LDperformance

PDM

Manual

Version: 22-01-2026

Table of Contents

Conventions used

- **Outputs** are numbered **1–12**
  - "Channel 0" appears in configuration as **Settings** and is **not a physical output**
- **Inputs** are numbered **1–10**
  - Inputs 1–5 have internal pull-down resistors
  - Inputs 6–10 have internal pull-up resistors

Introduction

What is LDperformance PDM?

This PDM (Power Distribution Module) is a 12-channel intelligent power distribution system designed to replace traditional fuses and relays in automotive applications. Every output measures current, can be controlled via CAN bus, and is fully protected.

Who is this for?

This PDM is designed for:

- ✓ Those looking to simplify wiring and add telemetry
- ✓ Car owners who want smart power management
- ✓ Professionals connecting to ECUs and data acquisition systems
- ✓ Anyone tired of hunting for blown fuses and failed relays

Key features

Power Distribution

- 12 independently controlled high-side outputs
- Outputs 1-8: 30A continuous (high current)
- Outputs 9-12: 10A continuous (medium current)
- Total capacity: 150A continuous
- All outputs measure current with 1A resolution

Protection

- Configurable overcurrent protection per output
- Hardware and software short-circuit protection
- Integrated temperature sensors on outputs 1-8
- Over-temperature shutdown (configurable)
- Under-voltage and ground fault detection

Control Options

- **Keypad control:** Up to 4 CAN keypads with tactile buttons and RGB LEDs
- **Analog inputs:** 10 inputs for switches and sensors

- **CAN bus control:** Connect to any CAN-enabled ECU
- **Mobile app:** iOS and Android apps for configuration and monitoring
- **Web interface:** Browser-based configuration from laptop/tablet

Monitoring & Telemetry

- Real-time current monitoring on all 12 channels
- 8 integrated temperature sensors
- Live voltage monitoring
- Export configuration for backup
- CAN bus telemetry output for data logging

Special Functions

- **Fuel prime:** Timed pump activation on ignition
- **Wipers:** Multi-speed with automatic park
- **Turn signals:** Synchronized blink mode
- **Soft-start:** Reduce inrush current for motors/fans
- **Sleep mode:** Ultra-low power consumption when inactive
- **Dual PDM:** Network multiple units together

What's in this manual

This manual covers installation, configuration, and operation of the PDM from a user perspective:

1. **Getting Started**: Quick power-up, safety guidelines
2. **Installation**: Mounting, wiring, pinout
3. **Configuration**: App usage, output setup, CAN, keypad
4. **Operation & Monitoring**: Faults, firmware updates, troubleshooting
5. **Reference**: Complete specifications

For low-level serial protocol or advanced integration, see the appendices.

System requirements

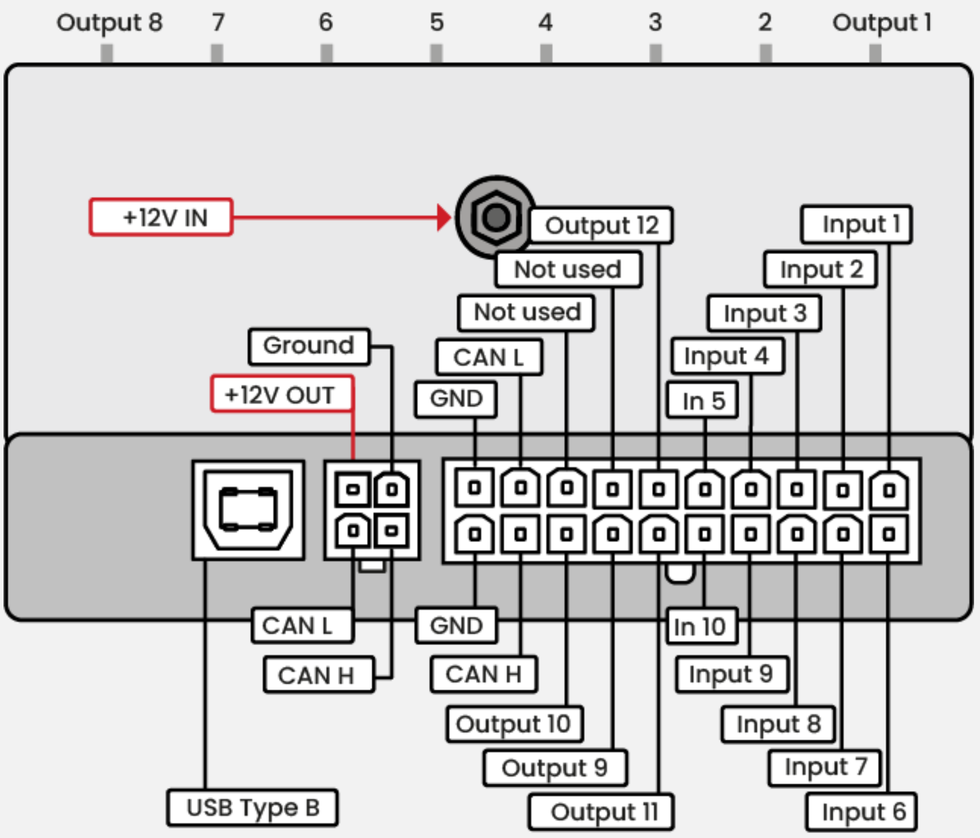
- **Power:** 6-16V DC (automotive 12V systems)
- **Configuration**: iOS or Android device with Bluetooth, OR laptop with WiFi
- **Optional:** CAN-enabled ECU for advanced control and monitoring
- **Keypad:** LDperformance CAN keypads (up to 4 units)

Quickstart

Before you start

✓ Confirm the unit is mechanically secured and cannot short against chassis.

Pinout:



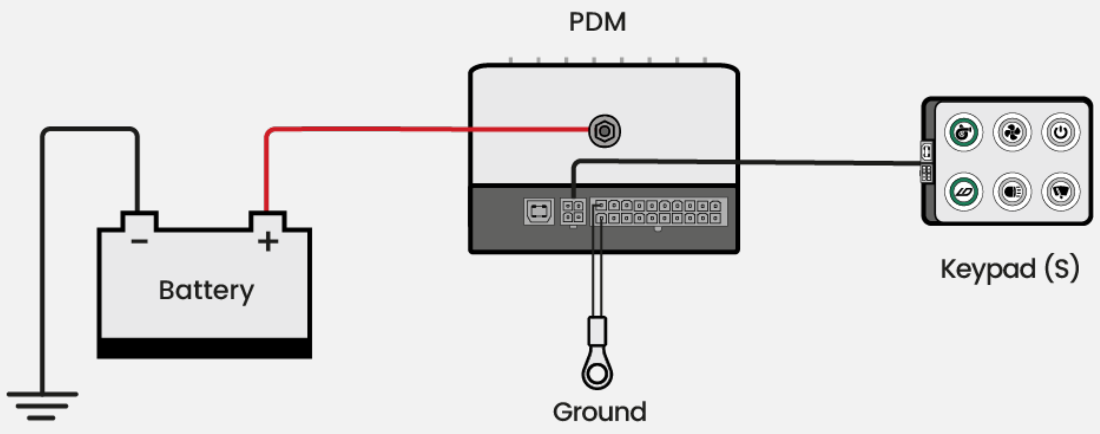
Inputs 1-5 are triggered by 5 or 12V. Inputs 6-10 are triggered by ground.

First power-up checklist

1. Connect **two separate ground wires** to a good chassis ground point **(≥ 0.75 mm² each).**
2. Connect main power (**VBAT+**) to the PDM (use appropriate wiring and fusing for your installation).
3. Connect your keypad(s).
4. Apply power and confirm the unit powers up normally:

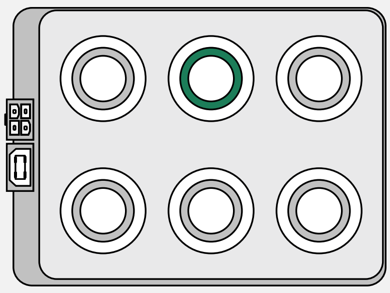
Normal power-up: output LEDs on the PDM will flash **red** → **green** → **blue**.

- If all output LEDs are **red**, recheck both ground wires and the main VBAT+ connection.



Connect with the app (recommended)

1. Install the **LDperformance PDM app** (iOS or Android).
2. Enable Bluetooth on the phone/tablet.
3. Open the app and connect to the PDM.
4. Confirm live telemetry updates (currents/temps/inputs change when you interact with the system).
5. Before making any changes, unlock the PDM:
   - Hold the **Settings/Unlock** button for ~3s (default mapping: **Keypad 1, Button 2, Hold**), or

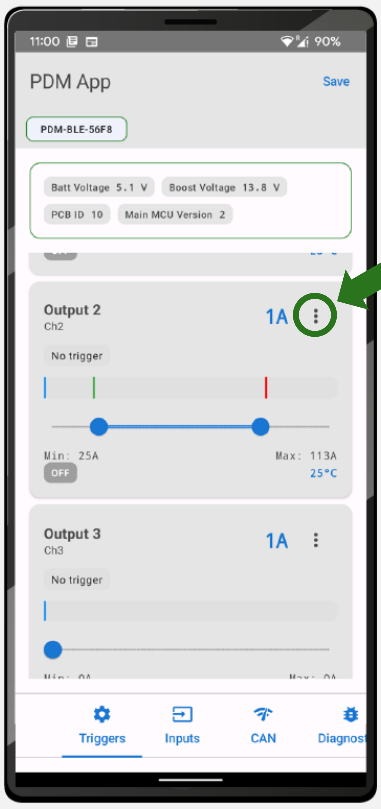


   - Bring your phone close to the PDM and press **Unlock** in the app.

* If you can't connect, see section Troubleshooting.

Verify one output

1. In the app, open one output's configuration.
2. Set Trigger 1 to a keypad button.



11:00 90%

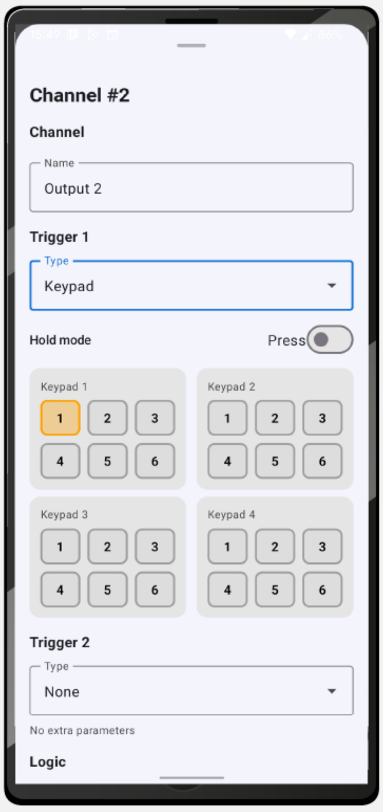
PDM App Save

PDM-BLE-56F8

Batt Voltage 5.1 V | Boost Voltage 13.8 V | PCB ID 10 | Main MCU Version 2

Output 2 Ch2 1A
No trigger
Min: 25A Max: 113A
OFF 25°C

Output 3 Ch3 1A
No trigger



Channel #2

Channel
Name: Output 2

Trigger 1
Type: Keypad

Hold mode: Press

Keypad 1: 1 2 3 4 5 6
Keypad 2: 1 2 3 4 5 6
Keypad 3: 1 2 3 4 5 6
Keypad 4: 1 2 3 4 5 6

Trigger 2
Type: None
No extra parameters

Logic

3. Press the keypad button and confirm:
   - ✓ The output turns on
   - ✓ Current is visible in the app

Save configuration

After you've made changes you want to keep, use the app's **Save** action so settings survive power cycles.

You can set the same keypad button as a trigger for 2 or more outputs.

Example: ignition + starter on one button

1. Edit **Output 1**, name it `IGN1`, set Trigger 1 = keypad button 1.
   - Connect ECU, ignition coils, injectors, etc.
2. Edit **Output 2**, name it `IGN2`, set Trigger 1 = keypad button 1.
   - Connect wideband controller, dash, etc.
3. Edit **Output 3**, name it `START`, set Trigger 1 = keypad button 1 and enable ***Hold mode***.
   - Connect the starter relay coil.

That way you are using just one keypad button which would do ignition on and start the engine (when holding it). Note that you can still crank the engine without ignition on if you want.

Set the same button as the ***Ignition button*** in Settings. Now you can turn off all outputs at once with the same button.

Remember to press ***Save*** from time to time to make changes permanent.

Fan example

Use **Output 4**, set name to `FAN`, Trigger 1 = keypad, select button 2. Verify operation.

Fuel pump example

Use **Output 5,** set name to `FuelPump`:
- Trigger 1 = Fuel prime
- Prime duration = 4000 ms (4s)
- Optional: select a keypad button for manual operation

Make sure you have selected an ignition button in Settings.
Verify that the fuel pump operates for 4 s every time you press the ignition button. If you selected a manual fuel pump button, you can use it to turn the fuel pump on/off manually.

CAN bus control

The PDM can be connected to any CAN-enabled ECU.

Wiring:

1. CANH to CANH and CANL to CANL using two wires twisted together.
2. The PDM does not have an internal CAN termination resistor.
   - If your CAN bus requires termination, add **120Ω** between CANH and CANL at each physical end of the bus.

ECUmaster example

1. Open PDM Settings and select **ECUmaster ECU.**
2. Make sure the configured CAN bus speed in the PDM and ECUmaster is the same.
3. Go to the CAN tab and verify correct values appear for RPM, coolant temp, etc.
4. Configure the fuel pump output:
   - If Trigger 1 is Fuel prime, set Trigger 2 = CAN and pick RPM.
   - ON threshold = 50, OFF threshold = 20.
   - This way when the engine starts (RPM > 50) the fuel pump turns on.
5. Configure the fan output:
   - Configure a trigger to use CAN and select CLT.
   - ON threshold = 60, OFF threshold = 50.
   - The cooling fan turns on when coolant temperature is >60 and runs until it becomes <50.

Megasquirt example

1. Open PDM Settings and select ***Megasquirt*** from the list of ECUs.
2. To enable an output for CAN bus operation set its **duty** to `1`.
   - Outputs with duty `0` will not accept CAN bus control from Megasquirt.
3. In TunerStudio, configure CAN outputs according to your ECU documentation.

Check ` Megasquirt` section for full details.

Safety and reliability

This product switches **high currents** in a vehicle environment. Professional installation is required.

General safety rules

- ✓ Disconnect the vehicle battery before wiring changes.
- ✓ Route wiring away from sharp edges, heat sources, and moving parts.
- ✓ Provide strain relief at connectors; vibration will fatigue wires.
- ✓ If using spade connectors, ensure they are fully seated and latched; vibration can loosen them.

Inductive loads and inrush

Relays, motors, pumps, solenoids, and fans can produce large inrush and inductive spikes. Configure current limits with appropriate headroom.

Fault overrides

Some protections can be overridden for troubleshooting. Treat overrides as **temporary**.

Track/vehicle operations

- ✓ Label outputs clearly (both in software and on the harness).
- ✓ Keep a known-good configuration backup.

Reliability notes

When configuring protection features you need to decide whether they are appropriate for your use case.

<u>Example: configuring a current limit for a fan:</u>

- Do a test run and note how much current the fan needs on startup and while running (e.g. 80 A peak, 20 A steady).
- In hotter conditions the fan might need more (e.g. 120 A). If you set the limit too low, the fan may fail to start when it matters most.

<u>Similar concept with over-temperature protection:</u>

- Set it too low and you risk outputs turning off unexpectedly.
- Disable it and you risk hardware damage if something goes wrong.

Do a proper test run and monitor temperatures/currents to choose sensible limits.

Overcurrent settings

The unit has several layers of overcurrent protection features both in software and hardware.
Set the overcurrent limit in software according to your needs and leave at least **30% overhead** so temperature or noise won't trip the limit accidentally.
Even if the overcurrent limit is not set (0) the system still has short circuit protection enabled in software and hardware.
If overcurrent occurs, the output is switched off and remains off until cleared (typically by switching its trigger OFF and then ON).

Installation (High Level)

This section covers practical installation guidance. For exact pinout and connector details, see `*Pinout-and-Connections*`.

Mounting

- ✓ Mount on a rigid surface with adequate airflow and vibration isolation as needed.
- ✓ Avoid direct exhaust/engine heat soak areas.
- ✓ Ensure water ingress protection matches your vehicle environment.

Power and ground

Connect the main 12V wire coming from battery positive or the main contactor to the PDM stud and ensure it cannot accidentally touch chassis.
Connect **two separate ground wires** to a good chassis ground point.
** Use appropriately sized cables for the expected continuous current.*
Ground wires should be at least **0.75 mm^2.**

Keypad wiring

Connect the keypad using a 4-wire straight-through cable from the PDM to the keypad.

- ✓ CANH and CANL should be twisted together.
- ✓ The keypad CAN bus does not require an external termination resistor for typical single-PDM + keypad installations.

CAN bus wiring to ECU (optional)

- ✓ Use twisted pair for CANH/CANL.

- ✓ Terminate the bus correctly (typically **120Ω at each physical end**) – the PDM has **no internal termination resistor**.
- ✓ Avoid stubs longer than necessary.

Power management options

The PDM offers two power management features for different use cases:

External power on/off switch

The PDM can be configured to turn off all outputs using an external switch. This is useful for master kill switches or dual-location power control.

Setup:

1. Connect a switch to ground on one of the analog inputs **6-10** (these have internal pull-up resistors)
2. The switch should pull the input to ground when "ON" position is desired
3. Use a low-current signal switch (high current not required)
4. In PDM settings, set the "on/off pin" to your chosen input (6, 7, 8, 9, or 10)
5. To disable this feature, set the on/off pin to 200

Multiple switches:

You can connect two switches in series to the same input. For example:

- One switch inside the car
- One switch outside the car (for pit/paddock access)
- Both switches must be ON for the PDM to power outputs

Sleep mode

Sleep mode automatically powers down the PDM and keypads after a period of inactivity, achieving ultra-low power consumption.

How it works:

- The PDM monitors output activity
- When all outputs are OFF (except those with "Default On" enabled) for the configured timeout period, the system enters sleep mode
- Current draw in sleep mode is extremely low (suitable for long-term storage)

To wake from sleep:

Press any button on the keypad

Configuration:

- Set the sleep timeout in the app (Settings section)

- Value is in **seconds** (shown as "*Autosleep after (s)*" in the app)
- Set to 0 to disable sleep mode

<u>Use case:</u>
Ideal for race cars stored between events, or vehicles that sit for extended periods.

Battery isolator and contactor wiring

⚠️ **CRITICAL SAFETY INFORMATION**

Understanding battery isolators

Many users install battery disconnect switches or contactors for safety. However, there's a common and dangerous misconception:
Battery isolators disconnect the battery — they do NOT turn off the engine!

Why this matters:

When the engine is running:
1. The alternator is generating power
2. Disconnecting the battery does NOT cut power to the electrical system
3. The engine continues running on alternator power alone
4. **Without the battery connected, the electrical system experiences**:
   - High voltage spikes and transients
   - Electrical noise
   - Voltage instability
   - Potential damage to sensitive electronics (ECU, PDM, sensors)

The correct approach:

The engine must be turned off BEFORE or AT THE SAME TIME as battery disconnection.

Proper wiring with PDM on/off control:

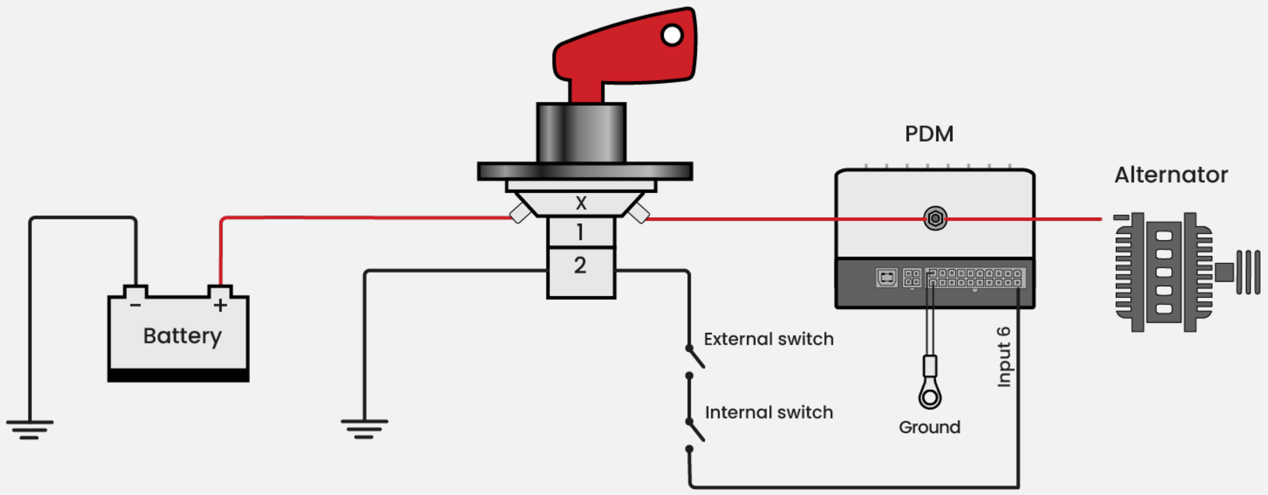
1. Connect the PDM's external on/off switch input (analog 6-10) **through** the battery isolator switch
2. When the battery isolator is activated:
   - First: The signal to the PDM on/off input is removed (pull-up goes open)
   - The PDM turns off all outputs
   - **The engine stops** (ignition/fuel/injectors powered down)
   - Then: The battery disconnects safely
3. This ensures proper shutdown sequence

Wiring options:

<u>Option A: Mechanical switch</u>

- Battery isolator switch controls both the battery contactor AND the PDM on/off input
- When switch opens: PDM sees open circuit on analog input → all outputs turn off → engine stops → battery disconnects

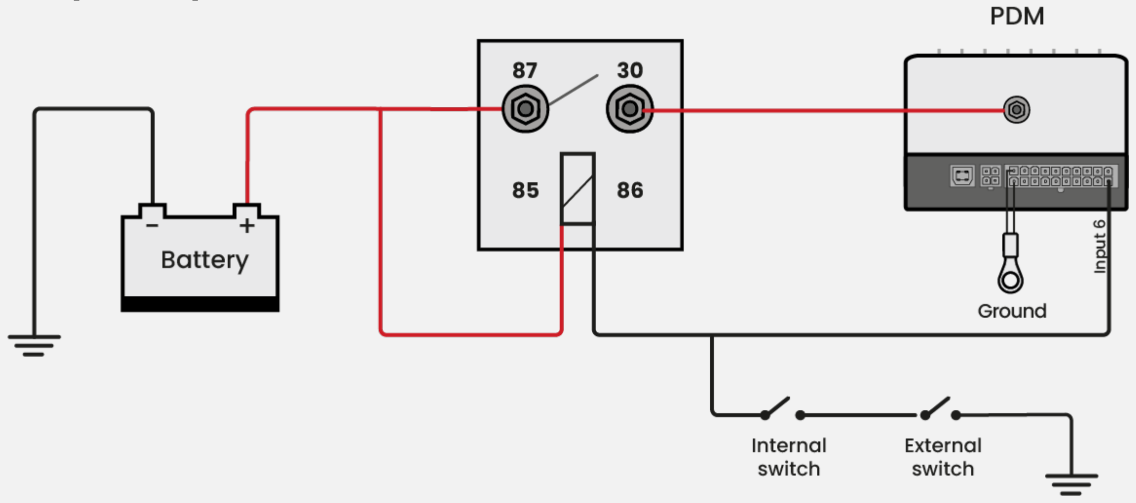
Mechanical Isolator:



<u>Option B: Relay or solid-state relay (SSR)</u>

- Control relay/SSR with the same signal that controls the PDM on/off input
- Ensures both the PDM shutdown and battery disconnect happen in the correct sequence
- Can be controlled remotely (e.g., from dashboard switch or external interlock)

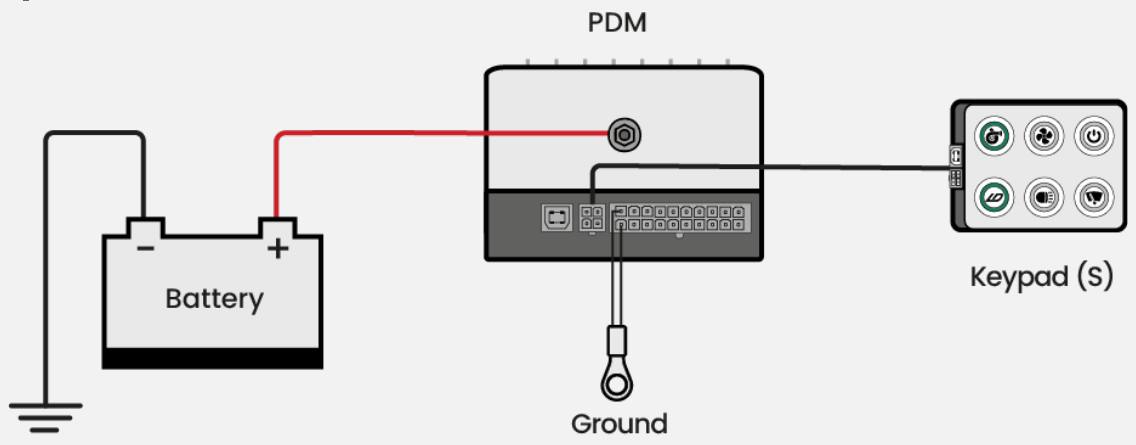
Relay Battery Isolator:



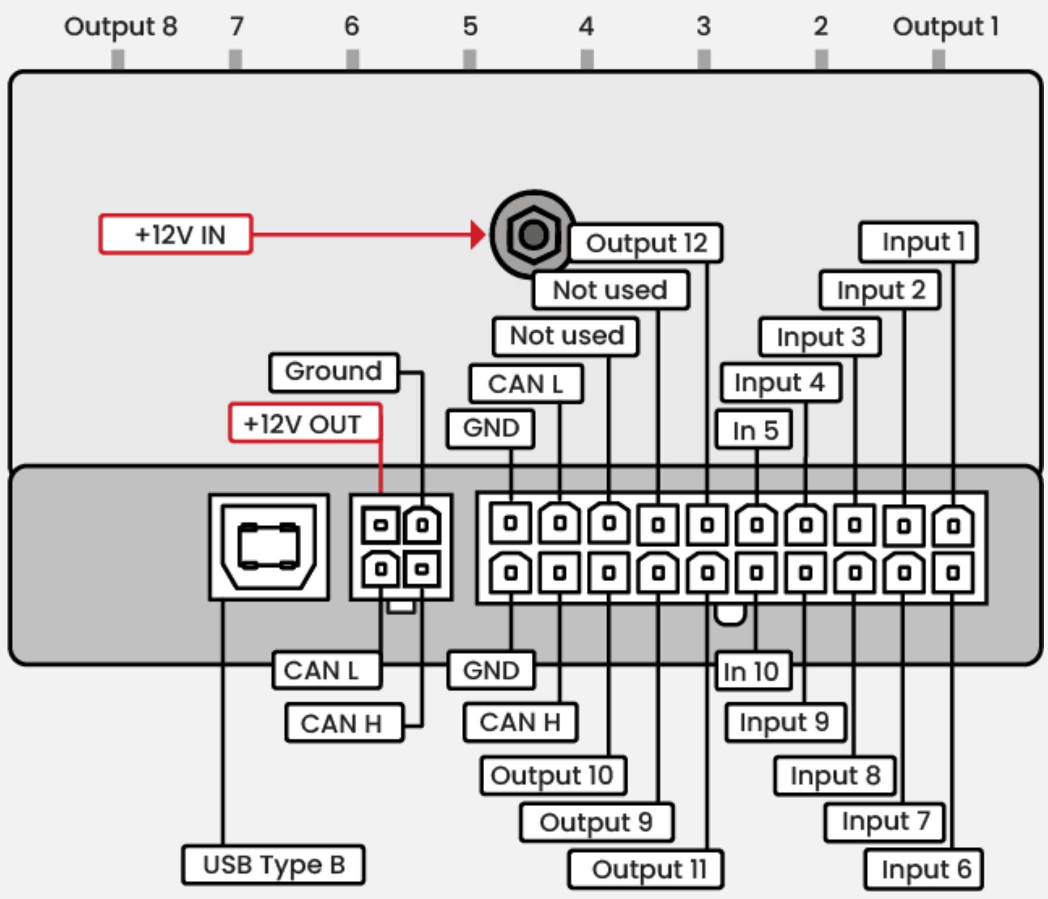
<u>Option C: Direct battery connection (simplest)</u>
The simplest and most reliable wiring is to connect the PDM directly to the battery with no isolator:

- Use the PDM's external on/off switch feature for shutdown control
- Use the PDM's sleep mode for long-term storage
- No risk of isolator-related wiring errors

Quick start:



Pinout



- Inputs 1-5 are triggered by +5V or +12V.
- Inputs 6-10 are triggered by ground.

Wiring Examples (Practical Patterns)

These examples show common wiring and configuration patterns.

Example 1: Ignition + engine start (single button)

Goal: one keypad button turns on "engine power" outputs; holding the same button cranks starter.

Wiring:
- Output 2 → ECU / ignition / injectors relay feed (or direct feed if appropriate)
- Output 3 → wideband / dash / engine ancillaries
- Output 4 → starter motor

App configuration:
- Output 2 name `IGN1`, Trigger 1 = keypad button 1 (toggle), set an appropriate max current.
- Output 3 name `IGN2`, Trigger 1 = keypad button 1 (toggle).
- Output 4 name `START`, Trigger 1 = keypad button 1 (hold/momentary).
- In Settings: set ***Ignition button*** to keypad button 1 (so "ignition OFF" shuts down non-default outputs).

Example 2: Fuel pump with prime + RPM safety

Goal: prime the fuel pump for a few seconds at ignition ON, then keep running only when RPM indicates the engine is running.

Wiring:
- Output X → fuel pump (+12 V feed)

App configuration (typical):
1. Trigger 1 = **Fuel prime**, prime duration = e.g. 4 s
2. Trigger 2 = **CAN parameter** = RPM
   - Direction: `>`
   - ON threshold: e.g. 50 RPM
   - OFF threshold: e.g. 20 RPM
- Logic = **OR** (prime OR running RPM)

Example 3: Cooling fan from coolant temperature (CAN)

Goal: automatically run a fan from ECU coolant temperature.

Wiring:

- Output X → fan (+12 V feed)

App configuration (typical):

- Trigger 1 = keypad (manual override) (optional)
- Trigger 2 = CAN parameter = CLT (coolant temp)
  - Direction: `>`
  - ON threshold: e.g. 90 °C
  - OFF threshold: e.g. 85 °C
- Logic = **OR** (manual OR temperature)

Tip: use **soft-start** for fans if inrush is high.

Example 4: Brake lights from a switch (input 6–10)

Goal: brake lights turn on when a pedal switch closes to ground.

Wiring:

- Input 6 (or any input 6–10) → brake pedal switch to ***ground***
- Output X → brake lights (+12 V feed)

App configuration:

- Output X Trigger 1 = Analog input, Input = 6, Direction = `<`
  - ON threshold: choose a value below the "released" reading (pull-up typically reads high when open)
  - OFF threshold: choose a value above the "pressed" reading

Example 5: Turn signals / hazard lights (blink module)

Goal: left and right indicators blink and synchronize for hazard lights.

Wiring:

- Output 9 → left indicator lights
- Output 10 → right indicator lights

App configuration:

- For both outputs: set trigger type to ***Blink/Timer***
- Start with: ON = 800 ms, OFF = 800 ms

Example 6: Wipers with park switch (input 6–10)

Goal: wipers always park in the same position.

Wiring:

- Input 7 (or any input 6–10) → wiper park switch to ***ground***
- Output 8 → wiper motor

App configuration:

- Configure the **Wipers** module per your setup.
- Assign a keypad button for slow/fast control.

Single press on the keypad button will run the wipers in slow mode.
Double press of the same button will run the wipers in fast mode.
Single press will stop (park) the wipers
ignition off will stop (park) the wipers.

Example 7: External master on/off switch (input 6–10)

Goal: a physical switch turns off all outputs.

Wiring:

- Input 6–10 → switch to ground (switch closed = “ON allowed”)

App configuration:

- In Settings: set the ***on/off pin*** to the chosen input number.

Example 8: CAN wiring checklist (ECU integration)

1. Use twisted pair CANH/CANL.
2. Keep stubs short.
3. Ensure the bus is terminated with ***120Ω at each end*** (PDM has ***no internal termination***).
4. Set CAN0 bitrate to match the ECU (see `CAN Integration`).
5. Use the CAN screen to confirm live values update before relying on CAN triggers.

App Guide (iOS/Android)

The recommended way to configure and monitor the PDM is the companion mobile app.

Connect (Bluetooth)

1. Power the PDM.
2. Open the app and connect to the device (typically shown as `PDM-BLE-xxxx`).
3. Confirm live telemetry updates (currents, temps, inputs).

If you see connection drops, see section `Troubleshooting`.

Security and unlock mode

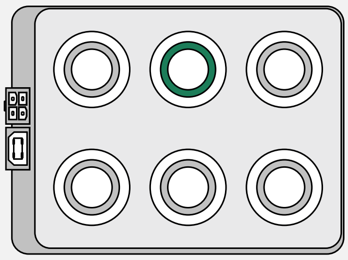
The PDM has built-in security to prevent unauthorized wireless modifications to your configuration.

Default state (locked):

- WiFi: OFF
- Bluetooth: monitoring/read-only

To unlock (enable configuration changes):

1. Keypad method: hold the *Settings/Unlock* button for 3 seconds (default mapping: *Keypad 1, Button 2, Hold*).



2. App method: press *Unlock* in the app while holding your phone very close to the PDM.

When unlocked:

- The app accepts configuration changes.
- The WiFi access point is enabled and the web interface becomes available.

** The PDM locks again after a period of time and always after power cycle. Unlock again before making changes.*

Main screens (typical)

- **Triggers / Outputs**: live channel cards, on/off state, current bars, quick edits.
- **Temps**: temperature readings and related status.
- **Inputs**: analog inputs *1–10* shown as raw counts *0–1023*
- *Inputs 1–5* can read 0-5V and have internal *pull-down* resistors.
- *Inputs 6–10* read 0-3.3V and have internal *pull-up* resistors (common pattern: switch to ground).
- **CAN**: configure up to 5 CAN parameters and view their live values.
- **Settings**: save/persist, export/import, firmware update.

Web interface (optional)

The web interface is useful for commissioning from a laptop/tablet.

1. Unlock the PDM (this enables WiFi).
2. Connect to the PDM WiFi network (`PDM-…`).
3. Enter password: `pdm123456`
4. Open a browser and go to `192.168.4.1`

If the page stops responding: unlock again, verify you are still connected to the PDM WiFi, then refresh.

Editing an output (1–12)

Common settings:

- **Name**: label shown in the UI (max *10 characters*).
- **Default On**: allowed to stay on after ignition is turned OFF (see `Keypad`).
- **Triggers**: up to two triggers with OR/AND logic (see `Channel Setup`).
- **Min/Max current**: undercurrent warning and overcurrent trip (see `Monitoring and Faults`).
- **Duty / soft-start / timeout**: load-dependent behaviour (see `Channel Setup`).

Saving changes

Most edits apply immediately, but will not persist across power cycles until you use *Save* in the app.

Backup / restore

Use *Export* to save a known-good configuration, and *Import* to restore it later.
See `Monitoring and Faults`.

Channel Setup (Outputs 1–12)

This section explains how to configure outputs in a user-facing way.

Outputs vs “Settings” (Channel 0)

- *Outputs 1–12* are the physical power channels.
- The UI may also show *Settings / Channel 0* which controls system behaviors (unlock/WiFi, protections, CAN settings, etc.). It is not a physical output.

Output groups (typical)

- *Outputs 1–8:* high-current channels (also have temperature sensors and on-board RGB LEDs).
- *Outputs 9–12:* lower-current channels.

Common per-output settings

Name

Give every output a meaningful name (`FUELPUMP`, `FAN`, `LIGHTS`, etc.).

- Max length: *10 characters*
- Tip: keep names short and consistent so telemetry and logs are easy to read.

Default On

If enabled, the output turns on when power is applied and is allowed to stay on after ignition is turned OFF (until sleep mode/external power-off switch turns the system off).

Current limits

- **Max Current (trip):** if exceeded, the output is shut down and latched as a fault.
  - Keypad LED (if mapped): typically flashes *red*.
  - On-board output LED (outputs 1–8): *red*.
  - Clear the fault by making the trigger go *OFF*, then *ON* again.
- **Min Current (warning):** if enabled, low current is flagged as a warning (output stays ON). Useful for detecting unplugged loads or broken grounds.

Practical tip: for inductive/inrush loads (fans, pumps, solenoids), leave headroom. A common starting point is *~30% above* the highest current you measured during a real test.

Triggers

Each output can have up to *two triggers*:

- Configure **Trigger 1** and **Trigger 2**
- Choose **Logic**:
  - **OR**: output turns on if either trigger is active
  - **AND**: output turns on only if both triggers are active

Triggers use hysteresis (separate ON and OFF thresholds) for stable behavior.

Trigger types (user view)

Keypad trigger

Turns the output on/off from a keypad button. Common modes:

- Toggle (press toggles state)
- Momentary/hold (active only while pressed)

Analog input trigger

Turns the output on/off based on an analog input (inputs 1–10) crossing thresholds.

Important: By default, *Analog Trigger 1 is ignition-gated* (it only works when ignition is ON). Analog Trigger 2 is not ignition-gated.

Threshold units: analog inputs are shown as raw counts **0–1023** (10-bit) representing **0–5V** for inputs 1-6 and **0–3.3V** for inputs 6-10.

Wiring patterns (most common):

- **Inputs 6–10 (pull-up):** connect a switch to *ground* → input reads LOW when pressed/closed (recommended choice for simple on/off switches).
- **Inputs 1–5 (pull-down):** input reads HIGH when switched to 5V or 12V*.

CAN parameter trigger

Turns the output on/off based on one of the 5 configured CAN parameters.

When the PDM is connected to an ECU over CAN, it can read values like RPM and coolant temperature. If you select a predefined ECU type, common CAN parameters may be available automatically.

Thresholds + direction (how a trigger "decides")

For analog/CAN triggers you typically set:

- **Direction**
  - `>` means the trigger becomes active when the value goes above the ON

threshold
  - `<` means the trigger becomes active when the value goes below the ON threshold
  - **ON threshold:** point where the trigger turns on
  - **OFF threshold:** point where the trigger turns off (hysteresis to prevent chatter)

Special functions (optional)

Some trigger types implement higher-level behaviors:

- **Fuel prime:** runs a configured output for a set duration when ignition turns on (and may support manual activation).
- **Wipers:** supports multi-click behavior (slow/fast/park logic).
- **Blink/timer**: toggles an output with ON/OFF durations.

See `Keypad` and `CAN` for related setup patterns.

Special function setup notes

Fuel prime

Typical pattern:

1. Choose the output that powers the fuel pump.
2. Set one trigger type to **Fuel prime**.
3. Set the **prime duration**.
4. Turn ignition ON and verify the pump runs for the prime duration.

You can also set a manual on/off keypad button for the fuel pump if needed.

Tip: fuel prime works well in combination with a CAN RPM trigger (OR), so the pump primes and then runs whenever RPM is above a specified limit.

Wipers

The wipers module supports slow/fast and always parks the wipers in the same position.

It requires a wipers park/position switch wired to an **input with internal pull-up**:

- Use **Inputs 6–10** and wire the switch to ground.

Wipers are typically controlled from a keypad button:

- Single press: slow/intermittent behavior
- Double press: fast
- Single press / Ignition OFF: they go to park position

Blink / timer

Blink mode toggles an output ON and OFF with configurable durations.

Typical turn-signal starting point:

- ON time: **800 ms**
- OFF time: **800 ms**

If you configure two outputs for left/right indicators, they will synchronize to behave like hazard lights.

Soft-start / PWM duty

Some loads benefit from soft-start or PWM control (fans, pumps, heaters). Use PWM sparingly: it increases heat in the PDM and in wiring.

Duty (%)

- `0` or `1` means full ON (no PWM).
- `2–100` sets a PWM duty percentage.

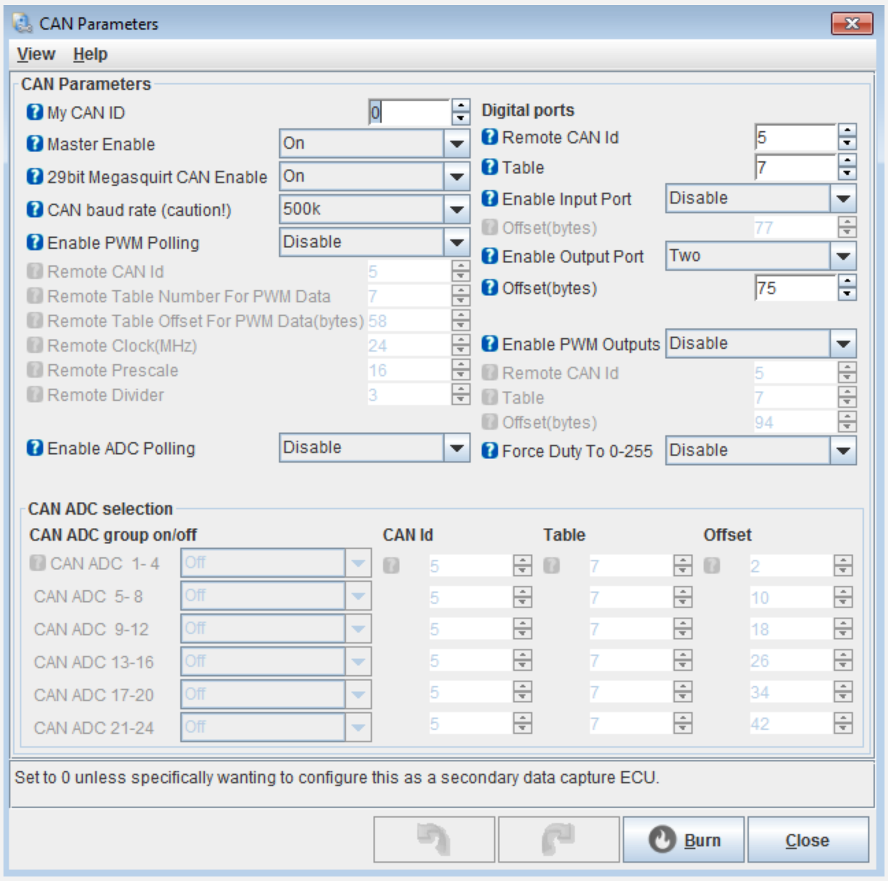
Soft-start

- Soft-start ramps PWM up from a low value to the configured duty.
- The **soft-start number** scales the ramp time (higher = slower).
  - Rough guide: each step is about **~20 ms** at 100% duty (ramp time depends on your configured duty and the load). 1 – very fast turn on, 255 – turn on takes around 5s.

Megasquirt Example

Those instructions apply for Megasquirt MS3.

Go to CAN bus/ Testmodes / CAN parameters. It should look like this:



CAN Parameters

My CAN ID	0
Master Enable	On
29bit Megasquirt CAN Enable	On
CAN baud rate (caution!)	500k
Enable PWM Polling	Disable
Remote CAN Id	5
Remote Table Number For PWM Data	7
Remote Table Offset For PWM Data(bytes)	58
Remote Clock(MHz)	24
Remote Prescale	16
Remote Divider	3
Enable ADC Polling	Disable

Digital ports

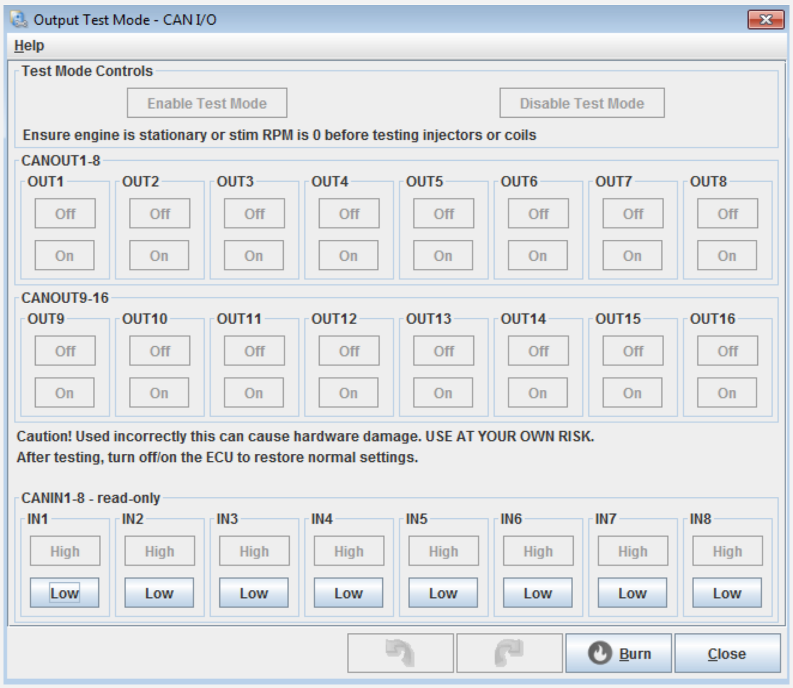
Remote CAN Id	5
Table	7
Enable Input Port	Disable
Offset(bytes)	77
Enable Output Port	Two
Offset(bytes)	75
Enable PWM Outputs	Disable
Remote CAN Id	5
Table	7
Offset(bytes)	94
Force Duty To 0-255	Disable

CAN ADC selection

CAN ADC group on/off		CAN Id	Table	Offset
CAN ADC 1-4	Off	5	7	2
CAN ADC 5-8	Off	5	7	10
CAN ADC 9-12	Off	5	7	18
CAN ADC 13-16	Off	5	7	26
CAN ADC 17-20	Off	5	7	34
CAN ADC 21-24	Off	5	7	42

Set to 0 unless specifically wanting to configure this as a secondary data capture ECU.

After that you can verify the ECU is receiving the data from the keyboard. Open CAN bus/Testmodes/ Output test mode – CAN I/O. Click on Enable Test Mode. When pressing a button on the keyboard it will trigger one of CAN IN 1-8. This can be used to activate switched inputs to the ECU, for example switching from low to high boost, enabling launch control, antilag etc.



Output Test Mode - CAN I/O

Help

Test Mode Controls

Enable Test Mode | Disable Test Mode

Ensure engine is stationary or stim RPM is 0 before testing injectors or coils

CANOUT1-8

OUT1	OUT2	OUT3	OUT4	OUT5	OUT6	OUT7	OUT8
Off	Off	Off	Off	Off	Off	Off	Off
On	On	On	On	On	On	On	On

CANOUT9-16

OUT9	OUT10	OUT11	OUT12	OUT13	OUT14	OUT15	OUT16
Off	Off	Off	Off	Off	Off	Off	Off
On	On	On	On	On	On	On	On

Caution! Used incorrectly this can cause hardware damage. USE AT YOUR OWN RISK.
After testing, turn off/on the ECU to restore normal settings.

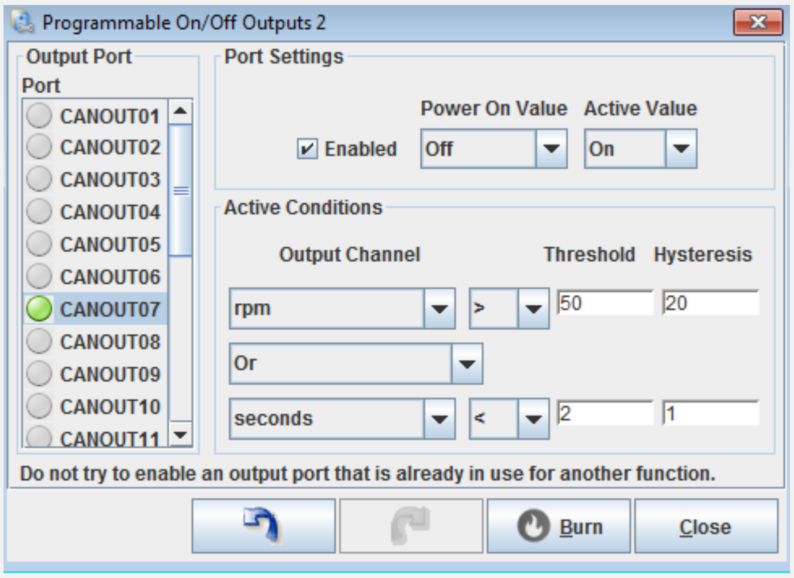
CANIN1-8 - read-only

IN1	IN2	IN3	IN4	IN5	IN6	IN7	IN8
High	High	High	High	High	High	High	High
Low	Low	Low	Low	Low	Low	Low	Low

Burn | Close

If you want to switch on an output of the PDM from the ECU, for example fan or fuel pump etc, follow this procedure:

- Connect the fuel pump to let's say CH7 of the PDM.
- In the PDM app make sure the settings for duty for CH7 is set to 1 (so it can be triggered though CAN)
- Now when you turn on CAN OUT 7 in Tunerstudio, the fan will turn on. For the test you can do that in the menu above. Otherwise you can program a CAN output from: Advanced Engine/ Programmable On/Off outputs 2. Here is an example for fuel pump settings:



Programmable On/Off Outputs 2

Output Port

Port: CANOUT01, CANOUT02, CANOUT03, CANOUT04, CANOUT05, CANOUT06, CANOUT07 (selected), CANOUT08, CANOUT09, CANOUT10, CANOUT11

Port Settings

☑ Enabled — Power On Value: Off — Active Value: On

Active Conditions

Output Channel		Threshold	Hysteresis
rpm	>	50	20
Or			
seconds	<	2	1

Do not try to enable an output port that is already in use for another function.

Burn | Close

CAN Integration

The PDM can:

1. **Monitor up to 5 CAN parameters** (for display and logic), and
2. Use those values as **triggers** to control outputs.

CAN parameter slots (1–5)

You configure up to five "virtual sensors" by defining:

- Name (max **11 characters**)
- CAN ID
- Byte offset and data type (U8, U16, S16, etc.)
- Scaling (multiplier/divider/offset/decimal places)
- Timeout behavior (value to use when CAN data is missing)

This is usually done in the app under the **CAN** tab.

Data types (common)

- **U8 / S8:** unsigned/signed 8-bit value from one byte
- **U16 / S16:** unsigned/signed 16-bit value from two bytes (big-endian or little-endian)
- **BIT:** extract a single bit (0/1) from a byte (bit position 0–7)

Scaling (how raw CAN becomes engineering units)

The PDM applies scaling in this order:

`Final = (((Raw × Multiplier) ÷ Divider) + Offset) ÷ (10^DecimalPlaces)`

Use this to convert raw values into °C, kPa, RPM, etc.

BIT type note

For **BIT** type, the **Offset** field is typically used as the **bit position** (0–7) within the selected byte.

Typical use cases

- Fan on/off from coolant temperature
- Fuel pump only when RPM > threshold
- Lights based on speed or gear

Using a CAN parameter as a trigger

1. Configure a CAN parameter slot with the correct CAN ID and scaling.
2. In the output's config, set a trigger type to *CAN parameter*.
3. Select which parameter slot (1–5) to use.
4. Set ON/OFF thresholds and direction (> or <).

Example: RPM-based fuel pump enable (outline)

1. Configure CAN parameter "RPM" in slot 1 (CAN ID + byte offset + U16 endianness + scaling).
2. On the fuel pump output, set:
    - **Trigger type:** CAN parameter
    - **Parameter slot:** 1
    - **Direction:** `>` (greater-than)
    - **ON threshold:** e.g. `300`
    - **OFF threshold:** e.g. `200`

CAN bus speed and wiring

See `Installation` for wiring best practices.

Supported bitrates and bus roles

- **CAN0 (ECU / vehicle CAN):** Configurable.
    - In the current UI, CAN0 bitrate is stored in **Settings / Channel 0**
    - `0` → **1 Mbps**
    - `3` → **250 kbps**
    - `4` or `1` → **125 kbps**
    - `2` (default) → **500 kbps**
- **CAN1 (Keypad CAN):** Fixed **500 kbps.**

Note: In *secondary PDM* mode (`set_keypad_bus,0`), keypad traffic is on CAN0 instead of CAN1.

Keypad

Keypads provide tactile control and status indication for outputs.

Basic operation

- A button can be assigned to an output as a trigger.
- Button LED color and behavior reflect the output status (normal / warning / fault).
- Up to 4 keypads can be connected to one PDM.

The PDM app also allows the user to customise:
  - Keypad button colours
  - Backlight colour
  - button brightness
  - backlight brighthness.

Keypad button will flash in red if overcurrent event occurs.

Ignition behavior (user view)

The PDM supports a logical **Ignition ON/OFF** state. When ignition is turned:
- **ON:** ignition triggers become active; any configured fuel-prime outputs can run.
- **OFF**: outputs without "Default On" are turned off in a controlled shutdown sequence (wipers are handled specially if configured).

Default-on outputs vs ignition

- Outputs marked **Default On** are allowed to stay on after ignition is turned OFF.
- If you want an output to always shut down with ignition OFF, keep **Default On** disabled and use ignition-gated triggers instead.

Settings / Unlock / WiFi enable (default mapping)

By default, the "Settings" channel is mapped to:
- **Keypad 1, Button 2, Hold mode** → unlock configuration / request WiFi.

Note: This mapping can be changed in configuration.

Dual PDM (two modules)

Dual PDM lets you run **two PDM units on the same vehicle**. This is useful when you have more loads than one unit can cover, or when you want to split wiring front/rear.

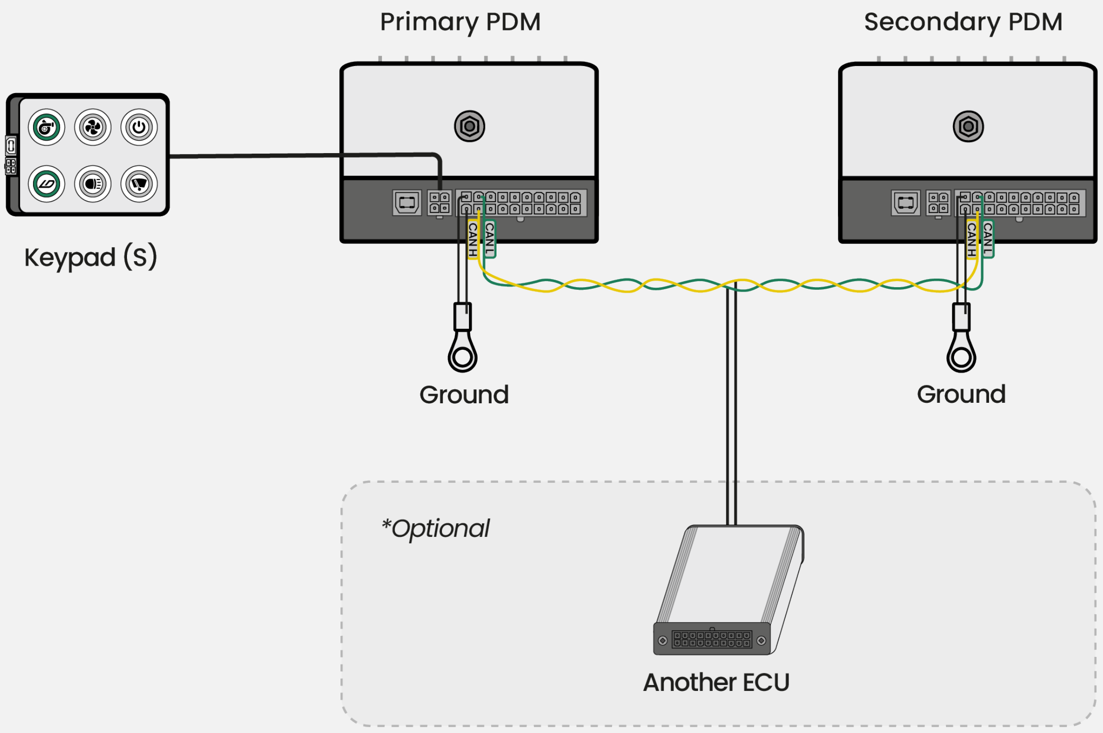
Roles

- **Primary PDM:** the unit that has the **keypad physically connected.**
- **Secondary PDM:** the unit that **follows the primary** for ignition and button presses over CAN.

Both units still run their own outputs independently — you are only sharing “control signals” (ignition + keypad).

Typical wiring (high level)

- Both PDMs connect to the vehicle **ECU CAN bus.**
- The keypad is wired to the **primary PDM.**
- The secondary PDM does **not** need a keypad connection.



Setup in the app

You must configure **each PDM separately** (connect to one, apply settings, then connect to the other).

1. Connect to the **primary** PDM.
2. Go to **Settings → Secondary PDM.**
3. Press **Configure as Primary PDM** and confirm.
    - This sets the keypad routing correctly and sets the default **base CAN ID** for the primary (typically `0x668`).
    - If the unit was previously set to "follow ignition from CAN", that ignition-follow setting will be cleared (so the primary uses the keypad again).
4. Connect to the **secondary** PDM.
5. Go to **Settings → Secondary PDM.**
6. Press **Configure as Secondary PDM** and confirm.
    - This sets the secondary to receive keypad/ignition via the ECU CAN bus and assigns it a different **base CAN ID** (typically `0x678`) so the two units don't collide.

Important: Primary and secondary must use **different base CAN IDs**. If both units use the same ID, their CAN messages will overlap and the system will behave unpredictably.

What you should see

- The keypad will control outputs on the primary as normal.
- The secondary will react to the same ignition ON/OFF events as the primary (so "Default On" behaviour is consistent across both).
- You can still name and configure outputs on each unit independently.

Common issues

- **Secondary doesn't respond to keypad / ignition:**
    - Confirm you configured one unit as **Primary** and the other as **Secondary** (not both the same).
    - Confirm both PDMs are on the same ECU CAN bus.
    - Confirm base CAN IDs are different (e.g. `0x668` and `0x678`).

Monitoring and Faults

Use the app to monitor:
- ✓ Output currents (per channel)
- ✓ Temperatures (for channels 1-8)
- ✓ Analog inputs (1–10)
- ✓ CAN values (up to 5 configured parameters)
- ✓ System voltage(s)

Common channel states

Normal ON

Current is within limits.

Overcurrent fault (trip)

If current exceeds the configured maximum, the output is shut down and flagged as a fault.
What to do:
1. Switch the output OFF (make the trigger go inactive).
2. Inspect wiring/load for short or stalled motor.
3. Increase max current only if the load and wiring support it.
4. After fixing the cause, switch the trigger OFF then ON to clear the latched fault.

Undercurrent warning

If current stays below the configured minimum, a warning can be shown (the output usually stays on).
Use cases: detecting unplugged loads, broken grounds.

Over-temperature fault (if enabled)

Outputs 1–8 are protected by temperature sensors. An over-temp fault will shut down the output until it cools.
It is the user's responsibility to verify normal operating temperatures of all outputs in all conditions and decide whether to use over-temp protection, and at what limit.

LED indication

The PDM provides visual feedback through:
- On-board output LEDs (outputs 1–8 only)
- Keypad button LEDs (if a keypad is used)

Startup sequence (normal power-up)

When the PDM powers up normally, all output LEDs will flash:
Red → Green → Blue
This sequence confirms the PDM is initializing correctly.

PDM output LED status (outputs 1–8)

Each output has an RGB LED on the PDM showing its state:

- **Green**: Output ON, operating normally
- **Red**: Overcurrent fault
- **Yellow**: Undercurrent warning
- **Magenta**: Over-temperature fault
- **Off**: Output not active

Keypad button LED status

Keypad LEDs can be configured to reflect the associated output status. Typical priority order:

- **Overcurrent fault** → red (flashing)
- **Normal ON** → configurable
- **OFF** → no light

Fault indicators

All PDM LEDs flashing red:

- **Meaning**: Ground fault detected - one or both ground connections are not properly connected
- **Action:**
  1. Check both GND1 and GND2 connections to chassis ground
  2. Check main 12V (VBAT+) connection
  3. Ensure connections are tight and corrosion-free

Orange flashing:

- **Meaning**: Firmware update in progress OR internal system issue
- **Action**:
  - If updating firmware: wait for completion (can take 10+ minutes)
  - If not updating: check `Troubleshooting`.

Keypad button flashing red:

- **Meaning**: The associated output has tripped due to overcurrent
- **Action**: Turn output off, inspect wiring/load, clear fault by cycling the trigger

System protections (voltage / ground fault)

The system may prevent outputs turning on if supply voltage is too low or a ground fault is detected.

Scripting (advanced)

The PDM supports an **advanced scripting mode** that lets you run a small, user-supplied program on the PDM. This is designed for simple automation tasks that don't fit neatly into the standard trigger system.

Typical uses:

- Custom keypad logic (e.g. **double-press**, toggle behaviors, step-through modes)
- Simple state machines (e.g. "if Channel 2 current is high, turn Channel 5 on")
- Extra safety/logic rules based on inputs and live telemetry

What you need to know

- Scripts are uploaded as a **WebAssembly (`.wasm`)** file.
- Only **one script** is supported at a time (uploading a new one replaces the old one).
- You compile the `.wasm` on your computer (laptop/desktop), then upload it to the PDM.
- The PDM still enforces its normal protections (current limits, faults, etc.). Scripting is for "logic", not bypassing safety.

Uploading a script

1. **Unlock** the PDM (this enables WiFi and allows configuration changes).
2. Open the **web interface** (`192.168.4.1`).
3. Go to the **Scripting** page and upload your `script.wasm`.
4. Send it to the **main MCU** when prompted.

When a new script is received successfully, **scripting is auto-enabled** and the new script replaces the previous one.

Enabling / disabling

In the mobile app, go to **Settings** → **Scripting**:

- **Enabled**: scripting is allowed to run
- **Active**: the script is currently running (ticking)

If you change Enabled/Disabled and want it to persist across power cycles, **use Save** in the app.

Recovery
If a script causes unexpected behavior:
1. Disable scripting in **Settings** → **Scripting**
2. Upload a corrected script to replace it

Getting started (developer kit)
In this firmware repository there is a `wasm_scripts/` folder with example scripts and a short guide for compiling a `.wasm`.

A standalone GitHub “starter” repo (API header + examples + build instructions) is availa-ble so users can get started quickly.

Firmware Updates

Firmware updates are typically performed from the mobile app.

Recommended workflow

1. Make a configuration **Export/Backup** first.
2. Note the current firmware version in the app **Settings**.
3. Ensure stable power during the update (avoid low battery and avoid cranking).
4. Start the update from the app allow **10+ minutes**.

Avoid updating right before an event. If everything is working reliably, consider leaving the current firmware in place.

<u>During the update:</u>
- Keep the phone screen awake (do not let it go to sleep).
- Stay close to the PDM to keep the wireless link stable.

Troubleshooting

All PDM LEDs are red (or flashing red)

Meaning: the PDM is in voltage/ground protection (common cause: bad ground connection).

Checks:
1. Verify **both** ground wires are connected to a clean chassis ground point (two separate wires).
2. Verify the main **VBAT+** connection is solid and cannot short to chassis.
3. Check for corrosion/loose hardware at battery and chassis points.

App can't find the device (Bluetooth)

1. Confirm the PDM is powered.
2. Confirm Bluetooth is enabled on your phone/tablet.
3. Move closer to the device and retry.
4. Make sure you have allowed all requested permissions for the app.
5. Hold the Unlock button (button2) for 6s and rescan for BT devices in app.
6. Power-cycle the PDM and retry.
7. Reboot the phone/tablet and retry.

If you can connect but cannot change settings, the PDM may be locked — see the unlock section in `App-Guide`.

Diagnostics & Warnings

If you are facing any issue check the **Diagnostics** or **Warnings** tab in the app - it identifies common issues with your PDM configuration.

For example:
- Ignition button not set
- Unlock button not set to button2
- Scripting enabled
- PWM, PWM module , softstart or timeout enabled for any output
- CAN ID different from 0x668
- Keypad set on **ECU CAN Bus** instead of **Keypad CAN Bus**
- Vref not 180 for any output

Web interface won't load

1. Unlock the PDM (WiFi is enabled only when unlocked) — see `App-Guide`.
2. Connect to the PDM WiFi network (`PDM-…`) and confirm your device stayed on that WiFi.
3. Open `192.168.4.1` in your browser.
4. If the page is stuck: unlock again, then refresh.

Output won't turn on

1. Check that the output is not in an **overcurrent** or **over-temp** fault (see `Monitoring-and-Faults`).
2. Check that triggers are configured correctly (and ignition is ON if the trigger is ignition-gated).
3. Check wiring and the load itself.
4. Simplify configuration to isolate the issue:
   - Temporarily remove Trigger 2
   - Use OR logic
   - Test again

Output turns off immediately

Most common cause: overcurrent trip or wiring short.

1. Reduce the load (disconnect the device) and test the output again.
2. Inspect harness for shorts to chassis.
3. Verify your max current setting is appropriate for the load and wiring.
4. After fixing the cause, clear the fault by switching the trigger **OFF then ON.**

CAN value never updates

1. Verify CANH/CANL polarity and wiring (twisted pair).
2. If there are only two CAN devices on the bus, add a **120Ω** termination resistor between CANH and CANL at each end (the PDM has **no internal termination**).
3. Verify CAN speed (bitrate) matches the ECU and other devices.
4. Confirm the CAN ID, byte offset, endianness, and scaling are correct.

Keypad not responding

1. Verify keypad power/ground.
2. Verify keypad CAN wiring (CANH/CANL twisted pair).
3. Restore the keypad to default settings:
   - Hold the two buttons furthest from the connector while turning on the keypad

- LEDs blink green when reset is accepted

Firmware update takes a long time or appears stuck

- Allow **10+ minutes** (updates can be slow).
- Keep the phone awake and close to the PDM.
- If the PDM indicates update activity (orange flashing), wait for it to complete.

Still stuck?

Capture and send:

- An exported configuration backup (from the app Settings).
- A short description of the wiring/load and what you expected to happen.
- Photos of the installation (power, grounds, CAN wiring) if possible.

Appendix A CAN Protocol

PDM CAN outputs (telemetry + keypad)

The PDM transmits **two families** of CAN messages:

1. **Telemetry frames** on the ECU/vehicle CAN (CAN0).
2. **Keypad LED/control frames** on the keypad CAN (CAN1 by default).

1) Telemetry frames (CAN0)

Base CAN ID = `data1.o[0].timeout` (channel 0 “timeout” field).
Default base ID is **0x668.**

Update rate: One CAN0 telemetry frame every **10 ms,** rotating through IDs.
Each individual ID updates about every **100–120 ms**.

All telemetry frames are **standard 11-bit IDs**, DLC = 8, and unused bytes are zero.

FRAME MAP (CAN0)

Offset	ID	Name	Notes
+0	BASE + 0	System data	Version + temps + voltages
+1	BASE + 1	Packed channel summary	Two channels per byte
+2	BASE + 2	Analog inputs (pulldown)	IN1–IN5
+3	BASE + 3	Analog inputs (pullup)	IN6–IN10
+4	BASE + 4	Currents 1–8	Raw amps (uint8, clamped 255)
+5	BASE + 5	Currents 9–12	Raw amps (uint8, clamped 255)
+6	BASE + 6	Output voltages 1–8	Raw ADC-scaled (uint8)
+7	BASE + 7	Output voltages 9–12	Raw ADC-scaled (uint8)
+8	BASE + 8	Temperatures 1–8	°C (uint8)
+9	BASE + 9	Full state 0–7	`o[0].state` … `o[7].state`
+10	BASE + 10	Full state 8–12	`o[8].state` … `o[12].state`

BASE + 0 — System data

```
byte 0: CAN output version (fixed 0x10)
byte 1: Keypad bus update period (ms, uint8, clamped to 255)
byte 2: VBAT sense (sysmeas[3], uint8 raw)
byte 3: Board temp 1 (temps[0], °C)
byte 4: Board temp 2 (temps[1], °C)
byte 5: Board temp 3 (temps[2], °C)
byte 6: Board temp 4 (temps[3], °C)
byte 7: 12V/boost sense (sysmeas[0], uint8 raw)
```

Notes:
- `sysmeas[]` values are **raw ADC-derived** values. Use your existing calibration to convert to volts.

BASE + 1 — Packed channel summary (legacy compact)

Six bytes encode **channels 1–12**, two channels per byte:

```
byte 0: CH1 (bits 7–4) + CH2 (bits 3–0)
byte 1: CH3 (bits 7–4) + CH4 (bits 3–0)
byte 2: CH5 (bits 7–4) + CH6 (bits 3–0)
byte 3: CH7 (bits 7–4) + CH8 (bits 3–0)
byte 4: CH9 (bits 7–4) + CH10 (bits 3–0)
byte 5: CH11 (bits 7–4) + CH12 (bits 3–0)
byte 6–7: 0
```

Per-channel nibble format:

```
bit 3: channel active (state > 0)
bits 2–0: state & 0x07  (lower 3 bits of channel state)
```

BASE + 2 — Analog inputs (pulldown)

```
byte 0: IN1  (raw, (inp[0] - 1) / 4)
byte 1: IN2  (raw, (inp[1] - 1) / 4)
byte 2: IN3  (raw, (inp[2] - 1) / 4)
byte 3: IN4  (raw, (inp[3] - 1) / 4)
byte 4: IN5  (raw, (inp[4] - 1) / 4)
byte 5–7: 0
```

BASE + 3 — Analog inputs (pullup)

```
byte 0: IN6   (raw, (inp[5] - 1) / 4)
byte 1: IN7   (raw, (inp[6] - 1) / 4)
byte 2: IN8   (raw, (inp[7] - 1) / 4)
byte 3: IN9   (raw, (inp[8] - 1) / 4)
byte 4: IN10  (raw, (inp[9] - 1) / 4)
byte 5–7: 0
```

BASE + 4 — Currents 1–8

```
byte 0–7: CH1–CH8 current (uint8, clamped to 255)
```

BASE + 5 — Currents 9–12

```
byte 0–3: CH9–CH12 current (uint8, clamped to 255)
byte 4–7: 0
```

BASE + 6 — Output voltages 1–8

```
byte 0–7: OUT1–OUT8 voltage (raw, (outps[1..8] - 1) / 4)
```

BASE + 7 — Output voltages 9–12

```
byte 0–3: OUT9–OUT12 voltage (raw, (outps[9..12] - 1) / 4)
byte 4–7: 0
```

BASE + 8 — Temperatures 1–8

```
byte 0–7: TEMP1–TEMP8 in °C (uint8)
```

BASE + 9 — Full state 0–7

```
byte 0: o[0].state (system flags: ignition/vbat/unlock)
byte 1: o[1].state
byte 2: o[2].state
byte 3: o[3].state
byte 4: o[4].state
byte 5: o[5].state
byte 6: o[6].state
byte 7: o[7].state
```

BASE + 10 — Full state 8–12

```
byte 0: o[8].state
byte 1: o[9].state
byte 2: o[10].state
byte 3: o[11].state
byte 4: o[12].state
byte 5–7: 0
```

---

2) Keypad LED/control frames (CAN1 by default)

These frames drive the **RGB LEDs** on the CAN keypad.

Keypad ID mapping (LED state frames):

Keypad	CAN ID
1	`0x215`
2	`0x214`
3	`0x213`
4	`0x212`

LED state frame (DLC 8):

```
byte 0: Red bitmask   (bit0=button1 … bit7=button8)
byte 1: Green bitmask (bit0=button1 … bit7=button8)
byte 2: Blue bitmask  (bit0=button1 … bit7=button8)
byte 3–7: 0 (reserved)
```

Keypad sleep command (same IDs):

```
byte 0–2: 0x00 (all RGB off)
byte 3:   0xAA
byte 4:   0x55
byte 5:   0xEE   (sleep command)
byte 6–7: 0x00
```

`` ``` ``

After sending sleep, the PDM sends a **clear frame** (all zeros) so the magic bytes don't persist.

Keypad lighting (Mode 2) frame
Sent only when `keypad_lighting` serial command is used:
```
ID = LED_ID + 1   (0x213–0x216)
DLC = 4
byte 0: LED brightness
byte 1: Backlight brightness
byte 2: Backlight color
byte 3: Persist flag (1=save)
```

In secondary PDM mode (`set_keypad_bus,0`), keypad LED frames are transmitted on **CAN0** instead of CAN1 so the main PDM can merge and forward them.

---

3) Keypad bridge forwarding (optional)

If **keypad bridge** is enabled (`set_keypad_bridge,1`), the main PDM forwards keypad button frames **unchanged** from CAN1 to CAN0:

- **IDs**: `0x192–0x195`
- **Payload**: unchanged (same as keypad source)

---

4) ECU-specific outputs (optional)

Depending on `ecutype`, the PDM can send ECU-specific frames:

Megasquirt (ecutype = 10)
- Sends an extended-ID "set" frame carrying `keybtocanstates` (keypad 1 button mask).
- **ID**: `0x02201838` (29-bit extended ID, MS "set", offset 136)
- **DLC**: 1
- **data[0]:** bitwise **NOT** of `keybtocanstates`
    - `keybtocanstates` bit 0–7 = keypad 1 buttons 1–8
- Intended to drive ECU inputs from keypad buttons.

- **Startup request**: One-time MS “request” frame at boot
    - **ID:** `0x02249838` (offset 137)
    - **DLC**: 3
    - **data[0]** = 7, **data[1]** = `offset >> 3`, **data[2]** = `(offset & 0x07) << 5 | 0x02`

EMU (ecutype = 12)

- **ID 0x662,** DLC 8.
- Encodes momentary keypad states in nibbles (0xF = pressed, 0x0 = released):
    - byte 0: keypad1 buttons 1–2 (upper/lower nibble)
    - byte 1: keypad1 buttons 3–4
    - byte 2: keypad1 buttons 5–6
    - byte 3: keypad2 buttons 1–2
    - byte 4: keypad2 buttons 3–4
    - byte 5: keypad2 buttons 5–6
    - byte 6–7: 0