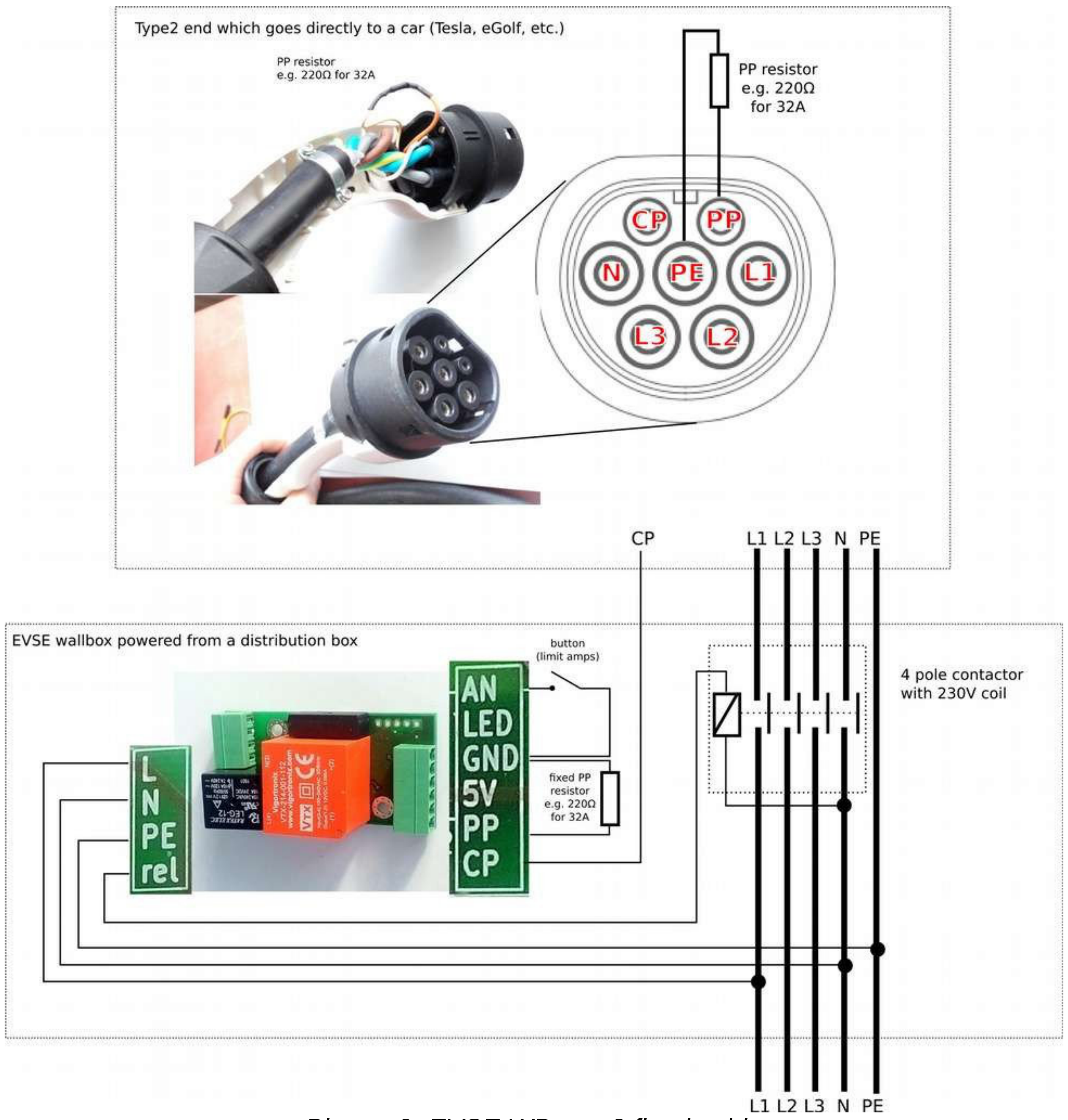
3phase wallbox including a cable and plug

In this example we make 3 phase Wallbox using DSIEC-2E cable. PWM duty will be limited by the size of R_{PP} (refer to the *Theory of operation* chapter). If you do not connect any R_{PP} current will be limited to only 6A. If your EVSE includes a cable which is fixed then you can hard-wire R_{PP} resistor for cable's nominal value.

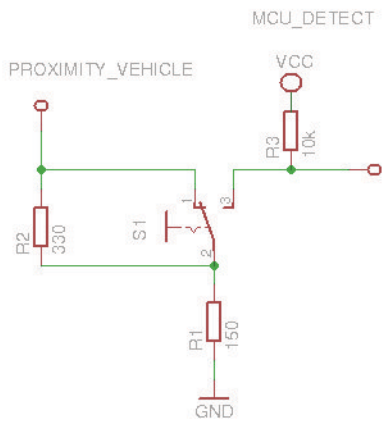

Type2 end which goes directly to a car (Tesla, eGolf, etc.)
PP resistor e.g. 220Ω for 32A
PP resistor e.g. 220Ω for 32A
CP
PP
N
PE
L1
L3
L2
CP
L1 L2 L3 N PE
EVSE wallbox powered from a distribution box
button (limit amps)
AN
LED
GND
5V
PP
CP
L
N
PE
rel
fixed PP resistor e.g. 220Ω for 32A
4 pole contactor with 230V coil
L1 L2 L3 N PE

Picture 9: EVSE WB type2 fixed cable

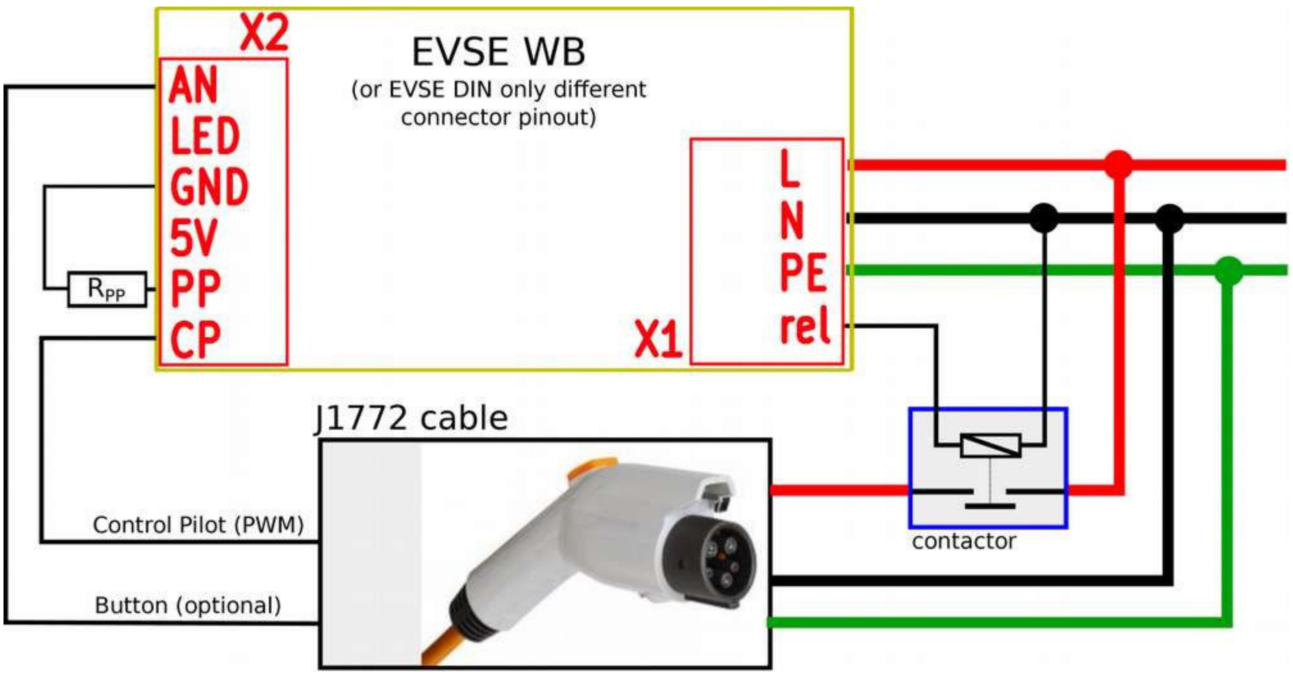
32A EVSE with connector J1772

With EVSE Wallbox board you can quickly build a charging station for your Nissan Leaf 6.6kW or any other vehicle equipped with J1772 plug.

Optional Analog Input connection:
The internal J1772 connection allows to use S1 proximity button as an auxiliary button for EVSE. With the help of this button you can easily change charging current with the smallest step of 1A (see Features - Precise current setting).



Picture 10: J1772 proximity button connection detail



Picture 11: J1772 connector - signal and power wires

Recommended contactors

You should use relays / contactors equipped with 230V coil and connect them directly to the board connector X1. Here are some examples which relays can be used. The most common one (4pole contactor 25A) can be easily obtained in local electrical accessories shop (Conrad, K&V Elektro...)

- 3-phase, maximum current 20A (e.g. Tesla 11 kW):
  Noark Ex9CH25 40, Elko VS420
- 3-phase, maximum current 40A (e.g. Tesla 16.5 / 22 kW, ZOE 22 kW):
  Elko VS440, Eaton Z-SCH 230V/25-40
- 1-phase, maximum current 16-20A (e.g. Peugeot iOn 3 kW, Tesla 3-4kW):
  Eaton Z-R230/S
- 1-phase, maximum current 40A (e.g. Nissan Leaf 6.6 kW):
  RELPOL R40N
  or use normal 4-pole contactor

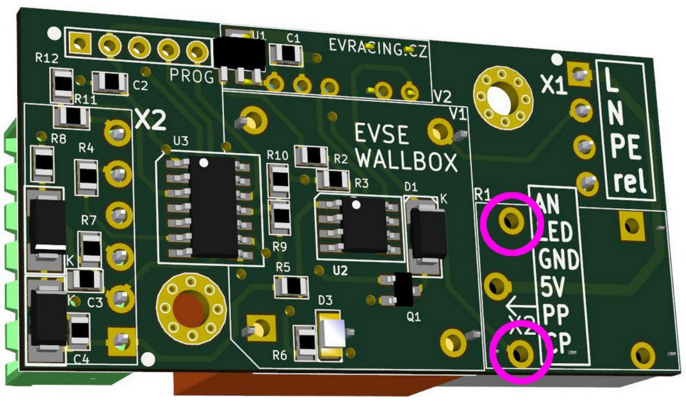
Picture 13: 40A 1phase relay

Picture 12: 20A 3phase relay

Using external 12V relay

In some cases it would be possible to use another 12V coil relay (maximum 0.8W coil consumption), however this approach requires desoldering of the original relay and is recommended only to advanced users. Automotive relays can handle sufficient currents however their voltage rating is not high enough.

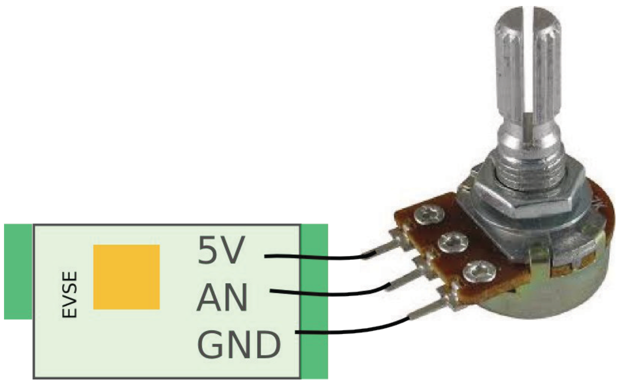
Pins marked by pink are connected to a 12V relay coil:



Picture 14: EVSE WB - connecting 12V relay directly

Continuous EVSE current regulation using a pot (e.g. 6-32A)

Use AN as EVSE voltage input and connect a potentiometer wiper. The other two terminals are to be connected to GND and 5V. Minimum 6A and maximum 32A can be adjusted in registers as needed.



- set 2003 = 0 analog input current regulation
- set 2000 = 32A maximum current (AN = 5V)
- set 2002 = 6A minimum current (AN = 0V)

See the register table for more details. If you do not know how to change register values, please check “Advanced configuration” chapter.

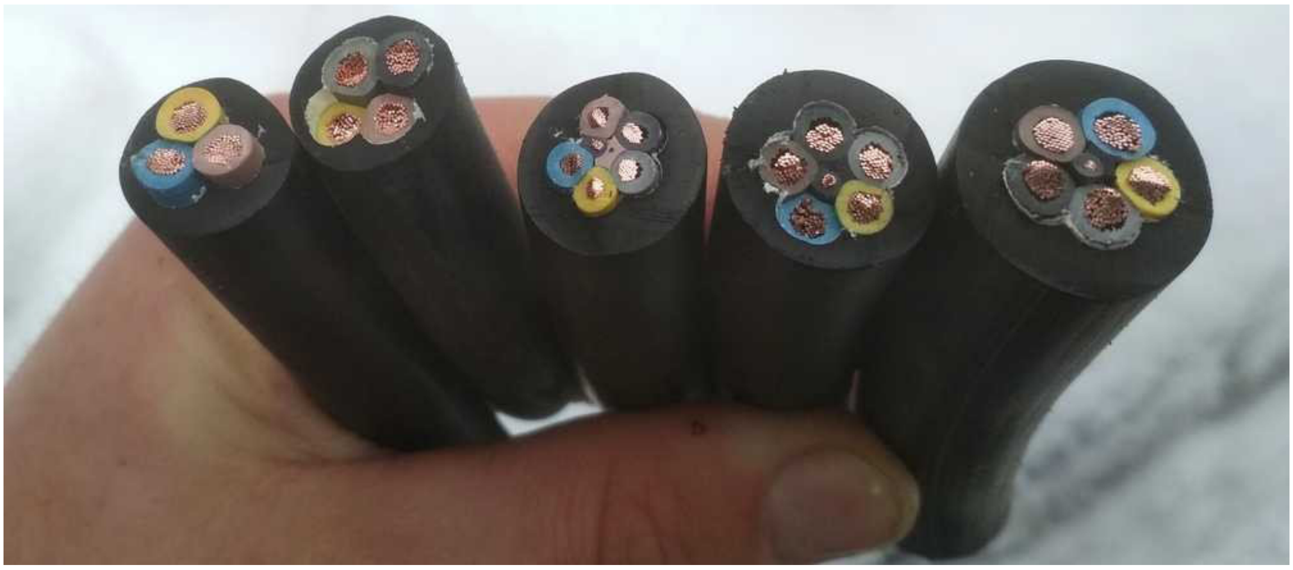
0 - 10V configuration

Add a 1k resistor R12. R11 is also 1k and it will create a divider 1:1. Please note that this setup is recommended to use only with I/O modules having **0-10V analog output galvanically isolated.**

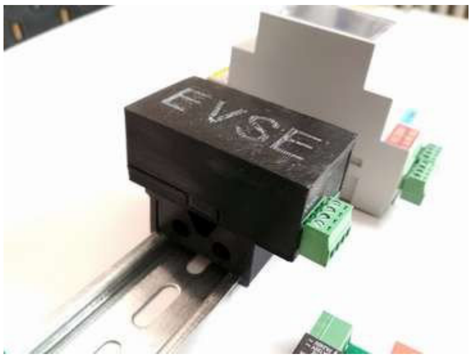
Cable selection guide

rubber cables suitable for electric vehicles (type H07RN-F 450/750V)				
type	diameter [mm]	current [A]	weight [kg/m]	application
5G6 + 0,75	18-22	38	0.7	Type2 female 3phase charger, Type2 male to Type2 female extension cable
5G4 + 0.75	16-20	30	0.5	
5G2.5 + 0.75	13-17	20	0.4	
4G6	16-20	37	0.6	Type1 1phase charger, Type2 male to Type1 extension cable
3G6	14-18	40	0.4	Type1 or Type2 1phase charger with EVSE built into the connector *)

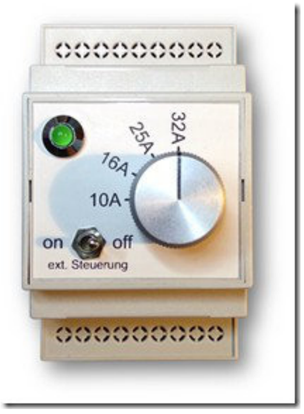
*) other variants: 3x2.5mm 1phase, 5x2.5mm 3phase


Picture 15: 3G6, 4G6, 5G2.5+0.75, 5G4+0.75, 5G6+0.75

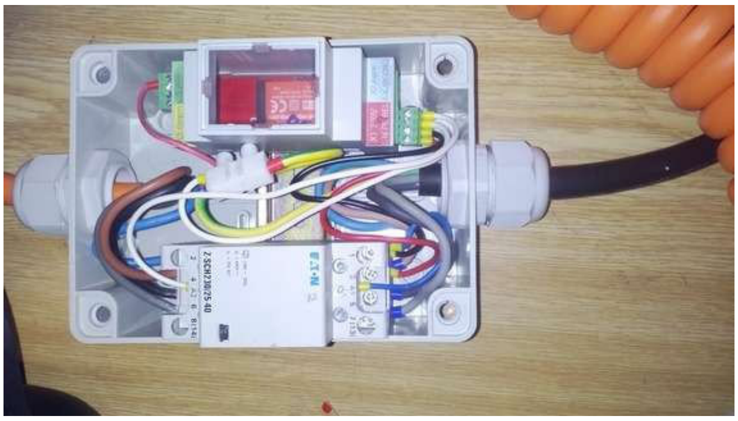
Customer solutions and projects

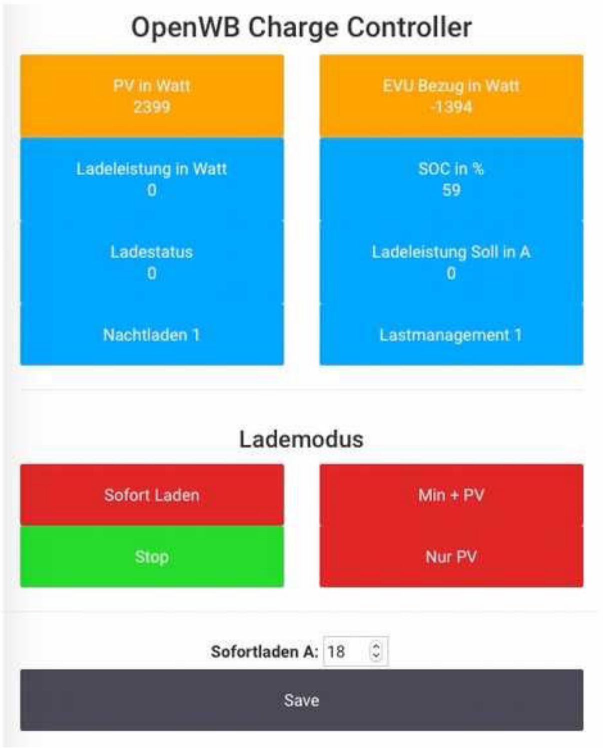


Picture 16: 3D printer box for EVSE WB in comparison to EVSE DIN



Picture 17: EVSE WB in custom box with 4position current switch


Picture 18: Charging cable made with EVSE DIN together with 4pole contactor in a box


OpenWB Charge Controller

PV in Watt
2399

EVU Bezug in Watt
-1394

Ladeleistung in Watt
0

SOC in %
59

Ladestatus
0

Ladeleistung Soll in A
0

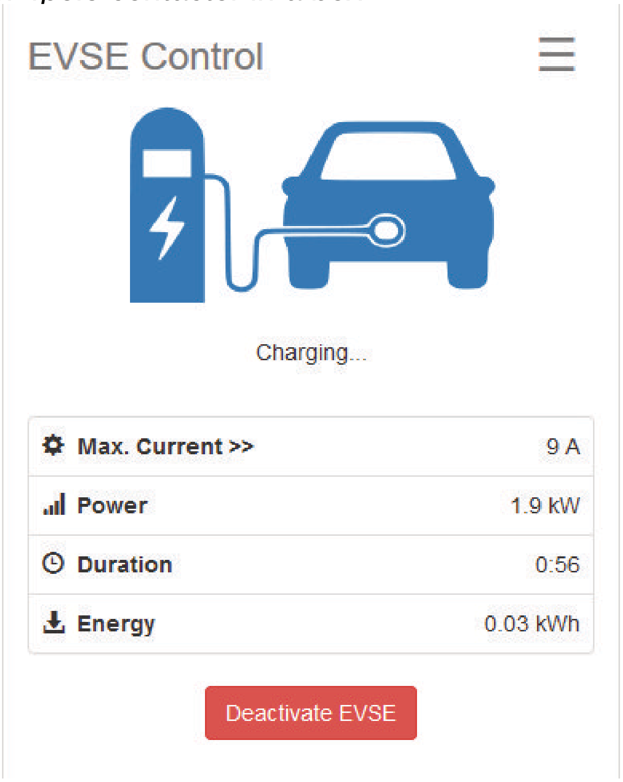
Nachtladen 1

Lastmanagement 1

Lademodus

Sofortladen A: 18

Picture 19: SDM630 electric meter + EVSE DIN RS485, web interface (http://openwb.de)


EVSE Control

Charging...

Max. Current >>	9 A
Power	1.9 kW
Duration	0:56
Energy	0.03 kWh

Picture 20: Wifi interface for EVSE WB (https://github.com/CurtRod/SimpleEVSE-WiFi)

FAQ

1) How many PP resistors do I need?

As described in more detail above in the schematic (page 17), there must be one resistor in Type2 female (car side) and other resistor connected to an EVSE. Both are usually part of the extension cable Typ2 female (car side) to Type2 male (EVSE side). When using a fixed cable (not possible to disconnect from the EVSE) then the size of EVSE PP resistor can be “hardcoded” (modbus versions only, see register 2007 below) or a resistor of a fixed size have to be connected (usually 220 ohm for 32A cable).

2) EVSE does not work (config problem)

Make sure that the EVSE is not turned off (2005 configuration register). If the board does not respond at all, make sure the the UART parameters are correct (9600 baudrate, RX/TX+GND wired correctly). Did not the Modbus address change by accident (reg. 2001)?

Any kind of software / firmware / settings problem can be solved using a PicKIT and reprogramming the board from scratch using MPLAB IPE (do not forget to erase the chip too!). PicKIT 4 can be bought on Farnell (https://uk.farnell.com/search?st=pickit%204). EVSE firmware is ready for download from this link: http://evracing.cz/evse/evse-wallbox/fw_pickit/ .

Parameters and options:
- chip name is PIC16F1825
- “power target circuit from tool” (use 2-2.5V)
- “use high voltage program mode entry”
- pin1 (Vpp) is always marked (both on Pickit and the board too)
- if “power target circuit from tool” does not work, power the board externally
- when everything is ready, select HEX file “Browse” and click “Program”
- after successful programming you will see “Programming complete” in log window

3) LED indication explained

Using the pin LED you can directly connect LED to indicate EVSE status. The output includes 1k resistor. External LED has the same indication function as on-board LED.

LED	**State description**
	Normal operation:
1x fast + pause	pilot signal is steady +12V, no vehicle connected (can be customized – always ON)
2x fast + pause	PWM signal is generated, vehicle is present
1x long + pause	vehicle requested power, contactor is ON
20x fast in 0.5s	enter or leave current setting mode
1x each 0.3s	current setting mode – one more amper set
	Error states (no charging):
3x fast + pause	EVSE is disabled in software (FW >= 6)
4x fast + pause	Charging with ventilation is disabled (FW >= 13)
5x fast + pause	Pilot signal check failed (FW >= 11)
6x fast + pause	Residual current check failure (FW >= 16)

4) How to keep current settings after grid failure?

- see “advanced configuration” and enable communication (if necessary), see NOTE#1
- see HC06 bluetooth module + Android app
- set 2004 = 1 to save current set by button OR
- set 2000 = “desired amp value” to keep this maximum current every time

Advanced configuration

Beginning January 2018 all boards can be configured by MODBUS. By default MODBUS interface is disabled and old functions of PROG pins are kept. However you can enable it ~~by grounding PIN4 and PIN5 of PROG connector while powering the board up~~ please see NOTE#1 instead.

Physical layer is not galvanically isolated UART (0-5V) or optionable isolated RS485 EVSE DIN only.

The default device ID is 1 (can be changed in register 2001).

Some values can be then read or written over MODBUS protocol. PROG pin header is used for this purpose with following pinout:

PROG connector	**connection**
pin3	GND
pin4	TX
pin5	RX

This feature is useful for further development and testing with EVSE Wallbox board and can be also a great way to interface other devices such as Raspberry PI, Ethernet UART bridges (e.g. WIZnet or USR serial-to-ethernet boards), various WiFi modules etc.

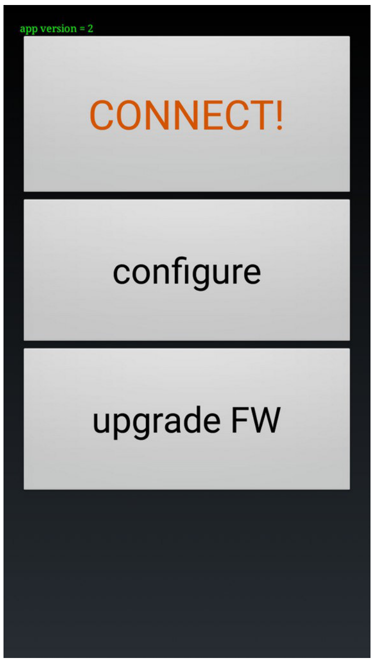
HC06 bluetooth module + Android app

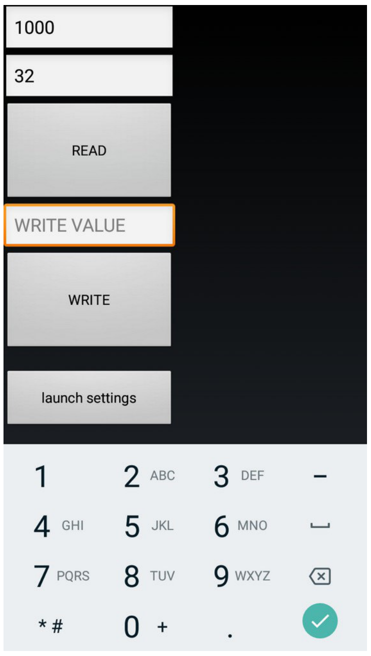
Preferred way is to use HC06 Bluetooth converter and Android application developed for this purpose. By default the communication is disabled (see **NOTE#1** below register table), because the original analog functions of RX,TX pins are kept (for changing current by resistors).

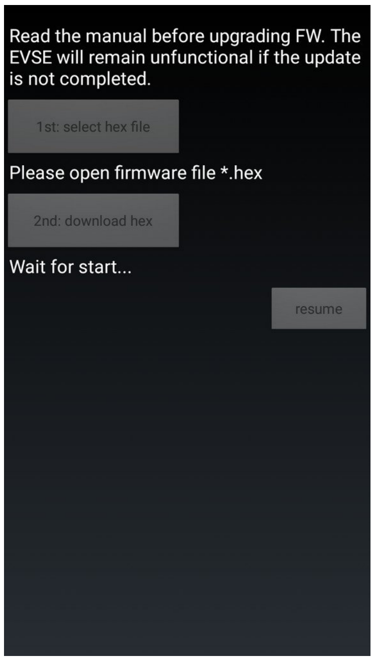
You can download and install the app from here (*.apk extension): http://evracing.cz/evse/evse-wallbox/

HC06 is connected to PROG connector.

1. power up the board, HC06 should start blinking
2. pair your Android device with the adapter (default pin 1234)
3. open the EVSE app and choose the paired bluetooth adapter in its settings
4. hit connect button and wait for connection
5. read & write registers you need






Read the manual before upgrading FW. The EVSE will remain unfunctional if the update is not completed.

Please open firmware file *.hex

Wait for start...

Flashing new firmware

Flashing new firmware using Android phone is supported (with bluetooth adapter) on all EVSE DIN devices and on EVSE Wallbox devices only with “modbus firmware”. In order to flash new version of firmware bootloader mode must be enabled (last bit of the register 2005 - see the below).

Bootloader mode

If bootloader mode is activated the LED is solid on or off and it toggles when there is active communication. Bootloader mode can be turned off by changing the configuration register or it happens automatically after successful flashing of new firmware or there is a timeout of approximately 4 minutes.

When bootloader mode is activated, modbus slave address is set to 1.

~~The other way to enter bootloader mode is to keep analog input down when the device is booting (before powering on) for about 12 seconds~~ (CHECK: or some more complicated way to prevent accidental bootloader turn on? CONCLUSION: if divider 1:1 used for AN => colision, result: this option disabled)

Bootloader change with board having initial revision 10: bootloader can be enabled by pulling MCLR (pin 1 PROG header) down for at least 12 seconds. This is needed for the case that software upgrade fails and there is no possibility to enable bootloader by changing the register.

Troubleshooting

If the download of a new firmware was not completed then you need to try it again, otherwise the board will not function properly. Last chance to fix badly programmed board is using PICKIT programmer (you can order PicKIT 4 on Farnell to get it fast).

How to determine firmware version?

If you power up the EVSE board and the LED is on for about 2 - 3 seconds you have Modbus firmware onboard and it is disabled. You can enable it this way NOTE#1. Firmware revision can be then read from register 1005.

Analog input control

Charging current can be changed based on analog input AN voltage (when register 2003 = 0). Minimum value is set in register 2002 and maximum in register 2000. Default supported range is 0 - 5V. Analog input is referenced to GND and is not isolated

Charging current ($\mathbf{I_{PWM}}$) for the vehicle is calculated:

$$I_{PWM} = U_{AN} / 5 * I_{MAX}$$

Where $\mathbf{U_{AN}}$ is the voltage measured at the input AN and $\mathbf{I_{MAX}}$ is actual maximum current. Default is 32A, however it can be adjusted by settings or “current boost” / ”current limit” feature or change in configuration register.

If the resulting current is smaller than 6A then the EVSE state will change to “pilot steady 12V” and vehicle stops charing. Minimum current can be also adjusted in the register.

Register table

Register address	**R/W**	**def ault**	**Description**
1000	R/W		Actual configured amps value (from reg 2002 to 80A)
1001	R		Actual amps value output
1002	R		Vehicle state: 1: ready 2: EV is present 3: charging 4: charging with ventilation 5: failure (e.g. diode check, RCD failure)
1003	R		6 / 13 / 20 / 32 / 63 / 80 A Maximum current limitation according to a cable based on PP resistor detection
1004	R/W		**bit0:** turn off charging now **bit1**: run selftest and RCD test procedure (approx 30s) **bit2:** clear RCD error **bit3 - bit15:** not used
1005	R		Firmware revision
1006	R		EVSE state 1: steady 12V 2: PWM is being generated (only if 1000 >= 6) 3: OFF, steady 12V
1007	R		EVSE status and fails **bit0**: relay on/off **bit1**: diode check fail **bit2**: vent required fail **bit3**: waiting for pilot release (error recovery delay) **bit4**: RCD check error **bit5**: **bit6-bit15**: reserved
1008	R		Error timeout if > 0 then waiting for automatic clear
1009	R		Selftest timeout if > 0 then waiting for a selftest to complete
1010	R		reserved
...			...
2000	R/W	32	**Default amps value** after boot (max 80A, min 6A)

Register address	**R/W**	**def ault**	**Description**
2001	R/W	0	**Function of PROG PIN 4 + 5, slave address** (default 0: current limit or boost functions available) 0: analog inputs enabled > 0: MODBUS communication enabled This value also means the slave address.
2002	R/W	5	Minimum amps value, allowed 0 - 13 if set to 0 the EVSE will completely stop charging during analog input mode (2003 = 0) and AN = 0V
2003	R/W	1	**Analog input config:** 0: analog input current regulation, input 0 - 5V corresponds to the range: minimum amps --> default amps note that there is a weak pullup resistor enabled for this input (no input connected --> 5V) 1: each blink 1 amp step (default), starts from 0 2: each blink 2 amps step, starts from 0 3: each blink 3 amps step, starts from 0 .... 10: each blink 10 amps step, starts from 0 11: mapping table: registers 2010 - 2017 1x blink = value from 2010 2x blink = value from 2011 ...
2004	R/W	0	Amps settings after power on (applies only to a changes made by the button), whether to save amps settings to eeprom each time it changes or not 0 - do not save amps value (default) 1 - save amps value to register 2000 every time it changes

Register address	**R/W**	**def ault**	**Description**
2005	R/W	1	**bit0**: Enable button for current change (no sense when 2003 = 0) 0: disabled 1: enabled (default) **bit1**: Stop charging when button pressed 0: disabled (default) 1: enabled charging will automatically start after you manually unplug and plug the cable to the vehicle **bit2:** Pilot ready state LED 0: blinks once (default) 1: is always ON **bit3:** enable charging on vehicle status D (ventilation required) 0: vehicle status D charging is disabled 1: vehicle status D charging is enabled (default) **bit4:** enable RCD feedback on MCLR pin (pin 4) 0: disabled, no RCD connected (default) 1: enabled **bit5**: auto clear RCD error 0: disabled (default, power cycle needed) 1: enabled (clear RCD error after <30s min timeout) **bit6: (rev16 and later)** 0: AN internal pull-up enabled (default) 1: AN internal pull-up disabled **bit7-12**: reserved **bit13**: disable EVSE after charge (write 8192) **bit14**: disable EVSE (write 16384) **bit15:** enable bootloader mode (write 32768) **NOTE:** if both bit0 and bit1 are enabled then “interrupt charging” will have higher priority, when charging
2006	R/W	0	RFU: Current sharing mode is active (two or more EVSEs connected to a single breaker) 0 - charging current is half of the default current amps > 0: this amps value will be used
2007	R/W	0	PP detection 0: PP detection enabled (default) value > 0: detection disabled, fixed PP limit entered [A]
2008	R/W		reserved
2009	R		Bootloader firmware revision
2010	R/W	6	Amps value 1
2011	R/W	10	Amps value 2
2012	R/W	16	Amps value 3
2013	R/W	25	Amps value 4

Register address	**R/W**	**def ault**	**Description**
2014	R/W	32	Amps value 5
2015	R/W	48	Amps value 6
2016	R/W	63	Amps value 7
2017	R/W	80	Amps value 8

Register addresses are in decimal format!

NOTE#1: By default MODBUS is NOT enabled, so the original analog input switches can be used (current limit and current boost functions). MODBUS can be enabled by pulling AN input down to GND while booting (= when you power the device up) at least 5 times within 3 seconds seconds (button activated). This will change register 2001 to the value 1, but it will not save the value permanently. Value will be saved after first successful R/W operation over MODBUS (register number >=2000).

NOTE#2: You can restore default settings of all registers by saving a value 65535 to the register 2010

NOTE#3: Only functions 03 (Read Holding Registers) and 16 (Preset Multiple Registers) are implemented. For more details please check: http://www.simplymodbus.ca/FAQ.htm

RFU: reserved for future use

EVSE selftest:

1) turn on +12V (0 seconds)
2) turn on PWM generation, 5th second
3) enable 220R pp resistor (max 32A, 50% duty cycle), 10th second
4) turn on contactor (15th second)
5) RCD test procedure (if configured, 20th second, performs 500ms pulse on TEST-IN)
6) selftest finished after 30 seconds

Selftest counter (reg 1009)	**Relay out**	**RCD test out**	**PP 220ohm**	**Action**
300	off	off	off	Steady +12V CP signal
250	off	off	off	PWM 1kHz start, default duty cycle
200	off	off	**ON**	PWM duty cycle change to 50%
150	**ON**	off	ON	Relay activation
100-95	ON	**ON**	ON	Test output RCD enabled ~500ms
0	off	off	off	Steady +12V CP signal

NOTE: if equipped with RCD sensor, the test must finish with RCD error (LED blinks six times). RCD error must be cleared manually or by timeout (depending on config)

Firmware updates

2020-06-25 revision 17:
- stricter CP signal check (pilot low decision level, shorted diode)

2020-04-27 revision 16:
- added RCD feedback support for EVSE-WBDIN (e.g. Bender RCMB121 and others)
- added configuration bit for AN pullup (can be disabled)

2020-02-15 revision 15:
- added minimum pilot_high condition (fix for Metrel A 1532 – status E/F test)

2019-09-01 revision 13:
- possible to change LED behavior in pilot steady state
- diode check fail increased error recovery delay
- EVSE status vent / without vent

2019-04-26 revision 12:
- RX buffer limit fix (mainly for RS485 systems)

2018-12-10 revision 11:
- improved CP signal check (implements status E), requires feedback resistor 100k (rev3 board R20, rev2 board R13, DIN version R21)
- if you do not upgrade feedback resistor, do not use firmware 11+

2018-10-30 revision 10:
- fix evse status register 1002 initial value (probably caused problems in OpenWB)
- modbus timeout set to 5 steps

2018-08-28 revision 9:
- combined code for EVSE DIN 485 and EVSE WB (but 2 different hex files)
- uart communication fix (RX overflow error check)
- weak pullup disabled for PP detection pin
- modbus timeout set to 3 steps

2018-02-06 bootloader v3:
- fix: removed reconfiguration from bootloader code (causing rewriting reg 2002)

2017-11-18 bootloader v2:
- added possibility to turn bootloader on by hardware pin 1 (Vpp/MCLR)

2017-11-18 Androi app v2:
- fixed .hex file parser (firmware upgrade), always use latest app version for firmware flashing!

2017-10-31 revision 8:
- added “disable EVSE after charge” bit (register 2005)
- fixed issue with voltage AN input - now follows minimum amps and does not turn off

2017-06-23 revision 7:
- interrupt routine rewritten (modbus communication)
- correct vehicle status update, if pwm is not being generated
- modbus timeout set to 20 steps

2017-06-07 revision 6:
- new configuration bits, register 2005 (disable EVSE, interrupt charging, change current)
- error code signalization

2017-05-03 revision 5:
- fixed current limit based on cable PP resistor (and analog pin input)

2017-04-26 revision 4:
- fixed amps value from AI (e.g. for max 32A reached only 31A and 5V input)
- fixed minimum amps value (was not taken into account)

2017-04-21 revision 3:
- added bootloader (Android application)
- fixed bug with analog input not working while modbus enabled
- added control register 1004

2017-04-13 revision 2:
- changeable slave address in reg 2001
- stop charging button function reg 2005 (no timeout, but wait for unplug & plug), bitfield introduced for more possible options
- added register 1005 (firmware revision number)

2017-02-17 revision 1:
- initial version

Communication examples

Connection parameters: speed 9600, bytesize 8, stop bit 1, no parity, byte order Motorola.

Read holding registers example

01 03 03 E8 00 02 44 7B (request)

bytes	description
01	The slave address (always 01 unless changed by customer)
03	Function code (03 is for read holding registers function)
03 E8	The data address of the first register to read
00 02	The total number of registers requested.
44 7B	CRC-16 Modbus (cyclic redundancy check)

01 03 04 00 20 00 20 FA 21 (answer)

bytes	description
01	The slave address (always 01)
03	Function code (03 is for read holding registers function)
04	The number of data bytes to follow (2 registers x 2 bytes each = 4)
00 20	The contents of register 1000 (32A)
00 20	The contents of register 1001 (32A)
FA 21	CRC-16 Modbus (cyclic redundancy check)

Reading the data out with Python (USB – serial adapter)

You can use almost any MODBUS master utility for Windows / Linux / Mac. For common users we recommend using **QModBus** (http://qmodbus.sourceforge.net/). There is also QModMaster (http://sourceforge.net/projects/qmodmaster/) utility, but there may be a problem with write multiple registers function (there is bug in the software which causes register shift) - so use it carefully.

Advanced users can use Python / pymodbus (https://github.com/bashwork/pymodbus) for further development or similar libraries.

In following example we read out first 7 registers

```
#!/usr/bin/python3
from pymodbus.client.sync import ModbusSerialClient
client = ModbusSerialClient(method = "rtu", port=”/dev/ttyUSB0, baudrate=9600,
stopbits=1, bytesize=8, timeout=1)
rq = client.read_holding_registers(1000,7,unit=1)
print(rq.registers)

#Reading the data out with Python (USB – serial adapter)
```