start/stop/reset timer, up to 2,147,483,647 ms (24.9 days).	

Specification

	PDM15/25/35	PDM28
Maths Channels		
Mathematical Operators	Addition, Subtraction, Multiplication, Division, Modulus	
Binary Operators	AND, OR, XOR, Left Shift, Right Shift	
Functions	Choose, Min, Max, Sin, Cos, Tan, Floor, Ceil, Abs, Asin, Acos, Ln, Log, Pow, Sqrt	
Scripts		
Scripts	5 scripts, 10 exports per script. Read PDM values, write counters and keypad states, or transmit CAN frames.	
Counters		
Type	Increment, decrement, and reset based on user defined conditions.	
Sensor Calibration		
Type	20-point Sensor Calibration Tables for sensor linearisation.	
2D Tables		
Type	20x20 2D map tables for advanced behaviour.	
IMU		
Rotation	3-axis rotational position measurement 0.1° resolution	
Gyroscope	2000°/s 3-axis measurement	
Accelerometer	-8g to 8g 3-axis measurement	
PCB Temperature		
Type	Single PCB temperature measurement.	6-Zone PCB temperature measurement.
RTC		
Type	Real Time Clock with battery backup for 30 days. Used to store the correct time and date for data logs.	
USB		
Type	USB 2.0 compliant with Mini-B	
Driver	Integrated driver with Windows 10/11	
PC App		
Name	Hardwire Electronics Pro PDM Configurator Software.	
Compatability	Windows 10/11 machines	
Connectivity	Easy connection to the PDM with USB and windows integrated drivers. No unreliable USB-serial converter needed. Alternatively connect via a Kvaser USB to CAN Device.	

System Overview Diagram

Figure 1 Typical installation of a PDM15/25/35 in a vehicle electrical system.

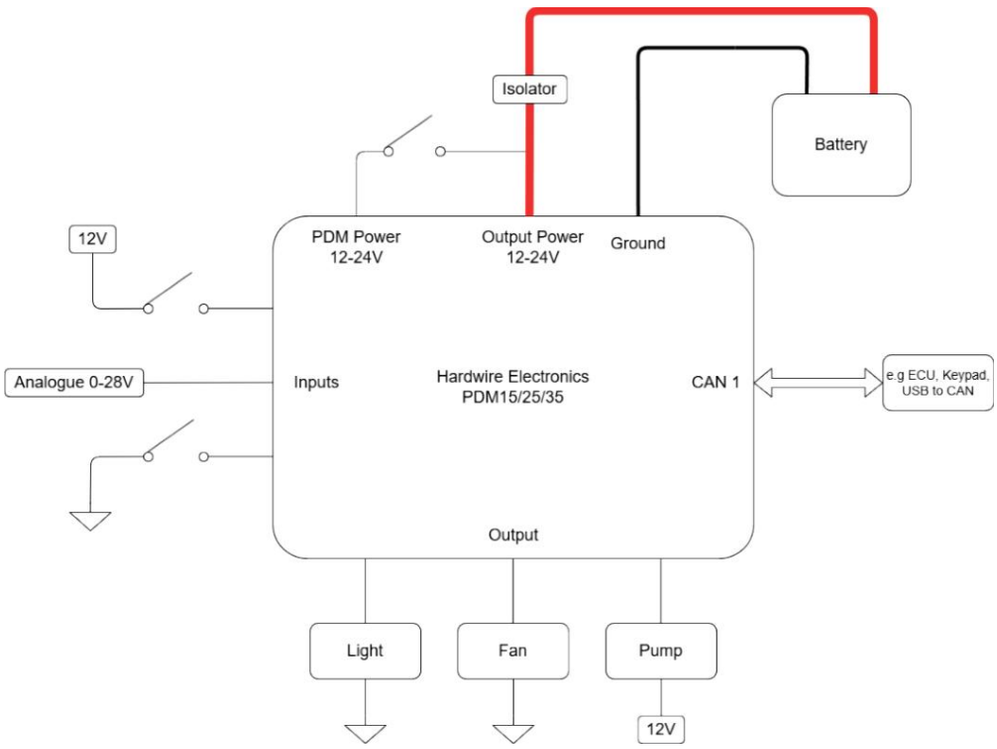
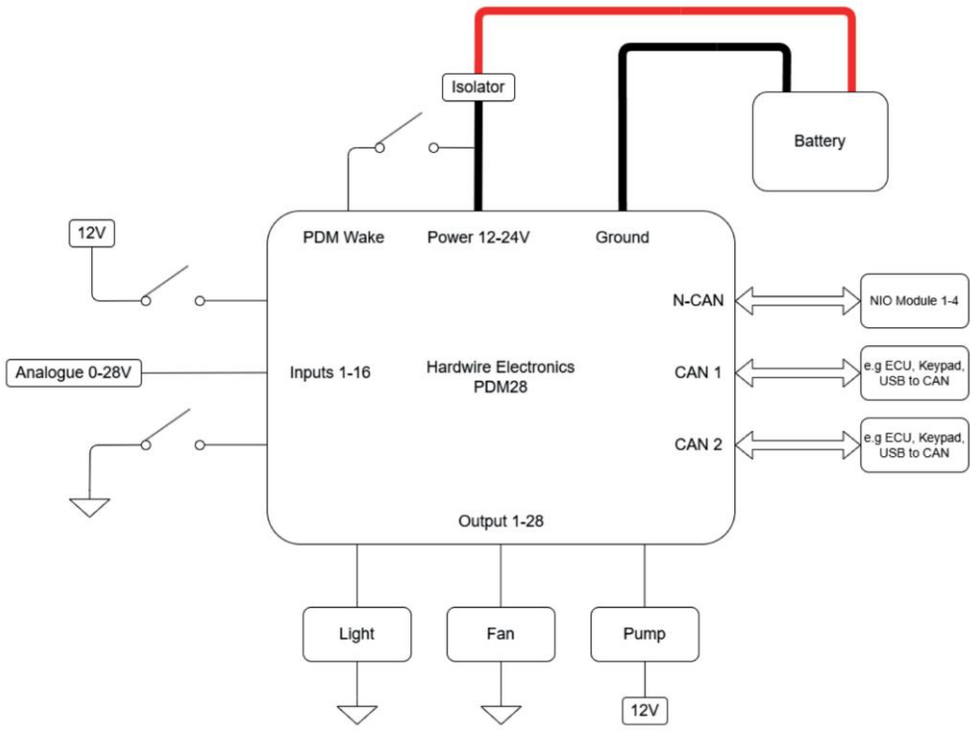


Figure 2 Typical installation of a PDM28 in a vehicle electrical system.



Installation

Mounting

The dimensional diagrams below show the physical dimensions of the PDM15/25/35 and the PDM28, as well as the hole size and spacing for mounting. The PDM should be mounted in a well-ventilated area to cool the unit when under heavy load. Care should be taken to mount the PDM in a location which minimises the exposure to water and dirt. In most cases, mounting the PDM next to the ECU is the most efficient method, as the wires running between both units are kept to a minimum to save on weight. Around 100mm of clearance should be maintained both on top and in front of the PDM to allow room for the wiring loom and USB to be connected without excess strain. It is recommended to mount the PDM on rubber mounts to minimise the vibration coupling from the vehicle to the PDM. This will help with IMU data.

Figure 3 PDM15/25/35 dimensions in mm.

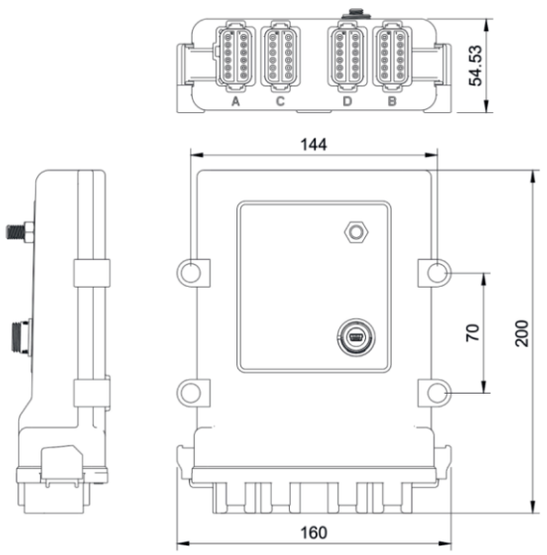
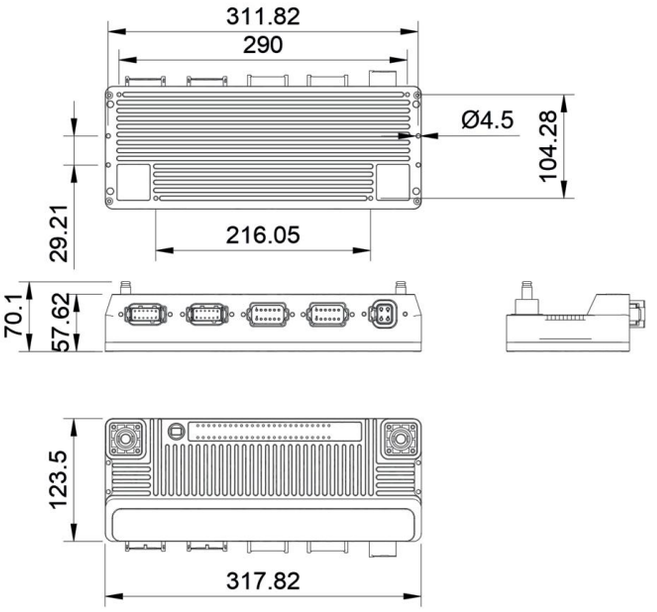
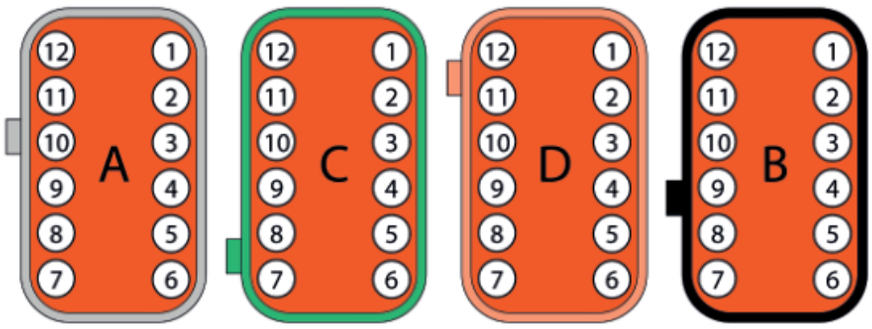


Figure 4 PDM28 dimensions in mm.



PDM15/25/35 Pinout



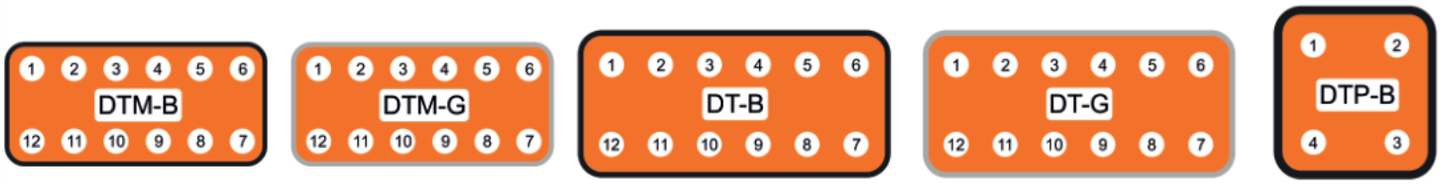
NC = not connected on that model. Pins shown as Input / Output are shared on the PDM35, and are inputs only on the PDM15 and PDM25.

Colour	Pin	PDM15	PDM25	PDM35	Notes
	A1	Input 12	Input 12	Input 12 / Output 31	0-27V, 40K pull down / 200K pull up. PDM35 output: 20A High Side, 1KHz PWM *
	A2	Input 10	Input 10	Input 10 / Output 29	0-27V, 40K pull down / 200K pull up. PDM35 output: 20A High Side, 1KHz PWM *
	A3	Input 8	Input 8	Input 8 / Output 27	0-27V, 40K pull down / 200K pull up. PDM35 output: 20A High Side, 1KHz PWM *
	A4	Input 6	Input 6	Input 6	0-27V, 40K pull down / 200K pull up
	A5	Input 4	Input 4	Input 4	0-27V, 40K pull down / 200K pull up
	A6	Input 2	Input 2	Input 2	0-27V, 40K pull down / 200K pull up
	A7	Input 1	Input 1	Input 1	0-27V, 40K pull down / 200K pull up
	A8	Input 3	Input 3	Input 3	0-27V, 40K pull down / 200K pull up
	A9	Input 5	Input 5	Input 5	0-27V, 40K pull down / 200K pull up
	A10	Input 7	Input 7	Input 7 / Output 26	0-27V, 40K pull down / 200K pull up. PDM35 output: 20A High Side, 1KHz PWM *
	A11	Input 9	Input 9	Input 9 / Output 28	0-27V, 40K pull down / 200K pull up. PDM35 output: 20A High Side, 1KHz PWM *
	A12	Input 11	Input 11	Input 11 / Output 30	0-27V, 40K pull down / 200K pull up. PDM35 output: 20A High Side, 1KHz PWM *
	C1	Input 16	Input 16	Input 16 / Output 35	0-27V, 40K pull down / 200K pull up. PDM35 output: 20A High Side, 1KHz PWM *
	C2	Input 15	Input 15	Input 15 / Output 34	0-27V, 40K pull down / 200K pull up. PDM35 output: 20A High Side, 1KHz PWM *
	C3	PDM Power	PDM Power	PDM Power	4V - 32V
	C4	Output 3	Output 3	Output 3	20A High/Low side, reverse current protected, 1KHz PWM *
	C5	Output 2	Output 2	Output 2	20A High/Low side, reverse current protected, 1KHz PWM *
	C6	Output 1	Output 1	Output 1	20A High/Low side, reverse current protected, 1KHz PWM *
	C7	Ground	Ground	Ground	
	C8	CAN 1 High	CAN 1 High	CAN 1 High	
	C9	CAN 1 Low	CAN 1 Low	CAN 1 Low	
	C10	5V Out	5V Out	5V Out	100mA maximum
	C11	Input 13	Input 13	Input 13 / Output 32	0-27V, 40K pull down / 200K pull up. PDM35 output: 20A High Side, 1KHz PWM *
	C12	Input 14	Input 14	Input 14 / Output 33	0-27V, 40K pull down / 200K pull up. PDM35 output: 20A High Side, 1KHz PWM *
	D1	NC	Output 17	Output 17	20A High Side, 1KHz PWM *
	D2	NC	Output 21	Output 21	20A High Side, 1KHz PWM *
	D3	NC	Output 24	Output 24	20A High Side, 1KHz PWM *

Colour	Pin	PDM15	PDM25	PDM35	Notes
	D4	Output 9	Output 9	Output 9	20A High Side, 1KHz PWM *
	D5	Output 8	Output 8	Output 8	20A High Side, 1KHz PWM *
	D6	Output 7	Output 7	Output 7	20A High Side, 1KHz PWM *
	D7	Output 4	Output 4	Output 4	20A High/Low side, reverse current protected, 1KHz PWM *
	D8	Output 5	Output 5	Output 5	20A High Side, 1KHz PWM *
	D9	Output 6	Output 6	Output 6	20A High Side, 1KHz PWM *
	D10	NC	Output 24	Output 24	20A High Side, 1KHz PWM *
	D11	NC	Output 20	Output 20	20A High Side, 1KHz PWM *
	D12	NC	Output 16	Output 16	20A High Side, 1KHz PWM *
	B1	NC	Output 19	Output 19	20A High Side, 1KHz PWM *
	B2	NC	Output 23	Output 23	20A High Side, 1KHz PWM *
	B3	NC	Output 25	Output 25	20A High Side, 1KHz PWM *
	B4	Output 15	Output 15	Output 15	20A High Side, 1KHz PWM *
	B5	Output 14	Output 14	Output 14	20A High Side, 1KHz PWM *
	B6	Output 13	Output 13	Output 13	20A High Side, 1KHz PWM *
	B7	Output 10	Output 10	Output 10	20A High Side, 1KHz PWM *
	B8	Output 11	Output 11	Output 11	20A High Side, 1KHz PWM *
	B9	Output 12	Output 12	Output 12	20A High Side, 1KHz PWM *
	B10	NC	Output 25	Output 25	20A High Side, 1KHz PWM *
	B11	NC	Output 22	Output 22	20A High Side, 1KHz PWM *
	B12	NC	Output 18	Output 18	20A High Side, 1KHz PWM *

*Limited by maximum current capacity of contact and pin

PDM28 Pinout



Colour	Pin	Description	Notes
	DTM-B-1	PDM Enable	10K Pull Down, 3-24V to activate
	DTM-B-2	LIN	
	DTM-B-3	CAN 2 Low	
	DTM-B-4	CAN 2 High	
	DTM-B-5	CAN 1 Low	
	DTM-B-6	CAN 1 High	
	DTM-B-7	N CAN Low	
	DTM-B-8	N CAN High	
	DTM-B-9	Input 1	10 kΩ to VBat or 10 kΩ to GND
	DTM-B-10	Input 2	10 kΩ to VBat or 10 kΩ to GND; 12 V max on 72–79
	DTM-B-11	Input 3	10 kΩ to VBat or 10 kΩ to GND; 12 V max on 72–79
	DTM-B-12	Input 4	10 kΩ to VBat or 10 kΩ to GND
	DTM-G-1	Input 5	10 kΩ to VBat or 10 kΩ to GND
	DTM-G-2	Input 6	10 kΩ to VBat or 10 kΩ to GND
	DTM-G-3	Input 7	10 kΩ to VBat or 10 kΩ to GND
	DTM-G-4	Input 8	10 kΩ to VBat or 10 kΩ to GND
	DTM-G-5	Input 9	10 kΩ to VBat or 10 kΩ to GND
	DTM-G-6	Input 10	10 kΩ to VBat or 10 kΩ to GND
	DTM-G-7	Input 11	10 kΩ to VBat or 10 kΩ to GND
	DTM-G-8	Input 12	10 kΩ to VBat or 10 kΩ to GND
	DTM-G-9	Input 13	70–71: 10 kΩ to VBat. 72–79: 10 kΩ to 5 V. Down: 10 kΩ to GND.
	DTM-G-10	Input 14	70–71: 10 kΩ to VBat. 72–79: 10 kΩ to 5 V. Down: 10 kΩ to GND.
	DTM-G-11	Input 15	70–71: 10 kΩ to VBat. 72–79: 1 kΩ to 5 V. Down: 10 kΩ to GND.
	DTM-G-12	Input 16	70–71: 10 kΩ to VBat. 72–79: 1 kΩ to 5 V. Down: 10 kΩ to GND.
	DT-B-1	Output 1	20A High/Low side, 1KHz PWM *
	DT-B-2	Output 2	20A High/Low side, 1KHz PWM *
	DT-B-3	Output 3	20A High/Low side, 1KHz PWM *
	DT-B-4	Output 4	20A High/Low side, reverse current blocking, 1KHz PWM *, Max voltage 36V
	DT-B-5	Output 5	20A High/Low side, 1KHz PWM *
	DT-B-6	Output 6	20A High/Low side, 1KHz PWM *
	DT-B-7	Output 7	20A High/Low side, 1KHz PWM *
	DT-B-8	Output 8	20A High/Low side, reverse current blocking, 1KHz PWM *, Max voltage 36V
	DT-B-9	Output 9	20A High/Low side, 1KHz PWM *
	DT-B-10	Output 10	20A High/Low side, 1KHz PWM *
	DT-B-11	Output 11	20A High/Low side, 1KHz PWM *
	DT-B-12	Output 12	20A High/Low side, 1KHz PWM *
	DT-G-1	Output 13	20A High/Low side, 1KHz PWM *
	DT-G-2	Output 14	20A High/Low side, 1KHz PWM *
	DT-G-3	Output 15	20A High/Low side, 1KHz PWM *

Colour	Pin	Description	Notes
	DT-G-4	Output 16	20A High/Low side, reverse current blocking, 1KHz PWM *, Max voltage 36V
	DT-G-5	Output 17	20A High/Low side, 1KHz PWM *
	DT-G-6	Output 18	20A High/Low side, 1KHz PWM *
	DT-G-7	Output 19	20A High/Low side, 1KHz PWM *
	DT-G-8	Output 20	20A High/Low side, reverse current blocking, 1KHz PWM *, Max voltage 36V
	DT-G-9	Output 21	20A High/Low side, reverse current blocking, 1KHz PWM *, Max voltage 36V
	DT-G-10	Output 22	20A High/Low side, 1KHz PWM *
	DT-G-11	Output 23	20A High/Low side, 1KHz PWM *
	DT-G-12	Output 24	20A High/Low side, reverse current blocking, 1KHz PWM *, Max voltage 36V
	DTP-B-1	Output 25	30A High/Low side, reverse current blocking, 1KHz PWM *, Max voltage 36V
	DTP-B-2	Output 26	30A High/Low side, reverse current blocking, 1KHz PWM *, Max voltage 36V
	DTP-B-3	Output 27	30A High/Low side, reverse current blocking, 1KHz PWM *, Max voltage 36V
	DTP-B-4	Output 28	30A High/Low side, reverse current blocking, 1KHz PWM *, Max voltage 36V

*Limited by maximum current capacity of contact and pin

PDM15/25/35 Device Power

Battery Positive: The Hardwire Electronics PDM Powers the connected devices via the M8 stud protruding from the top of the enclosure. The PDM current can peak at upwards of 180 Amps, therefore an appropriate gauge copper wire must be used to attach to the M8 stud to avoid excess heat and voltage drop. It is recommended to use at least 2 AWG wire with the supplied ring terminal. Care must be taken when tightening the retaining nut on the stud. The recommended torque to use is 8Nm – exceeding this could damage the internal circuit board.
Battery Negative: Under normal operating conditions, the current flow through the ground pin of the PDM is low. However, when a load dump occurs, the current flow can be much higher. Therefore, it is recommended that at least 16 AWG wire be used.
5V Output: One pin (C10) on the PDM15/25/35 is a dedicated 5V output. The 5V output is over current and short circuit protected. No more than 0.1 Amp should be drawn from the 5V output to avoid excessive heating of the internal circuitry. The 5V output can be used to connect to analogue voltage sensors such as thermal sensors.
Ignition Input: To power on the PDM, the Ignition Input must be switched to the battery positive voltage. This input can be wired directly to the accessory input of an ignition key barrel, or alternatively to another dedicated switch. The ignition input may be connected directly to the battery positive voltage - however, this is not advised as the battery will slowly discharge over time when the vehicle is not in use. A battery isolator may be used to combat this problem.

PDM28 Device Power

The PDM28 is powered by two Amphenol RADLOK connectors- Red (Battery Positive 12-24V) and Black (Battery negative)- which make connecting/disconnecting the PDM an easy process with no tools required. The RADLOK connectors are crimped using a standard battery cable crimper. It is recommended to use at least 2AWG/ 35mm2 wire for the Positive and Negative PDM supplies. It is preferred to ground the PDM28 to the vehicle chassis close to where the PDM28 is located. A battery isolator should be installed between the vehicle battery and the PDM28. The device is enabled (powered up) when >3V is applied to the PDM Enable Pin. This pin should be a switched live, typically coming from a key barrel or ignition switch. A 10K pull down resistor is used on the PDM Enable Pin. When the PDM is not enabled, the PDM28 current consumption will be approximately 6mA.

PDM28 Status LEDs

The PDM28 has 50x RGB Status LEDs.
Power: Green when the PDM has the correct supply voltage. Blue if the voltage is too low (<7V). Red if the voltage is too high (>28V)
Temperature: Green when the PDM temperature is below 70degC, red otherwise. **CAN:** Green if the CAN TEC and REC Values are below 100, red otherwise. **Output:** Green if either the HS or LS output is On. Red if either the HS or LS output has tripped due to Over
Current. Blue if either the HS or LS output has tripped due to Under Current.

Input: Green if the Input is triggered.

LED Type			
Power	Low Voltage <7V	Ok	Over Voltage>28V
Temperature	Not used	Ok	Too High > 70degC
CAN	Not used	Ok	Tx/Rx errors > 100
LIN	Not used	Ok	Error Detected
Input	Not used	Triggered	Not used
Output	Tripped Under Current	Ok	Tripped Over Current



Inputs

All models have 16 analogue inputs, which can also measure frequency up to 300 Hz. The great flexibility of the PDM inputs allows analogue, digital, switched 12V and switched ground inputs to be measured with ease.
PDM15/25/35: Each input can measure analogue voltages between 0-28V with a sample resolution of 12-bit or 0.007V. Each input has a fixed 40 kΩ pull-down and 200 kΩ pull-up resistor. On the PDM35, inputs 7 to 16 share their pins with outputs 26 to 35. See the pin table.
PDM28: The inputs have 16-bit resolution. Each input has selectable bias resistors for switched-voltage and switched-ground signals. See the pin table and Input Resistor section for the pull-up supply and resistor value. Observe the voltage limit for the selected input and hardware revision.

Outputs

PDM15/25/35: The PDM15, PDM25 and PDM35 each have four High/Low side outputs (Outputs 1 to 4), meaning that they can output the positive battery voltage to source current OR put the output to ground to sink current. Internal circuitry stops both the high and low side being activated at the same time on any given output. These outputs are also reverse current protected, meaning that when the voltage present on the output is greater than the battery voltage, the output does not allow current to flow back into the output. This is useful when driving wiper motors with low/high speed windings. The remaining outputs are high side only: 11 on the PDM15, 21 on the PDM25 and 31 on the PDM35. Each output has current sensing, thermal overload protection, static discharge protection, short circuit protection and over/under voltage protection. The outputs are able to switch on and off incredibly quickly, which reduces heat build up when using the output in PWM mode. The outputs have a continuous current rating of 20Amps (limited by the current capacity of the contact and pin to 13A at 125degC). Multiple outputs of the same type can be joined together to increase the current capacity.
PDM28: The PDM28 has 28 Outputs. All 28 outputs are High and Low sided. Ten outputs are reverse current protected, meaning that when the voltage present on the output is greater than the battery voltage, the output does not allow current to flow back into the output. This is useful when driving wiper motors with low/high speed windings. Each output has current sensing, thermal overload protection, static discharge protection, short circuit protection and over/under voltage protection. The outputs are able to switch on and off incredibly quickly, which reduces heat build up when using the output in PWM mode. Four of the outputs have a continuous current rating of 30Amps, (limited by the current capacity of the contact and pin to 25A at 125degC), and should be used for powering larger devices such as radiator fans and large fuel pumps. The remaining outputs have a continuous current rating of 20Amps (limited by the current capacity of the contact and pin to 13A at 125degC). Multiple outputs of the same type can be joined together to increase the current capacity.

CAN Bus Wiring

The CAN bus is designed to be extremely robust and fault tolerant, however to ensure the best reliability, it is necessary to follow best practices when wiring up a CAN bus network. The physical CAN bus is constructed with two wires, CAN high and CAN low. The wires should be twisted around each other at approximately 1 twist per 1”/2 cm of length, this is to help mitigate electrical interference on the CAN bus. Each end of the CAN bus **MUST** have a 120Ω termination resistor installed between the CAN high and CAN low lines. The two termination resistors are effectively wired in parallel, resulting in a total of 60Ω of resistance between the CAN high and CAN low lines. It is important to verify that there is approximately 60
Ω of resistance between the CAN high and CAN low lines. Some CAN bus devices will have a CAN

termination resistor installed internally. Each Hardwire PDM has a software selectable CAN termination resistor on each CAN bus port (CAN bus 1 on the PDM15/25/35; CAN bus 1 and CAN bus 2 on the PDM28). If the PDM CAN termination resistor is used, then the PDM must be placed at the end of the CAN bus.

Care must be taken to ensure that the physical wiring of the CAN bus meets the following requirements -

- The maximum bus length must not exceed 25m
- The CAN bus must be terminated at each end with a 120Ω resistor. Each PDM CAN bus port has a software selectable CAN termination resistor.
- CAN bus 'stubs' must be no longer than 30cm in length
- Twisted pair wire must be used for the CAN bus with a minimum of one twist per 3cm.

Configuration Software

The Hardwire Electronics Configuration Software has been designed with ease of use in mind. There are seven main sections to the software - Connection, Configuration, Monitor, Logging, Update, Loom View and Pin Manager. Each section can be accessed via the tabs at the left of the screen in the sidebar. Other important information such as USB/CAN connection status can also be found in the sidebar for easy viewing. Use the question mark beside a feature to open its instructions. The help panel can also be searched. The bottom console displays connection messages and errors. The software can be found on the supplied USB drive, or on the Hardwire Electronics Website in the downloads section. https://www.hardwire-electronics.co.uk/downloads

Connection

To prepare a configuration without connecting a PDM, open the Connection tab. Under Work offline, select the PDM model from the drop-down list.

Offline Mode

In Offline Mode, select the correct PDM model before preparing an offline configuration. The model determines the available pins, buses and functions. Check the connected model and firmware before sending the configuration.

If the PDM configuration has been locked with a password, you will be prompted to enter the password to gain access.

USB

Connect the PDM to the PC with the supplied USB cable. On the Connection tab, select USB under Connect a device. Select the PDM serial port and click Connect. The sidebar shows the connection status and the transmit and receive counts. Select USB autoconnect if you want the software to connect automatically when the PDM is available.

USB to CAN

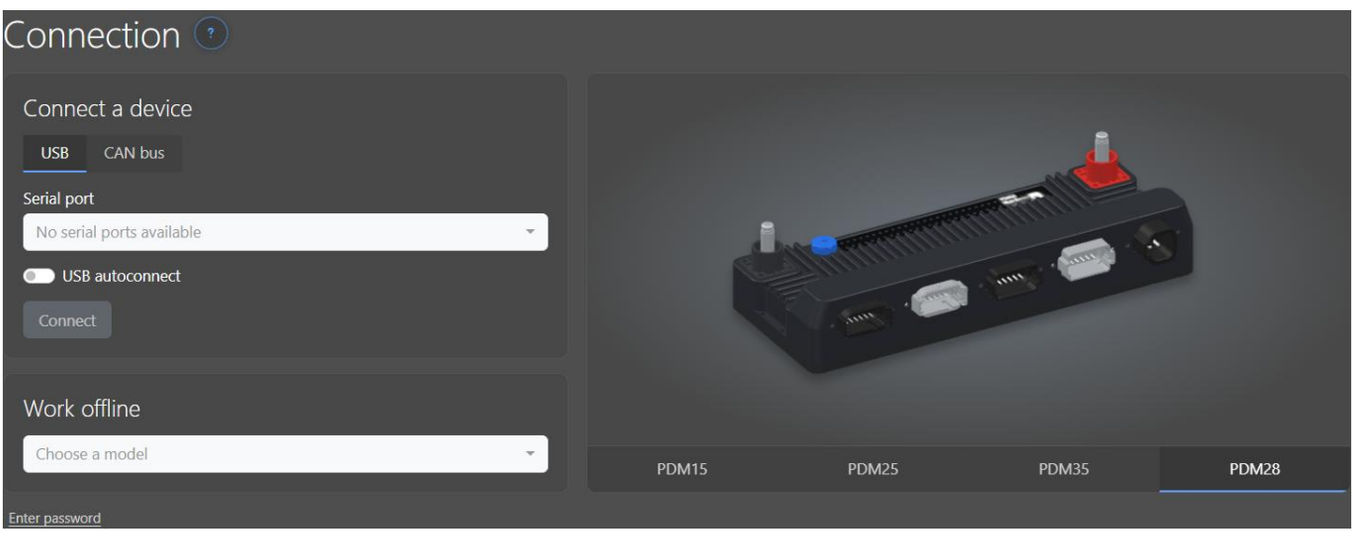


Figure 5 Connection screen.

To connect through CAN, install the driver for the supported USB-to-CAN adapter and connect the adapter to the PDM CAN bus. On the Connection tab, select CAN bus, choose the adapter and set the CAN speed to match the PDM. Check the bus termination. Open Detected PDMs to scan, select the PDM and click Connect. The connection status is shown on the screen.

Software and Firmware Versions

Normal operation requires matching Configurator and PDM firmware versions. A device in firmware update mode can be detected separately, but cannot be configured until the correct firmware is installed.

Configuration - Configuration Files

To configure the PDM, connect to the PDM via USB or CAN and navigate to the Configuration Tab.



Figure 6 Configuration file toolbar.

Save Configuration

Pressing Save As at the top of the screen will prompt the user to locate a directory on the PC in which to save the configuration file. Configuration files should have a .HWPDM file extension. Subsequent presses of the Save button will overwrite your currently saved file. The file indicator shows the configuration name and whether there are unsaved changes

Saving a file does not send the configuration to the PDM. Use Send to Device to apply the configuration to the connected unit.

Load Configuration

Likewise, pressing Open will prompt the user to navigate to a configuration file which can then be loaded into the configuration software.

Send Configuration

Press the Send button to send the current configuration to the PDM, and press Retrieve to retrieve the configuration currently on the PDM.

Retrieve Configuration

When retrieving the configuration from the PDM, if any changes are detected between the current configuration on the PC and the configuration on the PDM, then the user will be given a choice to either maintain the current configuration on the PC, or alternatively to overwrite the configuration on the PC and load in the configuration from the PDM onto the PC.

Undo

The Undo history allows configuration changes to be reversed. Click Ctrl + z to undo

Configuration Synchronisation

The synchronisation warning indicates that the configuration on the PC differs from the configuration on the PDM. Retrieve from Device reads the PDM configuration. Review the choice to keep or replace any unsaved PC changes.

Opening an Older Configuration

The Configurator Software converts supported older configuration files when they are opened. Save the converted configuration under a new name, review the settings and check Configuration Warnings before sending it to the PDM.

General

In the General tab, the user can configure the general settings for the PDM.

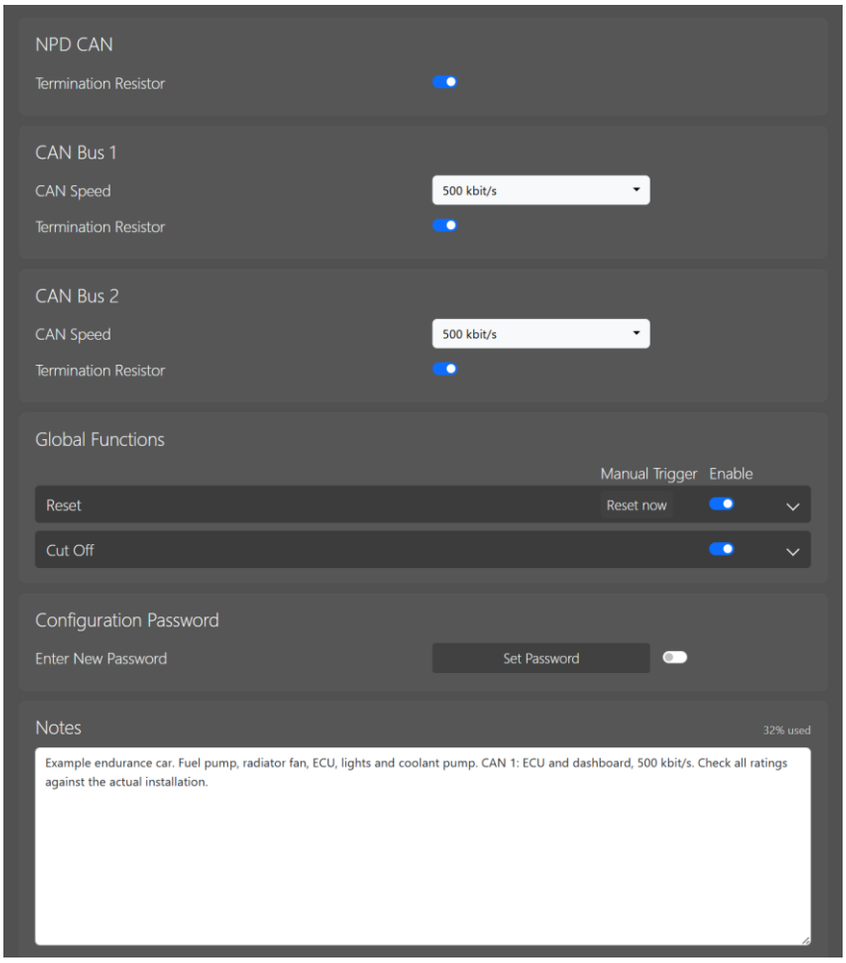


Figure 7 General settings.

CAN Bus Settings

At the top of the tab, the user can select the CAN termination resistors and CAN speed for each CAN bus on the PDM. The PDM15/25/35 has one CAN bus (CAN bus 1). On the PDM28, CAN bus 1 and 2 are for the user to use. These buses can interface with external CAN bus devices such as Keypads, Digital Dashboards and ECUs.

On the PDM28, the N-CAN bus is currently unsupported. Use CAN Bus 1 or CAN Bus 2 to connect external CAN devices.

Reset Function

The Reset function is used to reset tripped outputs, the global cut-off state and the IMU impact and rollover triggers. More information is given on configuring functions later in the instruction manual. When the Reset function changes from False to True, each tripped output on the PDM will reset, so that normal operation can resume. Triggering the reset function does not affect already operational outputs.

Reset Now

The Reset now button performs a Global Reset on the connected PDM. This clears tripped outputs, the global cut-off state and the IMU impact and rollover triggers. It can be used whether or not the Reset function is enabled.

Cut Off Function

The Cut Off function is used to turn off every output on the PDM. This is a useful safety feature, to disable the entire vehicle electrical system if a problem occurs. When the Cut Off function changes from False to True, each output is immediately turned off. The outputs will remain turned off until the PDM power is turned off and back on again, or until the Reset function is triggered.

When the Cut Off condition becomes False, the outputs remain cut off until a Global Reset is performed.

Password

Once you have completed your PDM setup, you can lock the configuration with a password. To add a password, click enter new password and check the box to enable it. Each time you connect to the PDM, you will be prompted to enter the password again.

If the password is lost, contact Hardwire Electronics for recovery instructions.

Notes

The Notes field stores information on the PDM as part of the configuration. This is a good place to add installation specific notes.

The note is saved with the HWPDM file and sent to the PDM when the configuration is sent.

Inputs

Input Label

Each input can be given a label of up to 19 characters. Use a name which describes the connected switch or sensor.

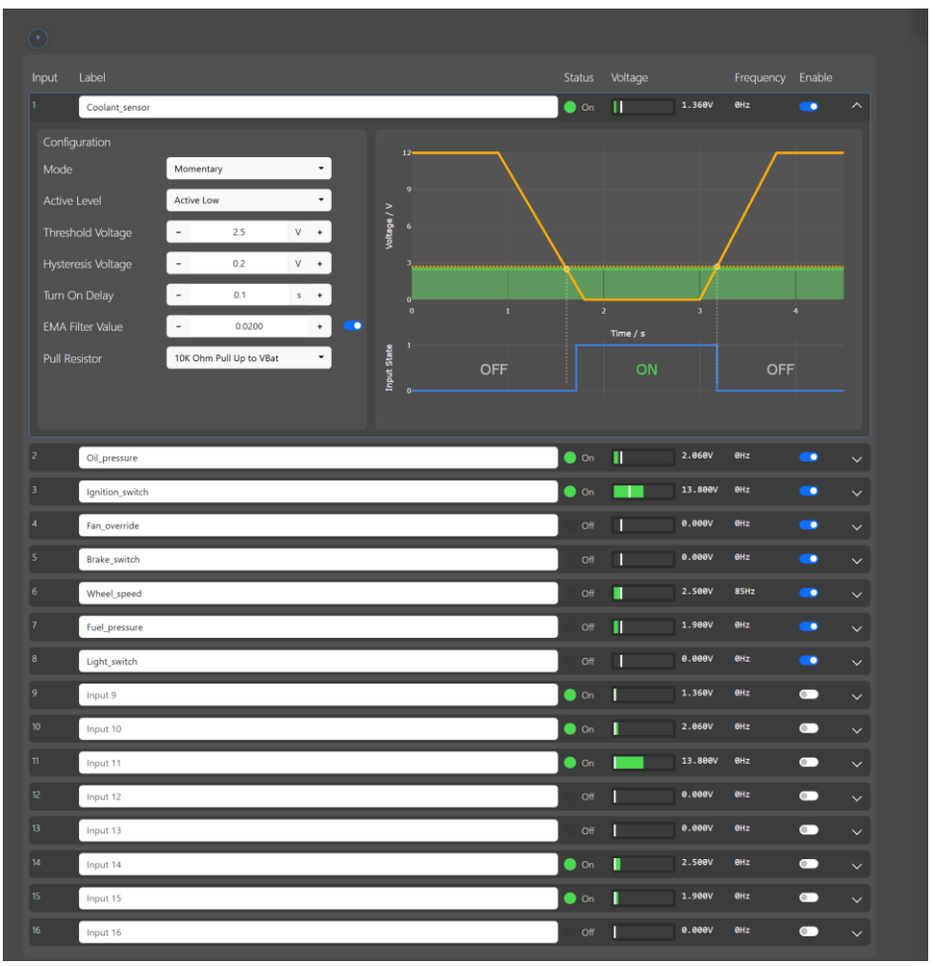


Figure 8 Input configuration.

Voltage/Frequency

The PDM inputs are able to measure both the voltage and frequency of the input signal. The inputs have a voltage measurement range of 0-28V, and a frequency measurement range of 0-300Hz. The frequency measurements are performed automatically, and will work with any periodic waveform between 0 and 300Hz, irrespective of the peak to peak voltage or DC offset value.

Mode

The inputs may be set to Momentary or Latched Mode. This feature makes it possible to use a simple push button switch to activate components such as fans and switches with one button press to turn the component on and off. In Momentary Mode, when the input voltage passes the threshold voltage, the input will turn on. In Latched Mode, when the input voltage passes the threshold voltage, the input will toggle its state from Off to On, or from On to Off.

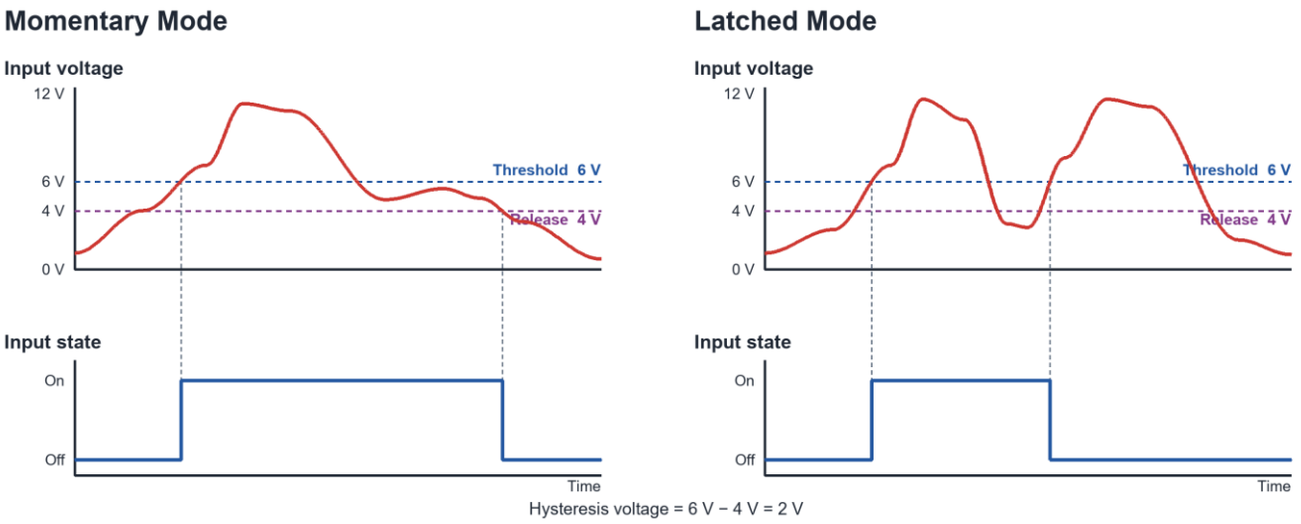


Figure 9 Momentary and latched input modes.

Active Level

Each input can be programmed as Active-High or Active-Low. Active-High causes the input to turn on when the voltage rises above the threshold. Active-Low causes the input to turn on when the voltage falls below the threshold.

Threshold Voltage

Each input operates as an analogue input. Inputs are active whenever the voltage present on the input passes the Threshold Voltage.

Hysteresis Voltage

For an Active High input, the input becomes active above the threshold and releases below the threshold minus hysteresis. For an Active Low input, it becomes active below the threshold and releases above the threshold plus hysteresis.

Turn On Delay

A turn on delay can be set, so that the input is triggered after the delay period has passed.

EMA Filter

An EMA (Exponential Moving Average) filter can be applied to the input voltage. This can help to smooth out noisy input voltages. A lower EMA value produces more smoothing and a slower response. A higher value follows changes more quickly.

Input Resistor

Select Pull Up for a switch connected to ground, or Pull Down for a switch connected to supply. On the PDM15/25/35, Pull Down has 40 kΩ resistor to ground and Pull Up has a 200 kΩ resistor. On the PDM28, a pull-down connects the input to ground through 10 kΩ. On PDM28 model versions 70 and 71, each pull-up is 10 kΩ to battery voltage. On versions 72–79, inputs 1–12 use 10 kΩ to battery voltage, inputs 13–14 use 10 kΩ to 5 V and inputs 15–16 use 1 kΩ to 5 V. Select the correct hardware revision in the Configurator before checking a pin.

Pull Up Resistor Example

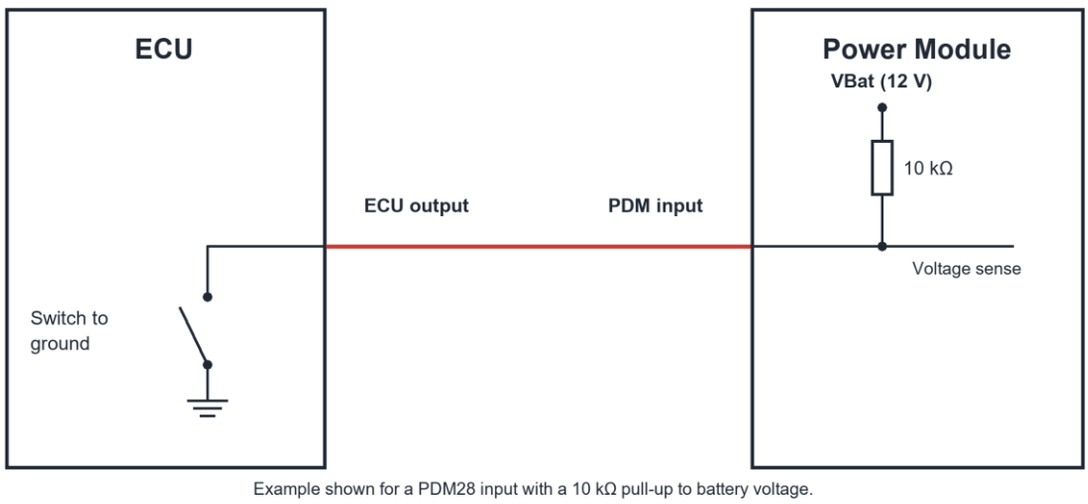


Figure 10 Pull-up resistor circuit.

Outputs

Output Label

Each high-side or low-side output can be given a label of up to 19 characters. The pin indicator identifies its physical connector pin.

This section will detail how to configure an output for the PDM. Navigate to the Output Tab to configure the outputs. Each PDM28 output, and Outputs 1 to 4 on the PDM15/25/35, is High/Low sided. This means that both the High and Low side need to be configured independently. HS denotes High side, LS denotes Low side. If both the high and low side on a single output are instructed to turn on, only the high side will turn on. The low side will remain off until the high side is turned off. The remaining PDM15/25/35 outputs are high side only, so only their high side settings are used.

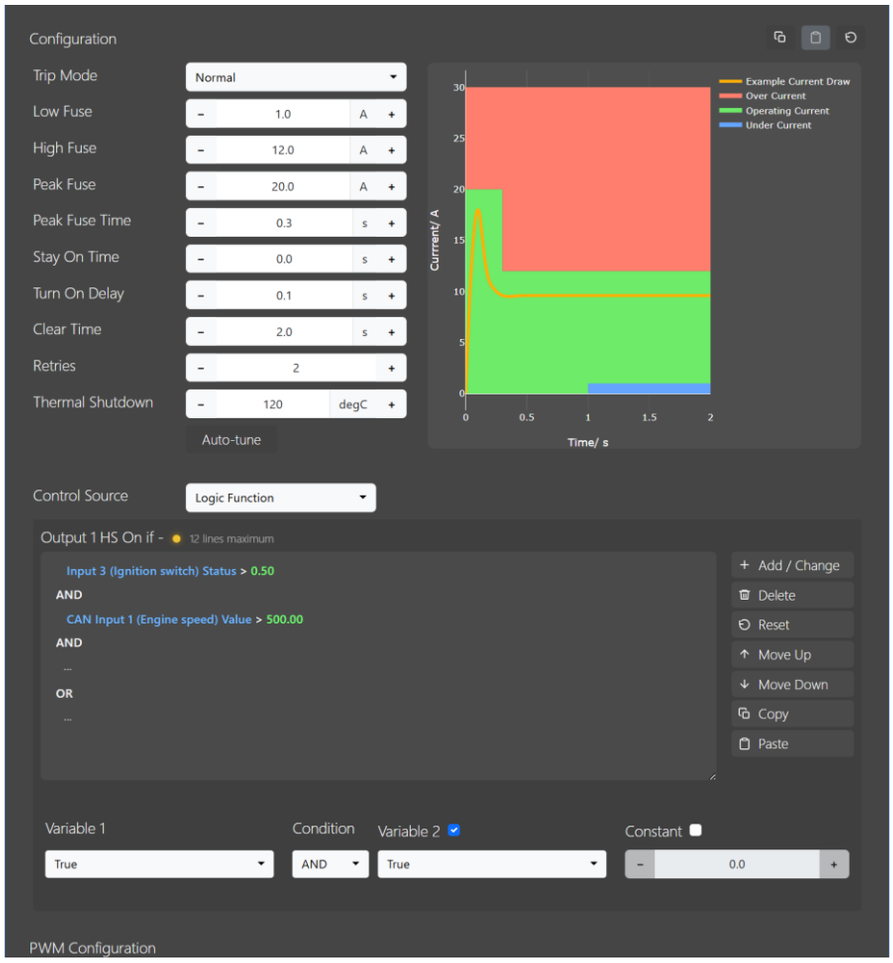


Figure 11 Output configuration.

Output Status

Off = 0 **On = 1** **Tripped Over Current = 2**
Tripped Under Current = 3 **Tripped Under Voltage = 4** **Tripped Over Temperature = 5**

Output Current

How much current is flowing through the output.

Output Voltage

The voltage on the output pin.

Output Load

The output load is the measured output current divided by the High Fuse setting, expressed as a percentage. For example, 6 A with a 12 A High Fuse gives a load of 50%.

Output Test

The user can manually test each output. The output must be enabled and configured in order to test it.

Output Enable

Enables or Disables the output. Pressing 'Expand' will open up the full configuration settings for that output.

Trip Mode

Instant Trip: This mode will instantly turn the output off if the current flowing through the output goes over the current threshold. This setting is good for devices that need instant overcurrent protection.

Normal Trip: This mode acts like a conventional fuse and allows small current spikes over the current threshold without tripping. This setting is useful for devices that draw current in short bursts, such as ignition coils.

Low Fuse
The low fuse setting will trip if the current falls below the threshold for over 1 second continuously. If this functionality is not needed then the value should be set to 0.

High Fuse
The high fuse setting comes into effect after the peak fuse time has elapsed, and sets the current limit for an output working at steady state. If the current exceeds this value, then the output will trip.

Peak Fuse & Peak Fuse Time
The peak fuse setting determines the maximum current that an output can provide before it trips in the initial turn-on phase of the output. The length of the turn-on phase is determined by the peak fuse time setting. The peak fuse setting is used to allow large inrush currents without tripping the output. When turning on loads such as radiator fans or fuel pumps, the initial current draw from the device can often double or triple the steady state current draw.

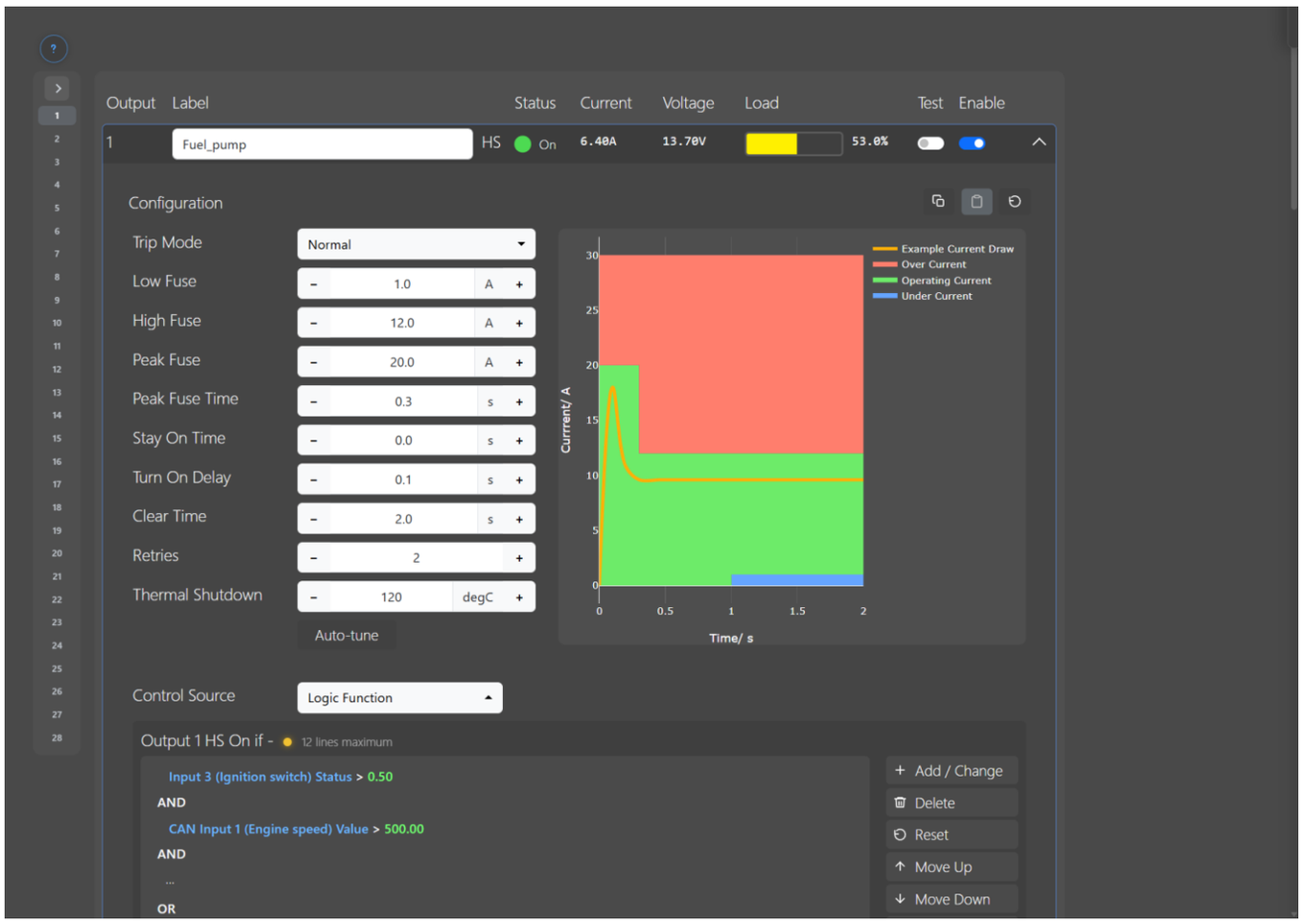


Figure 12 Output protection settings.

Thermal Shutdown
Thermal Shutdown sets the output temperature at which a thermal trip occurs. A thermal trip reports status 5 and remains latched. Cooling the output or waiting for an automatic retry does not clear this trip; a reset is required.

Auto-tune
Auto-tune operates the actual load and measures its current for ten seconds before proposing protection settings. Check the load and wiring before starting. Review the proposed settings before applying them; the measured current does not establish the safe rating of the wiring or connector. The battery voltage must be sufficient for the autotune to start.

Stay On Time
After an output is commanded to turn off, the output will remain switched on for the stay on time, before turning off.

Turn On Delay
This will delay the turn on of the output once the logic function evaluates to True.

Clear Time
If an output trips, the output will not be retried until the 'clear time' has passed. This is to allow time for an output device to perhaps cool down or reset itself before being turned back on.

Retries
Retries sets the number of automatic attempts after a recoverable trip. Zero disables automatic retries, values 1 to 9 provide a limited number of attempts, and 10 is displayed as Infinite. Select the retry behaviour to suit the load and fault response.

Functions
Output Functions are used to control if an output should be On or Off. The PDM evaluates the logic function; if the function evaluates to True then the output will turn On, if the output evaluates to False then the output will turn Off. Examples of how to use logic functions will be given later in the instruction manual.

Follow Output
To command one output from another, set Control Source to Follow Output and select the source output. The following output retains its own protection settings.

Copy and Paste
Copy, paste and reset apply to the selected high-side or low-side configuration. After pasting, check the fuse values, output function, PWM settings and references to other channels before sending.

PWM
All outputs on the PDM are PWM (Pulse Width Modulation) capable. High speed switching circuitry allows for PWM operation at up to 1KHz with minimal heat dissipation. PWM is used to vary the average voltage present on the output. This is useful for varying the speed or intensity of motors or lights. PWM works by switching the output on and off at high speed, with the ratio of ON to OFF time (duty cycle) dictating the average voltage on the output. A lower duty cycle means that the output will be OFF for longer than it is ON, resulting in a low average voltage on the output. Conversely, a higher duty cycle means that the output will be ON for longer than it is OFF, resulting in a higher average output voltage. The duty cycle can vary between 0% (output always off) to 100% (output always on).

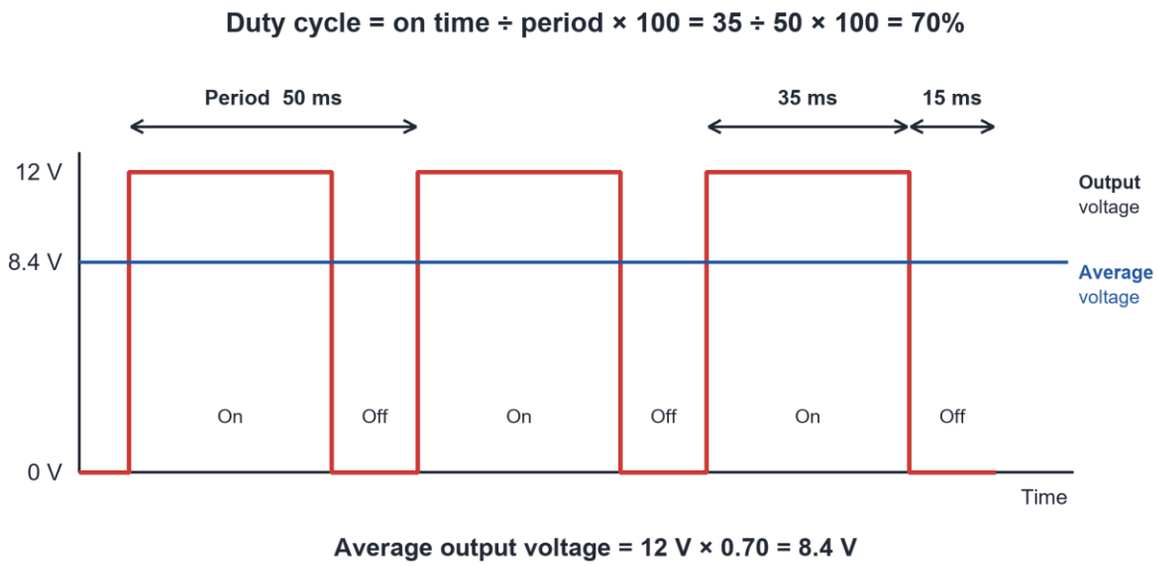


Figure 13 PWM duty cycle and average voltage.

Frequency
The PWM frequency dictates the speed at which the outputs are turned on and off. A higher frequency can result in smoother operation of the device which is being driven. Frequency is equal to 1/Period.

Mapping

To vary the PWM duty cycle, it is necessary to 'map' the required value of duty cycle to a PDM variable. As the PDM variable changes, the PWM duty cycle changes. This allows the user to vary the PWM duty cycle on the output as desired. To configure the PWM Mapping, first select a PDM variable from the drop down list. For each value of duty cycle, a value of the PDM variable can be chosen. Eleven variable values must be chosen for the different values of PWM duty cycle. The duty cycle value is linearly interpolated between the values given. The variable value can either increase or decrease with duty cycle. In the example, the mapped variable is coolant temperature. The fan reaches 50% duty at 87 °C and 100% at 94 °C.

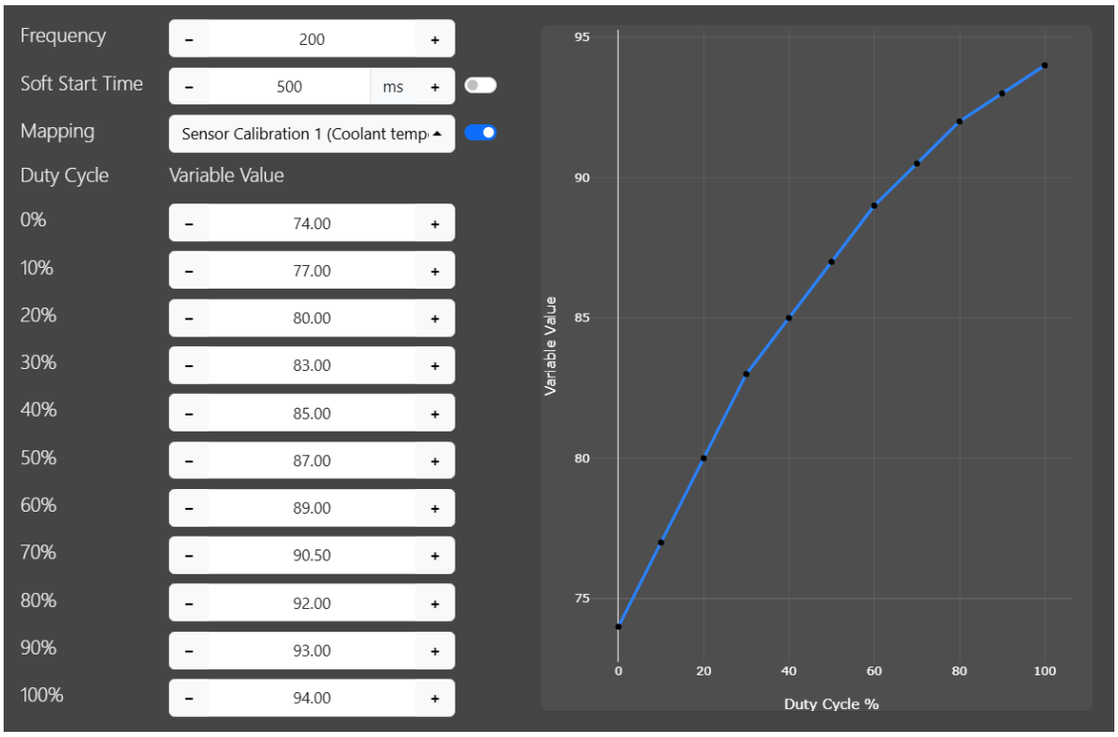


Figure 14 Radiator fan PWM mapping.

Soft Start Time

Soft Start is a feature used to slowly turn on a device, by ramping the duty cycle on the output from 0% to 100% over a specified period. This results in the average voltage on the output increasing from 0% to 100%, slowly turning the device on. This feature is useful for starting devices such as radiator fans, which inherently have a large mechanical inertia. This large mechanical inertia means that the fan draws a large current initially to accelerate the fan blade, before the current drops back down to a base level when the fan reaches full speed.

Disable Diagnostic Pulse

While an output is using PWM, the PDM turns it fully on for 1 ms every 100 ms to measure its current more accurately, then shortens later pulses to maintain the average duty cycle. This can affect devices which use the individual pulse widths as a control signal.

The Disable Diagnostic Pulse option is available for PWM mapping or soft start when the High Fuse and Peak Fuse settings are both 1 A or less. Leave the diagnostic pulse enabled for normal loads.

Timers

The PDM has up to 30 individual Timers, which can be used in output functions to achieve advanced functionality such as Flashing indicator lights, Intermittent wiper motors, and more. To configure a Timer, go to the Timer Tab and click 'Add.' This will add a Timer into the current configuration. The Timer can then be given a label and be enabled.

Pulse Train

In Pulse Train mode, the timer has an ON and an OFF Time, given in milliseconds. When the timer is enabled, it will run automatically in the background, switching state from ON to OFF at the given intervals. It is up to the user to use the state of the timer in the output function.

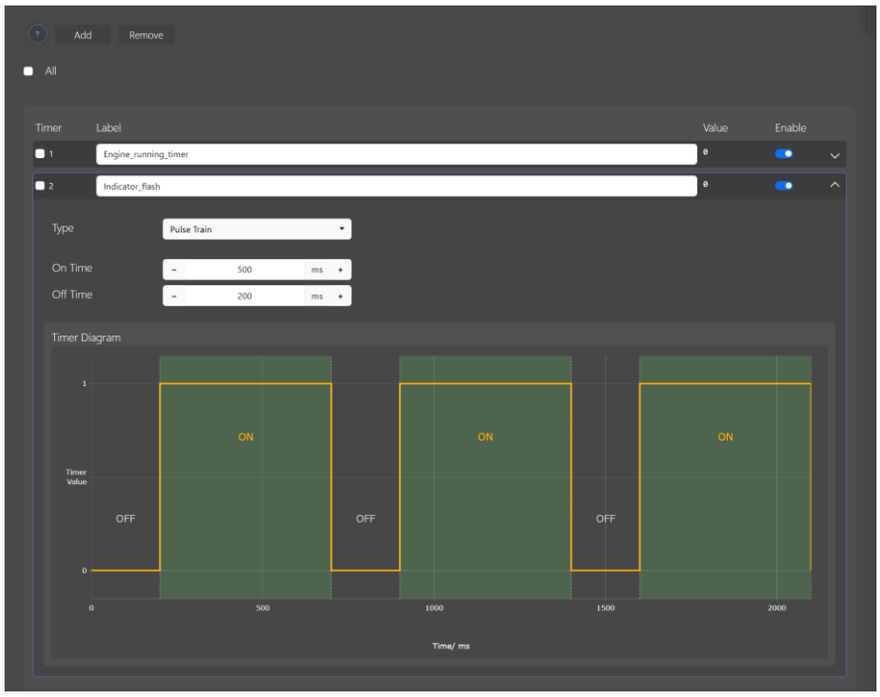


Figure 15 Pulse Train.

Duration

In Duration mode, the timer has a start and reset condition. When the start condition is met, it will begin counting. The value of the timer will be the milliseconds elapsed since the start. The Timer will continue running until it reaches the duration value which is set, or until the reset condition is met. If Reset on End is selected, the timer will go back to 0 once the duration has elapsed. If the Reset on End is not selected, the timer value will stay at the duration value until the reset condition is met.

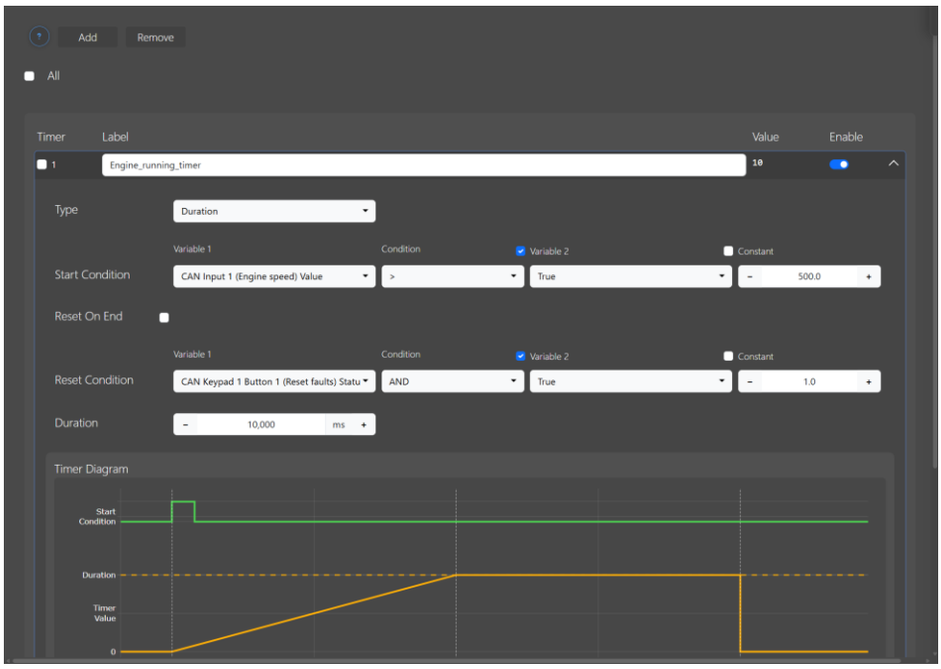


Figure 16 Duration.

Timer Range

Timer settings are entered in milliseconds. The maximum value is 2147483647 ms, approximately 24.9 days.

Counters

The PDM has up to 30 individual Counters. Counters are variables which can be incremented or decremented, and are triggered by logic expressions, such as when an input status equals true. Counters can be used for more advanced functionality, such as incrementing a number every time a button is pressed, and then using the value of the counter to produce different PDM output behaviors. To configure a counter, go to the Counter Tab and click 'Add.' This will add a counter into the current configuration. The counter can then be given a label and enabled. Click 'Expand' to expand the configuration for the counter. Counters are always given a Start Value. This is the value which the Counter starts from when the device is powered up, or when the counter is reset. The Loop Around Value is the maximum or minimum value that the counter will count to before looping back to the Start value. The Counter can be Incremented, Decremented, or Reset, based on three logic expressions. When the logic expression evaluates to 'True,' the counter will Increment, Decrement, or Reset.

In the example below, the counter selects a mode using two momentary CAN keypad buttons. Set Start Value to 0 and enable Loop Around Value at 4. Set Increment to Step up Status Equals 1 and Decrement to Step down Status Equals 1. Press Step up to advance the counter, or Step down to move back one step. On the fourth step up from 0, the counter returns to 0.



Figure 17 Counters.

Maths Channels

The PDM has up to 30 Maths Channels. Maths Channels are an advanced feature which allow the user to create their own mathematical expressions which are evaluated by the PDM. The value of the Maths Channels can then be used as needed.

In Configuration, select Maths Channels and click Add. Enter a label, enable the channel and expand its row to edit the equation. Each channel supports up to 50 equation lines. The completed result is available as a variable elsewhere in the configuration.

The Mathematical Equation is built up by adding lines to the equation. To add a line, select a line in the editor window. This will select that line to edit. The line can be configured by choosing the required mathematical operation from the drop down menu below. Click 'Add/Change' to add the line into the equation.

Equation lines can also be deleted, or the equation can be reset to default by clicking the 'Delete' and 'Reset' button respectively. There are a number of mathematical operations that can be selected from the Operation drop down list,

Mathematical Operators	
Constant/Variable	A constant is just a number which is fixed, and can be used in the mathematical equation. A variable is a number which changes, and can be selected from the list of variables inside the PDM i.e Output 1 Current, Input 1 Voltage
Addition	The Addition operator adds the two operands together. Each operand can be either a constant number or a PDM variable
Subtraction	The Subtraction operator subtracts the second operand from the first. Each operand can be either a constant number or a PDM variable
Division	The Division operator divides the equation section by the divisor. The divisor can be either a constant number or a PDM variable.
Modulus	The Modulus operator performs the modulo operation on the equation with the divisor. This gives the remainder when the equation section is divided by the divisor. The divisor can be either a constant number or a PDM variable.
Bitwise Operators	
AND	Returns the bitwise AND operation of the two operands.
OR	Returns the bitwise OR operation of the two operands.
XOR (Exclusive OR)	Returns the bitwise XOR operation of the two operands.
Left Shift <<	Shifts the equation section to the left by between 0-32
Right Shift>>	Shifts the equation section to the right by between 0-32
Function Operations	
Choose	If the condition is true, the Choose function will return the first value. If the condition is false, the Choose function will return the second value.
Min	Returns the smallest of the two values
Max	Returns the largest of the two values.
Sin	Returns the sine of the value (Radians)
Cos	Returns the cosine of the value (Radians)
Tan	Returns the tangent of the value (Radians)
Floor	Returns the nearest whole number below the value
Ceil	Returns the nearest whole number above the value
Abs	Keeps the value of the number, but always returns it as a positive value
Asin	Returns the arc sine of the value (Radians)
Acos	Returns the arc cosine of the value (Radians)
Ln	Returns the natural logarithm of the value
Log	Returns the Log 10 of the value
Pow	Returns the base value to the power of the exponent value
Sqrt	Returns the square root of the value

In the example below, a Maths Channel converts the calibrated coolant temperature from degrees C to Fahrenheit.

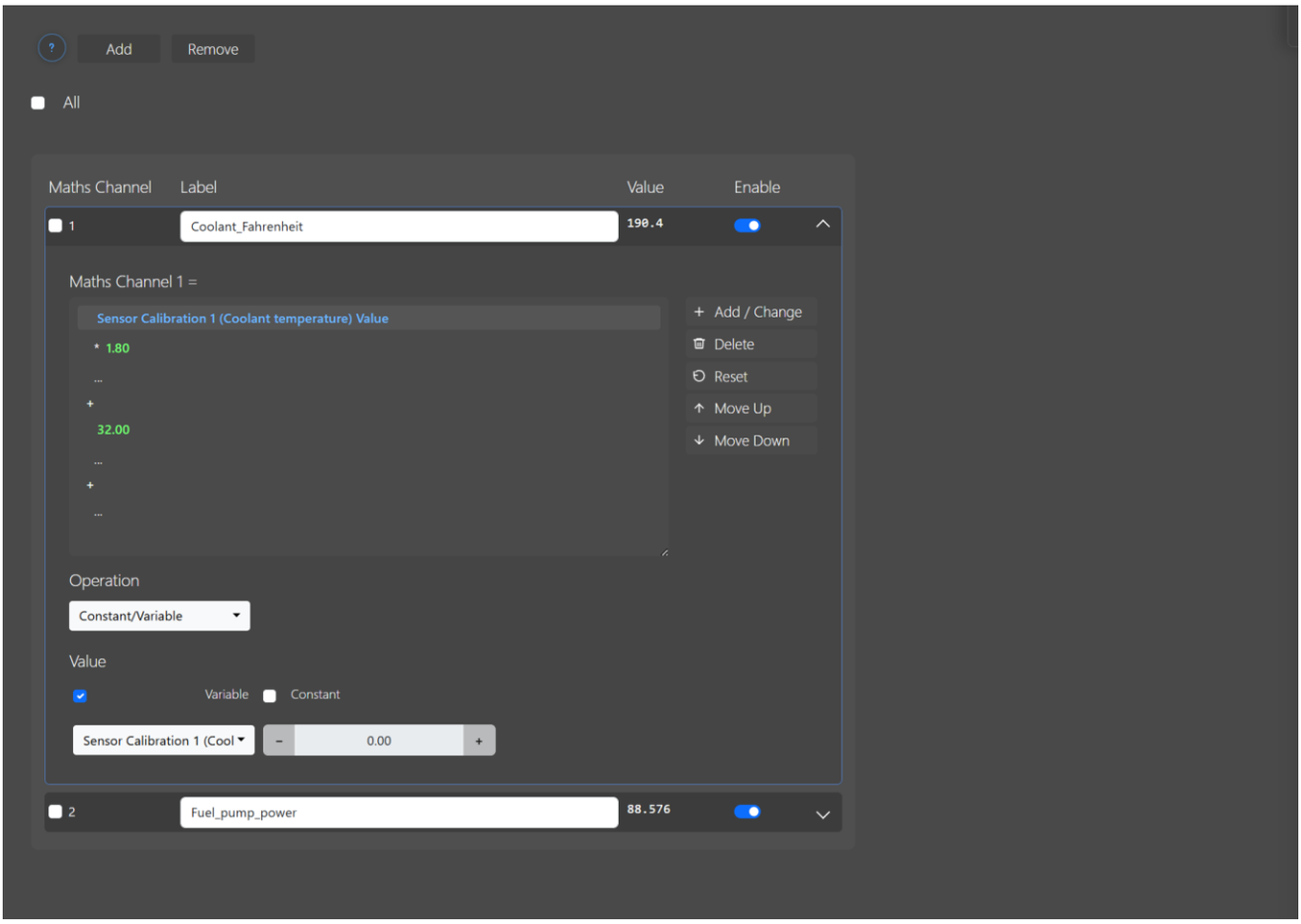


Figure 18 Maths Channels.

Scripts

Scripts are small programs that run on the PDM. Use them when the built-in features cannot do what you need on their own. A script can:

- read any PDM value, such as battery voltage, an input or a CAN signal
- make decisions and perform calculations, including ones that depend on time
- publish results that other features can use, such as output logic or **CAN Outputs**
- change a counter value or a CAN keypad button state
- send its own CAN messages

How a script is built

Every script has the same basic shape:

```
export let result = 0;   // An export: other features can read this value.
let count = 0;           // A private variable: only this script can see it.

function setup() {
    // Runs once when the script starts.
}

function loop() {
    // Runs again and again for as long as the script is enabled.
    count += 1;
    result = device.Battery_Voltage * 2;
}
```

Create and run a script

1. Open **Configuration > Scripts**, add a script and expand its row.
2. Write your script in the editor, or select all the existing code and paste a complete script over it. A script must always contain both `setup()` and `loop()`.
3. Check the status line below the editor. It reports any errors and shows how much of the script's resources you have used.
4. Enable the script. New scripts are disabled by default.
5. Select **Send to Device**.
6. While connected, expand the script row and open **Live Variables** to watch the script running.

Example — Low battery warning

This example watches the battery voltage and raises a warning if it stays low. It is a practical script you can adapt, and it demonstrates most script features in one place.

The script:

- turns on a warning when the battery has been below **11.8 V** for **5 seconds**, so that a brief dip while cranking does not trigger it
- turns the warning off again only when the battery recovers above **12.4 V**
- switches off a non-essential load (a heated screen on CAN keypad button 3) when the warning comes on
- sends a CAN message to a dash or logger whenever the warning turns on or off
- publishes `lowBattery` and `secondsLow` so that other features can use them

```
// Low battery warning
// Warns when the battery has been below 11.8 V for 5 seconds,
// and clears the warning once it recovers above 12.4 V.

export let lowBattery = false;  // Other features can use this.
export let secondsLow = 0;      // How long the voltage has been low.

let warnVolts = 11.8;
let clearVolts = 12.4;
let warnDelay = 5;              // Seconds.
let sendPending = false;        // True while a CAN message is waiting to be sent.

// Send the warning state and battery voltage to the dash.
function sendStatus(volts) {
    let centivolts = round(volts * 100);  // 12.34 V becomes 1234.
    return CANTransmit(1, CAN_STANDARD, 0x321,
        lowBattery,                       // Byte 0: 1 = warning on.
        (centivolts >>> 8) & 0xFF,        // Byte 1: voltage, high byte.
```

```
        centivolts & 0xFF);               // Byte 2: voltage, low byte.
}

function setup() {
    sendPending = true;  // Report the starting state once at power-up.
}

function loop(dt) {
    let volts = device.Battery_Voltage;

    // 1. Time how long the voltage has been low.
    if (volts < warnVolts) {
        secondsLow += dt;
    } else {
        secondsLow = 0;
    }

    // 2. Turn the warning on or off.
    if (!lowBattery && secondsLow >= warnDelay) {
        lowBattery = true;
        sendPending = true;
        set(device.CAN_Keypad_1_Button_3_Status, 0);  // Switch off the heated screen.
    } else if (lowBattery && volts > clearVolts) {
        lowBattery = false;
        sendPending = true;
    }

    // 3. Tell the dash when the warning changes.
    if (sendPending) {
        if (sendStatus(volts)) {
            sendPending = false;  // Queued. If not, try again next time.
        }
    }

    // 4. Make the new values available, then wait 100 ms.
    publish();
    sleep(100);
}
```

How the example works

The variables at the top. `lowBattery` and `secondsLow` are export variables, so they appear in the variable list used in the rest of the PDM configurator. `warnVolts`, `clearVolts` and `warnDelay` are private variables. Keeping them together at the top makes the thresholds easy to find and change. `sendPending` remembers that a CAN message still needs to be sent. Because all of these are declared outside the functions, they keep their values from one loop to the next.

`setup()` runs once when the script starts. It sets `sendPending` so that the first loop sends the starting state. This means the dash is up to date as soon as the PDM powers up.

`loop(dt)` runs repeatedly. The `dt` parameter gives the time, in seconds, since the previous loop started. The loop works through four steps:

1. **Time the low voltage.** The script reads the battery voltage once into the local variable `volts`. While the voltage is below `warnVolts`, it adds `dt` to `secondsLow`. As soon as the voltage is above the threshold, the timer resets to zero. Using `dt` keeps the timer accurate even though the time between loops varies slightly.
2. **Turn the warning on or off.** The warning turns on once `secondsLow` reaches `warnDelay`. It turns off only when the voltage rises above `clearVolts`. Between 11.8 V and 12.4 V the warning stays as it is, so it does not flicker on and off when the voltage hovers near a single threshold. This technique is called *hysteresis*.

When the warning turns on, `set()` switches keypad button 3 off. This happens only once, at the moment the warning starts, so the driver can switch the button back on if they need to.

3. **Send the CAN message.** Whenever the warning changes, `sendPending` is set to `true`. The helper function `sendStatus()` then builds and queues the message. `CANTransmit()` returns `true` if the PDM accepted the frame. If the transmit queue was full, it returns `false`, `sendPending` stays `true`, and the script tries again on the next loop.
4. **Publish and wait.** `publish()` makes the new values of `lowBattery` and `secondsLow` available straight away. `sleep(100)` then pauses the script for 100 milliseconds, so the check runs about ten times a second.

The CAN message contains three bytes:

Byte	Contents	Example
0	Warning state: `1` = on, `0` = off	`1`
1	Battery voltage × 100, high byte	`0x04`
2	Battery voltage × 100, low byte	`0x7E`

For example, 11.50 V becomes 1150, which is `0x047E`. The script rounds the voltage to a whole number first, then splits it into two bytes.

Use the result elsewhere

Each export appears in the variable list under its script, for example **Script 1 Export 1 lowBattery**. Select it anywhere the Configurator lets you choose a variable. For example, use it in an output's logic to light a warning lamp or shed other loads, or include it in a **CAN Outputs** message.

Writing scripts

Variables

There are three kinds of variable:

Kind	Declared	Keeps its value between loops?	Visible to other features?
Export	`export let name = 0;` outside any function	Yes	Yes
Private global	`let name = 0;` outside any function	Yes	No
Local	`let name = 0;` inside a function	No — it starts again on each call	No

Global variables are set up once, in the order they appear, before `setup()` runs. Restarting the script, for example by sending a new configuration, resets every variable. Script variables are not saved when the PDM powers off.

Numbers and true/false

All variables are numbers. A variable can hold a whole number or a fraction, and all calculations use 64-bit floating-point precision.

`true` and `false` are stored as `1` and `0`. Comparisons such as `volts < 11.8` and logical operators such as `&&` (and), `||` (or) and `!` (not) also return `1` or `0`. In an `if` statement, `0` counts as false and any other number counts as true.

You can write numbers in decimal (`12.5`), hexadecimal (`0x321`) or binary (`0b1010`).

Making decisions and repeating

Use `if` and `else` to make decisions, and `for` or `while` to repeat code:

```
if (volts < 11.8) {
    lowCount += 1;
} else if (volts > 14.8) {
    highCount += 1;
}

for (let i = 0; i < 4; i += 1) {
    total += samples[i];
}
```

`break` leaves a loop early and `continue` skips to its next pass. `switch`, `do`/`while` and `try`/`catch` are not supported.

Maths functions

Call these directly by name:

Function	Result
`abs(x)`	The value without its sign.
`round(x)`, `floor(x)`, `ceil(x)`	Round to the nearest whole number, down or up.
`min(a, b)`, `max(a, b)`	The smaller or larger of two values.
`sqrt(x)`, `pow(a, b)`	Square root, and `a` to the power of `b`.
`sin(x)`, `cos(x)`, `tan(x)`	Trigonometry, with angles in radians.

Arrays

An array holds a fixed number of values. This example keeps the last four voltage readings and averages them:

```
export let average = 0;
let samples = [12, 12, 12, 12];  // Four positions, numbered 0 to 3.
let next = 0;

function setup() {}

function loop() {
    samples[next] = device.Battery_Voltage;  // Replace the oldest reading.
    next = (next + 1) % 4;                   // 0, 1, 2, 3, 0, 1, ...
    average = (samples[0] + samples[1] + samples[2] + samples[3]) / 4;
}
```

An array cannot change size and cannot be exported. To make an array value available elsewhere, copy it, or a result calculated from it, into an exported variable. Using a position outside the array, such as `samples[4]` above, stops the script with a fault.

Helper functions

You can write your own functions to keep a script tidy, like `sendStatus()` in the example. Declare them outside `setup()` and `loop()`:

```
function toFahrenheit(celsius) {
    return celsius * 9 / 5 + 32;
}
```

A helper can read and change global variables, call other helpers, and return a number. It cannot call itself.

Timing

How often loop() runs

The time between loop calls is not fixed. It depends on what else the PDM is doing. If a calculation depends on time, use `loop(dt)`. The `dt` parameter is the time in seconds since the previous call to `loop()` started. It is `0` on the first call.

The PDM may also pause a long-running script briefly to let other tasks run. The script then continues from where it stopped, with its variables unchanged.

Pause a script

Call	Effect
`sleep(ms)`	Pauses the script for at least `ms` milliseconds, then continues on the next line. `ms` must be a whole number from `0` to `2147483647`. Use `sleep()` only in `loop()` or in a helper called from `loop()`.
`yield()`	Lets the PDM run other tasks, then continues without a set delay. Most scripts never need this.

Use `sleep()` to set how often a script does its work. In the example, `sleep(100)` means the voltage is checked about ten times a second. Do not use a loop that does nothing while waiting for time to pass.

Publish values

Changes to exports are not visible to the rest of the PDM straight away. The PDM publishes all exports together when `loop()` finishes, so other features never see a half-finished set of results.

Call `publish()` to make the current values available earlier. Always call `publish()` before `sleep()`. Otherwise, other features see the new values only after the pause.

If other features need values before the first loop, set them in `setup()` and then call `publish()`.

Read device values

Type `device.` and select a value from the autocomplete list.

Write to a counter or keypad button

Use `set()` to change the value of a configured counter or the state of a CAN keypad button. Type `set(device.` and choose the target from autocomplete.

```
set(device.Counter_1_Value, 0);               // Reset Counter 1 to zero.
set(device.CAN_Keypad_1_Button_3_Status, 0);  // Switch button 3 off.
```

Target	Allowed values
Counter value	Whole numbers from –2,147,483,646 to 2,147,483,646
Momentary or latching button	`0` or `1`
Multi-state button	Whole numbers from `0` up to the button's number of states

The write takes effect immediately. The counter or button then carries on working normally: a counter keeps counting from its new value, and a button can still be pressed.

Write once, not every loop. Placing `set()` directly in `loop()` repeats the write on every pass, so the button or counter can never change from that value. Call `set()` only when something changes, as the example does when the warning turns on.

Send a CAN message

Use `CANTransmit()` to send a classic CAN data frame. You can call it from `setup()`, `loop()` or a helper.

```
// Standard ID 0x123 on CAN 1, with two data bytes.
CANTransmit(1, CAN_STANDARD, 0x123, 0x11, 0x22);

// Extended ID on CAN 1, with no data bytes.
CANTransmit(1, CAN_EXTENDED, 0x18FF0100);
```

Argument	Accepted values
Bus	`1` on all models. The PDM28 also supports `2`.
ID type	`CAN_STANDARD` for an 11-bit ID, or `CAN_EXTENDED` for a 29-bit ID.
Message ID	`0` to `0x7FF` for a standard ID, or `0` to `0x1FFFFFFF` for an extended ID.

Argument	Accepted values
Data bytes	Zero to eight bytes, each a whole number from `0` to `255`. The PDM sets the data length from the number of bytes you supply.

The bus speed and termination come from the PDM configuration. `CANTransmit()` sends exactly the bytes you give it. If the receiving device expects scaling, a rolling counter or a CRC, your script must include them. For messages sent at a regular rate, **CAN Outputs** is usually simpler.

Check whether the message was accepted

`CANTransmit()` does not wait for the message to be sent. It places the frame in a queue and returns straight away:

Returns	Meaning
`true` (`1`)	The frame was queued. This does not confirm that another device received it.
`false` (`0`)	Nothing was queued, because the queue was full or the bus was temporarily unavailable. Try again later if the message is still needed.

Queue and rate limits

- Each script can queue up to four frames per bus. Frames from one script are sent in the order they were queued.
- All scripts together can send up to 100 frames per second on each bus, with at least 10 ms between frames. A busy bus can add further delay.
- Queued frames are discarded if the script is disabled, restarts or faults, if a new configuration is sent, or if the bus becomes unavailable.

Split a number into bytes

A CAN byte holds a value from 0 to 255. To send a larger number, split it into bytes:

```
let value = 1150;                    // 0x047E
let highByte = (value >>> 8) & 0xFF;  // 0x04
let lowByte = value & 0xFF;           // 0x7E
```

`& 0xFF` keeps the lowest eight bits. `>>> 8` moves the next eight bits down into that position first. Use `>>> 16` and `>>> 24` for the third and fourth bytes of a larger number.

These operators discard any fraction, so convert measurements to whole units with `round()` before splitting them. For example, `round(11.5 * 100)` gives `1150`.

Monitor a running script

While connected, expand the script row and open **Live Variables**. It shows the script's device inputs, exports, private variables, local variables and each array element. Hover over a row to see its recent history.

Exports and private variables appear only when the script in the editor matches the one on the PDM. If they show a dash, select **Send to Device**. A dash can also mean that a value has not been set yet.

Each script also adds three status variables to the variable list:

Status variable	Meaning
Enabled	`1` when the script is enabled in the configuration.
Error Flag	`1` when a fault has stopped the script.
Time Since Update	Seconds since the script last published its exports, or `-1` if it has not published yet.

You can use **Error Flag** in your configuration, for example to light a warning lamp if a script stops.

Find and fix a fault

If a script hits a problem it cannot continue from, the PDM stops that script, sets its **Error Flag** to `1` and sets all of its exports to `0`. Other scripts and the rest of the PDM keep running. Any `set()` writes already made stay in effect.

Common causes:

Fault	Typical cause
Division by zero	Dividing by a value that is zero. For example, `dt` is `0` on the first loop.
Array index	Using a position outside the array, or a fractional position.
Arithmetic	A calculation produced an invalid result, such as `sqrt()` of a negative number.
Export range	An export is larger than ±9,007,199,254,740 when published. The highlight points to the `publish()` call or the end of `loop()`, not the line that calculated the value.
No progress	A loop ran for more than 100,000 instructions without finishing `loop()`, sleeping or publishing.
Sleep	An invalid delay, or `sleep()` called during `setup()`.
Device write	A `set()` value outside the allowed range for that counter or button.
CAN transmit	An invalid `CANTransmit()` argument, or a bus this model does not have.

Script limits

Resource	Limit
Scripts per configuration	5
Exports per script	10
Device values per script	16, including `set()` targets
Script size	4,096 bytes, including comments and spaces
Compiled code	768 instructions and 64 different numeric constants
Variable storage	64 values. Each variable, array element and helper parameter uses one.
Functions	16, including `setup()` and `loop()`
Function call depth	8 functions in a chain
Export range	−9,007,199,254,740 to +9,007,199,254,740

For the complete language rules and further examples, open Scripts in Configurator Help and select PDM Script Language Reference.

Generic Functions

Logic Lines

Generic functions operate in a similar fashion to the functions found in the global reset, global cutoff, and output sections. The function is evaluated constantly by the PDM, and the result (either 1 or 0) can be used elsewhere in the PDM for more advanced behaviour.

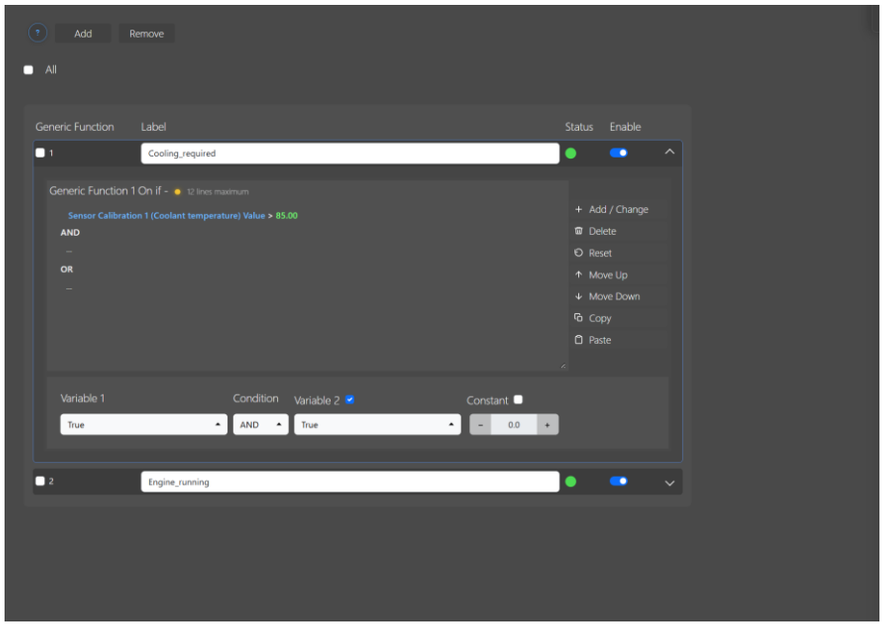


Figure 19 Logic Lines.

Sensor Calibrations

Sensor calibrations allow one to take a raw value/sensor reading and scale/offset its value at different points. This can be used for example, to linearise a temperature sensor, or to scale a throttle position sensor between 0 and 100%. Select the raw sensor value (this can be any variable in the PDM), and then use the arrows to adjust the calibrated value to your desired place. Each raw sensor value will map to the calibrated value according to the graph, with linear interpolation between the points.

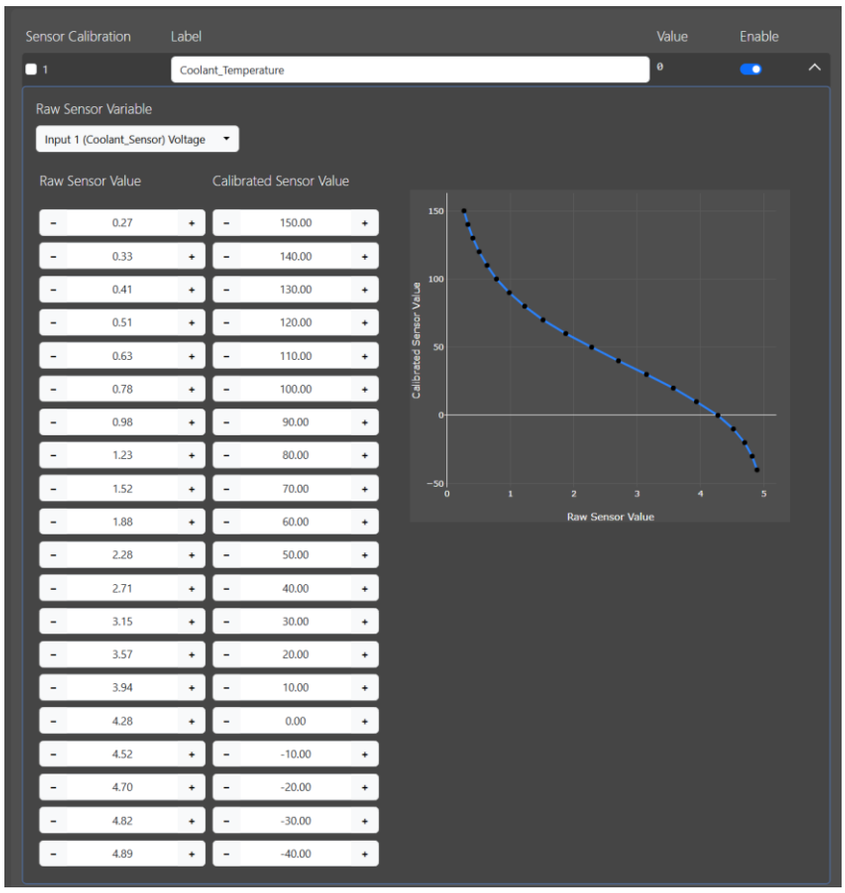


Figure 20 Sensor Calibrations.

IMU

Use the IMU page to align the PDM with the vehicle and configure impact and rollover triggers.

Mounting Orientation

Park the vehicle on level ground and keep it stationary. Select Configure Mounting, capture Vehicle Up, adjust Heading until the PDM model matches its installed forward direction, then select Save Reference. Clear Reference removes the saved vehicle frame.

Rollover Trigger

Set Maximum safe tilt to the largest angle considered upright. The mini globe shows the safe and trigger regions, while the live tilt value shows the current vehicle angle.

Impact Trigger

Each X, Y, and Z axis has separate negative and positive acceleration thresholds and can be enabled independently. Acceleration Total monitors the vector magnitude sqrt(X^2 + Y^2 + Z^2), which is useful for impacts that are not aligned with one axis.

Trigger state and reset

Impact and rollover settings create PDM variables. Add the required trigger variables to Global Cut Off or other logic, then send the configuration. Triggers are cleared by Global Reset. Check Data Valid and the reference-frame status before relying on the trigger values.

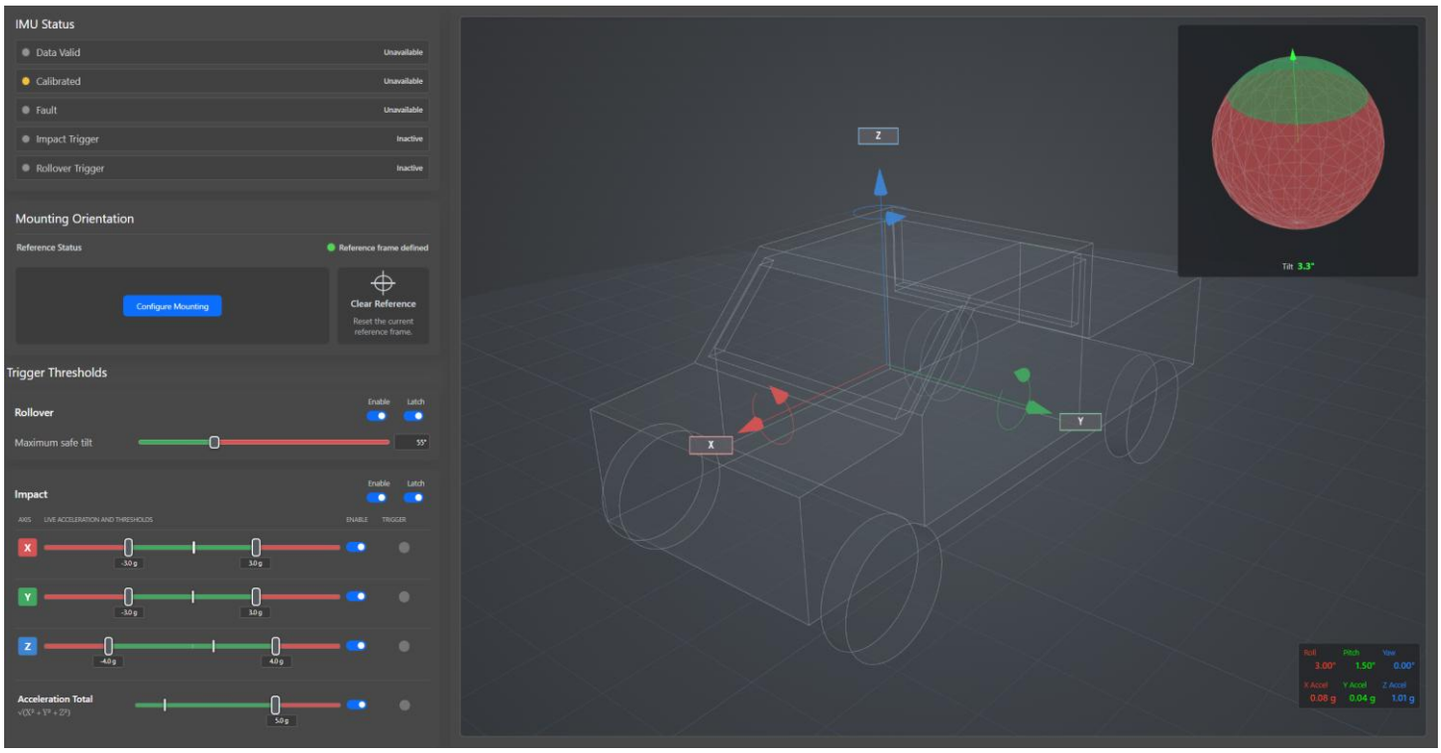


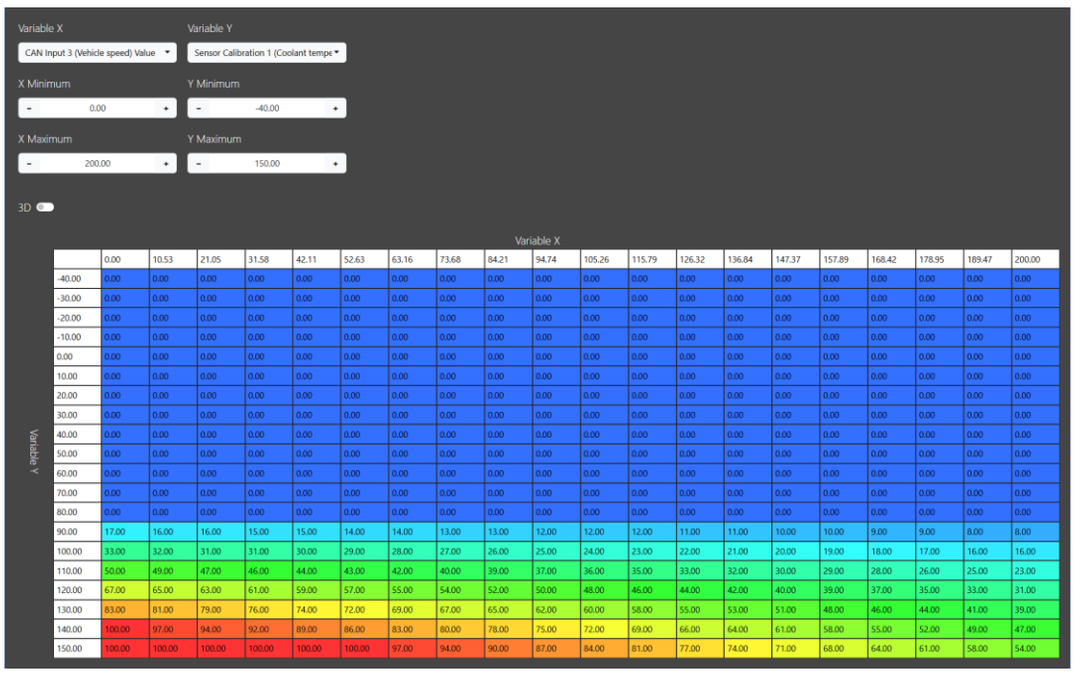
Figure 21 IMU mounting and trigger settings.

2D Tables

The PDM has 5 individual 20x20 2D Tables. 2D tables allow you to take two input variables and map their values to an output. First start by selecting your two input variables. In this made-up example, vehicle speed and coolant temperature has been selected, and the output value can be used for radiator fan PWM duty cycle %. You can then select the range that these variables will fall within. In this case, we would only be interested in a vehicle speed greater than 0 and less than 200, and a temperature greater than -40degC and less than 150degC. The values at each point in the table can then be input as desired.

Hover over the 'I' Icon to see the keyboard shortcuts which can be used for interpolation and mass data change.

In the example below, Variable X is vehicle speed and Variable Y is coolant temperature. The table keeps the fan off at low coolant temperatures. As the coolant gets hotter, demand increases. For the same hot coolant temperature, the table reduces fan demand as vehicle speed rises.



Variable Y \ Variable X	0.00	10.53	21.05	31.58	42.11	52.63	63.16	73.68	84.21	94.74	105.26	115.79	126.32	136.84	147.37	157.89	168.42	178.95	189.47	200.00
-40.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
-30.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
-20.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
-10.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
70.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
80.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90.00	17.00	16.00	16.00	15.00	15.00	14.00	14.00	13.00	13.00	12.00	12.00	12.00	11.00	11.00	10.00	10.00	9.00	9.00	8.00	8.00
100.00	33.00	32.00	31.00	31.00	30.00	29.00	28.00	27.00	26.00	25.00	24.00	23.00	22.00	21.00	20.00	19.00	18.00	17.00	16.00	16.00
110.00	50.00	49.00	47.00	46.00	44.00	43.00	42.00	40.00	39.00	37.00	36.00	35.00	33.00	32.00	30.00	29.00	28.00	26.00	25.00	23.00
120.00	67.00	65.00	63.00	61.00	59.00	57.00	55.00	54.00	52.00	50.00	48.00	46.00	44.00	42.00	40.00	39.00	37.00	35.00	33.00	31.00
130.00	83.00	81.00	79.00	76.00	74.00	72.00	69.00	67.00	65.00	62.00	60.00	58.00	55.00	53.00	51.00	48.00	46.00	44.00	41.00	39.00
140.00	100.00	97.00	94.00	92.00	89.00	86.00	83.00	80.00	78.00	75.00	72.00	69.00	66.00	64.00	61.00	58.00	55.00	52.00	49.00	47.00
150.00	100.00	100.00	100.00	100.00	100.00	100.00	97.00	94.00	90.00	87.00	84.00	81.00	77.00	74.00	71.00	68.00	64.00	61.00	58.00	54.00

Figure 22 Fan demand table.

Clicking the 3D button allows you to view the table in 3-Dimensions.

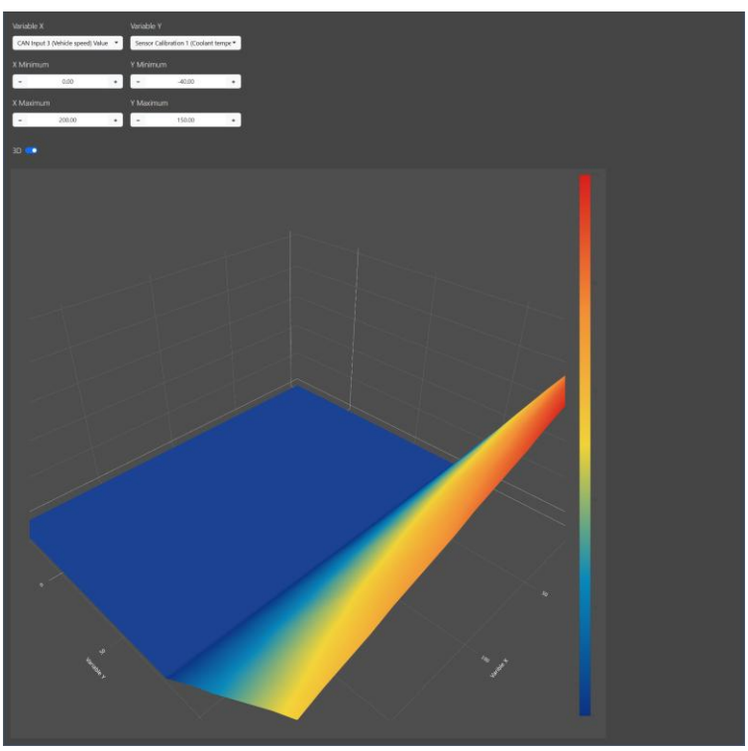


Figure 23 Fan demand surface in 3D.

CAN Keypads

Using CAN bus 1 (and CAN bus 2 on the PDM28), the PDM can communicate with up to 4 individual CAN bus Keypads. The PDM is able to communicate with the following keypad devices -

• Blink Marine CANOpen 4-15 button keypads.

• Grayhill CANOpen 6-20 button keypads.

• Motec 6-20 button keypads.

• Marlin Technology J1939 8 button keypad. Keypads are a great way to control the functionality of the Power Distribution System. The Keypad buttons can be used to control the state of multiple outputs, and the LED indicators on each button provide visual indicators to the user.

To install a Keypad into the system, make sure that the CAN bus lines are connected, and that the wiring is done in accordance with the instructions given in the CAN bus wiring section. Up to 4 Keypads can be used on the same CAN bus. It is important that the Keypads do not have the same ID. If the Keypads have the same ID, then the system will not work. It is also important that all of the Keypads are configured to run at the same CAN bus speed.

Keypad Configuration

To configure a Keypad, go to the CAN Keypad tab and select one of the four available Keypads to configure. Click the enable button to enable the keypad, and select the appropriate CAN bus which the Keypad is installed on. Choose the keypad which is being used from the drop down menu. Choose the appropriate ID which the keypad is configured to. Make sure to observe that the Keypad ID is entered in the decimal format in the configuration software.

If the keypad is to be powered from a PDM output, on a PDM, set up that output and enable it. Press 'Send' to send the configuration to the PDM. The Keypad should now light up with the backlight colour given in the Keypad configuration tab, and the Connection status should turn Green and say 'Connected.'

If the Keypad does not light up, check the following is correct -

• The CAN bus speed of the Keypad and PDM is identical

• The CAN bus is wired correctly in accordance with the recommendations given in the CAN bus wiring section.

• The correct CAN bus is selected in the Keypad configuration

• The ID of the Keypad is the same as the ID in the Keypad configuration.

Once the Keypad has successfully connected with the PDM, the Keypad functionality can be configured.

Remember Button States

Enable Remember Button States to retain the keypad button states across a PDM restart. Check that restoring a previously selected state is appropriate for the circuits controlled by the keypad.

The Button Brightness, Backlight Colour, and Backlight Brightness can be selected from the drop down menus. The Dimming Variable is used to dim

the keypad lights when the Dimming Variable is equal to True. This is useful for night time operation. The Dimming Percentage is the brightness of the Buttons and Backlight as a percentage of the Button Brightness and Backlight Brightness settings.

Keypad Button Configuration

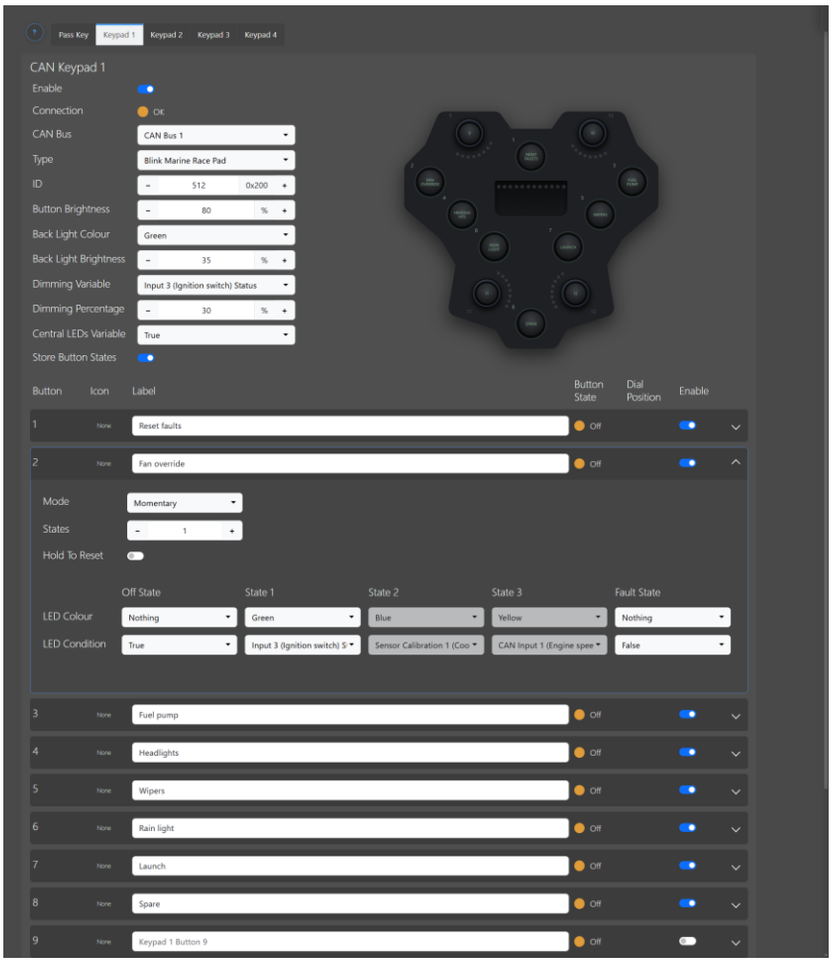


Figure 24 Keypad Button Configuration.

Each Keypad Button can be given a label and be enabled. Click the 'Expand' Button to configure the operation of the Keypad Button.

The Mode of the button can be either Momentary, Latched, or States. In Momentary Mode, the Keypad Button State will be True when the button is pressed, and False when it is not pressed.

In Latched Mode, the Keypad Button State will toggle between True and False each time the Button is pressed.

In States Mode, each time the Button is pressed, the Button State will increment. The number of states can be set between 1 and 3. When the Button State reaches the limit, it will return back to 0. Hold To Reset is used to reset the Button state back to 0 when the button is held for one second. For each Button State, the Button Colour and functionality can be set. The first drop down menu selects which colour the Button will be in that state. The second drop down menu selects a variable which is used to say when the button is that colour.

For instance, if the colour is selected to be Green, and the variable is selected to be 'True' - when the button is in that state, the button will be Green constantly. If the colour selected is Green, and the variable selected is 'Output 1 Status,' then the button will be Green when in that state AND when the state of 'Output 1 Status' is True. This is a useful feature, as the Keypad Button Colour can be changed based on the state of a variable. This, for example, allows the button to flash in time with a turn signal output, indicating to the user that the output is flashing.

Keypad Button Icons

Click the icon beside a keypad button to open the software icon picker. Use its categories or search to choose an icon, or remove the selection to leave the button without one. Selecting a software icon does not change the physical insert fitted to the keypad button.

CAN Keypad Dial Configuration

Some supported CAN keypads include rotary dial functions. When a keypad type with dial support is selected, additional dial configuration options become available for the relevant buttons.

To configure a keypad dial, navigate to Configuration > CAN Keypads, select the required keypad, then expand the required button. If the selected keypad type supports a dial on that button, a Keypad Dial section will be shown.

The available dial settings are Minimum, Maximum, LED Offset, and Hold To Turn.

Minimum is the lowest dial value. Maximum is the highest dial value. LED Offset shifts the LED start position around the dial ring. Hold To Turn requires the dial to be pressed before it can be rotated.

These settings define the usable dial range and how the LED ring behaves. The live dial position is shown in the main keypad table while monitoring.

Dial options are only displayed for keypad types and buttons that support them. For example, Blink Marine 15 key CAN Open w/ Dial provides dial-enabled buttons with a range up to 16, while Blink Marine Race Pad provides dial-enabled buttons with a range up to 8.

If no dial section is shown for a button, that button does not support dial configuration for the selected keypad type.



Figure 25 CAN Keypad Dial Configuration.

Pass Key

As a layer of security for the Power Distribution System, a Pass Key can be added to the Keypads. If the Pass Key is enabled, then the Keypads will not be operational until the correct sequence of Keypad Buttons has been pressed. This can stop unwanted users from potentially starting up a vehicle and driving it away.

CAN Bus Basics

This section provides a brief overview of the CAN bus. It is not necessary to understand exactly how the CAN bus works to successfully use it in the Power Distribution System. Skip to the next section to learn how to configure the PDM CAN inputs and outputs. For a full detailed video explanation of the CAN Bus, check out our beginners guide which can be found in the QR code below. Controller Area Network (CAN) Bus is a communication protocol designed

to allow different devices to communicate with each other in a robust manner. CAN bus covers both the physical wiring, and how the data is sent/received.

Figure 26 CAN Bus Basics.

For a full detailed video explanation of the CAN Bus, check out our beginners guide which can be found in the following QR code

CAN Inputs

The PDM15/25/35 has one user accessible CAN bus port and the PDM28 has two. These can be used to interface with other devices in the vehicle - such as ECUs, Digital Dashboards, Sensors, and more. Up to 100 individual CAN Inputs can be configured, spread across the available CAN bus ports. The CAN bus on a vehicle can become heavily crowded with CAN messages from different devices on the bus. Configurable hardware filters are used in the PDM to decipher frames of interest so that the processor is not overloaded with processing unwanted CAN messages. These filters only allow CAN frames with specific IDs to pass through for the processor to deal with, others are discarded automatically. Once the CAN message has passed through the acceptance filters, the CAN message is then processed by a CAN Input with matching ID.

CAN Configuration Files

To store the configuration of the CAN inputs on the PDM, the configuration can be saved as a file. A CAN Configuration file can also be loaded into software if needed. The CAN Configuration files are specific to the Hardwire Electronics PDMs.

DBC File Loader

.DBC Files are a standard used to specify how CAN messages are sent on a CAN bus. For example, an ECU manufacturer may have a DBC file which describes how the CAN data it sends out is formatted. The DBC file can then be used by another device to set up its own CAN inputs automatically.

The configuration software allows one to load up a DBC file, and choose which signals are required. Press ‘Import .DBC File,’ and select the DBC file which needs to be loaded. A new window will appear which shows all of the available signals in the DBC file. The correct CAN bus can be selected at the top, and each DBC file signal can be selected with the checkboxes below. Once all the required signals have been selected, press Import. This will automatically configure the PDM CAN Filters and CAN Inputs. By default, the CAN Filters and CAN Inputs from the DBC file will be added onto the existing CAN Filters and CAN Inputs in the configuration. To overwrite the current CAN Filters and CAN Inputs, click the ‘Overwrite Existing CAN Input Data’ checkbox before pressing Import. The software will attempt to optimise the CAN filters based on the CAN Inputs you have chosen in the DBC file.

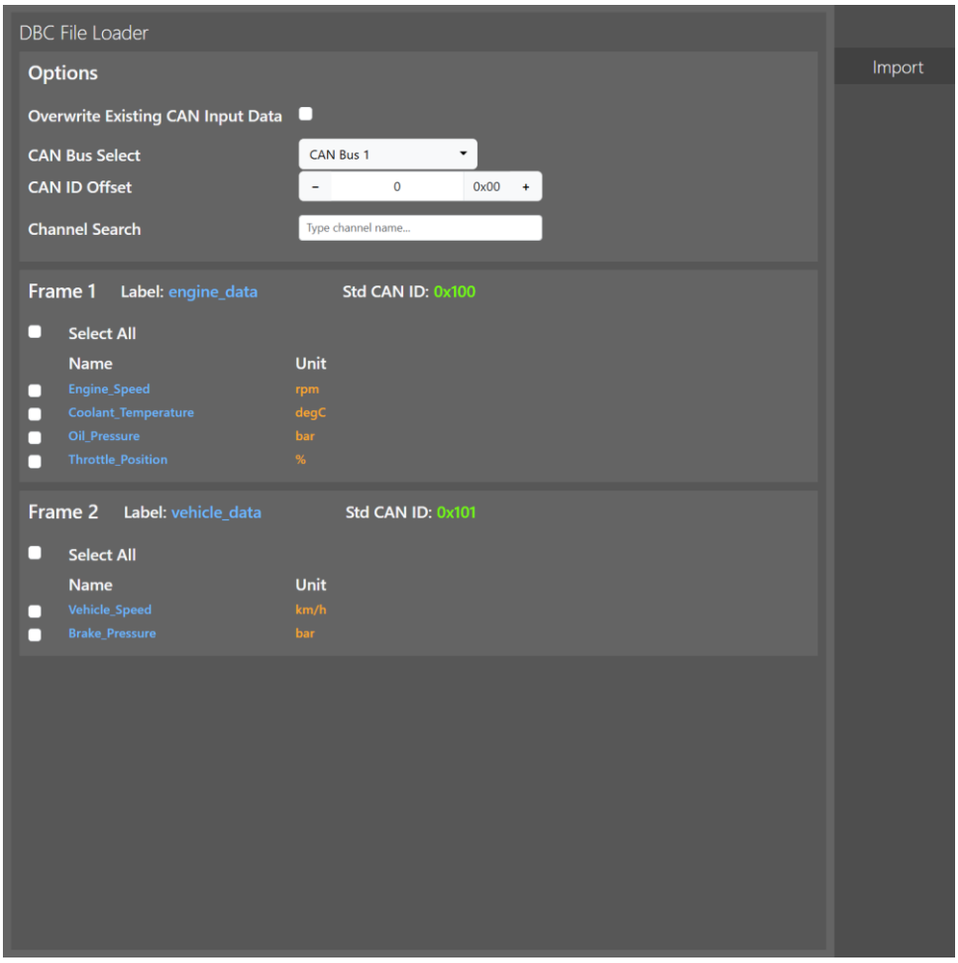


Figure 27 Importing signals from a DBC file.

CAN Bus Filter Configuration

Each CAN bus has ten hardware filters. A filter accepts a range of standard or extended CAN IDs. In Automatic mode the Configurator builds the ranges from the enabled CAN Inputs on each bus. When more than ten ranges are required, it merges the closest IDs into ranges where possible.

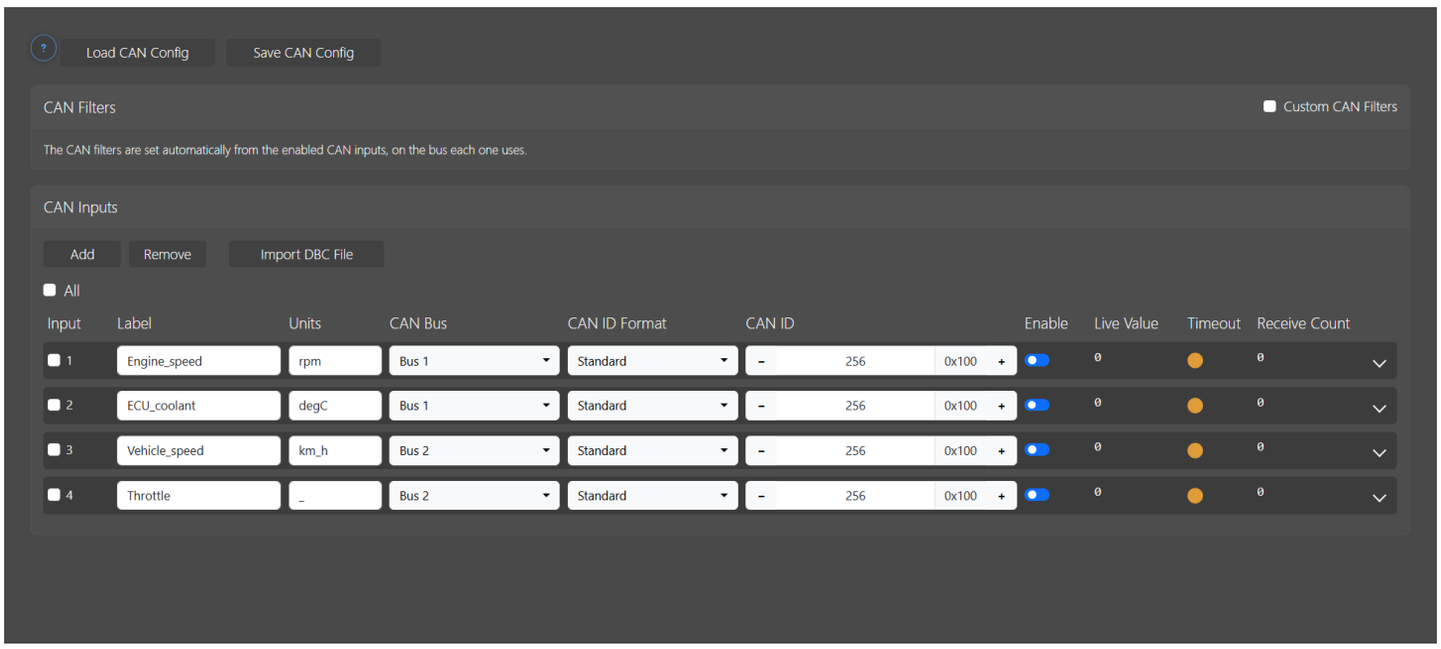


Figure 28 CAN Inputs with automatic CAN filters.

Optimise CAN Filters

To edit the ranges yourself, tick Custom CAN Filters. The Configurator starts with the automatic ranges. Click Optimise CAN Filters to rebuild those ranges from the enabled CAN Inputs. Configurations created before automatic filtering open in Custom mode with their saved ranges.

CAN Bus Selection

Select Bus 1 or Bus 2 in the CAN input row to receive the message on that bus only. Select Either to listen for the same ID on both buses. Automatic mode adds the ID to the filter list for every bus the input uses. If two devices send the same ID, select the bus carrying the required device. Check that the ID format and payload size match the frame being sent. On the PDM15/25/35, which has one CAN bus, use Bus 1.

CAN Input Configuration

To add a CAN Input, click the 'Add' Button. Up to 100 CAN Inputs can be added to the configuration. The CAN Input can be given a Label and a Unit, useful for viewing data in the Monitoring Tab. The CAN ID Format can be selected as either a Standard or Extended ID. If the CAN ID is valid, it will have a green tick next to it. If the CAN ID is invalid, it will have a red tick next to it. The CAN ID is valid if it falls within one of the CAN Filter ranges.

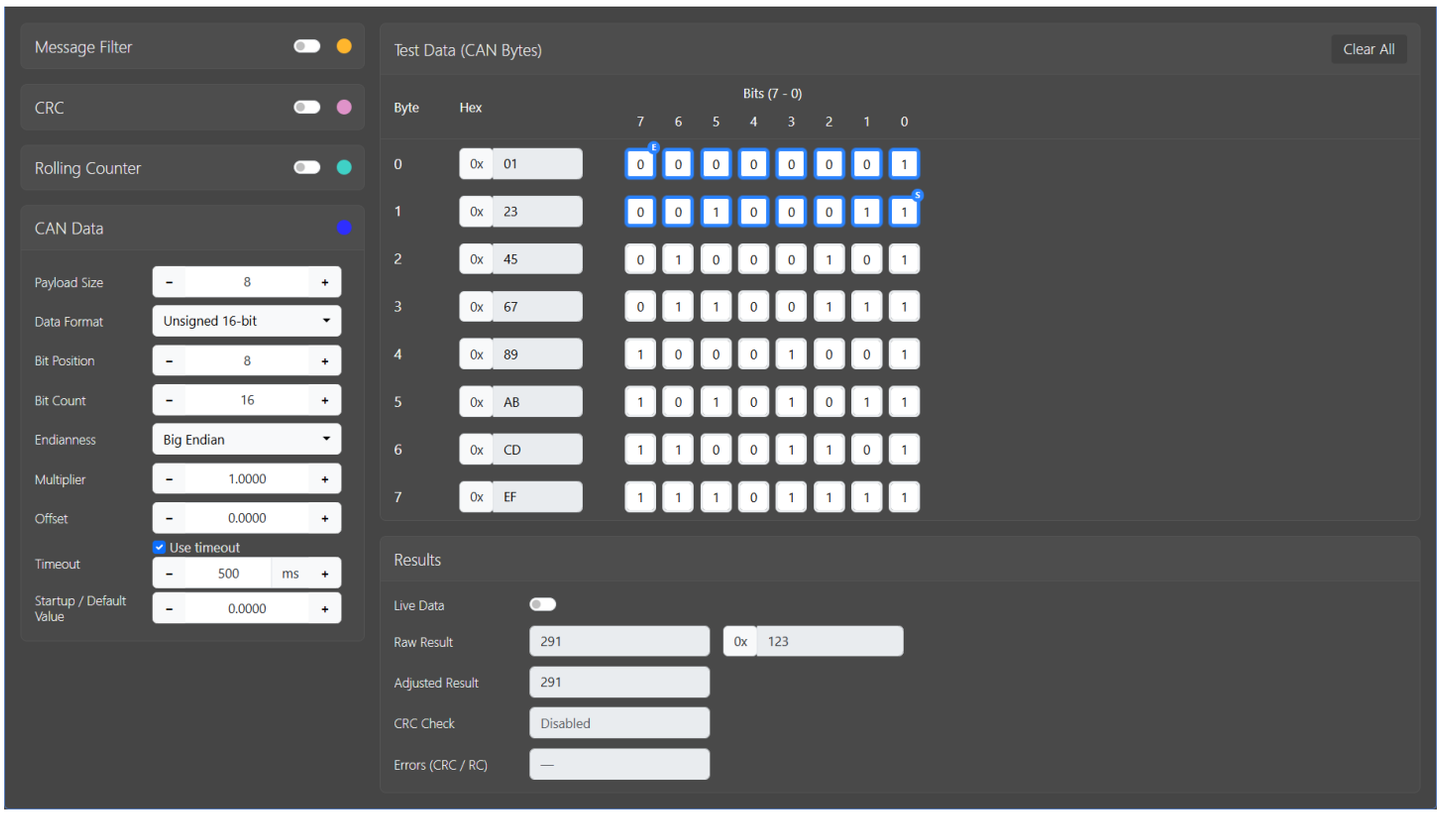


Figure 29 CAN input settings with Message Filter, CRC and Rolling Counter switched off.

Click Expand to configure the remainder of the CAN Input settings. In the dropdown, all of the parameters for the CAN Input can be configured. The aim of the CAN Input is to parse off the appropriate data from the payload bytes of the CAN message. A CAN Message can have up to 8 payload bytes. Within the payload bytes may be a number of different variables. For example, the first byte may contain engine RPM, the second byte may contain water temperature etc. The CAN Input is configured to parse off the specific variable value which is required.

Test Data

To make configuring the CAN Inputs easier, test data has been added to the bottom of the window. The test data is simply 8 data bytes which mimic the data that would reach the CAN Input. The Blue squares represent the data which is being parsed from the payload. The Orange Squares represent which bits the Message Filter uses. The resulting CAN Input variable value is shown below. Once an appropriate CAN Input Filter has been enabled, the 'Live Data' check box can be clicked. This will show the live data on the CAN bus in the test data bytes. The CAN Input can then be configured to correctly parse the data from the payload in real time.

Message Filter
In addition to filtering the CAN messages by ID, the PDM can also further filter the CAN messages by data in the payload. This is useful when a device on the CAN bus uses compound or segmented CAN messages to send data. Compound CAN messages are used to send different types of data with the same CAN ID. Usually, the first byte of the payload acts as a 'sub ID,' and increments in value for each data type being sent. The Filter Size in bits, the Offset in bits, and the numerical value to filter out, can all be configured with the plus and minus arrows. The test data at the bottom of the configuration window shows how the bits of the payload which will be filtered, highlighted in orange.

Payload Size
A CAN message can have between 0 and 8 data bytes. The CAN Input payload size must exactly match the payload size of the CAN Input for the message to be received.

Data Format
The format of the data in the payload can change depending on the type of data being sent. Data which can have negative values is called 'signed' data, and data which is positive only is called 'unsigned' data. The data can also have a different size, between 8 and 32 bits.

Bit Position
The Bit Position chooses the bit which the data is parsed from. Bit 0 is the least significant bit, and bit 7 is the most significant bit.

Bit Count
The Bit Count sets how many bits are parsed, starting from the Bit Position Value.

Endianness
Endianness describes the order that data bytes are sent for multiple byte variables. Big Endian order means that the most significant byte is sent first with the least significant byte after. Little Endian order means that the least significant byte is sent first with the most significant byte after. For example, consider the following CAN Message payload -

Payload Bite	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Value	**0x01**	**0x23**	**0x45**	**0x67**	**0x89**	**0xAB**	**0xCD**	**0xEF**

The variable to parse is a 16-bit variable on Byte 1 and 2. If the data is in Big Endian format, the 16-bit varia- ble would be 0x2345. If the data was Little Endian format, the 16-bit variable would be 0x4523.

Multiplier
The previous CAN settings are responsible for parsing the appropriate data bits from the CAN message payload. The Following settings are responsible for scaling and offsetting the raw data so that it is in the correct format. The Multiplier takes the raw variable value and multiplies it with a constant. The Multiplier value can range from 0.0001 to 1000. A multiplier is often used to preserve greater accuracy in the data being sent. For example, if an ECU needs to send a value of 8.36 over the CAN bus, then it would first need to multiply the value by 100 and send the resulting 836 instead. Then the receiving device can apply a multiplication of 0.01 to the raw data to return the 836 value back to the original 8.36, reserving the two decimal places of precision.

Offset
After the raw value has passed the multiplier step, an offset can be added. This can be used to adjust the value for calibration purposes. For example, if the value of 8.36 is actually supposed to be 9.00, then an offset of 0.64 can be applied.

Timeout
The Timeout value is how long the current CAN value is valid for in milliseconds. For example, If it is known that data should be received every 0.5s, but no data is received for 1s, then it can be assumed that an error has occurred. The CAN variable value will then return to a default value.

Use Timeout
Use timeout enables the CAN Input timeout check. The interval is 10 to 1000000 ms when enabled. Turning it off stores a timeout of zero and changes the value field to Start up value. The last received value is then retained until

another matching message arrives. The start-up value is used before the first valid message and is not retained across power cycles.

Timeout Flag

When timeout is enabled and expires, the input and monitor displays use the configured Default Value. Check Timeout Flag to distinguish a timeout from valid received data. With timeout disabled, the timeout flag remains clear even if messages stop arriving.

Default Value

The Default Value is the initial value of the CAN Input on device start up, or when the timeout has occurred.

CRC

Use these checks when the sending device adds a CRC or a rolling counter to its messages. Turn each on with the switch in its section header. The settings must match the sender.

CRC Type sets how the CRC and rolling counter are calculated, and must match the sender.

SAE J1850 uses polynomial 0x1D, initial value 0xFF, no reflection and final XOR 0xFF.

AUTOSAR E2E Profile 1 uses polynomial 0x1D with initial value and final XOR 0x00, covers every byte except the CRC byte, always includes the Data ID (8-bit LOW or 16-bit BOTH mode, low byte first) and counts 0 to 14.

Custom sets the polynomial, initial value, final XOR, covered bytes, Data ID size, position and byte order, and counter range, to match another device's documented CRC-8.

Check Value shows the CRC of the text 123456789 with the type's parameters and no Data ID: 0x4B for SAE J1850 and 0x37 for Profile 1. Compare it with the check value in the sender's documentation or a CRC catalogue.

CRC checks the CRC byte. Set CRC Byte and, for SAE J1850 or a Custom range, Covered From and Covered To for the bytes it is calculated over. The covered bytes must all be on one side of the CRC byte.

Data ID adds a value to the CRC calculation that is not part of the frame. It is optional for SAE J1850 (8-bit, before the data) and Custom; Profile 1 always uses it.

A frame with a wrong CRC is discarded: the value, Receive Count and rolling counter are not updated, and CRC Errors increases by one. If CRC errors continue, the input times out.

Rolling Counter

Rolling Counter checks a 4-bit counter in the Low (bits 0-3) or High (bits 4-7) nibble of Counter Byte. Each frame should increase it by one, wrapping within the CRC Type's range: 0 to 15, 0 to 14 for Profile 1, or the Custom range. A value outside the range, such as 15 with Profile 1, adds one to RC Errors. When the CRC check is also on, the counter must be inside the covered bytes.

A skipped or repeated counter value increases RC Errors by one, however many frames were missed, but the frame is still used. The first frame after power-up, sending the configuration or a timeout starts the sequence without an error.

Both checks run on every frame with this CAN ID and payload size, before the Message Filter, so each input sharing an ID sees the whole counter sequence. Inputs that share an ID should use the same settings.

CRC Test Results

In the test data grid, the CRC byte is pink and the counter nibble is teal. Red marks bits that the CRC or counter shares with the data or the message filter. CRC Check in Results compares the calculated CRC with the CRC byte in the test or live data.

CRC Errors and RC Errors count from power-up and are available as variables for logging, monitoring and scripts. Errors (CRC / RC) in Results shows them while connected. Use these variables in a script or function to implement the required fault response.

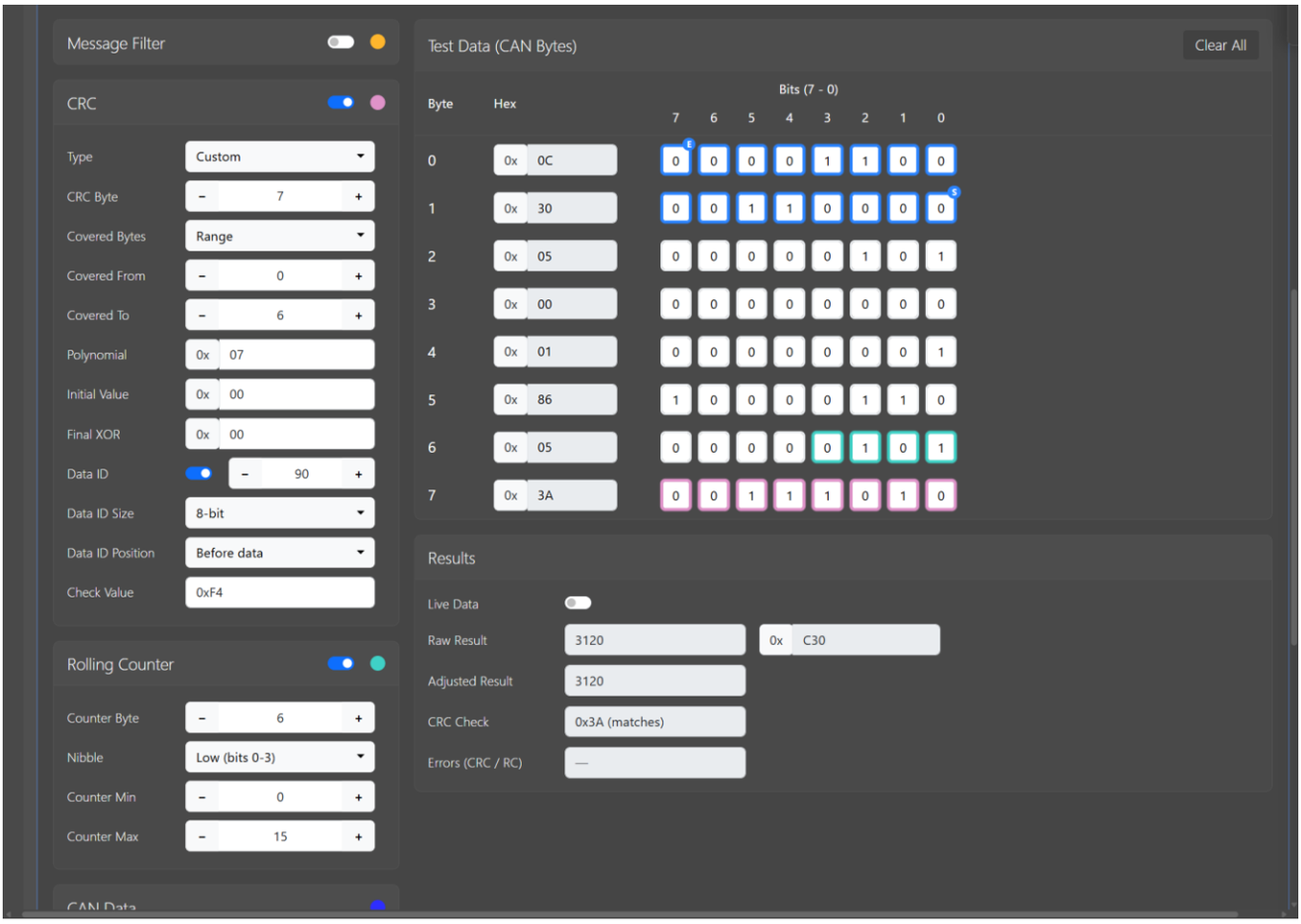


Figure 30 CAN input CRC and rolling counter.

CAN Outputs

The PDM has up to 100 individual user defined CAN Outputs. These can be configured on CAN bus 1, or on either CAN bus 1 or 2 on the PDM28. In addition to the user defined CAN Outputs, the user can enable the CAN Output stream, which sends preformatted data at a regular interval.

User CAN Outputs

A CAN Output can be added by clicking the ‘Add’ Button. A Label can then be given to that CAN Output. Click ‘Expand’ to further configure the CAN Output.

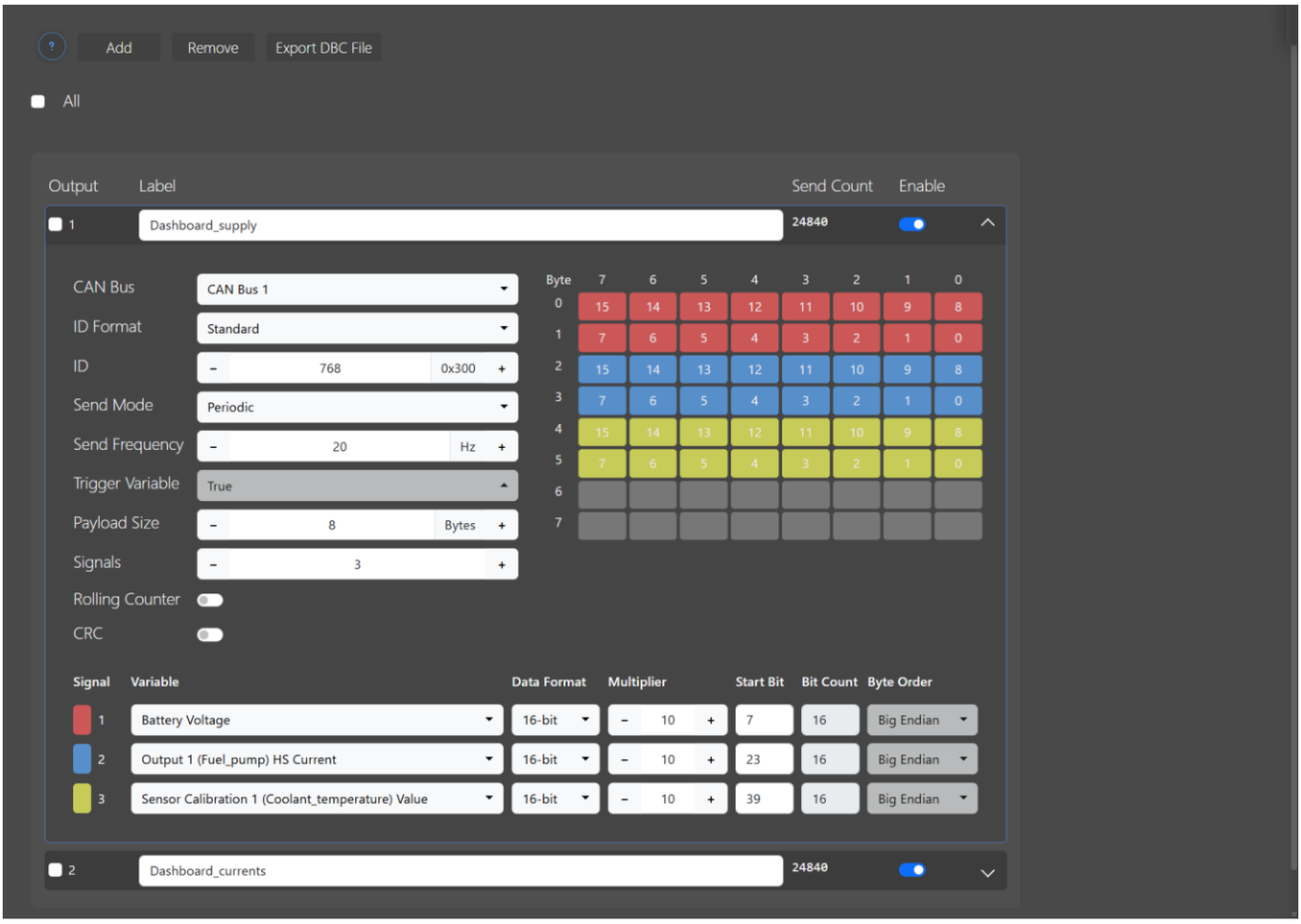


Figure 31 User CAN Outputs.

CAN Bus Selection

The CAN Bus that the CAN Output message will be sent on can be selected from the drop down menu.

CAN ID

The CAN Message ID format can then be selected as either Standard or Extended, and the value for the ID can be chosen.

Send Mode

The CAN Outputs can either be sent periodically, or be sent on a trigger. When the Send Mode is set to Periodic, the CAN Output will send at the interval defined by Send Frequency. When the Send Mode is set to Triggered, the CAN Output will send every time the Trigger variable goes from False to True. This is useful for sending a CAN Message when a certain event takes place.

Payload

A CAN Message can have between 0 and 8 bytes. This is selected with the Payload Size setting. The Payload is filled with PDM Variables.

Signals

A CAN Output can contain up to 32 signals in a payload of up to eight bytes. Set Payload Size and Signals independently. Adding a signal does not automatically extend the payload, and changing the payload does not move the existing signals.

Custom Signed and Unsigned Fields

Each signal has its own Variable, Data Format, Multiplier and Start Bit. Custom Unsigned fields support 1 to 32 bits. Custom Signed fields support 2 to 32 bits using two's complement. Custom integer formats truncate fractions towards zero and clamp values outside their range. The fixed integer formats wrap overflowing values, so choose a format wide enough for the scaled value.

Flag and Float Fields

A 1-bit Flag sends zero for a zero variable and one for any finite non-zero variable. Its multiplier is ignored. Float sends a 32-bit IEEE 754 value. The live Value column previews the converted value; the hexadecimal display for a float shows its IEEE 754 bits.

Start Bit and Byte Order

For Big Endian, Start Bit identifies the most-significant bit. Packing moves towards bit 0 within a byte and then continues at bit 7 of the next byte. For Little Endian, Start Bit identifies the least-significant bit and packing follows increasing payload positions. Use the colour-coded packing grid to check the selected bits. Red cells indicate an overlap; signals outside the payload must also be corrected before sending.

Remote Frames

An output with a zero-byte payload and no signals sends a remote frame. A script CANTransmit call with no bytes instead sends a zero-DLC data frame.

Rolling Counter

Turn on Rolling Counter or CRC to add fields that let the receiver detect lost, repeated or corrupted messages. Each adds a signal row that is positioned like any other signal; its variable, format, bit count, byte order and multiplier are fixed.

Rolling Counter is 4 bits in the high or low nibble of one byte. It starts at the lowest value of the CRC Type's range at power-up and increases by one on every send, wrapping from the highest: 15 for SAE J1850, 14 for AUTOSAR E2E Profile 1 (which never sends 15), or the Custom range. It also advances when a send fails, so the receiver can see the missed frame.

CRC

CRC is one CRC-8 byte calculated with the CRC Type: SAE J1850 (polynomial 0x1D, initial value 0xFF, no reflection, final XOR 0xFF), AUTOSAR E2E Profile 1 or Custom, as described for CAN Inputs. Check Value shows the type's check value. The PDM calculates the CRC last, after packing the signals and the rolling counter.

A new CRC goes in the last payload byte and a new rolling counter in the low nibble of the byte before it. Change their Start Bit to move them; it steps by a whole nibble or byte.

CRC Bytes sets which payload bytes are included in the checksum. With SAE J1850 or a Custom range, choose the first and last byte using positions 0 to 7. The range must stay on one side of the CRC byte. For example, with the CRC stored in byte 7, select bytes 0 to 6. If the frame also has a rolling counter, include its byte in that range. AUTOSAR E2E Profile 1 includes every payload byte except the CRC byte automatically.

CRC Data ID adds a value to the CRC calculation. It is not sent; the receiver must use the same value. It is optional for SAE J1850 (8-bit) and Custom; AUTOSAR E2E Profile 1 always uses it, as 8 bits (LOW) or 16 bits (BOTH).

The Value column shows Set when sent for these rows, because the PDM fills them in as each message is transmitted.

DBC Export

Export DBC File describes the visible, enabled CAN Outputs for use in a receiving logger, dashboard, ECU or analysis tool.

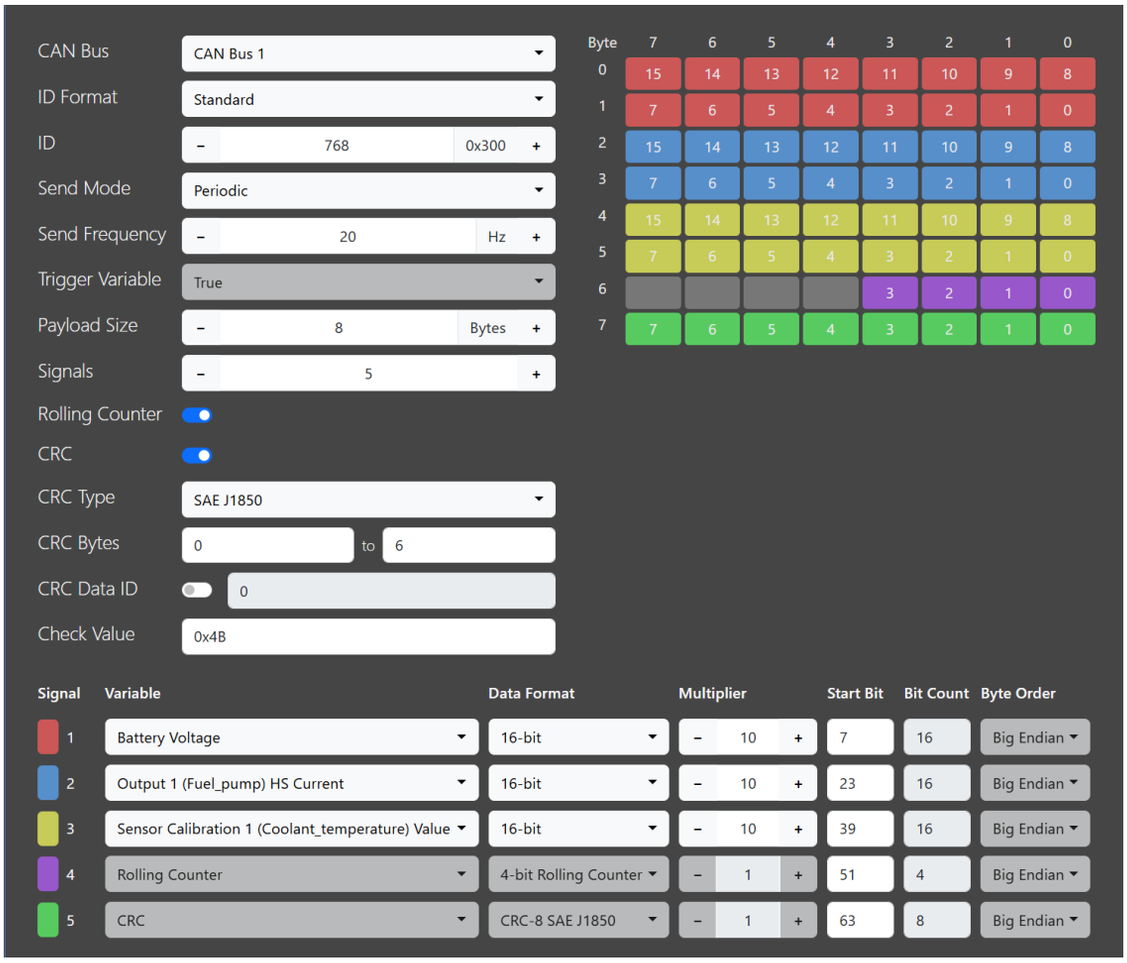


Figure 32 CAN output signal packing.

CAN Output Stream

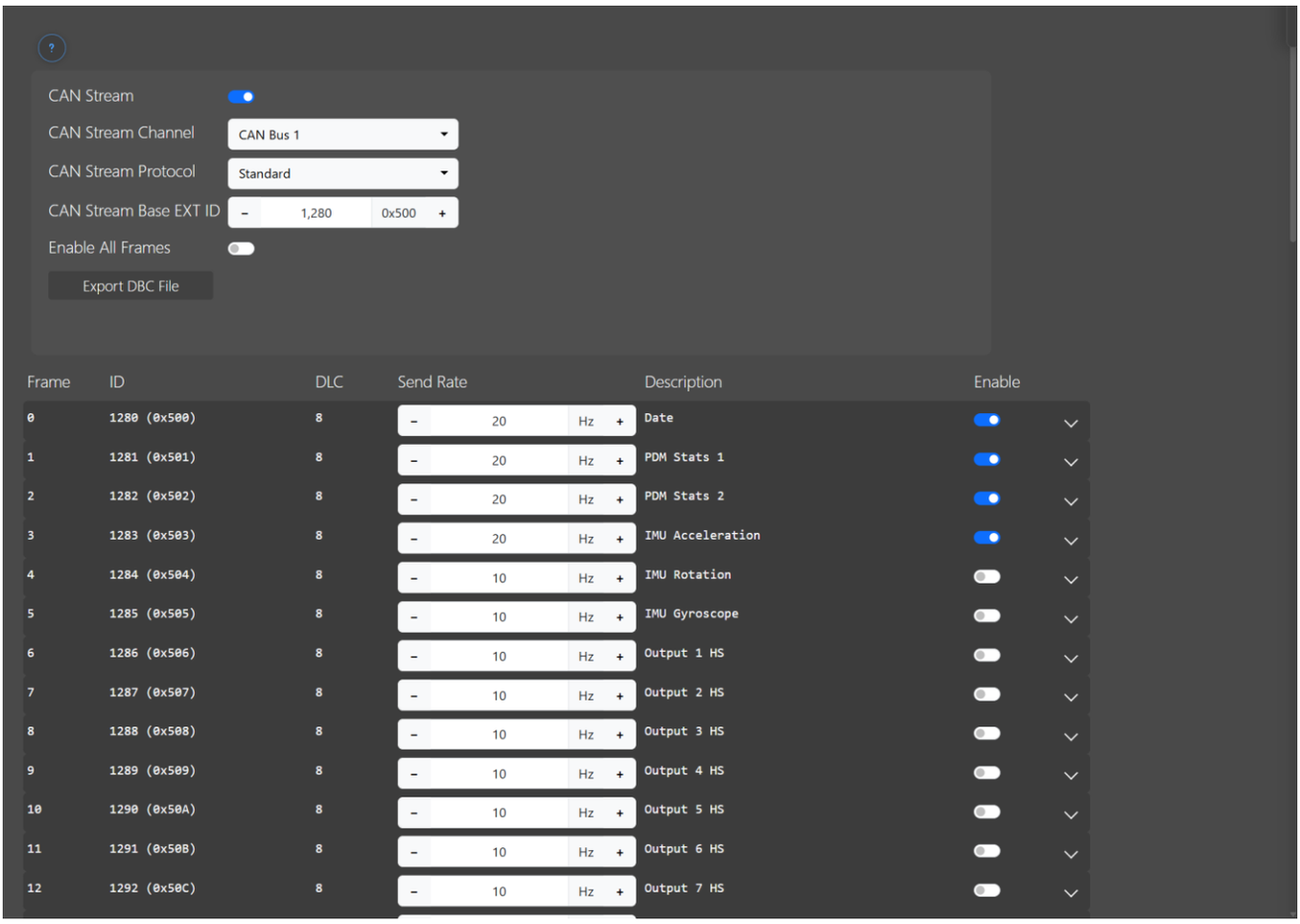


Figure 33 CAN Output Stream.

In addition to the customisable CAN outputs, there is an option to enable the PDM CAN stream. The PDM CAN stream is a collection of PDM variables which have been placed in a user friendly format which can be exported as a DBC file and used by other devices on the bus. There is lots of data which can be sent in the CAN output stream, so you can select the data you want to send out to minimise bus load. You can also vary the send rate of the data for the same effect. An offset can be applied to the CAN stream IDs if needed, to help avoid any CAN ID clashes.

CAN Stream DBC Export

The CAN Stream DBC export function creates a .dbc file describing the currently configured CAN output stream frames. This allows the stream to be imported into third-party CAN analysis or calibration software.

To export a DBC file, navigate to Configuration > CAN Output Stream. Configure the CAN stream as required, then press Export DBC File. Choose a save location and file name when prompted.

The exported DBC file contains the enabled CAN stream frames only. Each frame uses the IDs currently shown in the CAN Output Stream table, and the signal layout is generated from the stream definition in the configurator.

The exported DBC is useful for decoding CAN stream traffic in external CAN tools, sharing a signal definition with other systems, and verifying frame IDs and signal packing during commissioning.

If changes are made to the CAN Output Stream configuration, the DBC file should be exported again so that the external definition matches the current setup.

CAN J1939 Output Stream

The CAN Output Stream can operate in either Standard mode or J1939 Proprietary B mode.

To configure J1939 output stream mode, navigate to Configuration > CAN Output Stream. In CAN Stream Protocol, select J1939 Proprietary B. When this mode is selected, the standard base ID setting is hidden and two J1939 settings are shown: J1939 Priority and J1939 Source Address.

J1939 Priority sets the arbitration priority of the transmitted frames, where 0 is the highest priority and 7 is the lowest. J1939 Source Address sets the source address used in the J1939 frame ID.

In J1939 mode, each CAN stream frame is transmitted as a Proprietary B message using PF = 0xFF, PS = frame number, the configured Source Address, and the configured Priority.

This allows the stream to be used on J1939-based systems while still providing the same preformatted stream data. The calculated frame IDs are shown in the frame list and update automatically as the J1939 settings are changed.

LIN Bus (PDM28 only)

LIN bus is available on the PDM28 only. The PDM15/25/35 does not have a LIN bus port.

LIN Bus is a low cost communication protocol primarily used for interfacing with sensors and actuators. The speed of the bus is low when compared to CAN bus, and the data rate is hence limited. The low speed however means that only a single wire is needed, reducing wiring complexity significantly. Unlike CAN bus, LIN bus is very structured in the way that data is sent and received. There is no bus arbitration, and so each LIN 'node' sends its data in a predetermined window of time to avoid bus conflicts.

The structured nature of LIN bus necessitates the use of an 'LDF' file. An LDF file is analogous to a DBC file. It describes the following: - Which nodes are on the bus - Which signals are used - How the signals are encoded/decoded - How the signals are combined into frames - Which nodes send/receive each frame - What order and time period are each frame sent.

In most cases, the manufacturer of the LIN bus node will have an LDF file made up, which can be imported into the Hardwire Electronics configurator software. To help explain each part, an LDF file for a water pump has been loaded into the configurator software.

General

In the general tab, you can load in an LDF file. It is not completely necessary to use an LDF file, but it is difficult to not use one, as you will need to know the LIN node specific parameters. Click enable to enable LIN bus on the PDM. The status of the LIN bus, and various error counters are listed below. If the checksum, frame overrun or incorrect bytes error counters continue to increase, the LIN bus status flag will turn red - indicating a problem. The corresponding LED on the PDM will also turn red.

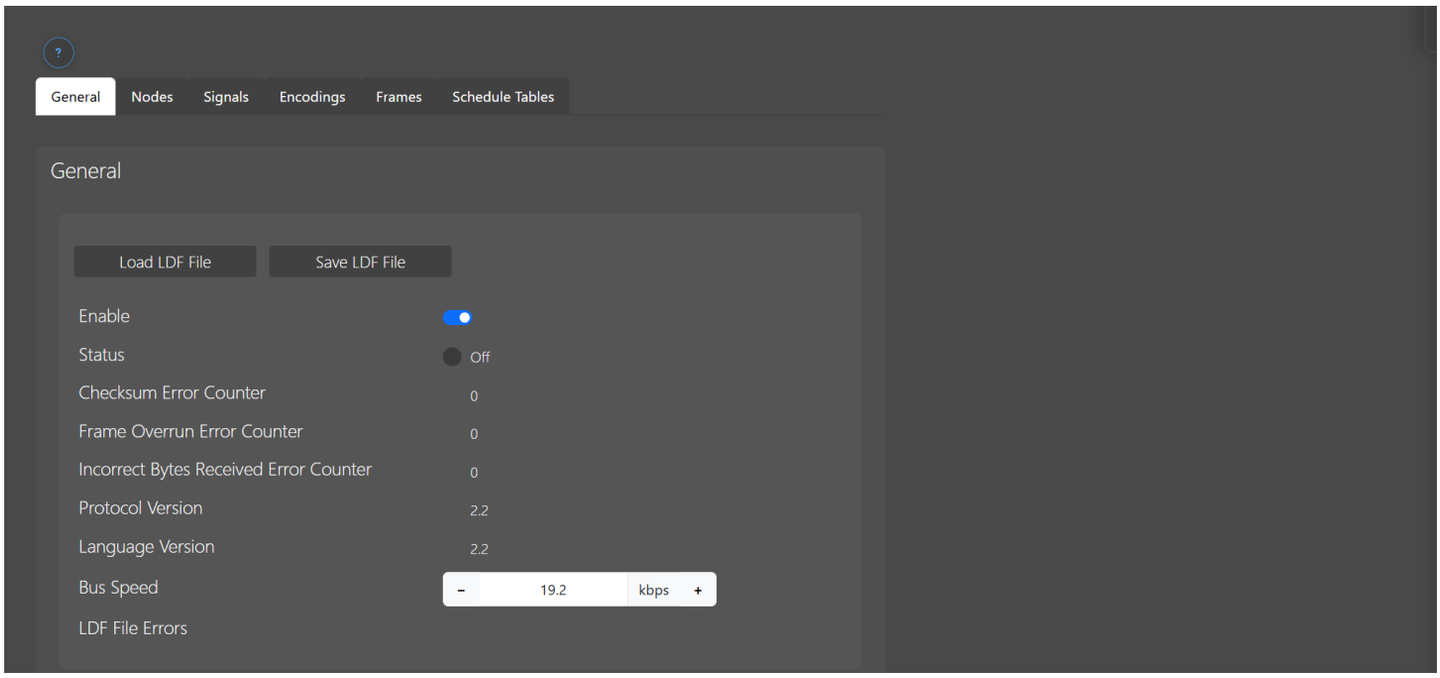


Figure 34 General.

Nodes

In the Nodes tab, we first have the Master node which is the PDM. Below this we have each slave which is on the Bus. Multiple slaves can be on the bus. The parameters for the nodes is taken from the LDF file for the water pump.

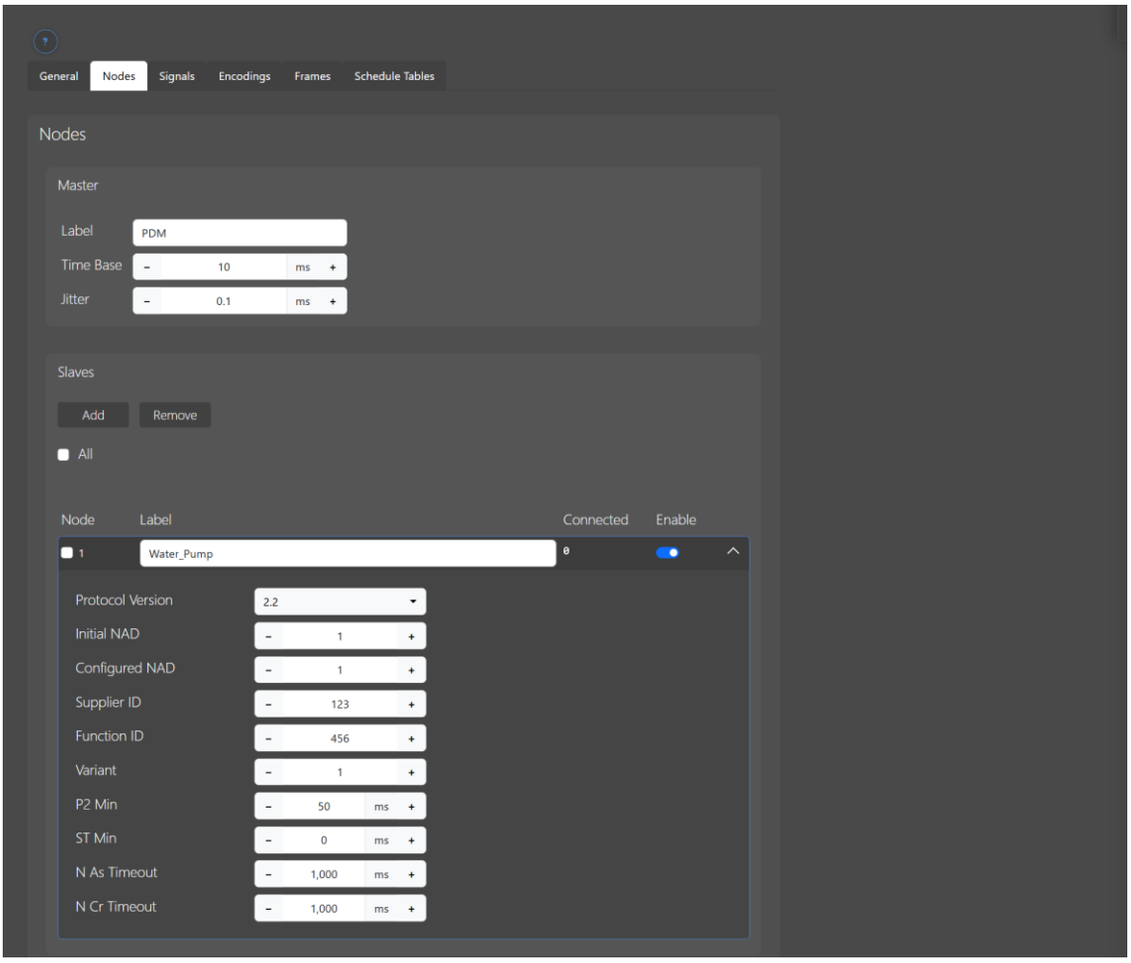


Figure 35 Nodes.

Signals

The signals tab is where the LIN bus signals are defined. The signals can go from the PDM to the pump (to control the pump), or from the pump to the PDM (to get information from the pump). In this example the first signal goes from the pump to the PDM, and contains a communication error status message. The second signal goes from the PDM to the pump, and contains the desired RPM the pump motor should run at. With signals that travel from the PDM to another LIN node, you must select the data which you wish to be sent. In this case, a maths channel which has the desired pump RPM is sent.

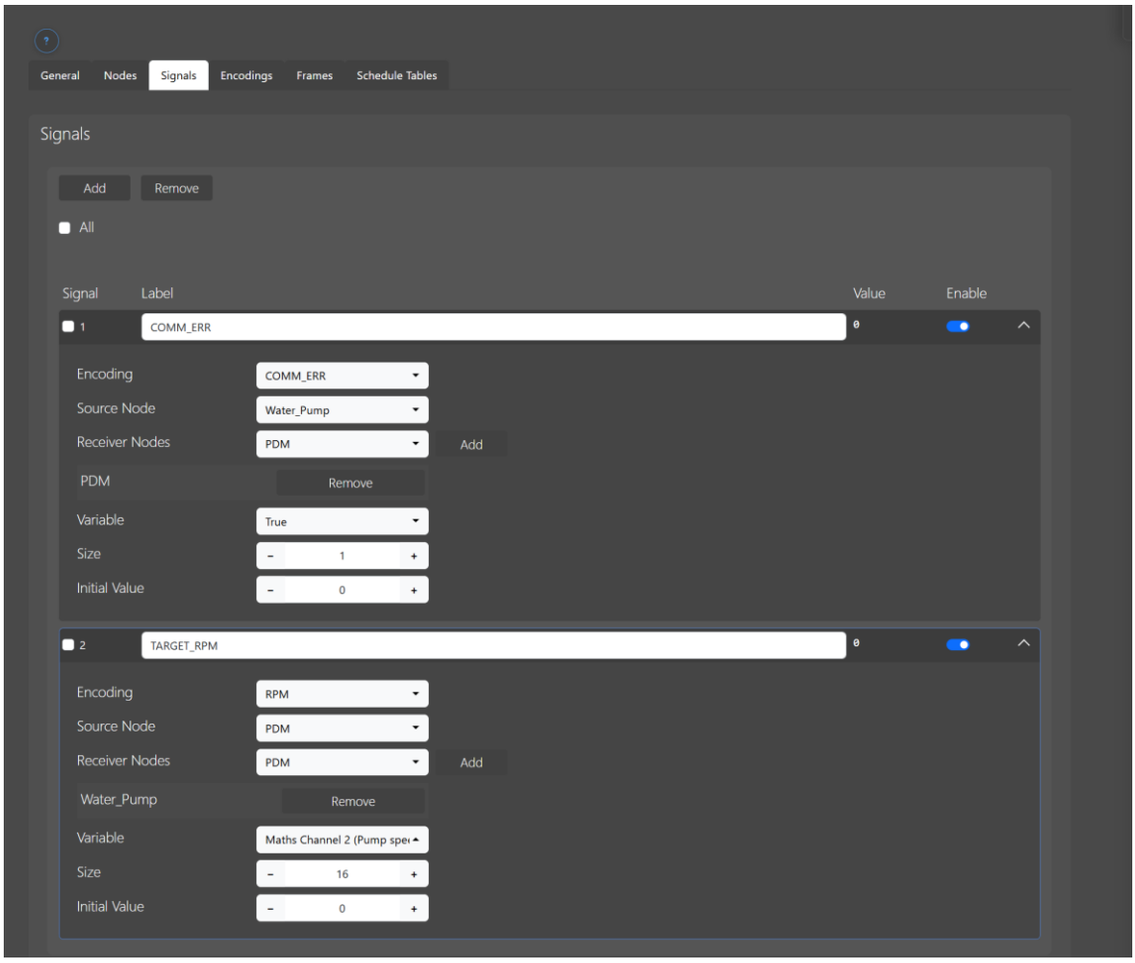


Figure 36 Signals.

Encodings

Each signal sent on the LIN bus has an associated encoding. This tells us how the raw data should be interpreted. Continuing on from the previous section on signals, the encodings for the communication status error, and pump RPM are shown below. The communication status error is a logical value of 1 or 0, representing error or no error respectively. The pump RPM is a physical value with a varying value. It is scaled and offset to get the actual RPM value, and windowed between a minimum and maximum of 0-5000. The raw value which is sent, is the maths channel variable shown in the signals tab.

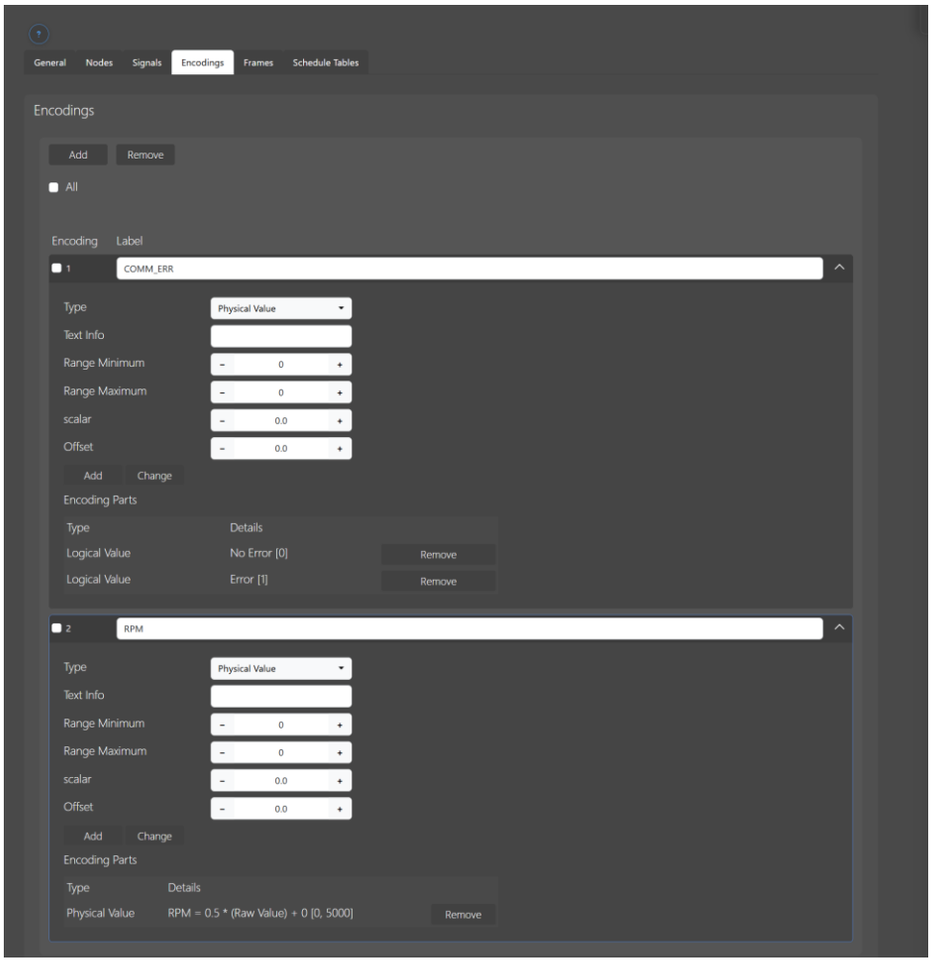


Figure 37 Encodings.

Frames

In this simple example with a signal water pump and a PDM28 on the CAN bus, only two frames are sent. One frame is sent from the PDM to the pump, and the other frame is sent from the pump to the PDM. Each frame is constructed of individual signals with offsets - piled together into the 8 byte LIN bus packet payload. The frames are unconditional frames, as they are sent unconditionally in a repeated fashion according to the schedule table.

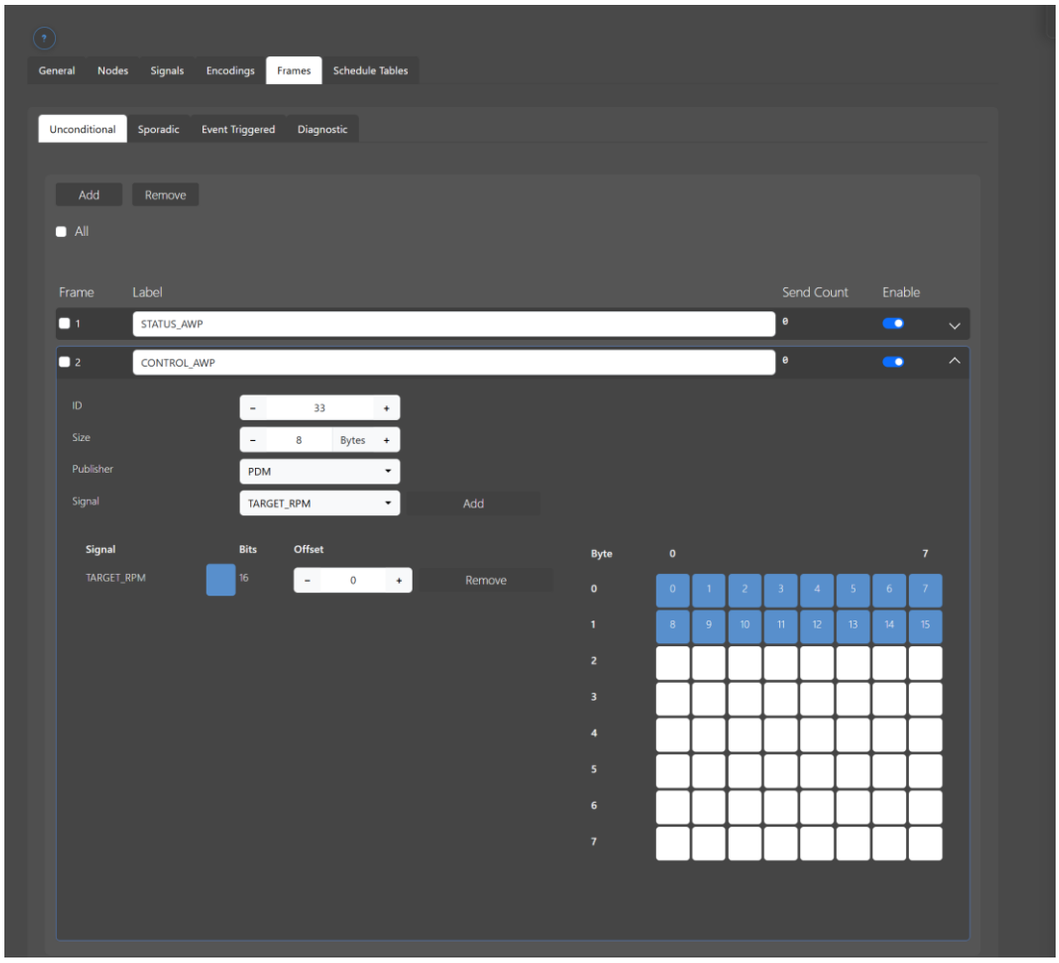


Figure 38 Frames.

Schedule Table

In this example with only two frames sent between the PDM and pump, the schedule table is simple. The schedule table dictates the timing in which each frame is sent. The schedule table runs constantly, in this case with a 15ms delay between each message.

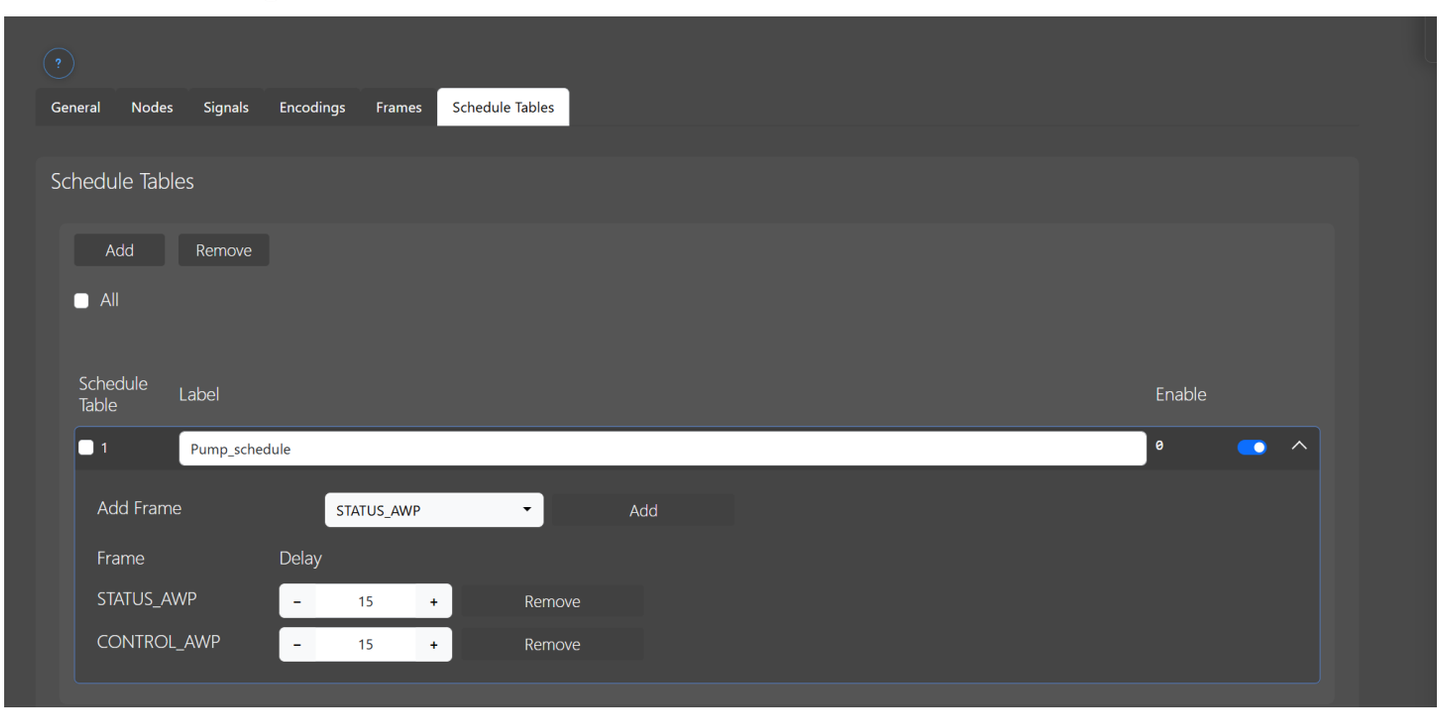


Figure 39 Schedule Table.

Frame Timing

Configure the schedule using supported unconditional frames, and allow enough time for each transfer. Other frame headings shown in the editor do not indicate that those transmission types are supported.

Data Logging

The PDM has 128Mb of on-board flash storage for logging data. This logged data can be downloaded and viewed on the PC to diagnose issues, and ensure that each electrical component in the vehicle is functioning correctly.

To enable data logging on the PDM, navigate to the Configuration tab and then to the Data logging tab. A Logging Group can be added by clicking the 'Add' Button. A Label can then be given to that Logging Group. Click 'Expand' to further configure the Logging Group.

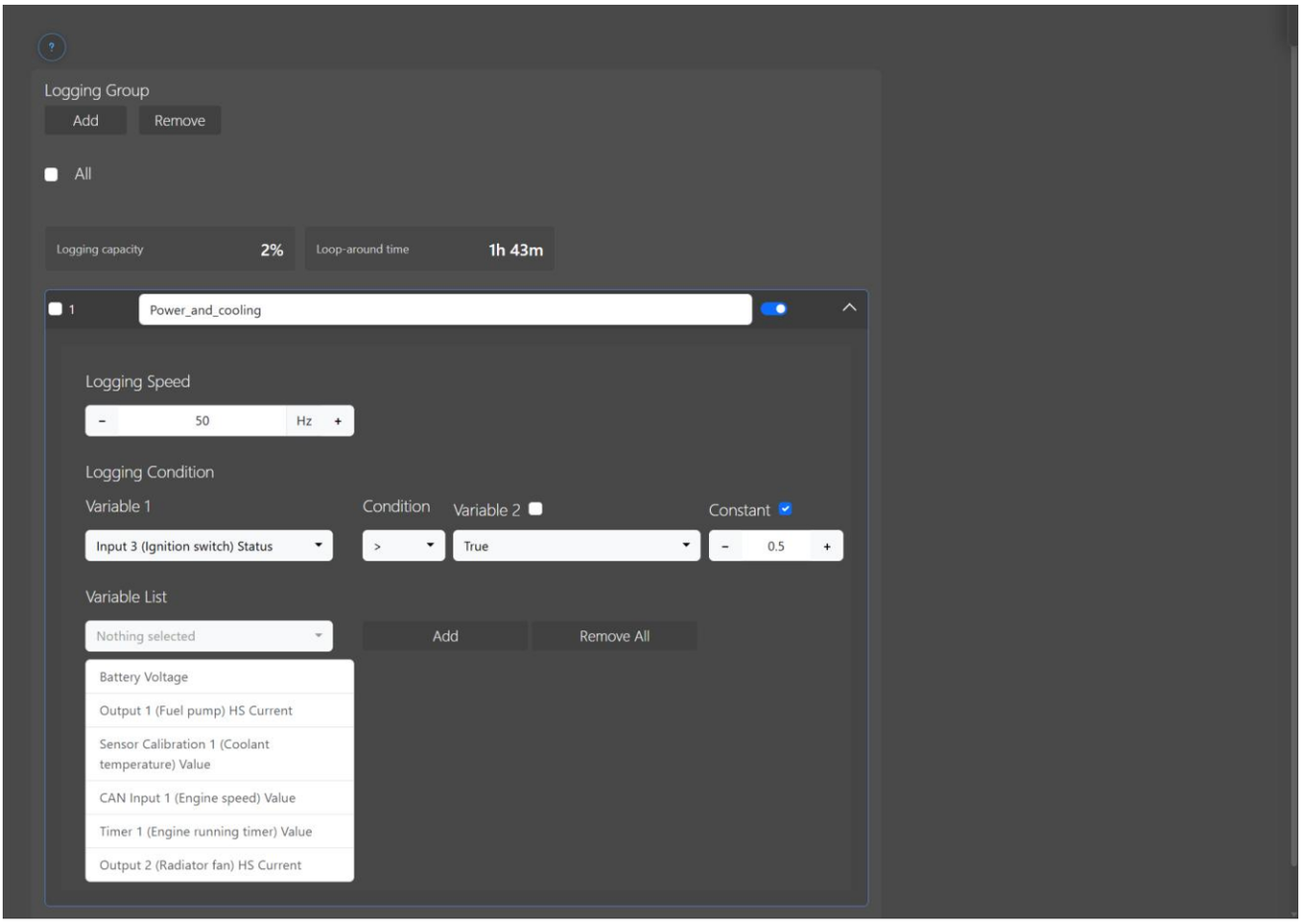


Figure 40 Data Logging.

Each Logging Group is responsible for data logging a collection of variables. These variables are logged at the Logging Speed, which can be changed between 1-1000Hz. The variables are logged whenever the Logging Condition evaluates to True. In this example, the Logging Condition uses the ignition input, so data is logged while the ignition is on. Variables to log can be selected below. Logging more variables at higher speed will consume the data logging memory more quickly. Fast changing variables such as output currents and input voltages should be logged at higher speeds, whereas slow changing variables such as PCB temperature should be logged at low speeds. Each time the PDM is restarted, or a new configuration is sent, a new data log will start. Data logs are numbered in ascending order. When the data logging memory is full, the data log will start to overwrite the oldest data stored in memory.

Configuration Warnings

The Configuration Warnings page provides a quick way to check the current setup for common configura- tion issues before sending it to the device.

To use this feature, navigate to the Configuration page and select the Configuration Warnings tab. Press the Update button to scan the current configuration. The software will then generate a list of warnings and display them in the table.

The warnings list should be reviewed before sending a finished configuration to the device. Warnings highlight settings that may lead to poor performance, unreliable operation, or unexpected behaviour. Correct any unexpected warnings and update the list again.

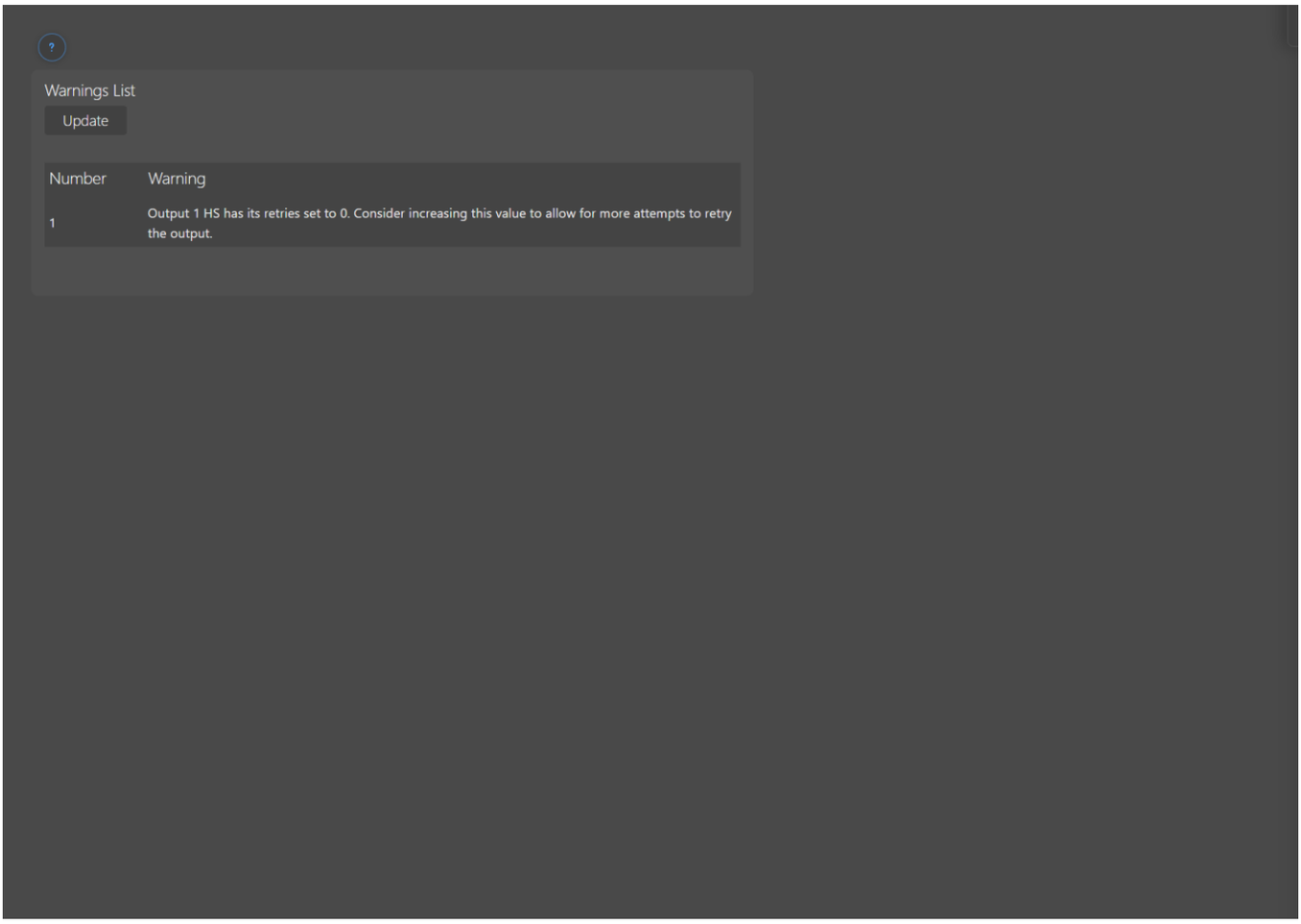


Figure 41 Configuration warnings.

Monitoring

One of the distinct advantages of using a Solid State Power Distribution System over a conventional relay and fuse system, is that the exact state of the electrical system can be monitored and logged in real time– a necessity in high reliability, high performance applications. The Configurator software allows one to monitor the value of each variable in the system, such as the Power Module Input Voltages, Output Currents and more.

Overview

The Overview tab displays live PDM values in panels. The Outputs panel shows output voltage, current, fuse load and state. The Keypads panel shows the configured button labels, icons and live states.

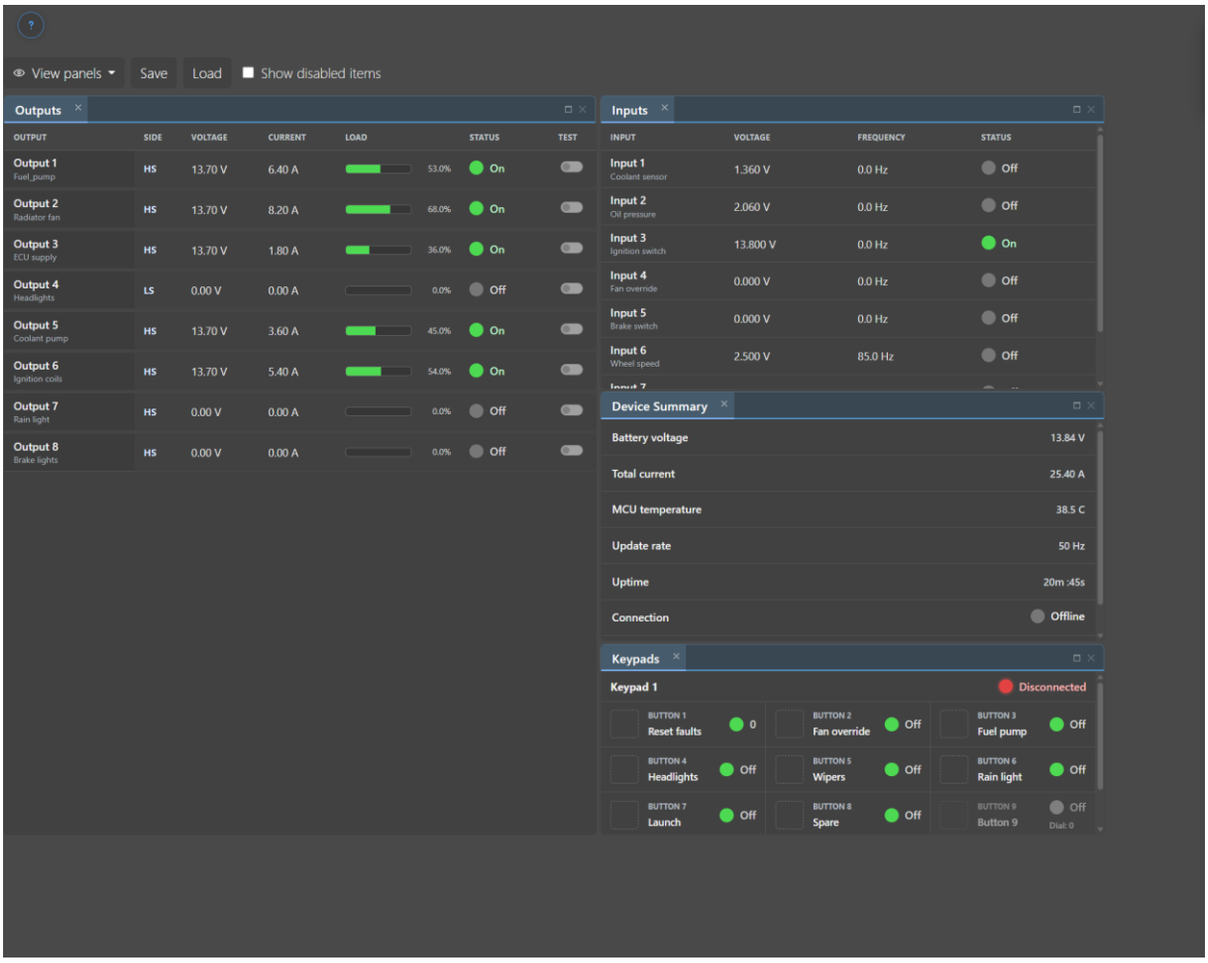


Figure 42 Overview.

View Panels

Open View panels and select the panels to display. The available panels include inputs, outputs, device information, IMU, CAN diagnostics, timers, counters, maths channels and generic functions. Close a panel using the button in its title bar, or reopen it from View panels.

Arrange Panels

Drag a panel title bar to move or dock the panel. Drag the boundaries between panels to resize them. Reset layout restores the Outputs, Inputs, Device Summary and Keypads panels.

Custom Variables

Select Custom from View panels to create a panel containing your chosen PDM variables. Add the required variables, drag their handles to change the order, and use the remove button to remove a variable from the panel.

Overview Graphs

Select Graph from View panels to add a graph panel. Select the variable to plot and the time window to display. Additional graph panels can be added to compare different variables.

Save and Load View

The Save button in Overview stores the panel layout and its selected variables in a file on the PC. Load restores a saved view. These controls change the Overview display; they do not send a configuration to the PDM.

Show Disabled Items

Enable Show disabled items to include disabled channels in the Overview panels. This only changes which channels are displayed; it does not enable them in the PDM configuration.

Graph View

One of the easiest ways to visualise the different variables is with a Live Graph. To add a graph, navigate to the Monitor Tab and to Graph View and click ‘Add.’ This will insert a graph window. To add a variable to the graph, click on the drop-down menu and select an appropriate variable. Then press ‘Add’- the variable will be shown on the graph, as well as in a list below. To remove a variable, double click on the variable in the list. To adjust the amount of data shown on the graph, change the value in the X-Time box. The X-Time is the number of seconds of data shown on the graph. The Y-Min and Y-Max values can be changed to adjust the vertical scale of the graph to better fit the data onto the screen. Alternatively pressing the auto-scale button to the top right of the graph will automatically adjust the scaling of the graph.

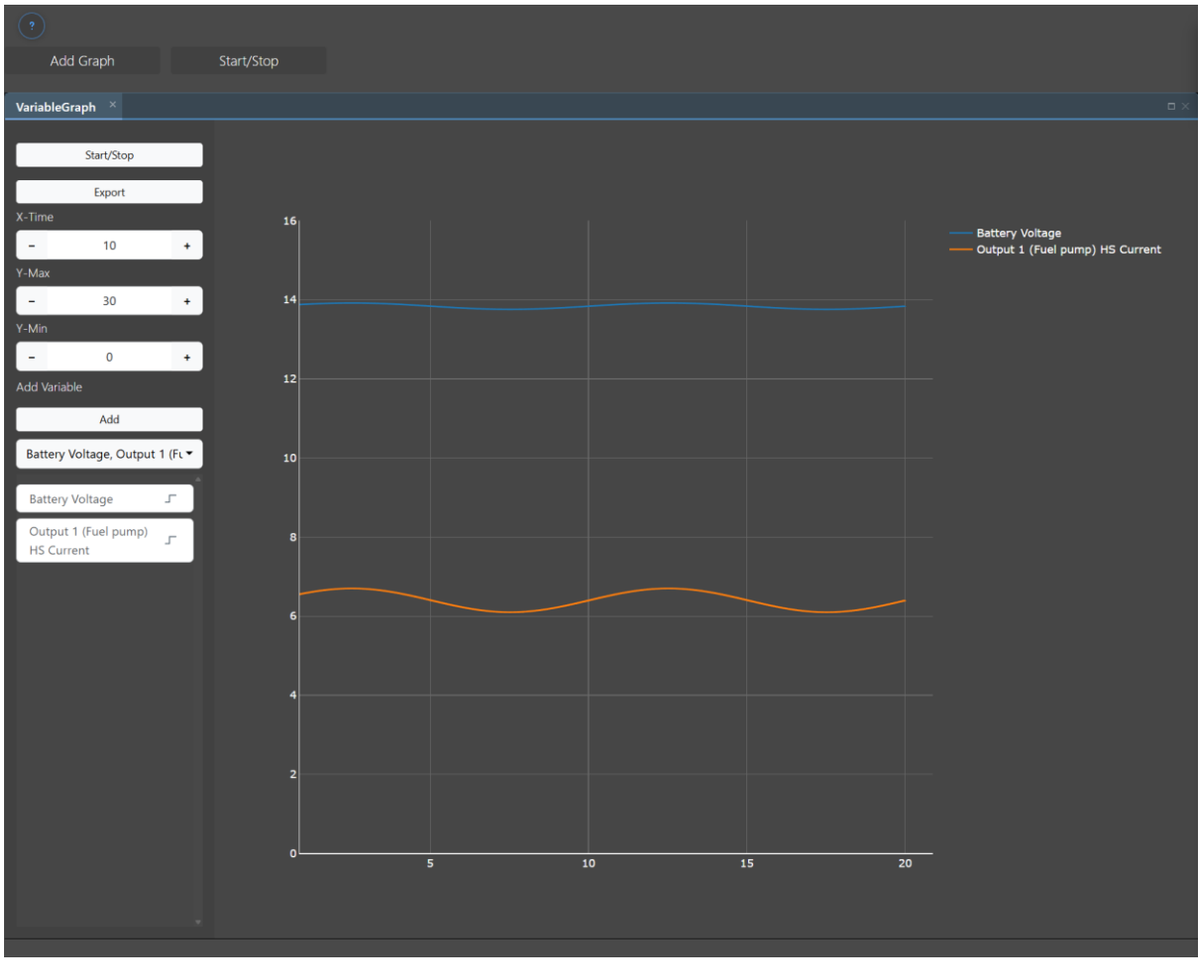


Figure 43 Graph View.

Graph Interpolation

Graph View can interpolate between samples to improve display continuity. Interpolation does not add measurement detail.

Live data gathering from the PDM can be halted at any moment by pressing the ‘Stop’ button at the top of the graph window. With the live graphing of variables stopped, the buttons at the top of the graph may be used to zoom in/out, save images of the graph, or to export the graph as a .csv file for viewing on external software. The graph windows can be resized, docked, and tabbed by clicking and dragging on the top of the graphing window.

Variable View

To view the numerical value of different PDM variables, navigate to the Variable View Tab. To make it easier to look at multiple collections of variables, one can add a variable ‘Group.’ Each group can be given a name. Click expand, and then add variables to the group in the drop down menu. Variable Groups can be saved to a file to make it easier to get the collection of variables you want to look at as quickly as possible. Each individual variable can be given a different colour as the user wishes.

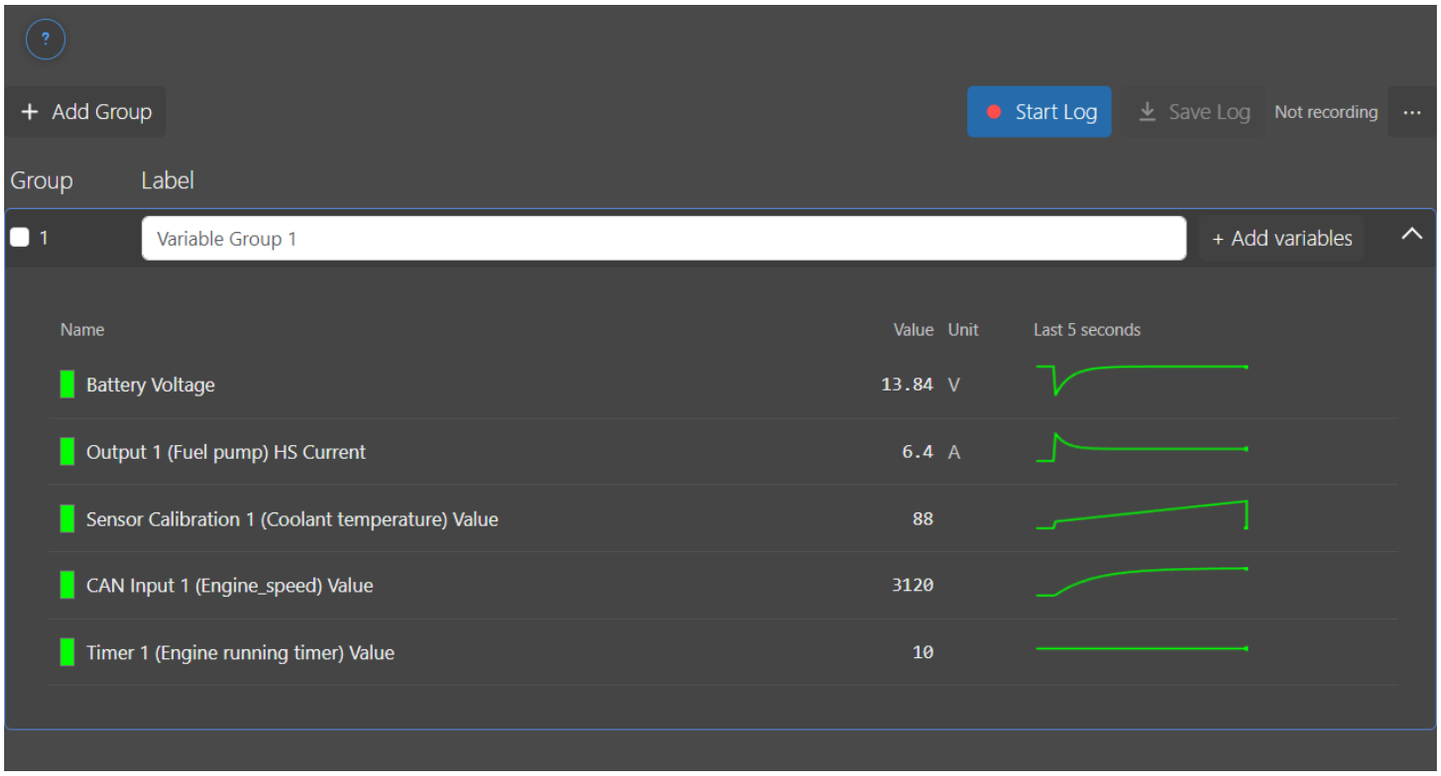


Figure 44 Live variables and recent history.

History Window

Select a 5, 20 or 60 second history window to change the period displayed in the Variable View sparklines.

Variable Logging

Start Log records the selected variables to a CSV file on the PC while the connection is operating. This is separate from the PDM onboard logging.

Capture View

The Capture View allows the user to record a short, high-speed snapshot of selected live variables and display the result on a graph. This is useful for investigating short events that may be difficult to observe in the normal Graph View.

To use the Capture View, navigate to the Monitor page and select the Capture View tab. Add the variables to be captured using the variable selector and Add button. Up to 10 variables can be captured at one time. Variables can be removed by double-clicking them in the capture list, or the entire list can be cleared with Remove All.

Two trigger modes are available: Manual: the capture starts immediately when Run is pressed. Conditional: the capture arms first, then waits for the selected trigger condition to become true before

recording.

When using Conditional trigger mode, select Variable 1, the comparison Condition, and either Variable 2 or a Constant.

The Duration setting defines how long data is recorded for. In Conditional mode, a Pre-Trigger time can also be entered so that data from before the trigger point is included in the capture.

Press Arm or Run to begin. When the capture is complete, the recorded data is plotted automatically. The graph displays time on the X-axis in seconds, with the trigger point used as time zero. The captured data can then be exported as a .csv file using Export CSV.

If no trigger occurs, the capture will time out and reset automatically.

Cancel Capture

Cancel stops an armed capture. In an exported capture CSV file, the trigger point is time zero.

In the example, the trigger is Output 1 (Fuel pump) HS Status Equals True. Set Pre-Trigger Time to 1 s and Post-Trigger Time to 2 s. The graph shows the previous second of battery voltage and pump current, then the start-up current and voltage dip from time zero.

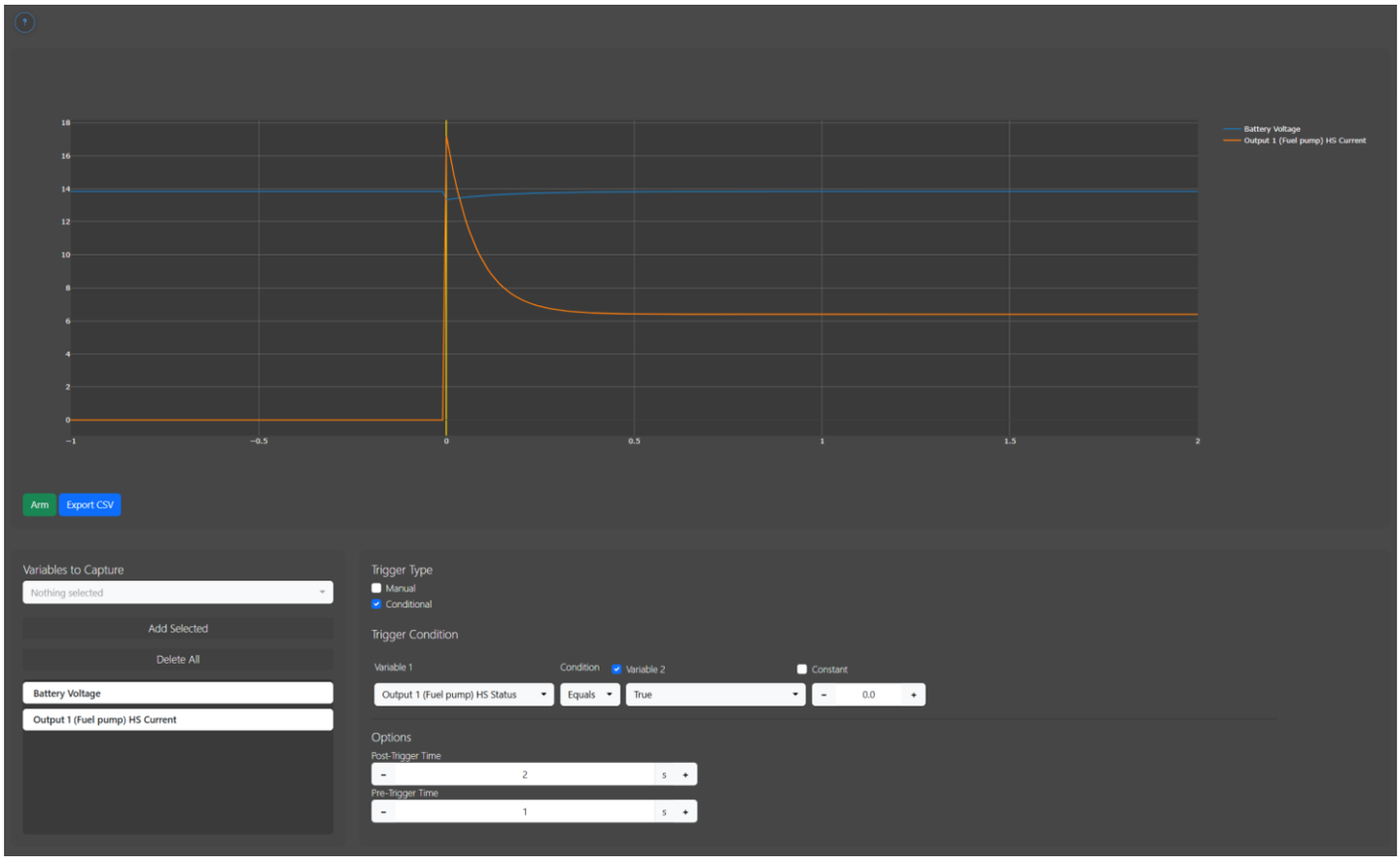


Figure 45 Conditional capture on Output 1.

CAN Bus Analyser

Being able to easily see what CAN bus messages are on the CAN bus is important when configuring a Power Distribution System. This can be difficult to do if CAN filters need to be set up before any CAN messages can be received. Hardwire Electronics have solved this problem by introducing a CAN Bus analyser feature into the software. To start the CAN Bus Analyser, navigate to the monitor tab and to CAN Bus analyser, then click start. The CAN Bus Analyser removes all CAN Bus Filters on the PDM CAN Bus ports, allowing all CAN messages to be received. In the image below, the CAN Bus Analyser is running in Continuous View mode. In Continuous View Mode, the CAN messages are received and shown in order. As more CAN bus messages are received, the display will scroll upwards to show the latest message.

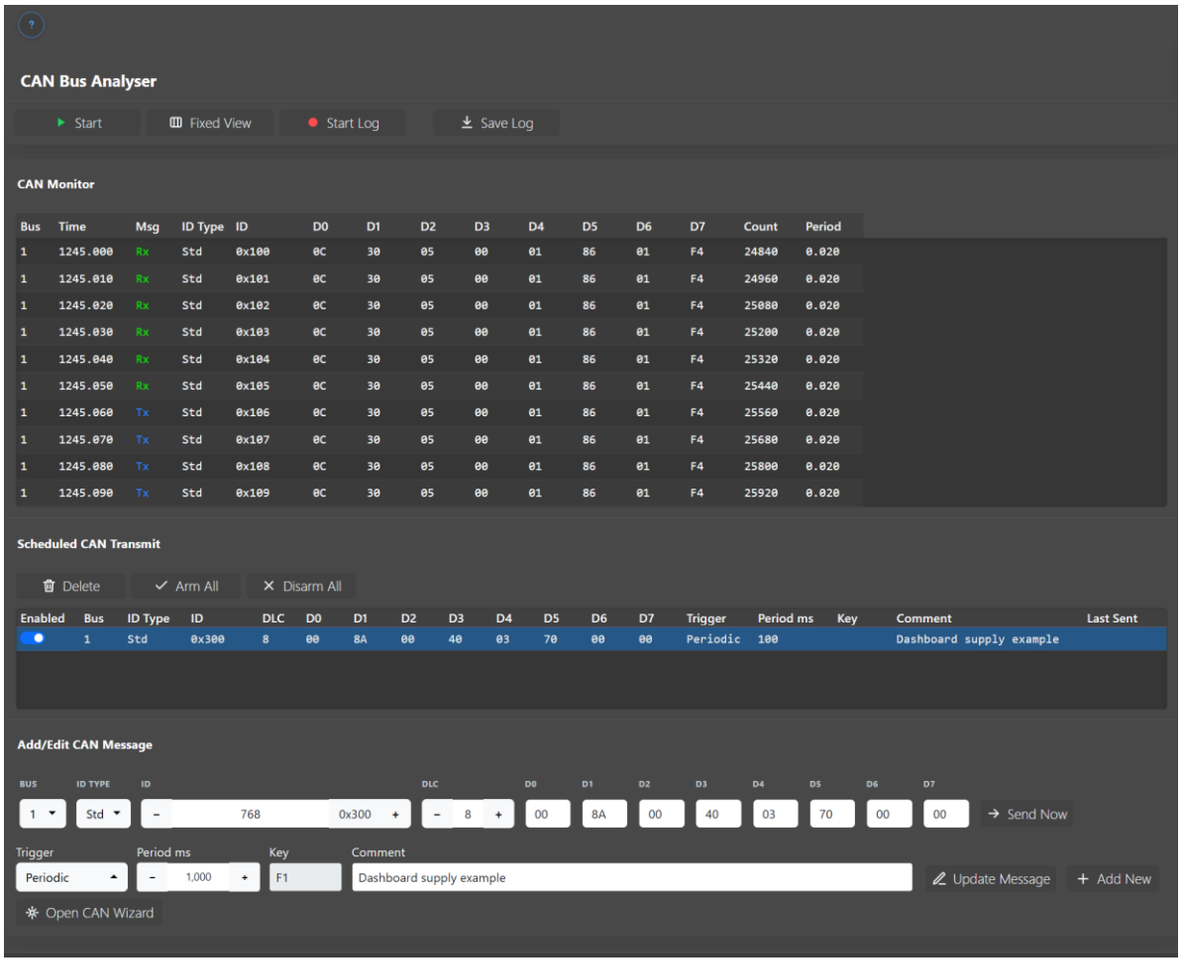


Figure 46 CAN Bus Analyser.

The image below shows the CAN Bus Analyser in Fixed View Mode. In Fixed View Mode, only the latest CAN Bus message with each unique ID is displayed. CAN Bus messages can be sent onto the CAN Bus using the CAN Transmit options at the bottom. The user can select the appropriate CAN Bus to send on, the ID Type and Value, the data length and finally the value of each data byte.

DBC Decoding

Load a DBC file in the CAN Bus Analyser to decode recognised messages into named signals.

Scheduled Transmit

Add or Edit CAN Message sets the frame ID, bus and payload. Scheduled transmit sends the selected frames repeatedly at their configured intervals. Stop scheduled transmissions when testing is complete.

When the CAN Bus Analyser is running, the data can be logged in real time to a CSV file. This data can then be viewed in an external program at a later date. Press Start Log to start a data log. Press Stop Log to stop logging CAN data. Once a data log has been stopped, the user can save the Log to the PC.

CAN Wizard

The CAN Wizard is a quick frame-building tool built into the CAN Analyser. It is intended to simplify the process of sending common CAN configuration commands to Blink Marine Keypads without manually entering all frame fields.

To use the wizard, navigate to Monitor > CAN Analyser and click Open CAN Wizard. Enter the Current Node ID, then select the required Action.

The currently available actions are Set Baud Rate, Set Node ID, and Set Mode.

For Set Baud Rate, select the required bus speed. For Set Node ID, enter the new node ID. For Set Mode, the current implementation supports J1939 to CANopen, and a J1939 Destination Address must also be entered.

The wizard can be used in two ways. Build Frame fills the CAN transmit fields in the analyser without sending the message. Build + Send fills the CAN transmit fields and sends the message immediately.

After building a frame, the CAN Analyser transmit section is populated automatically with the correct ID type, CAN ID, DLC and data bytes. This allows the user to review the message before sending if required.

The wizard is especially useful when changing keypad or node parameters, as it avoids the need to manually construct the CAN command frame each time.

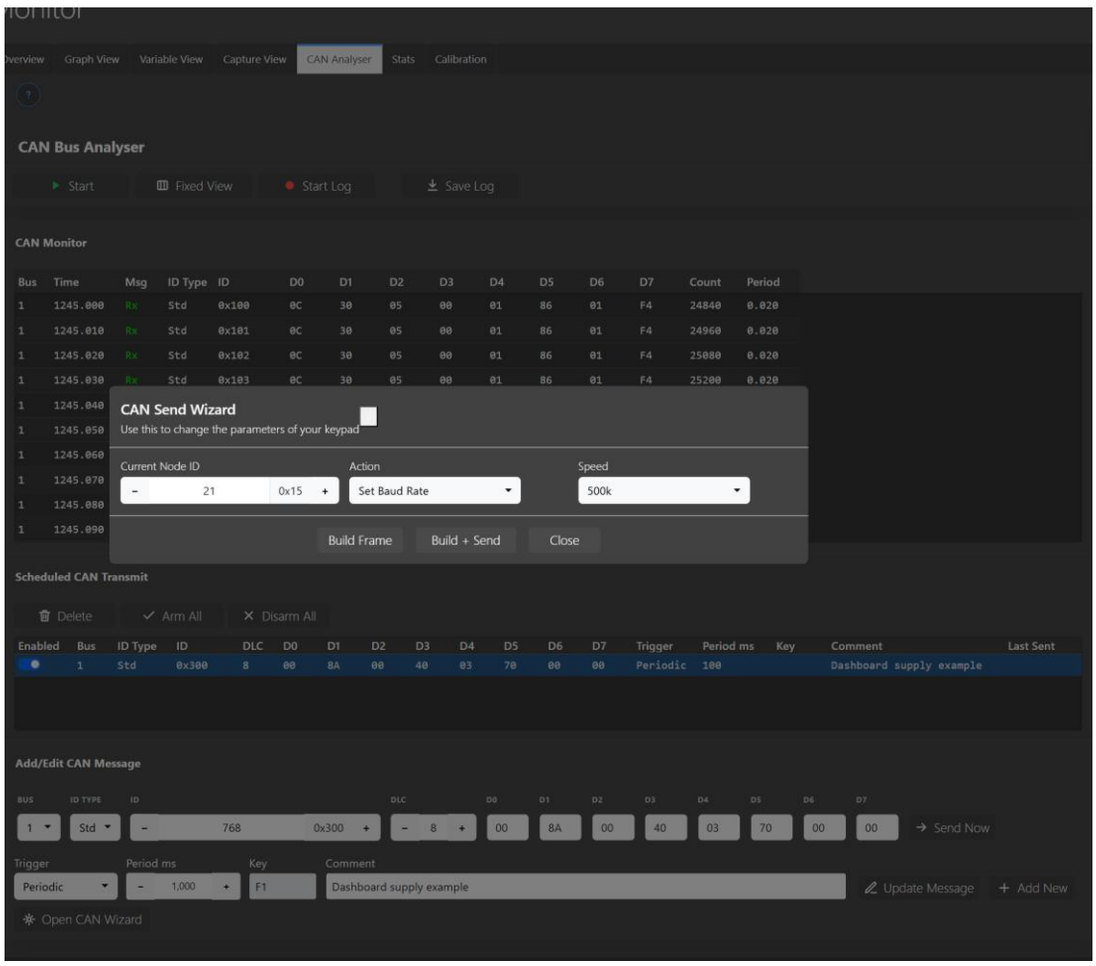


Figure 47 CAN Wizard.

Device Statistics

Stats shows Long Term Device Stats and Short Term Device Stats side by side. These records are maintained on the PDM, including while the configurator is disconnected.

Long term record

The lifetime record includes power-ups, on-time, flash-write statistics and recorded maximum voltages, currents, temperatures and IMU measurements. It has no user reset control.

Short term record

Short-term statistics cover the period since the last reset. Press Reset Short Term Stats and confirm to start a new measurement period. The PDM saves the reset before success is reported. The date comes from the PC and is displayed in local time. Not reset yet means that no user reset has been recorded.

After reset

Maximum values may become non-zero immediately because measurement continues. Power-ups count subsequent starts. Configuration and calibration are not reset. The flash-write figure follows the statistics journal; it is not a count of every configuration or log write.

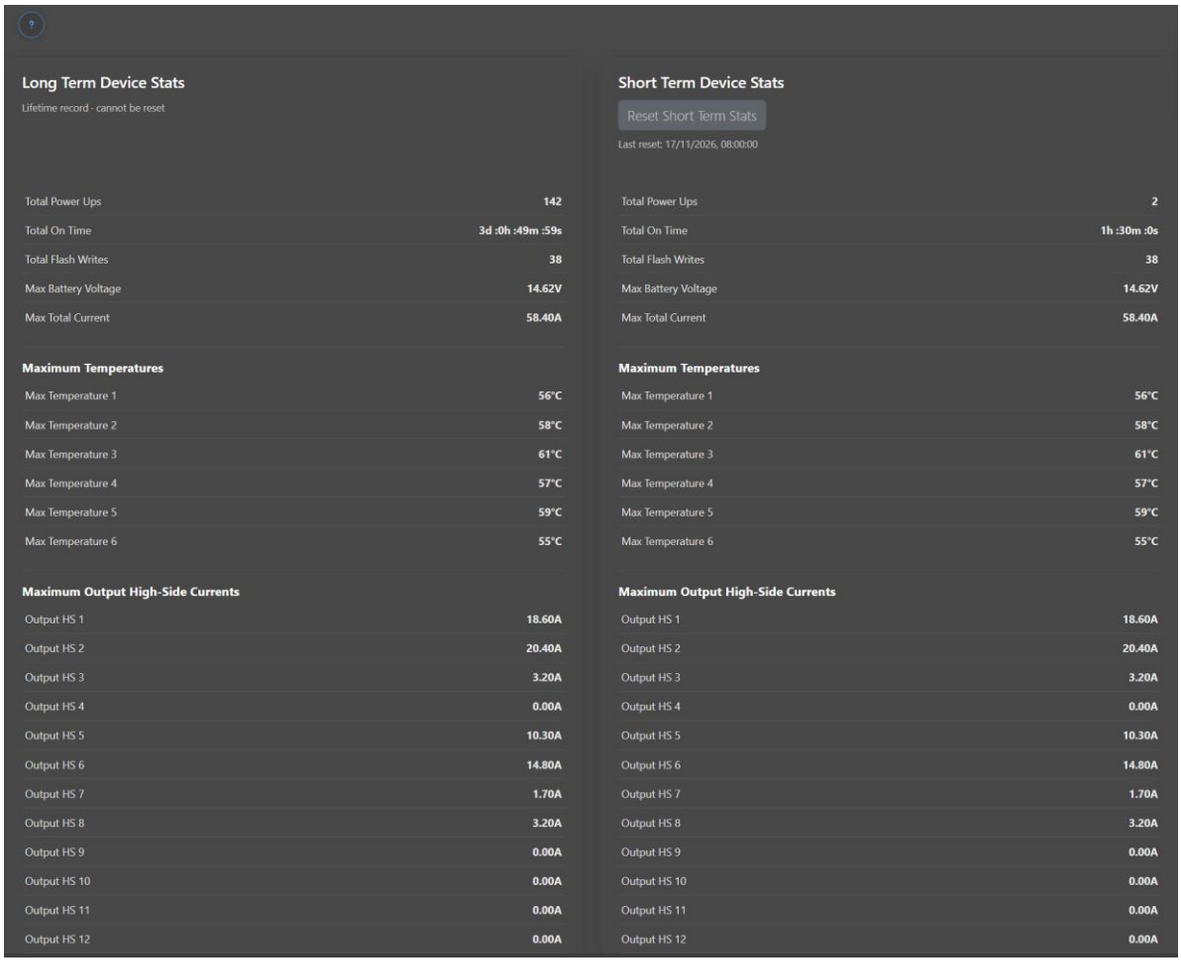


Figure 48 Device Statistics.

Calibration

Only alter PDM calibration if you are trained and authorised to do so. Always save and back up the existing calibration before making changes.

Use Calibration to correct output-current zero, voltage-dependent zero shift, and loaded-current scaling, or to save and restore a device-specific calibration backup.

Save

Saves every calibration point from the connected PDM to a JSON backup, including scaler, offset, high-side voltage compensation, device identity, and calibration date.

Load

Restores a saved calibration JSON to the connected PDM. Files are restricted to the same device ID, model version, and channel layout, and loading overwrites the calibration currently stored on that device.

Temporary setup

Set up output for calibration removes the output function and applies a temporary 15 A fuse setting. This is a service setup, not the final installation configuration. Turn the output off and restore the intended protection and function when finished.

Zero Calibration

Connect to the PDM and save its calibration before making changes. Select the output current point, use Set up output for calibration, disconnect the load and turn the output on. Follow the voltage conditions shown on screen. For a high-side output, capture the low-voltage point at 9-15 V, then raise the supply to 16-32 V and at least 8 V above the low point before completing calibration.

Scaler Calibration

Connect a stable 5-10 A load and an accurate reference meter. Operate at 12-15 V, enter the current measured by the reference meter and calibrate the scaler. Turn the test output off and restore the required output configuration afterwards.

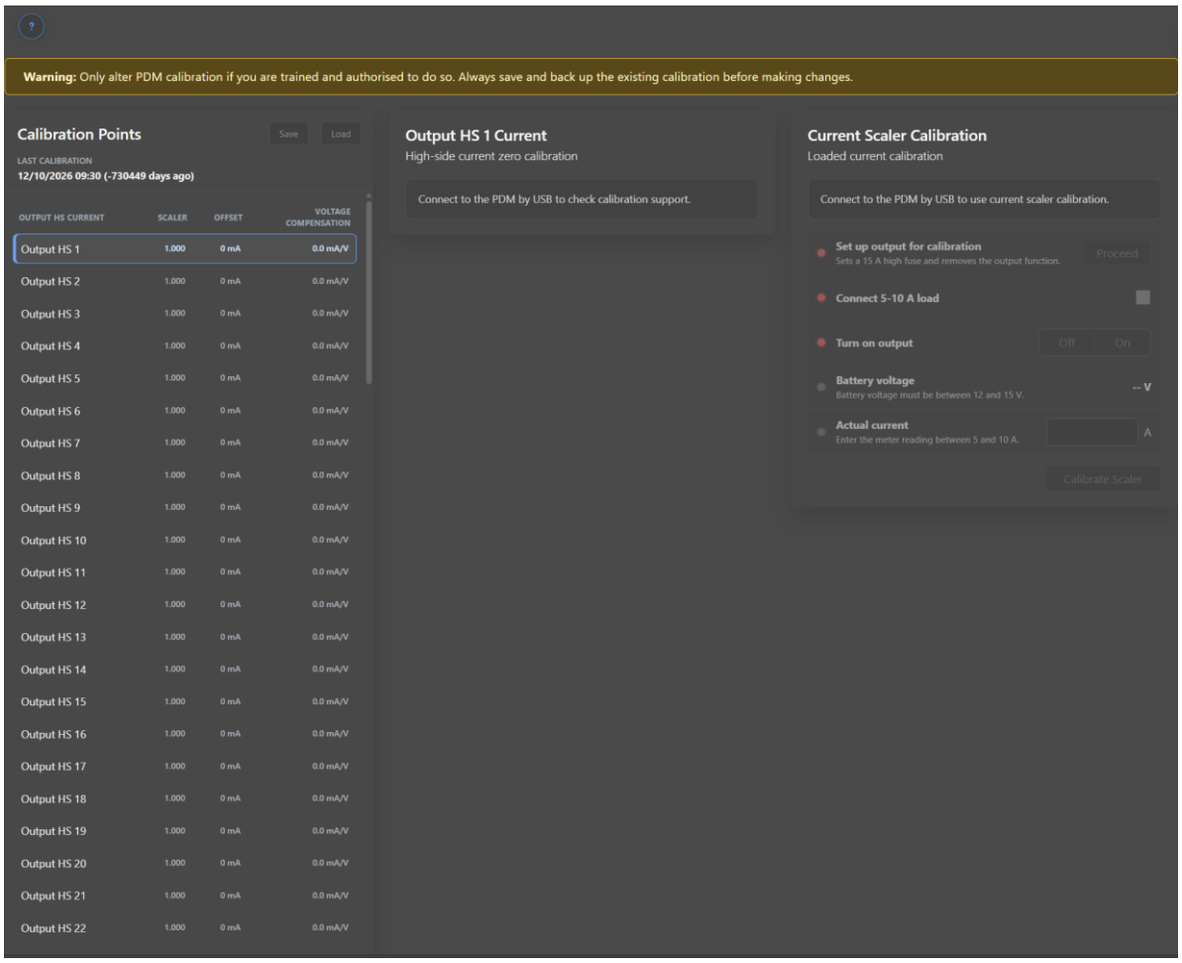


Figure 49 Scaler Calibration.

Logging

Log Library

Open Logging to use the Log Library and graph workspace. Connect to the PDM and use Refresh to retrieve the device log information. Select a recording and choose Download. The progress indicator shows the transfer; once downloaded, the recording is kept in the local library for later viewing.

Resume Download

If a download is interrupted, select the same recording after reconnecting to resume it. The incomplete file is not treated as a complete recording. Resume may be unavailable if the device recording has changed.

Import Log

Use Import to open a native log file or compatible exported CSV. Native files preserve information such as timing and metadata. Deleting a local library copy does not erase the corresponding recording from the PDM.

Graph Workspace

Create graphs in the workspace, then select signals to display. Use channel groups and search to find related values. Graph panels can be resized, docked and arranged. Assign signals to the left or right Y axis to display different scales, and choose elapsed seconds or 24-hour time for the X axis.

Export CSV

Export creates a CSV containing the selected channels and recording time.

Format Log Storage

Use Manage to format the connected PDM log storage. Formatting deletes the recordings on the device; download and export anything needed first.

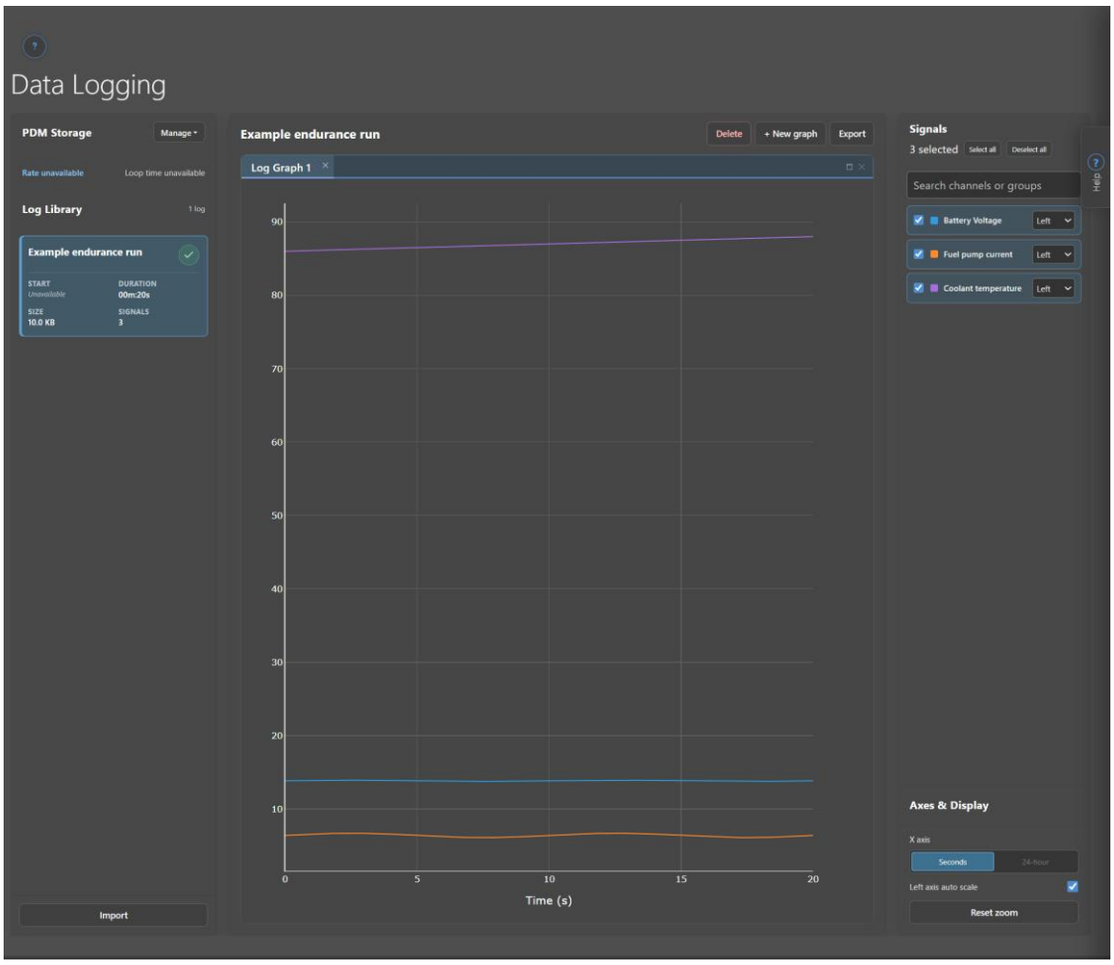


Figure 50 Log Library and graph workspace.

Firmware Updates

Firmware updates allow the user to take advantage of the newest PDM features. Connect the PDM to the PC by USB before starting the update.

PDM Firmware Update

Firmware Selection

Connect by USB and open Update. Check the connected model and installed firmware, then select the matching image in Compatible firmware. The Selected firmware field shows the version to be installed. Choose Update firmware or Reinstall firmware as displayed, and maintain power and USB throughout programming.



Figure 51 Firmware Selection.

Update Progress

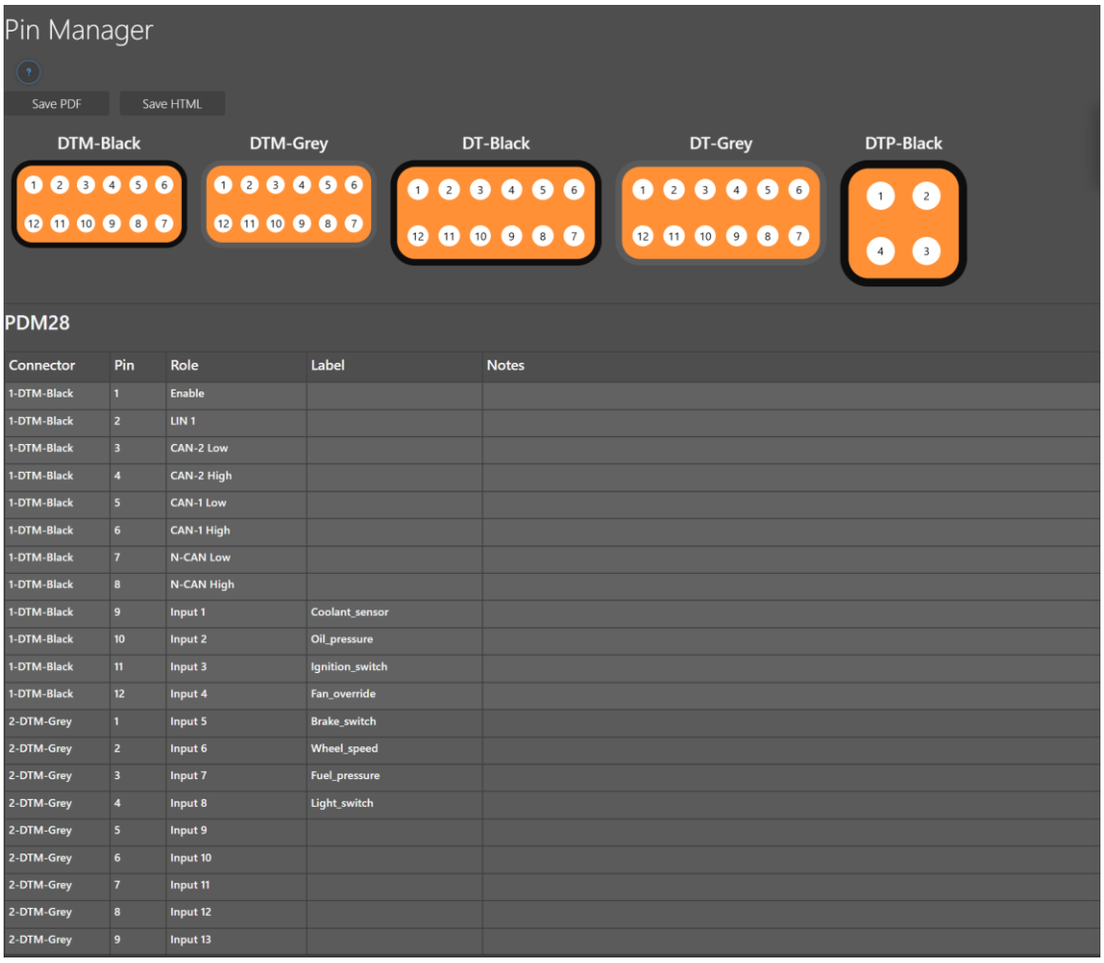
Wait for programming, finalisation, restart and verified reconnection.

Configuration Backup

Before changing firmware, retrieve and save the existing configuration with its matching Configurator and save important logs and calibration data. A firmware upgrade may initialise storage when the previous format cannot be safely identified. Use the saved configuration with the new matching Configurator, review its conversion and warnings, and send it before returning the vehicle to service.

Pin Manager

Pin Manager shows the connector drawings and pin assignments for the selected PDM model. Click a pin or table row to highlight the corresponding entry. Labels follow the input and output configuration. Check the connector orientation and any shared input/output function against the actual unit before wiring.



Connector	Pin	Role	Label	Notes
1-DTM-Black	1	Enable		
1-DTM-Black	2	LIN 1		
1-DTM-Black	3	CAN-2 Low		
1-DTM-Black	4	CAN-2 High		
1-DTM-Black	5	CAN-1 Low		
1-DTM-Black	6	CAN-1 High		
1-DTM-Black	7	N-CAN Low		
1-DTM-Black	8	N-CAN High		
1-DTM-Black	9	Input 1	Coolant_sensor	
1-DTM-Black	10	Input 2	Oil_pressure	
1-DTM-Black	11	Input 3	Ignition_switch	
1-DTM-Black	12	Input 4	Fan_override	
2-DTM-Grey	1	Input 5	Brake_switch	
2-DTM-Grey	2	Input 6	Wheel_speed	
2-DTM-Grey	3	Input 7	Fuel_pressure	
2-DTM-Grey	4	Input 8	Light_switch	
2-DTM-Grey	5	Input 9		
2-DTM-Grey	6	Input 10		
2-DTM-Grey	7	Input 11		
2-DTM-Grey	8	Input 12		
2-DTM-Grey	9	Input 13		

Figure 52 Pin Manager.

Pin Notes

Enter notes for wire colours, circuit names and installation details. Save the HWPDM configuration to retain these notes.

Export Pin Report

Save PDF produces a printable pin report, and Save HTML produces an interactive reference. The exports include the configuration name, date, connector roles, labels and notes. Changes made to an exported reference do not change the configuration.

Loom View

Loom View opens an exported wiring-documentation package inside the configurator. Loom3D is used to create and edit the wiring project; it does not need to be installed simply to read a compatible .loomview package.

Opening a package

Select Loom View in the sidebar and click Open a Loom View file. Choose a .loomview file exported by Loom3D. After loading, use Open Loom View to choose another package. The package determines which schematic, harness, parts, connection and documentation views are available.

Inspecting wiring

Use the package viewer to select components, pins, wires and the available documentation pages. The configurator view is read-only. To alter routing, parts or connections, edit the source project in Loom 3D and export a new package.

Live diagnostics

While connected, supported package mappings can display PDM input and output states, currents and voltages. Check the connected device and package mapping before interpreting the overlay. On PDM15/25/35, an output voltage overlay is inferred from battery voltage and output state; those models do not measure each output pin voltage.

Connection loss

The live overlay is marked unavailable when current telemetry is lost. The wiring document remains readable. A coloured wire or old value is not proof that a circuit is electrically isolated.

Examples

This section aims to provide a number of examples of how the Power Distribution System can be set up. The examples assume that the PDM is connected and working properly.

1. Basic Input Setup

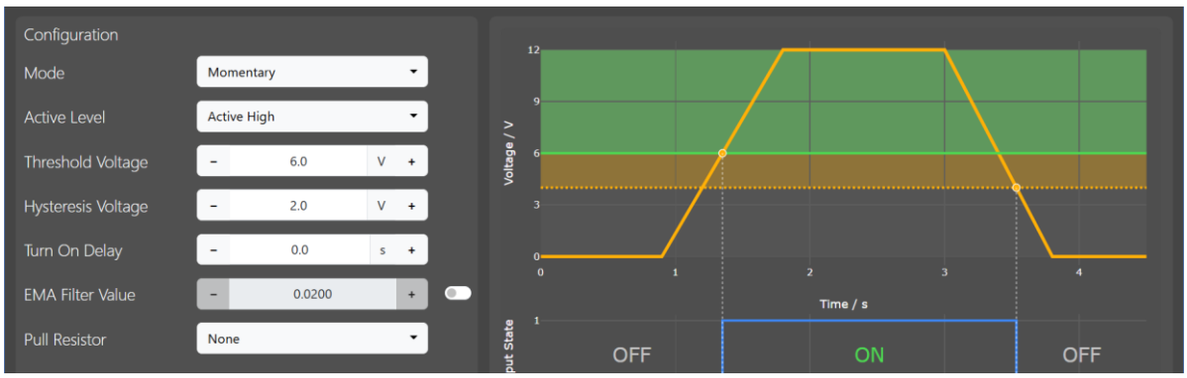


Figure 53 1. Basic Input Setup.

This example shows a basic setup for a PDM Input, so that the input triggers when 12V is applied. This configuration does not use a Pull-down resistor, but if you are using a push button switch to trigger the input, then it is recommended to select the pull down resistor from the menu.

From the graph below, one can see that the input triggers when the input voltage rises above 6V. With a Hysteresis voltage of 2V, the input does not turn off until the voltage drops below 4V (6V-2V). The Hysteresis voltage can help to eliminate false triggering of the input if the signal is noisy.

2. ECU Triggered Input

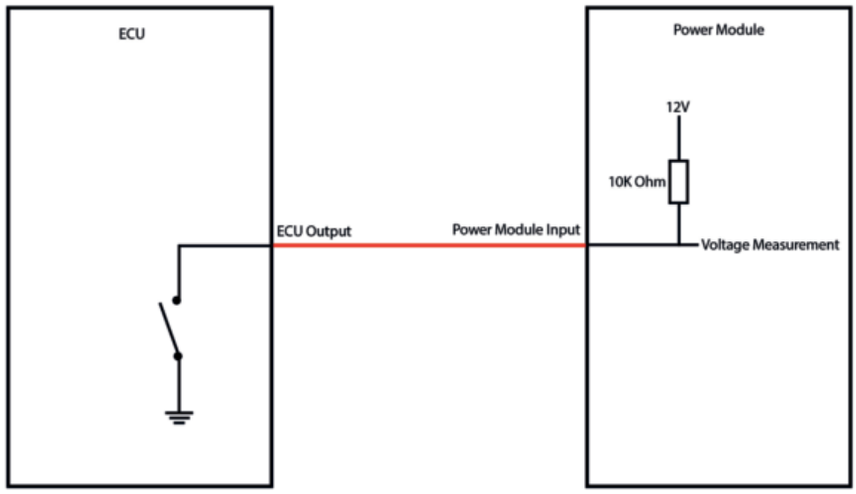


Figure 54 2. ECU Triggered Input.

This example demonstrates the typical procedure to have an ECU output trigger a PDM input. Most ECU outputs are low-sided, meaning that they switch ground onto the output. When the output is not triggered, this leaves the output 'floating,' meaning that the voltage on the output is undefined. If this is tied to a PDM input, the voltage measurement in this case can be sporadic. To combat this problem, it is important to enable the Pull-up resistor on the PDM input. This means that when the

ECU output is not triggered, the voltage on the output is 'pulled' up to 12V, or the voltage powering the PDM. Some ECUs have a Pull-up resistor internally on the output, but many do not. It is therefore recommended to always use the PDM Pull-up resistor when interfacing with Low Sided ECU Outputs.

The PDM Input is configured to be active low, so that when the ECU output is driven to ground, the input on the PDM triggers. The Threshold Voltage is set to 6V, this is half of the vehicle battery voltage. The Hysteresis voltage is set to 2V, but in most cases this will not be

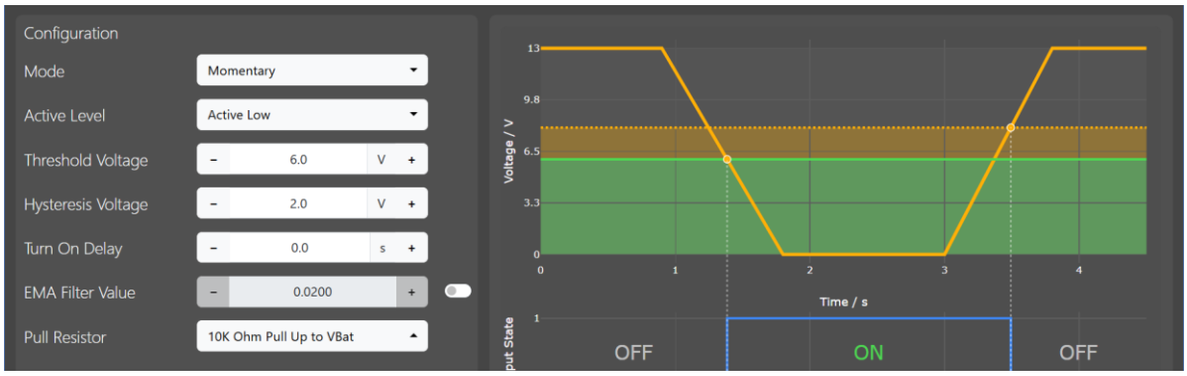


Figure 55 2. ECU Triggered Input.

necessary with ECU outputs as they are hard on/off. Finally, the Pull-up resistor is selected as discussed before. If a High-sided ECU output is used, then everything can be kept the same except from the Active Level and Pull resistor. They should be set to Active High and Pull-Down in this case.

Below is the voltage on the input as the ECU Low Side output triggers.

3. Latching Input Setup

This example shows a set up for an input configured in Latching Mode. In Latching Mode, the input status toggles everytime the input is triggered. In this case, the threshold voltage is set to 6V and the input is set to

active high, so each time the voltage rises above 6V, the input toggles state. This setup is useful for latching an input on when zpressing a momentary button switch.

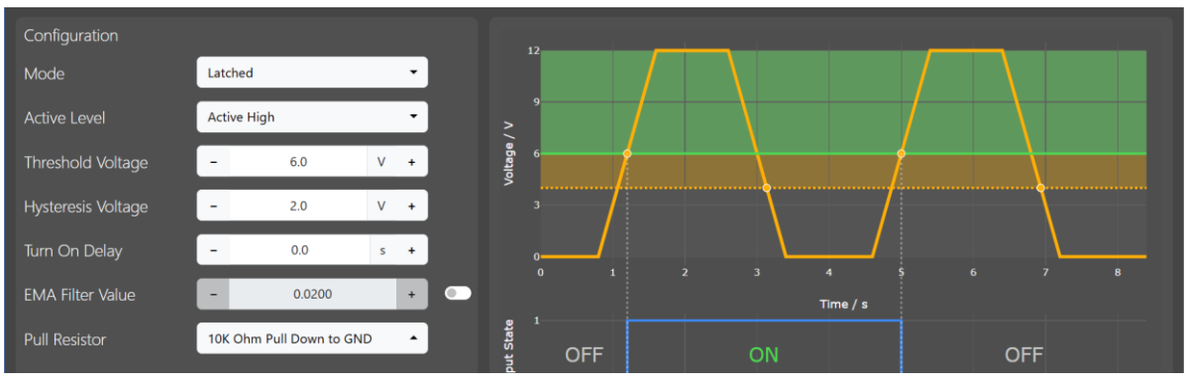


Figure 56 3. Latching Input Setup.

4. Basic Output Setup

This example will show how to set up a simple output that turns on when a button on a CAN bus Keypad is pressed. This method is also applicable for switching an output when an input is triggered. This example is based on the configuration of a typical Radiator Fan with the following parameters.

Nominal Current Draw	**17A**
Start Up Current Peak	**36A**

A Radiator Fan is a mechanical object which has inertia to overcome. On start-up, the fan motor has to accelerate and overcome this inertia to reach full speed. This acceleration of the fan motor results in a high start-up current, which then settles down to normal operating current once the motor is at full speed.

The configuration settings of the PDM Output are shown in the adjacent image. The High Fuse, Peak Fuse, and Peak Fuse Time have been carefully selected based on the current draw characteristics of the Fan. The Peak Fuse is set to 40A which is 4A above the peak current draw of the Fan to allow some room for error. The High Fuse has been set to 20A which is 3A above the nominal current draw of the Fan. The Peak Fuse Time has been set to 4 seconds. This is to allow time for the current draw to come down to below the High Fuse threshold. When setting up an output, it may take some time to tune these values in. Effects such as temperature and component age/condition can all affect the current draw. It is therefore necessary to add approximately 10-20% margin on the current trip limits, as seen in this example.

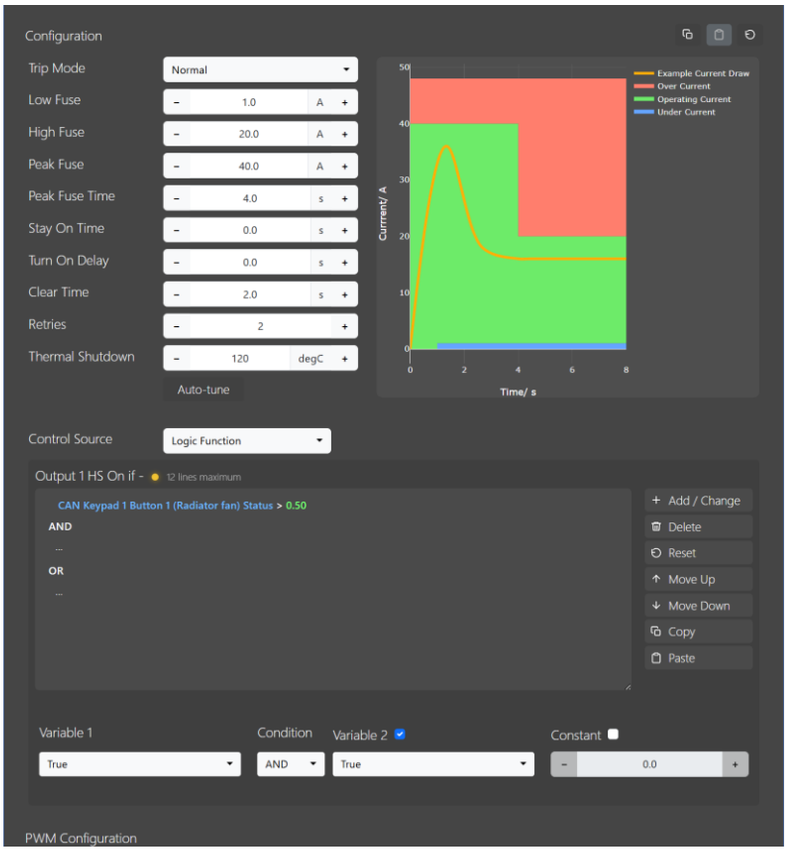


Figure 57 4. Basic Output Setup.

Indicator/Turn Signal Setup

To set up left and right Indicators with Hazard light functionality, first add a timer to the configuration and set the On and Off time to 400ms. Configure the CAN Bus Keypad or Inputs so that the Indicator Button and Hazard Button both latch on when the button is pressed. The Output for the Indicator can then be configured as shown here -

Timer 1 simply alternates between the On and Off state every 400ms. The status of the Timer can be AND'd with the status of the Indicator button. When both the Timer and Button status is true, the output will be on, and when either or both are false, the output will be off. The same can be repeated for the other indicator output.

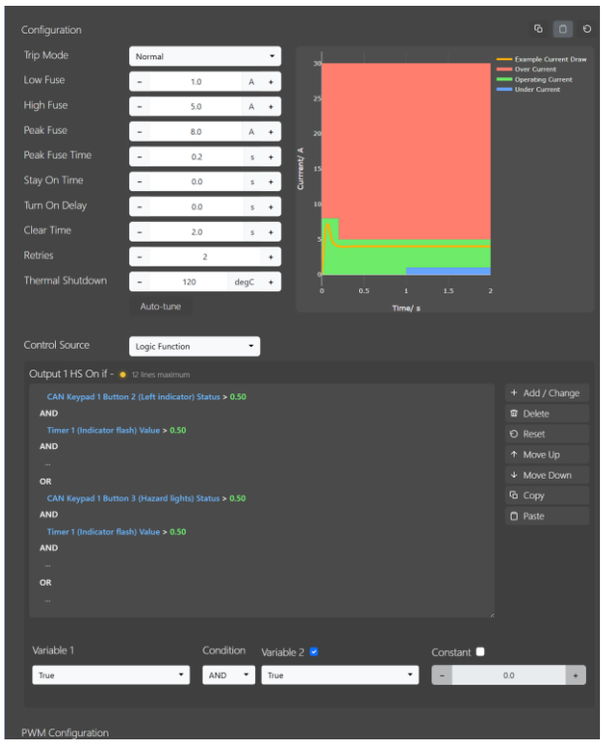


Figure 58 Indicator/Turn Signal Setup.

PWM Fan Control Setup

In this example, an output has been configured to control a radiator fan using PWM. For most applications, PWM is not required for radiator fans, but for users that need finer control, this is a good method to achieve that. The first thing to note is that the output function is set so that the output will turn on if either CAN Keypad 1 Button 1 Status is equal to true, or if the coolant temperature (received via CAN) is above 80 degrees C. If the output function does not evaluate to True, then the output will not turn on, no matter the PWM mapping.

Enable PWM Mapping and select the calibrated coolant temperature as the mapped variable. The table in the example gives 50% duty at 87 °C and 100% duty at 94 °C. The fan output must also satisfy the output function before it can run.

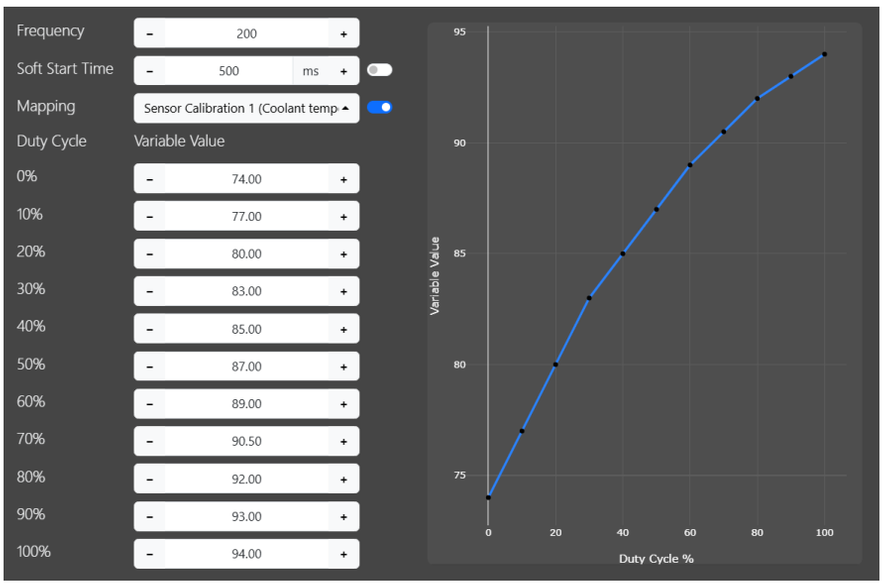


Figure 59 Radiator fan PWM mapping used in the example.

Script example for cooling demand

The following script converts a calibrated coolant temperature to a demand between 0 and 100. At 80 degrees C the demand is zero; at 100 degrees C it is 100. Select the actual calibrated-temperature variable from autocomplete in

place of the example name. The exported demand can then be selected as a PWM source. Configure the output function separately so that the output is enabled when cooling is required.

```
export let fanRequest = 0;

function setup() {
    fanRequest = 0;
}

function loop() {
    let temperature = device.Sensor_Calibration_1_Co;
    fanRequest = min(max((temperature - 80) * 5, 0), 100);
}
```

With a calibrated value of 88 degrees C, the exported demand is 40. Saving the script stores the draft on the PC; Send to Device applies it to the PDM. Verify the input value, exported demand and output command together during commissioning.

Thermostat with hysteresis

The fan request switches on at 90 °C and switches off at 85 °C. Between those temperatures it holds its previous state. Plot temperature and fanOn to see the effect. For a live installation, select the temperature sensor using device autocomplete and use fanOn in the fan output function.

```
export let temperature = 87.5;
export let fanOn = false;
export let thresholdOnly = false;
let onTemperature = 90;
let offTemperature = 85;
let elapsed = 0;

function setup() {}

function loop(dt) {
    elapsed += dt;
    // A slow temperature cycle with a small, faster ripple.
    temperature = 87.5 + 5 * sin(elapsed * 0.5);
    temperature += 0.4 * sin(elapsed * 17);
    thresholdOnly = temperature >= onTemperature;
    if (temperature >= onTemperature) {
        fanOn = true;
    } else if (temperature <= offTemperature) {
        fanOn = false;
    } // Between 85 and 90, leave fanOn unchanged.
    publish();
    sleep(50);
}
```

Accumulated current usage

The script adds the charge used by a changing load and exports the result in amp-hours. It clears the total when the reset request first becomes active. To use the example with a real load, replace the sample current and reset request with the corresponding PDM variables. The total starts at zero each time the script restarts.

```
export let currentAmps = 5;
export let usedAh = 0;
export let resetRequest = false;
let loads = [5, 20, 0, 10];
let elapsed = 0;
let previousAmps = 5;
let wasReset = false;

function setup() {}

function loop(dt) {
    // Attribute elapsed time to the previous sampled load.
    usedAh += max(0, previousAmps) * dt / 3600;
    elapsed += dt;
    currentAmps = loads[floor(elapsed / 5) % 4];
    resetRequest = elapsed >= 25 && (elapsed % 25) < 1;
    if (resetRequest && !wasReset) {
        usedAh = 0; // Reset once, not on every loop while held.
    }
    wasReset = resetRequest;
    previousAmps = currentAmps;
    publish();
    sleep(50);
}
```

Transmit a CAN counter

This example queues one standard CAN frame per second on CAN 1 with ID 0x123. The first byte increments after the frame is queued and the second byte is always 0xA5. The accepted and rejected exports count successful and unsuccessful queue requests. Choose a CAN ID that is available on your bus before enabling the script.

```
export let accepted = 0;
export let rejected = 0;
export let counter = 0;

function setup() {}

function loop() {
    if (CANTransmit(1, CAN_STANDARD, 0x123, counter, 0xA5)) {
        accepted++;
        counter = (counter + 1) & 0xFF;
        publish();
        sleep(1000);
    } else {
        rejected++;
        publish();
        sleep(20);
    }
}
```

Video Examples

We have invested in creating a number of educational videos on our Hardwire Electronics Youtube Channel. Some of the following videos may be useful, especially for CAN bus related problems.

Additional notes:

Additional notes:

Additional notes:

Hardwire electronics

