

DPL Classic & DPL-VU

Manual v2.0.0

Body-Lights boards for Astromech droids

Firmware DPL Firmware_Nano-v1.1
Platform Arduino Nano (ATmega328)

Acknowledgements

The firmware is based on the excellent **AstroCAN BodyLights** codebase by **RealNobser** from the R2 Builders community. A huge thanks to RealNobser for the great code that makes the DPL Classic and DPL-VU boards possible.

Doku v2.0.0 — 2026-04-26

(c) Printed-Droid · alle Rechte vorbehalten / all rights reserved

Disclaimer and Liability Notice

DPL Classic & DPL-VU is a DIY product for Astromech droid builders. Assembly, wiring, configuration and operation involve work that, if done incorrectly, can cause damage to property or persons.

By using this documentation, hardware and firmware you acknowledge:

- You are solely responsible for all work on your system.
- You have basic soldering and electronics knowledge or seek qualified help.
- You verify voltage, polarity and wiring before every power-on.
- You use only matching components (5V supply, correct LED strips, etc.).

Printed-Droid and all contributors accept no liability for damage caused by incorrect wiring, unsuitable components, improper operation, or insufficient safety measures.

Documentation and firmware are provided "as is". If in doubt, seek qualified help before working on the system.

This documentation: (c) **Printed-Droid, all rights reserved**. No permission granted for commercial reuse, duplication or modification without explicit consent. Firmware (AstroCAN BodyLights) by RealNobser, all rights with the original author.

Table of Contents

Disclaimer and Liability Notice	2
Table of Contents	3
DPL Classic & DPL-VU — User Manual	5
■ Quickstart in 5 minutes	5
Table of Contents	5
1. What is DPL Classic / DPL-VU	6
External VU extension for the DPL Classic	8
Other variant: RGB-DPL	10
2. First setup	10
Setup steps	10
3. Hardware connections & pin map	11
3.1 System architecture — DPL as central control	11
3.2 External LED modules	11
3.3 PCB breakouts on the DPL boards	13
3.4 Complete Arduino Nano pin map	14
3.5 Notes for own development	15
4. Flashing the firmware	16
5. Command syntax	16
How to configure a module — step by step	17
6. Modules and tokens	17
Main modules	17
DPL sub-areas (part of the DPL's MAX7219 matrix)	18
7. Animation reference per module	18
DP (DPL — container, coordinates all sub-areas + VU)	18
CB (CBI)	18
DC (DCBI)	19
LD (LDPL)	19
UA (UAL)	19
VU (VUBar)	19
DPL sub-areas	20
8. Configuring the voltage monitor	20
Activate	20

Voltage divider population	20
Set thresholds	21
Display	21
9. Door sensors (LeftDoor / RightDoor)	21
How to activate the switches	22
What if no switches and no jumper are connected?	22
Wiring tip	23
10. System commands	23
11. Troubleshooting	23
Command is not accepted (no "OK" response)	23
LEDs do not light up	23
Wrong colors (e.g. red shows as green)	23
VU bar does not respond to sound	24
Voltage display shows wrong value	24
EEPROM settings lost after firmware update	24
12. FAQ	24
13. Writing your own firmware	25
14. Cheat sheet (quick reference)	26
Module tokens	26
Command format	26
Parameters	26
System commands	26
Common animations (AN number)	26
Command examples (copy-paste)	27
Pin map (on-board Arduino Nano)	27

DPL Classic & DPL-VU — User Manual

Firmware DPL Firmware_Nano-v1.1 · Platform Arduino Nano (ATmega328) · Doc 2.0.0 · 2026-04-26

This manual is for users of the Printed-Droid DPL Classic and DPL-VU boards. It covers setup, hardware connections, operation, all commands, all animations and troubleshooting.

■ Quickstart in 5 minutes

If you've just unpacked the board and want to see it running:

1. **5V** to the **5V IN** screw terminal — a USB cable from your PC is enough
2. **USB cable** to the on-board Arduino Nano
3. **Bridge the door switch headers** — solder a short wire jumper between the two pins of both **Left Door** AND **Right Door** headers (otherwise DPL and CBI stay dark!)
4. **Flash the firmware** — load **DPL Firmware_Nano-v1.1.hex** with XLoader (see *Section 4*)
5. **Open a serial terminal** at **115200 baud, line ending CR** and send e.g. **/DP/BR/12** → DPL lights up

Working? Great. For details, external modules (CBI/LDPL/UAL), animations and the voltage monitor, read on.

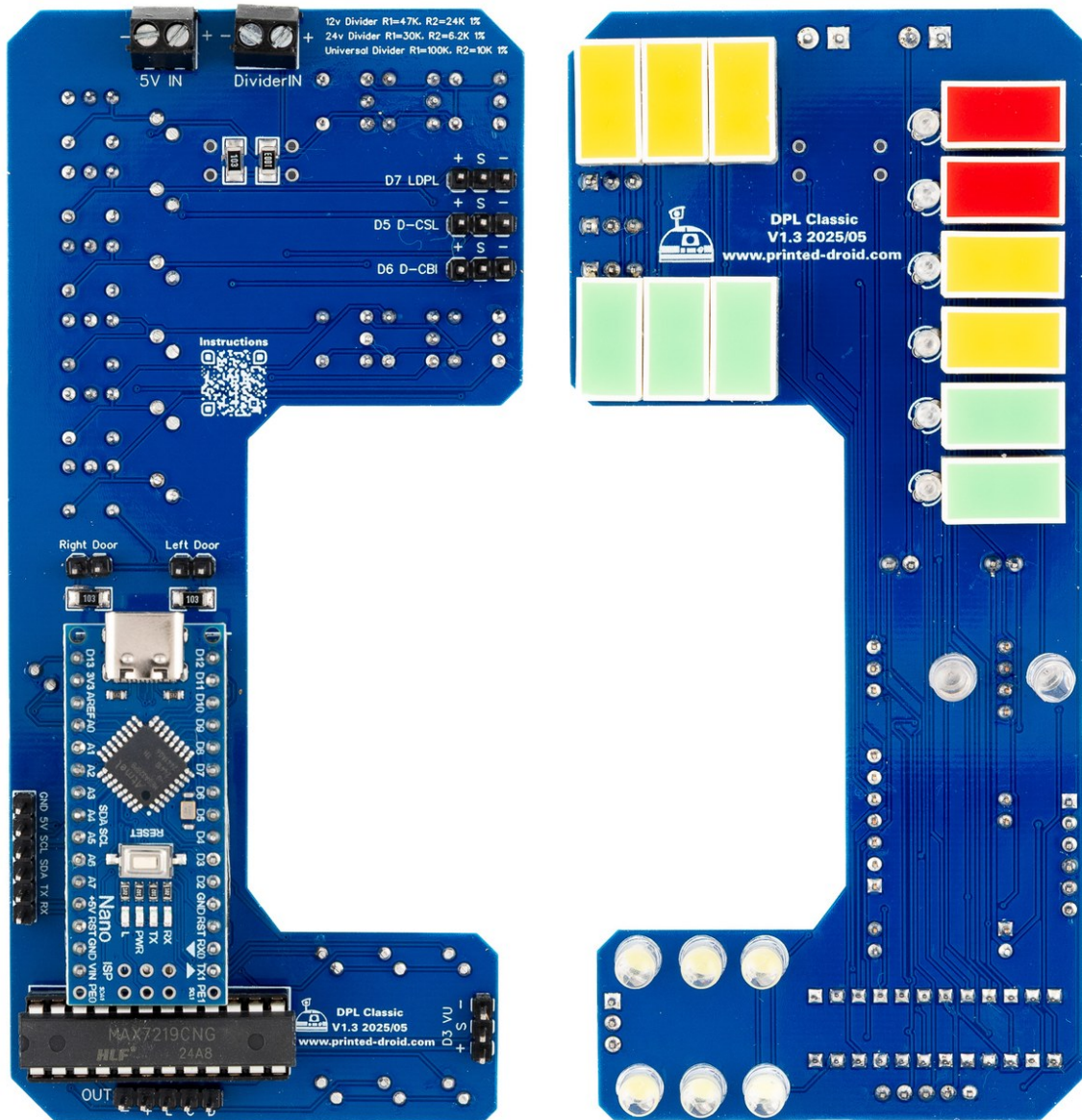
Not working? → *Section 11 Troubleshooting*, check the door switch jumpers first.

Table of Contents

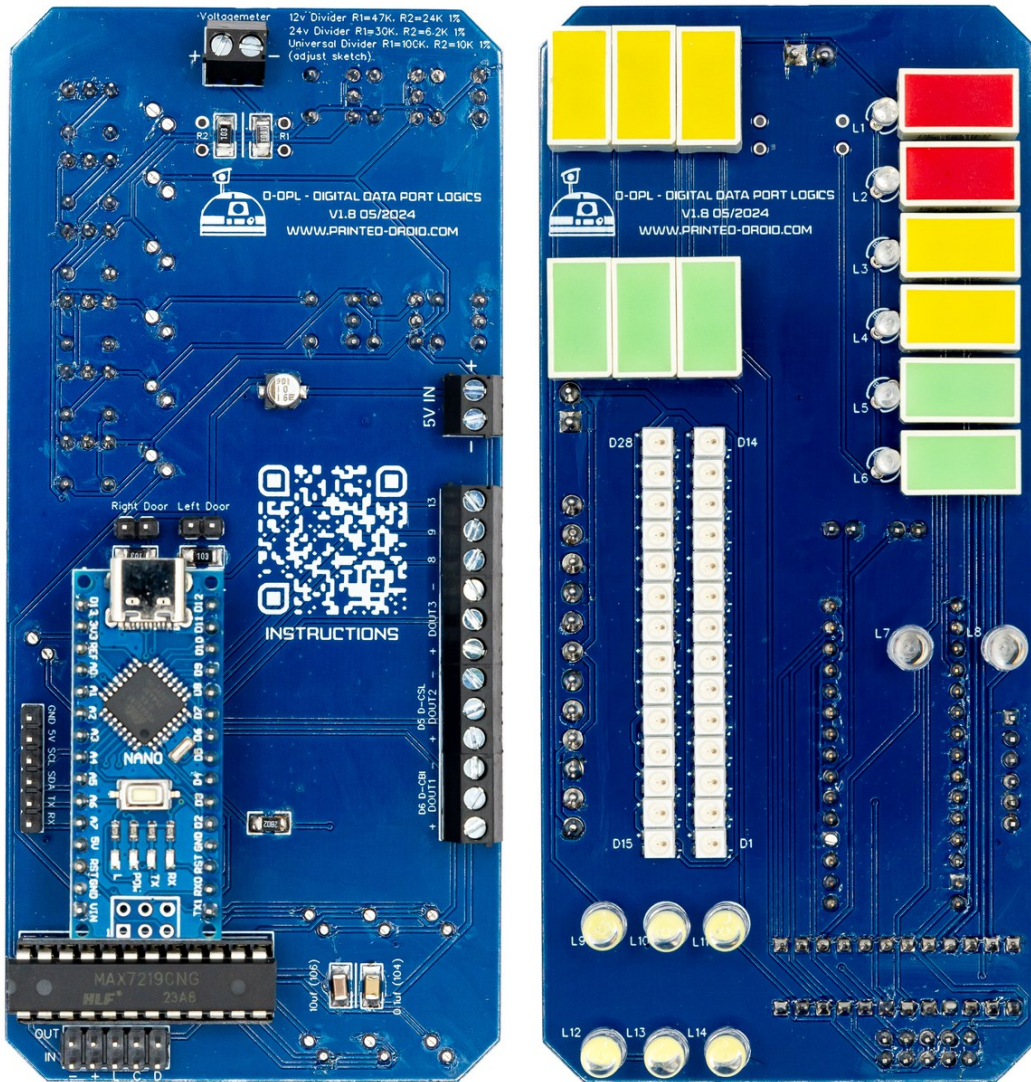
1. *What is DPL Classic / DPL-VU*
 2. *First setup*
 3. *Hardware connections & pin map*
 4. *Flashing the firmware*
 5. *Command syntax*
 6. *Modules and tokens*
 7. *Animation reference per module*
 8. *Configuring the voltage monitor*
 9. *Door sensors (LeftDoor / RightDoor)*
 10. *System commands*
 11. *Troubleshooting*
 12. *FAQ*
 13. *Writing your own firmware*
 14. *Cheat sheet (quick reference)*
-

1. What is DPL Classic / DPL-VU

Both are LED controller boards for Astromech droids (R2-D2 and similar). They replace older single-purpose boards for DPL, CBI, CSL, UAL with a combined board that has an Arduino Nano on-board.



DPL Classic — back side (left) with connectors, front side (right) with the DPL LEDs.



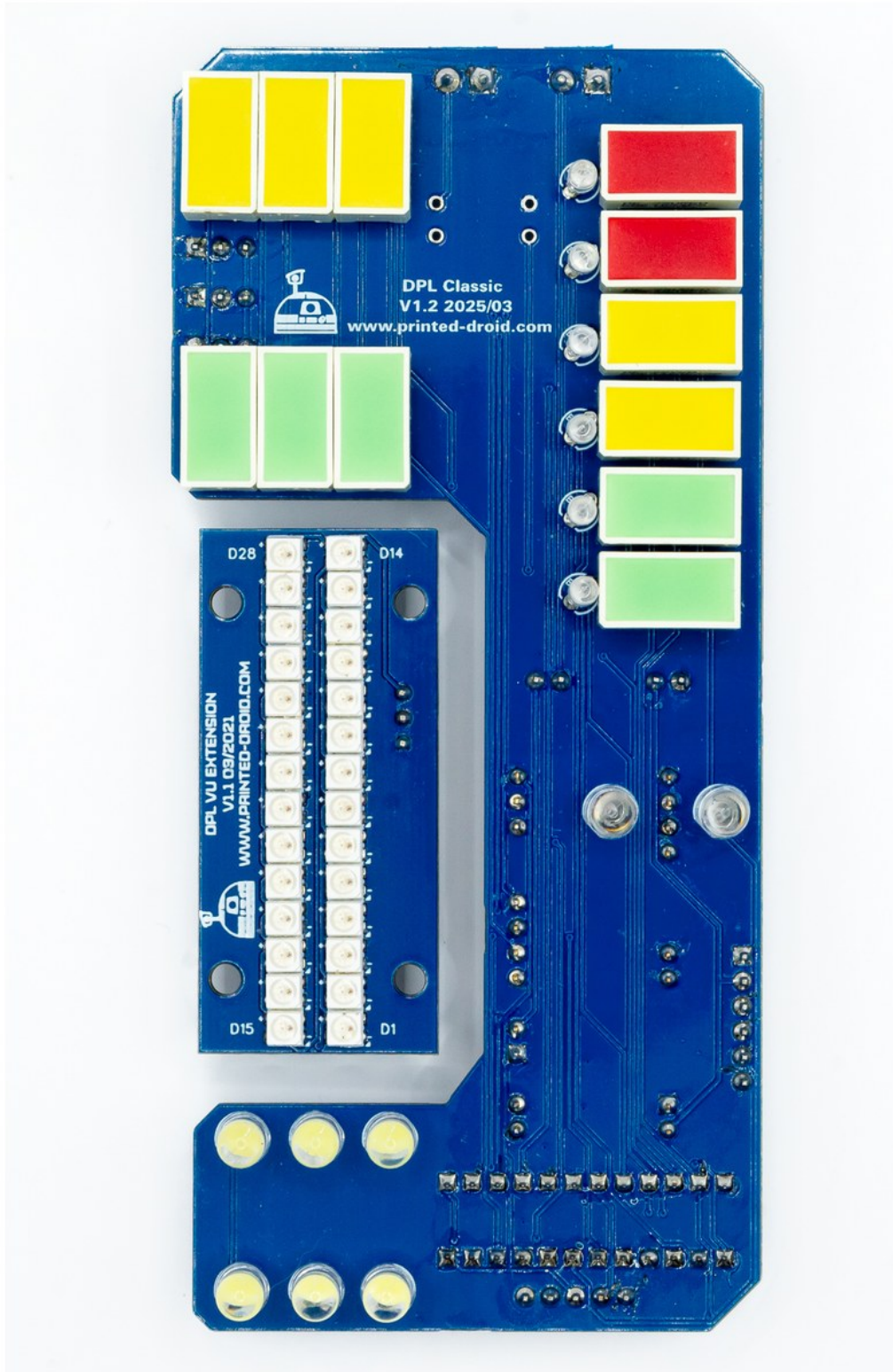
DPL-VU — back side (left) with connectors, front side (right) with DPL LEDs plus the integrated VU row (28 WS2812B in the middle area, D28-D14 / D15-D1).

Feature	DPL Classic	DPL-VU
Microcontroller	Arduino Nano on-board	Arduino Nano on-board
MAX7219 driver for DPL front-side LEDs	■	■
Connector for external CBI Plus / D-CBI	■	■
Connector for LDPL strip	■	■
Connector for UAL strip	■	■
VU LEDs populated on board	■	■ (28 LEDs in 2x 14-row layout)
External VU extension can be soldered to D3	■	■ (not needed)
Sound-reactive effects	■ (see note)	■ (see note)
4 voltage thresholds (battery indicator)	■	■
Door switch inputs	■	■
USB upload via on-board Nano	■	■

Important note about the "VU" display: The name "VU" suggests audio reactivity, but **the firmware does not use any audio input**. The effects (VU meter, BPM pulse, juggle sweep) are based on a synthetic 62 BPM sine wave. It looks like a VU meter but does not respond to actual sound.

External VU extension for the DPL Classic

For the DPL Classic, a separately available **VU Extension** (Printed-Droid) is offered:



DPL Classic with the external VU extension attached (28 WS2812B). The extension connects to the 3-pin header [D3 VU](#) on the DPL Classic — the firmware drives the pin anyway, no firmware update required.

Other variant: RGB-DPL

If you want **all panels in full RGB color** instead of the MAX7219 mix documented here, Printed-Droid also offers the **RGB-DPL** board. It uses WS2811/WS2812 LEDs throughout (DPL, CBI, LDPL, CSL) and ships with its own firmware (profiles, color schemes, personality modes). Different command syntax (`BRIGHTNESS 50` instead of `/DP/BR/15`).

Documentation & firmware: <https://github.com/PrintedDroid/RGB-DPL-Firmware>

2. First setup

The DPL board is the **central control unit**. All other modules (CBI Plus, D-CBI, LDPL, UAL, optional VU extension) connect to it — see *Section 3* for details.

The DPL has two door-switch inputs (`Left Door`, `Right Door`). As long as these are **not** connected, the firmware keeps **DPL and CBI dark** — as if the hatches were closed. **That is intentional**, but accounts for 80% of all "doesn't work" reports.

So when testing for the first time: either connect reed switches, **or** set a **wire jumper** between the two pins of each header. Details in *Section 9*.

Setup steps

1. Power supply

5V to the `5V IN` screw terminal (e.g. from your R2's 5V rail). The board runs **exclusively on 5V** — no 12V on this input!

2. Optional **battery voltage** (12V / 24V battery voltage) to the `Voltage IN` screw terminal. **This is ONLY a measurement input** for the battery indicator, not an additional power feed.

3. **Current draw**: With default brightness levels, a **standard USB cable directly from your PC** is sufficient as the only power source — even with DPL + analog CBI (CBI Plus) + digital D-CBI all connected at the same time. Only at very bright setups (all WS2812B strips at full brightness) does a separate 5V power supply with a few amps make sense.

4. **Connect LED modules** (all optional, depending on your build)

CBI Plus (analog CBI with its own MAX7219) → 5-pin header `-/+L/C/D` on the DPL

5. **D-CBI** (digital CBI with WS2812B) → 3-pin header `D6 D-CBI` (5V / Signal / GND)

6. **LDPL** strip → 3-pin header `D7 LDPL` (5V / Signal / GND)

7. **UAL** strip → signal-only pin `D8` (5V/GND from a free power pin)

8. For DPL Classic with retrofitted external **VU Extension** → 3-pin header `D3 VU`

9. **Door switches** (see warning above — DPL and CBI stay dark otherwise!)

Reed switch / micro-switch / hall sensor to the 2-pin headers `Right Door` and `Left Door`

10. Or for testing: wire jumper between the two pins of each header

11. Details and all variants in *Section 9*

12. **Plug a USB cable into the on-board Arduino Nano** (for configuration via serial terminal)

13. First function check

On first start, a boot sequence (DPL startup animation) plays

14. After that, all modules run with their default animations

15. If nothing lights up: check the door switches or send `/RESTART`

3. Hardware connections & pin map

This section first explains the system architecture (which modules exist, how they connect), then the actual connectors on the DPL board, and finally the complete Nano pin map for users with their own code.

3.1 System architecture — DPL as central control

The DPL board is the **master board** with on-board Arduino Nano. All other LED modules are separate boards that connect to the DPL:

External module	Connector on the DPL	Command token
D-CBI (digital Charge Bay Indicator, WS2812B)	3-pin header D6 D-CBI (5V / Signal / GND)	<code>/DC/...</code>
CBI Plus (analog Charge Bay Indicator, MAX7219)	5-pin header <code>-/+/L/C/D</code>	<code>/CB/...</code>
LDPL (Large DataPanel Logics)	3-pin header D7 LDPL (5V / Signal / GND)	<code>/LD/...</code>
UAL (Utility Arm Lights)	Signal-only pin D8 (5V/GND separate)	<code>/UA/...</code>
VU Extension (DPL Classic only)	3-pin header D3 VU (5V / Signal / GND)	<code>/VU/...</code>

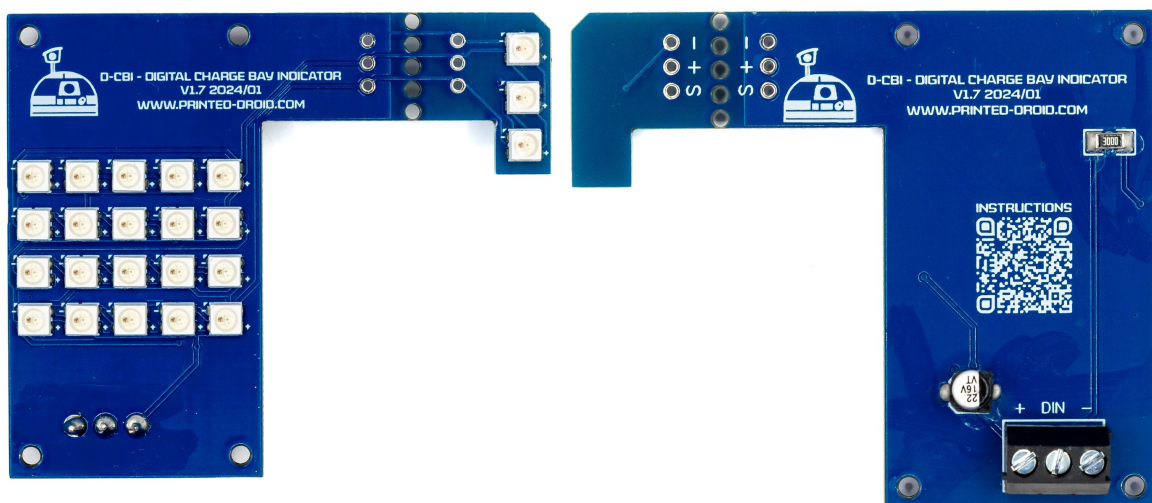
What this means for usage:

- You don't need to have all modules connected — the DPL works standalone
- If an external module is not connected, the corresponding animation simply runs into the void — no error, nothing visible
- CBI Plus and D-CBI can also be connected **at the same time** — they sit on different connectors and are configured via separate tokens (`/CB/...` for CBI Plus, `/DC/...` for D-CBI)

3.2 External LED modules

The following modules are sold separately by Printed-Droid and connect to the DPL:

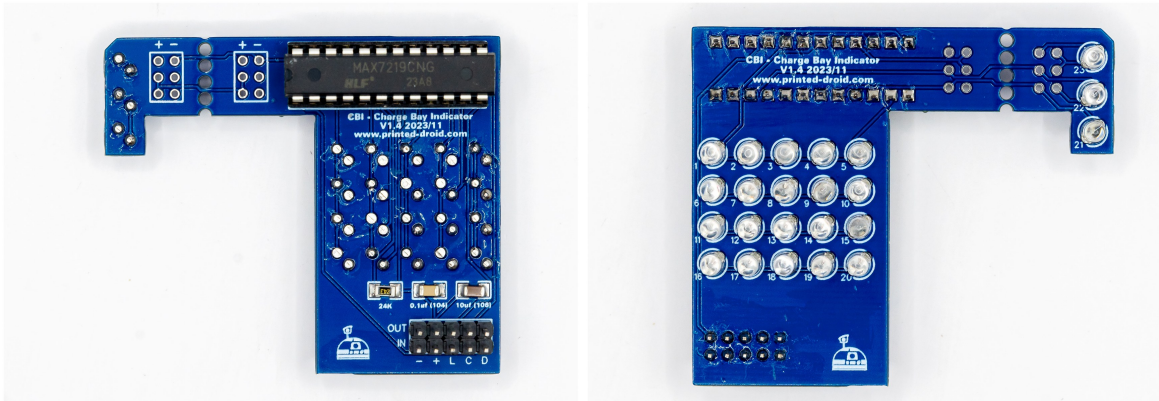
D-CBI (digital Charge Bay Indicator)



D-CBI — back side (left) with DIN screw terminal (+ DIN -) for the main connection to the DPL. Front side (right) with 23 WS2812B LEDs (20 main area + 3 status LEDs).

- **Full RGB color depth:** WS2812B LEDs allow **any color** (16.7 million colors per LED) — base color via `/DC/CO/R/G/B` freely selectable, animations like rainbow, color-fade, heart-beat etc. use the full color palette
- Connects to the DPL via the 3-pin terminal `D6 D-CBI` (5V / Signal / GND), or via the DIN screw terminal on the D-CBI board
- Commands: `/DC/...` (see *Section 6*)

CBI Plus (analog Charge Bay Indicator)



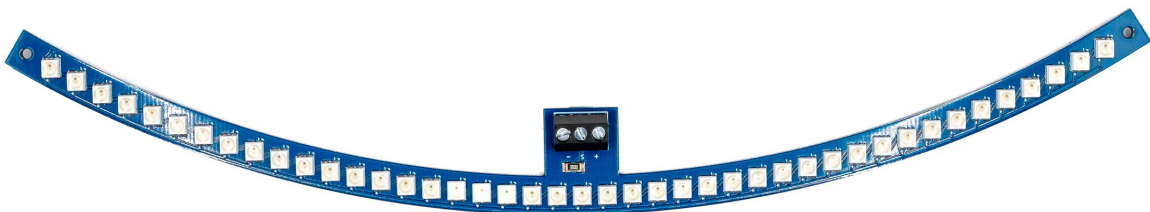
CBI Plus — back side (left) with its own MAX7219CNG + 5-pin `IN/OUT` header for daisy-chain. Front side (right) with the classic 3 mm LED indicators.

- **Fixed individual colors:** The front is populated with classic analog 3 mm LEDs — each LED has its color hard-soldered (red/yellow/green/blue in their respective areas). There is no RGB control; the `/CB/CO/...` command does not affect individual LEDs
- Connects to the DPL via the 5-pin header `-/+/L/C/D`
- Commands: `/CB/...`

Which CBI module? If you want the classic R2 look with fixed LED colors, choose **CBI Plus**. If you want more flexibility with colors and animations, choose **D-CBI**. Both can also be operated in parallel — they hang on different pins and are configured separately.

Snap-off PCB area for very old R2 bodies: Both D-CBI and CBI Plus have an area with a perforated line ("break line") where the PCB can be split into two halves if needed (for very old R2 body generations with different layout). After splitting, the two halves are reconnected electrically using the two adjacent 3-pin headers (top of the back, each labeled `+ S -`) with short wires or pin jumpers. If you keep the standard PCB form factor, leave this area unchanged.

LDPL (Large DataPanel Logics)



LDPL — curved WS2812B strip with 43 LEDs for the belly hatch lighting. Center has a 3-pin screw terminal (`- S +`) for connection to `D7` of the DPL.

- Connects via the 3-pin terminal `D7 LDPL` on the DPL
- Commands: `/LD/...`

UAL (Utility Arm Lights)

- WS2812B strip with 19 LEDs for the utility arms
- Connects via the **signal-only pin D8** on the DPL — get 5V/GND from a free power pin (e.g. I2C header or MAX7219 header)
- Commands: `/UA/...`

VU Extension (DPL Classic only)

The DPL-VU has the VU LEDs on board already. The DPL Classic can optionally be upgraded to VU — see picture in *Section 1* (DPL Classic with attached VU extension). Connects to the 3-pin header [D3](#) [VU](#) on the DPL Classic.

3.3 PCB breakouts on the DPL boards

Both DPL variants (Classic + VU) expose the same functions, but in different form: on the DPL-VU the LED outputs are **screw terminals**, on the DPL Classic only Power is a screw terminal — all other connectors are **pin headers**.

Function	Nano pin(s)	DPL-VU	DPL Classic
Power In: 5V + GND	—	Screw terminal	Screw terminal
Voltage In (battery measurement, NOT power!)	A0 via voltage divider	Screw terminal	Screw terminal
3-pin LED-Out DCBI (5V / Signal / GND)	D6	Screw terminal	Pin header
3-pin LED-Out (D5, reserved) (5V / Signal / GND)	D5	Screw terminal	Pin header
3-pin LED-Out LDPL (5V / Signal / GND)	D7	Screw terminal	Pin header
Signal-only UAL	D8	Pin header	Pin header
Signal-only free #1	D9	Pin header	Pin header
Signal-only free #2	D13	Pin header	Pin header
Door switch Left (D2 + GND)	D2	2-pin header	2-pin header
Door switch Right (D4 + GND)	D4	2-pin header	2-pin header
I2C breakout (5V / SDA / SCL / GND)	A4 (SDA), A5 (SCL)	Pin header	Pin header
UART breakout (5V / RX / TX / GND)	D0 (RX), D1 (TX)	Pin header	Pin header
MAX7219 / CBI-Plus daisy-chain : -, +, L, C, D	GND, 5V, D12, D11, D10	5-pin header	5-pin header

Pin map of the MAX7219 / CBI header (5-pin row for daisy-chain to CBI Plus):

Header label	Meaning	Nano pin
-	GND	GND
+	5V	5V
L	LOAD (CS)	D12

Header label	Meaning	Nano pin
C	CLOCK	D11
D	DATA	D10

3.4 Complete Arduino Nano pin map

This table is helpful for users who want to write their **own firmware** for the board. It shows every pin of the Arduino Nano, what is connected to it, and which pins are unused by the original firmware.

Nano pin	Default function (Arduino)	DPL firmware usage	Component on PCB	Free for own code?
D0	UART RX	—	RX/TX pin header	■ shared with USB serial
D1	UART TX	—	RX/TX pin header	■ shared with USB serial
D2	INT0	SWITCH_LEFTDOOR (INPUT_PULLUP)	2-pin door-switch header (left / data port door)	only if doors logic not needed
D3	INT1, PWM	VU_DATA (WS2812B Out)	VU LED row (DPL-VU on-board) or 3-pin header D3 VU for external VU extension (DPL Classic)	only if no VU LEDs / external VU
D4	—	SWITCH_RIGHTDOOR (INPUT_PULLUP)	2-pin door-switch header (right / charge bay door)	only if doors logic not needed
D5	PWM	reserved WS2812B-Out	3-pin connector (5V / Signal / GND)	■ free for own code
D6	PWM	DCBI_DATA (WS2812B Out)	3-pin connector D6 D-CBI (5V / Signal / GND)	only if no DCBI strip
D7	—	LDPL_DATA (WS2812B Out)	3-pin connector D7 LDPL (5V / Signal / GND)	only if no LDPL strip
D8	—	UAL_DATA (WS2812B Out)	Signal-only pin (power separate)	only if no UAL strip
D9	PWM	— (unused)	Signal-only pin	■ free
D10	SPI SS, PWM	MAX7219_DATA (bit-bang)	MAX7219 header pin D (to CBI-Plus daisy-chain)	only if no MAX7219 displays (DPL LEDs / CBI Plus)
D11	SPI MOSI, PWM	MAX7219_CLOCK (bit-bang)	MAX7219 header pin C	only if no MAX7219 displays
D12	SPI MISO	MAX7219_LOAD (bit-bang CS)	MAX7219 header pin L	only if no MAX7219 displays
D13	SPI SCK, LED_BUILTIN	— (unused)	Signal-only pin	■ free (caution: on-board LED on this pin)

Nano pin	Default function (Arduino)	DPL firmware usage	Component on PCB	Free for own code?
A0	ADC	<code>ANALOGPIN</code> (voltage divider)	Voltage divider 100kΩ/10kΩ (SMD; THT pads beside for custom values) → Voltage In terminal	only if no voltage measurement
A1	ADC	—	—	■ free
A2	ADC	—	—	■ free
A3	ADC	—	—	■ free
A4	ADC, I2C SDA	—	I2C pin header	■ free (intended for I2C extensions)
A5	ADC, I2C SCL	—	I2C pin header	■ free (intended for I2C extensions)
A6	ADC only	—	—	■ free (note: analog only, no digital I/O)
A7	ADC only	—	—	■ free (note: analog only, no digital I/O)
RST	Reset	—	Reset pad / auto-reset via USB	(hardware reset)
5V	Power bus	—	5V rail (fed from 5V IN terminal)	—
3V3	Power Out (50 mA max)	—	only available at the Nano pin	—
GND	Ground	—	GND rail	—
VIN	Power In (raw, Nano pin)	—	do not use for battery voltage — the DPL's "Voltage In" terminal is SEPARATE, ADC measurement input only	—
AREF	ADC reference	— (default = AVcc)	—	change for custom ADC scaling

3.5 Notes for own development

- **MAX7219 occupies D10/D11/D12 as bit-bang SPI** (LedControl library, no hardware SPI). This means hardware SPI is not usable without conflicts. Two options for additional SPI devices: Remove LedControl and use hardware SPI on D10 (SS) / D11 (MOSI) / D13 (SCK) — D12 would then be MISO.
- Use software SPI / I2C / UART for additional devices.

D13 has the on-board user LED of the Nano. If you use D13 as input or as output that contradicts the LED behavior, the LED may flicker along, or it acts as a ~1 kΩ pull-down. For sensor inputs, prefer A1-A3.

A6/A7 are ADC-only — they cannot be used as digital I/O (hardware limitation of the ATmega328P in the TQFP package).

D0/D1 (RX/TX) are reserved for the serial console. If you write your own firmware and don't need a serial console, they are free as digital I/O pins. **Important:** the USB upload protocol uses RX/TX — after flashing

there is no conflict, but during upload these pins must not be driven externally.

GND distribution: All GND pins (screw terminal, headers, door switches, MAX7219 header, I2C, UART) are connected together — your own devices can use any GND pin.

4. Flashing the firmware

Both boards use the same firmware. The current file is named: `DPL Firmware_Nano-v1.1.hex`

Recommended tools:

- **XLoader** (Windows) — recommended by Printed-Droid (<https://www.printed-droid.com/kb/dpl-classic-data-port-logics/> → Downloads)
- **avrdude** (Linux/macOS) — command line

XLoader configuration:

- Hex file: `DPL Firmware_Nano-v1.1.hex`
- Device: **Nano (ATmega328)**
- COM port: the port of the connected Nano (check Device Manager)
- Baud rate: **115200** (some older Nanos need 57600)
- Click Upload — the Nano blinks during upload, finished in approx. 5-10 seconds

After flashing:

- EEPROM settings are usually preserved
- If a firmware update changes the internal memory layout, user settings (brightness, colors, animations) are reset to defaults automatically — this will be noted in the changelog of the affected version

5. Command syntax

Commands are sent via the **serial interface** of the Arduino Nano:

- **Baud rate:** 115200
- **Line ending:** **CR (\r)** — some terminals send only LF (\n), which is ignored. CRLF works (LF is ignored, CR triggers parsing).
- **Maximum command length:** 32 characters including slashes

Format:

```
/MODULE/PARAMETER/VALUE [ /VALUE2/VALUE3 . . . ]
```

If the board responds with `OK`, the command was accepted. No response means: the command is malformed or the module does not exist in the current firmware.

Examples:

Command	Effect
<code>/DP/BR/15</code>	DPL brightness 15 (scale 0–15 for MAX7219)
<code>/LD/BR/100</code>	LDPL brightness 100 (scale 0–255 for WS2812B)
<code>/LD/CO/255/0/0</code>	LDPL base color red

Command	Effect
<code>/CB/AN/3</code>	CBI heart animation
<code>/DC/AN/6</code>	DCBI rainbow animation
<code>/DP/VL/11.5/12.0/12.5/13.0</code>	Voltage thresholds red/yellow/green/charge
<code>/DP/RR/100000.0/10000.0</code>	R1=100kΩ, R2=10kΩ (default)
<code>/DP/VM/1</code>	Activate voltage monitor
<code>/RESTART</code>	Soft reboot
<code>/FACTORY</code>	Reset EEPROM to defaults + reboot

How to configure a module — step by step

Example: make **LDPL** run in a glowing red at medium brightness.

1. Set brightness:

→ response: `OK`. Brightness range is 0-255 for WS2812B modules.

2. Set base color (R, G, B, 0-255 each):

→ pure red. Examples: `/LD/CO/255/255/0` = yellow, `/LD/CO/0/0/255` = blue.

3. Pick an animation:

→ `fadeOutAnimation` (Knight-Rider with fade). Other options in *Section 7*.

4. **Done.** Settings are saved to EEPROM automatically and survive a reboot. The next time you power up, LDPL starts straight away with these values.

The same pattern works for **all modules**: `BR` for brightness, `CO` for color (WS2812B modules only — D-CBI, LDPL, UAL, VU), `AN` for animation.

Tip: If you've configured something into a corner and lost track, just send `/FACTORY`. That resets everything to defaults and you start fresh.

6. Modules and tokens

The firmware knows these modules:

Main modules

Token	Module	Where physically?	Hardware driver	Brightness scale
<code>DP</code>	Data Port Lights (DPL) — container for all sub-areas + VU	On the DPL board itself	on-board MAX7219 (<code>dev0</code>)	0–15
<code>CB</code>	Charge Bay Indicator (CBI)	External CBI Plus module (on the MAX7219 header)	CBI Plus's own MAX7219 (<code>dev1</code>) via daisy-chain	0–15
<code>DC</code>	Digital Charge Bay Indicator (D-CBI)	External D-CBI module (on D6)	WS2812B 23 LEDs	0–255
<code>LD</code>	Large DataPanel Logics (LDPL)	External LDPL strip (on D7)	WS2812B 43 LEDs	0–255
<code>UA</code>	Utility Arm Lights (UAL)	External UAL strip (on D8)	WS2812B 19 LEDs	0–255

Token	Module	Where physically?	Hardware driver	Brightness scale
VU	VU Bar	DPL-VU on-board / for Classic optional VU Extension on D3	WS2812B 28 LEDs	0–255

What happens with modules that aren't connected? The firmware drives all pins regardless of whether a module is connected. If e.g. CBI Plus is missing, the `/CB/...` animation only runs "in code" — no error message, nothing visible. That's OK and no reason for concern.

DPL sub-areas (part of the DPL's MAX7219 matrix)

These tokens control individual rows/columns of the DPL matrix in isolation. Normally not needed because `/DP/AN/...` coordinates all sub-areas — but useful for special effects:

Token	Area	Position on the DPL matrix
TB	Top Bar	Rows 4+5
BL	Blue Lights	Row 0
BG	Bar Graph	Rows 2+3 (left block)
RL	Red Lights	Rows 2+3 column 6
WL	White Lights	Row 1

7. Animation reference per module

Each module has its own list of animations. Set with `/MODULE/AN/NUMBER`.

DP (DPL — container, coordinates all sub-areas + VU)

AN	Name	Description
0	none	Everything off
1	defaultAnimation	All sub-areas in default mode, VU cycling through VU effects
2	randomVUAnimation	Sub-areas in default, VU in random bargraph mode
3	startupAnimation	Boot sequence (default after first power-on or <code>/FACTORY</code>)

CB (CBI)

AN	Name	Description
0	none	Off
1	defaultAnimation	Standard animation sequence
2	randomAnimation	Random LED pattern
3	heartAnimation	Heart symbol sequence

AN	Name	Description
4	chargingAnimation	Charging symbol (automatically activated when the measured voltage rises above the charge threshold)

DC (DCBI)

AN	Name	Description
0	none	Off
1	defaultAnimation	Standard sequence in the configured base color (<code>/DC/CO/...</code>)
2	randomAnimation	Random LED pattern
3	heartAnimation	Heart symbol
4	randomFadeAnimation	Random points with fade-out
5	colorFadeAnimation	Random LEDs in rainbow colors with fade-out
6	rainbowAnimation	Classic rainbow gradient
7	heartBeatAnimation	Pulsating heartbeat
8	chargingAnimation	Charging symbol

LD (LDPL)

AN	Name	Description
0	none	Off
1	defaultAnimation	Knight-Rider scanner (hard, no fade)
2	fadeOutAnimation	Knight-Rider with fade-out (default)
3	rainbowAnimation	Scanner in rainbow colors

UA (UAL)

AN	Name	Description
0	none	Off
1	defaultAnimation	Knight-Rider with fade in the configured base color (<code>/UA/CO/...</code> , default)
2	movieAnimation	Movie sequence: 18 LEDs in the base color, then a green LED at the end. Stops automatically after position 26.
3	abortAnimation	Red LED at position 18 as abort indicator

Note: On the Printed-Droid boards there is no dedicated UAL switch pin. The movie animation is therefore **only triggered by serial command** (`/UA/AN/2`), not automatically by a hardware trigger.

VU (VUBar)

AN	Name	Description
0	none	Off
1	defaultAnimation	Automatically cycles every 5 seconds through three effects: VU meter, BPM pulse, juggle sweep
2	randomVUAnimation	Classic VU bargraph with random drift, green/yellow/red depending on level

DPL sub-areas

All sub-areas have at least **0** (none) and **1** (defaultAnimation):

Token	AN 0	AN 1	AN 2	AN 3
TB	none	defaultAnimation	—	—
BL	none	defaultAnimation	staticTimeAnimation	bargraphAnimation
BG	none	defaultAnimation	—	—
RL	none	defaultAnimation	—	—
WL	none	defaultAnimation	—	—

Note: When `/DP/AN/1` is set, the DP container overrides the sub-animations. Direct intervention (`/BL/AN/2` etc.) is overwritten on the next DP animation change.

8. Configuring the voltage monitor

The DPL board can measure an external battery voltage and indicate the charge state. **Important:** The `Voltage IN` measurement input is **separate** from the `5V IN` supply — the board runs on 5V, and the Voltage IN input is exclusively a high-impedance measurement input (via voltage divider to A0).

The status display runs through the CBI Plus (three MAX7219 LEDs in column 5) and/or the D-CBI (three dedicated WS2812B LEDs).

Activate

```
/CB/VM/1    Voltage monitor on
/CB/VM/0    Voltage monitor off
```

`VM` acts globally on all modules with VCC functionality (CBI, DCBI). The token can also be `/DP/VM/1` etc. — the effect is the same.

Voltage divider population

On the Printed-Droid boards (DPL Classic + DPL-VU) the voltage divider is **factory-populated with R1 = 100 kΩ and R2 = 10 kΩ as SMD components** and is suitable for 0 – ~30 V battery measurement. These default values are pre-configured in the firmware — you usually do **not** need to change them.

If different values are needed (e.g. to improve the resolution at higher voltages or to better resolve very low voltages):

1. **Remove the SMD resistors** — carefully de-solder the two SMD components (R1 and R2) on the PCB or tilt them off with a fine soldering iron

2. **Solder your own THT resistors** — directly **next to the SMD pads** there are through-hole pads (THT) for standard leaded resistors. Solder any values there

3. **Adjust the software scaling** — tell the firmware about the new values with

`/DP/RR/<R1_in_Ohm>/<R2_in_Ohm>` (or `/CB/RR/...` etc.), otherwise the displayed voltage will be wrong

```
/DP/RR/100000.0/10000.0    R1=100k, R2=10k (default – for SMD population)
/DP/RR/220000.0/22000.0    Example: for a higher voltage range
```

Important: The `/RR` command only changes the software scaling (what the firmware computes from the ADC value), **not** the hardware. If the SMD resistors are still on and you set `/RR` to different values, the firmware will display nonsense. If you accidentally messed up the values, run `/FACTORY` — that restores the defaults.

Sizing: Whatever the battery voltage, the ADC pin (A0) must see at most 5 V. Formula: $V_{ADC} = V_{battery} \times R2 / (R1+R2)$. With default values: $12\text{ V} \times 10 / (100+10) = 1.09\text{ V}$ ✓; $24\text{ V} \times 10 / (100+10) = 2.18\text{ V}$ ✓. Always plan for headroom below 5 V.

Set thresholds

```
/DP/VL/11.5/12.0/12.5/13.0    red / yellow / green / charge
```

Threshold	Meaning
red	Below this voltage: red LED on (critically low)
yellow	Between red and yellow: yellow
green	Between yellow and green: green (normal)
charge	Above this voltage: charging animation active (charger detected)

The default values (11.5/12.0/12.5/13.0) are tuned for a 12V lead-acid battery. For 3S LiPo (~10.5–12.6V operating voltage) the thresholds should be set lower, e.g. `/DP/VL/10.8/11.4/12.0/12.6`.

Display

- **On CBI Plus (when connected):** three dedicated LEDs in column 5 of the MAX7219 matrix (red/yellow/green)
- **On D-CBI (when connected):** three dedicated WS2812B LEDs (index 20=green, 21=yellow, 22=red)
- **Charging animation** is started automatically when the voltage rises above `charge` (e.g. charger connected). When it drops below `charge`, the firmware switches back to `defaultAnimation`.

9. Door sensors (LeftDoor / RightDoor)

The board has two door switch inputs on dedicated 2-pin headers. Both have internal pull-ups — a switch only needs to **pull the pin to GND**.

Input	Header	Nano pin	Pin LOW (to GND)	Pin HIGH (not connected)
LeftDoor (data port door)	2-pin left	D2	Door open → DPL + VU run	Door closed → DPL + VU all off

Input	Header	Nano pin	Pin LOW (to GND)	Pin HIGH (not connected)
RightDoor (charge bay door)	2-pin right	D4	Door open → CBI + DCBI run	Door closed → CBI + DCBI all off

Logic: LOW = "door open" = LEDs on. HIGH = "door closed" = LEDs off. Saves power and prevents light leakage through closed hatches.

How to activate the switches

There are several options depending on your droid build:

Option 1 — Reed switch with magnet (recommended, R2 builders standard)

- Use a **NO (Normally Open)** reed contact — closes when a magnet comes near
- Wiring at the 2-pin header: one pin to D2 (or D4), the other to GND — polarity doesn't matter, a reed is not directional
- Position the magnet so that it **sits next to the reed when the door is open** → reed closes → pin = LOW → DPL runs
- When the door closes, the magnet moves away → reed opens → pin = HIGH → DPL off

Tip — printed parts: On Makerworld / Thingiverse there are ready-made magnet and reed holders for the R2 doors that fit directly onto the servo mechanism.

Option 2 — Mechanical micro switch

- A switch with a lever, actuated by the door mechanism
- Wiring at the header: NO contact between D2/D4 and GND
- Door open: lever pressed → pin = LOW
- Door closed: lever released → pin = HIGH

Option 3 — Hall effect sensor

- Digital hall sensor (e.g. A3144 with open-drain output)
- VCC = 5V from the board, GND = GND, output to header at D2/D4
- Magnet detection like a reed but contact-less and more durable

Option 4 — External controller / servo controller

- If another microcontroller (e.g. body master, bridge) knows the door state, it can pull D2/D4 directly to GND or leave it floating
- Caution: both sides must share the same GND

Option 5 — Skip the switch entirely, LEDs always on

- Solder a wire jumper between the D2 (or D4) pin and GND on the 2-pin header → pin permanently LOW → DPL/CBI run continuously (even with closed door)
- Useful for show / display setups without movable doors, or for testing

What if no switches and no jumper are connected?

Pins stay HIGH due to internal pull-up → DPL and CBI are **always off**. That's the as-shipped state. On first setup: either connect switches or set a jumper, otherwise nothing lights up.

Wiring tip

For typical R2 wiring it's practical to bridge the two GND pins of both door-switch headers and only run two signal wires to the doors. If you have fixed reed holders with JST connectors, just solder matching JST sockets onto the 2-pin headers.

10. System commands

These commands have no module token, only the slash prefix:

Command	Effect
<code>/RESTART</code>	Soft reboot of the Arduino Nano. EEPROM is preserved. Useful after settings changes that require a restart.
<code>/FACTORY</code>	Reset EEPROM to defaults, then restart. All user settings are lost. Useful if the configuration is corrupt or after a firmware update with changed memory layout.

Settings are saved to EEPROM automatically on every `/MODULE/PARAM/VALUE` command — you don't need to save them manually.

11. Troubleshooting

Command is not accepted (no "OK" response)

1. **Check line ending:** the terminal must send CR. In the Arduino IDE serial monitor: select "Both NL & CR".
2. **Check baud rate:** 115200, otherwise only garbage arrives.
3. **Check command length:** maximum 32 characters. Long values (e.g. `/DP/VL/11.5/12.0/12.5/13.0`) push the limit — no extra spaces.
4. **Check the module token:** typo? Tokens are case-sensitive (uppercase always works).
5. **Module does not exist:** Check the list of valid module tokens in Section 6.

LEDs do not light up

- Door switches HIGH? → DPL/VU off (LeftDoor) or CBI/DCBI off (RightDoor)
- Brightness at 0? → set `/DP/BR/10`
- Animation at 0 (none)? → set `/DP/AN/1`
- LED strand power supply? Long WS2812B strings (LDPL = 43 LEDs) need up to 2.5 A at full brightness
- Data line broken? Check order: the first LED has to be on the pin (DCBI=D6, LDPL=D7, UAL=D8, VU=D3), then daisy-chain.

Wrong colors (e.g. red shows as green)

WS2812B strips come in several color orderings (RGB, GRB, BGR). The firmware is compiled for **GRB** (default for WS2812B). Strips with a different ordering show swapped colors — solution: a different strip type or firmware modification (no user setting).

VU bar does not respond to sound

The VU effect is **synthetic** (62 BPM sine wave), not audio-reactive. See Section 1.

Voltage display shows wrong value

1. Check voltage divider values (R1/R2) — default is 100k/10k SMD-populated. If you replaced with custom THT resistors, correct via `/DP/RR/<R1>/<R2>` to the actual values.
2. Check threshold values — `/DP/VL/RED/YELLOW/GREEN/CHARGE`. Order must be ascending (red < yellow < green < charge).
3. ANALOGPIN A0 must see at most 5 V — the voltage divider must scale the battery voltage below 5 V. With defaults: $12V \times 10/(100+10) = 1.09V$ ✓, $24V \times 10/(100+10) = 2.18V$ ✓.

EEPROM settings lost after firmware update

When a firmware update changes the internal memory layout, all user settings (brightness, colors, animations, voltage thresholds) are reset to defaults automatically. Note your settings before updating so you can re-enter them after flashing.

12. FAQ

What's on the DPL board itself, what is external?

On the DPL board (Classic + VU) are the LEDs for the data port display itself (TopBar, BlueLights, BarGraph, RedLights, WhiteLights). On the DPL-VU additionally the VU row. **Not** on the DPL: the charge-bay indicator display (D-CBI or CBI Plus), the LDPL belly-hatch strip, the UAL utility-arm strip — these are separate boards that connect to the DPL.

What happens if I don't connect a D-CBI / CBI Plus / LDPL?

Nothing bad — the firmware drives the pins anyway, the connected module (or none) decides what lights up. Commands like `/CB/AN/3` are still valid, you just don't see the animation.

D-CBI or CBI Plus — what's the difference?

D-CBI is the digital WS2812B variant with **full RGB color depth** — base color via `/DC/CO/...` is freely selectable, animations like rainbow, color-fade, heart-beat etc. use the full color palette. **CBI Plus** is the analog variant with classic 3 mm LEDs in **fixed individual colors** (red/yellow/green/blue, hard-soldered at the factory) — looks like the classic CBI in the original R2. Which one you choose is taste — many builders also combine both.

Can I connect D-CBI AND CBI Plus at the same time?

Yes. They sit on different connectors (CBI Plus on the 5-pin header, D-CBI on the 3-pin header at D6) and are configured via separate tokens (`/CB/...` and `/DC/...`). Both animate in parallel with their own settings.

Can I run both DPL boards (Classic + VU) in parallel, one per dataport door?

Yes, that's the typical use case for many R2 builds. Each board has its own Arduino — they can be configured independently.

Can the DPL Classic be upgraded to VU later?

Yes — an external VU Extension is available separately from Printed-Droid. It connects to the 3-pin header `D3 VU` on the DPL Classic. The firmware does not need to be adjusted.

Does the firmware work with any Arduino Nano?

With ATmega328-based Nano yes (original Nano or compatible clones). NOT with Nano Every (ATmega4809) or Nano ESP32 — those are different microcontrollers with different firmware requirements.

How many LEDs can I connect to LDPL / D-CBI / UAL?

The firmware expects the standard counts (LDPL = 43 LEDs, D-CBI = 23 LEDs, UAL = 19 LEDs, VU = 28 LEDs). More LEDs are not driven (extras stay dark), fewer LEDs are no problem (excess indices are written but don't physically exist).

Is there a Web UI?

No — the DPL Classic / DPL-VU configure exclusively via the serial interface.

Where do I find current firmware HEX files?

<https://www.printed-droid.com/kb/dpl-classic-data-port-logics/> → Downloads. New versions are provided by Printed-Droid.

Can I use the I2C header on the DPL, e.g. to connect a display?

Not with the DPL firmware — it doesn't use I2C. For your own extensions the header (A4=SDA, A5=SCL, 5V, GND) is freely available, but you would have to write your own firmware for that.

My D-CBI / CBI Plus doesn't fit into my old R2 body — can the board be split?

Yes. Both boards have an area with a perforated line ("break line") where the PCB can be split. The two halves can then be reconnected via the two adjacent 3-pin headers (+ S – each) on the back with short pin jumpers or wires. If you keep the standard form factor, leave that area unchanged.

Where do I get help if something doesn't work?

Three places:

- **Product pages** with FAQ updates: <https://www.printed-droid.com>
- **Facebook group** (community + direct contact): <https://www.facebook.com/groups/printeddroid/>
- **GitHub repo**: https://github.com/PrintedDroid/DPL-VU_and_DPL-Classic

13. Writing your own firmware

The DPL firmware uses the Arduino Nano only partially. If you want to write your own code or modify the firmware, the necessary information is in this manual:

- **Pin assignment** of the board — see *Section 3.4*
- **Which pins are free** — D9, D13, A1-A7 as well as the I2C bus (A4/A5) and the UART pins (D0/D1) are completely unconnected
- **Which pins are hard-wired** — MAX7219 (D10/11/12), WS2812B strips (D3/D6/D7/D8) and voltage divider (A0). These components are permanently connected — your code should either deliberately drive them or leave the pins as inputs without pull-up
- **Conflicts** — see *Section 3.5* (hardware SPI vs. MAX7219 bit-bang, A6/A7 ADC-only, D13 with on-board LED, D0/D1 USB conflict during upload)

Uploading your own firmware: same procedure as the firmware update in *Section 4* — XLoader for `.hex` files, or directly from the Arduino IDE via USB serial bootloader (no ICSP programmer required, the on-board Nano has a bootloader pre-installed).

14. Cheat sheet (quick reference)

A one-page summary to print and keep next to your soldering bench / PC.

Module tokens

Token	Module
DP	Data Port Lights (DPL on-board)
CB	CBI Plus (analog Charge Bay, external module)
DC	D-CBI (digital Charge Bay, external module)
LD	LDPL (Large DataPanel Logics, external module)
UA	UAL (Utility Arm Lights)
VU	VU bar
TB BL BG RL WL	DPL sub-areas

Command format

```

/<MODULE>/<PARAM>/<VALUE>[ /<VALUE2>/<VALUE3>... ]
Line ending: CR @ 115200 baud Max. 32 characters

```

Parameters

Token	Meaning	Range
BR	Brightness	0-15 (DP/CB) or 0-255 (DC/LD/UA/VU)
AN	Animation number	0-N
AS	Animation speed	ms
CO	Base color	R/G/B 0-255 each (WS2812B modules only)
VM	Voltage monitor	0/1
VL	Voltage thresholds	4 floats: red/yellow/green/charge
RR	Voltage divider	R1/R2 in ohms (default 100000/10000)

System commands

Command	Effect
/RESTART	Soft reboot
/FACTORY	Reset EEPROM + reboot

Common animations (AN number)

Module	0	1	2	3	4	5	6	7	8
DP	none	default	randomV U	startup					

Module	0	1	2	3	4	5	6	7	8
CB	none	default	random	heart	charging				
DC	none	default	random	heart	randFade	colorFade	rainbow	heartBeat	charging
LD	none	default	fadeOut	rainbow					
UA	none	default	movie	abort					
VU	none	cycle	randomBar						

Command examples (copy-paste)

```

/DP/BR/15           DPL bright
/LD/CO/0/0/255     LDPL blue
/LD/AN/2           LDPL Knight-Rider with fade
/DC/AN/6           D-CBI rainbow
/CB/AN/3           CBI heart
/UA/AN/2           UAL movie sequence
/VU/AN/1           VU cycle through effects
/DP/VM/1           Voltage monitor on
/DP/VL/11.5/12.0/12.5/13.0  Thresholds 12V lead-acid (red/yellow/green/charge)
/DP/VL/10.8/11.4/12.0/12.6  Thresholds 3S LiPo

```

Pin map (on-board Arduino Nano)

Pin	Function	Pin	Function
D2	Door switch left	D8	UAL Out
D3	VU Out	D9	free
D4	Door switch right	D10	MAX7219 Data (header D)
D5	reserved (3-pin)	D11	MAX7219 Clock (header C)
D6	D-CBI Out	D12	MAX7219 Load (header L)
D7	LDPL Out	D13	free (on-board LED)
A0	Voltage IN	A1-A3	free (ADC)
A4	I2C SDA (free)	A5	I2C SCL (free)
A6/A7	free (ADC only)	D0/D1	UART (USB serial)