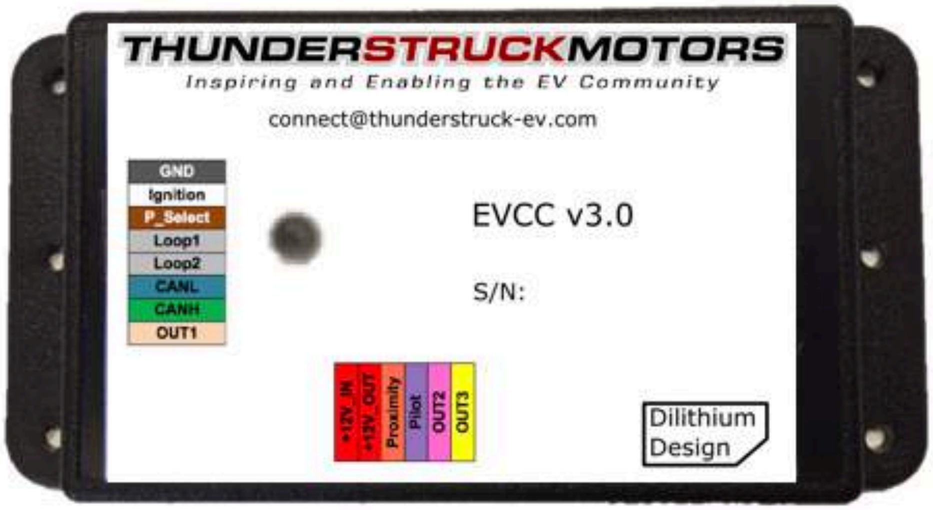
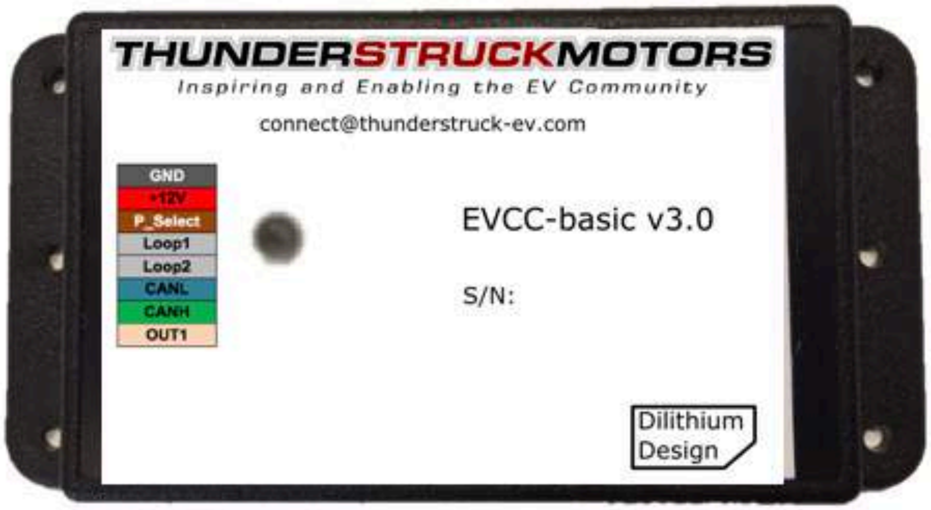
Thunderstruck Motors
EV Charge Controller v3.0

EVCC and EVCC-basic





Contents

Figures

Overview

The Electric Vehicle Charge Controller (EVCC) controls the operation of a CAN enabled charger. Parameters including maximum voltage, maximum current, and total charge time are configured, saved in nonvolatile memory, and used during charging.

There are two models of the EVCC 3.0. This document describes both the standard EVCC 3.0 and the EVCC-basic 3.0. The standard EVCC has been optimized for use in Electric Vehicles and has support for the commonly used J1772 Level 2 charge protocol. The EVCC-basic is optimized for non-J1772 applications. There is a large overlap in capability and in this document, the term "EVCC" used without qualification applies to both models.

The EVCC has support for several CAN enabled chargers, including the Thunderstruck TSM2500 charger. Charge voltage, charge current and overall charge time are controlled by the EVCC using the CAN interface to the charger. A constant current/constant voltage charge curve is supported for Lithium Batteries, and a three-phase charge cycle is supported for Lead Acid Batteries.

The EVCC supports up to four parallel chargers for faster charging. When multiple chargers are configured, each one is individually CAN addressed. Work is divided evenly between the chargers and statistics are gathered and recorded on each charger separately.

Determining cell overvoltage errors and cell undervoltage errors within the pack are functions of a Battery Management System (BMS). The EVCC can be configured to interface with a BMS either by a cell loop or by CAN messages. When a CAN BMS is used, the EVCC can also be configured to lower the charging current when a cell exceeds a configurable threshold.

Charging normally completes at the end of a charge cycle, which, for Lithium batteries, occurs when the pack voltage approaches the target maximum voltage and the charging current drops below a minimum configured charge current. The EVCC will also stop charging if a cell overvoltage error occurs, there is loss of communication between the EVCC and Charger, or the maximum configured charge time is reached. The standard EVCC will stop charging if the J1772 plug becomes unplugged.

Charging history is provided which records the reason that charging stopped, total charge time, maximum voltage, maximum current, final current, and watt hours delivered.

The standard EVCC and the EVCC-basic are both housed in the same enclosure wth a serial port jack, an LED and an 8-pin connector. The standard EVCC has an additional 6-pin connector, which provides the connections necessary for J1772.

EVCC-basic

The figures below show the system diagram and connections for the EVCC-basic:

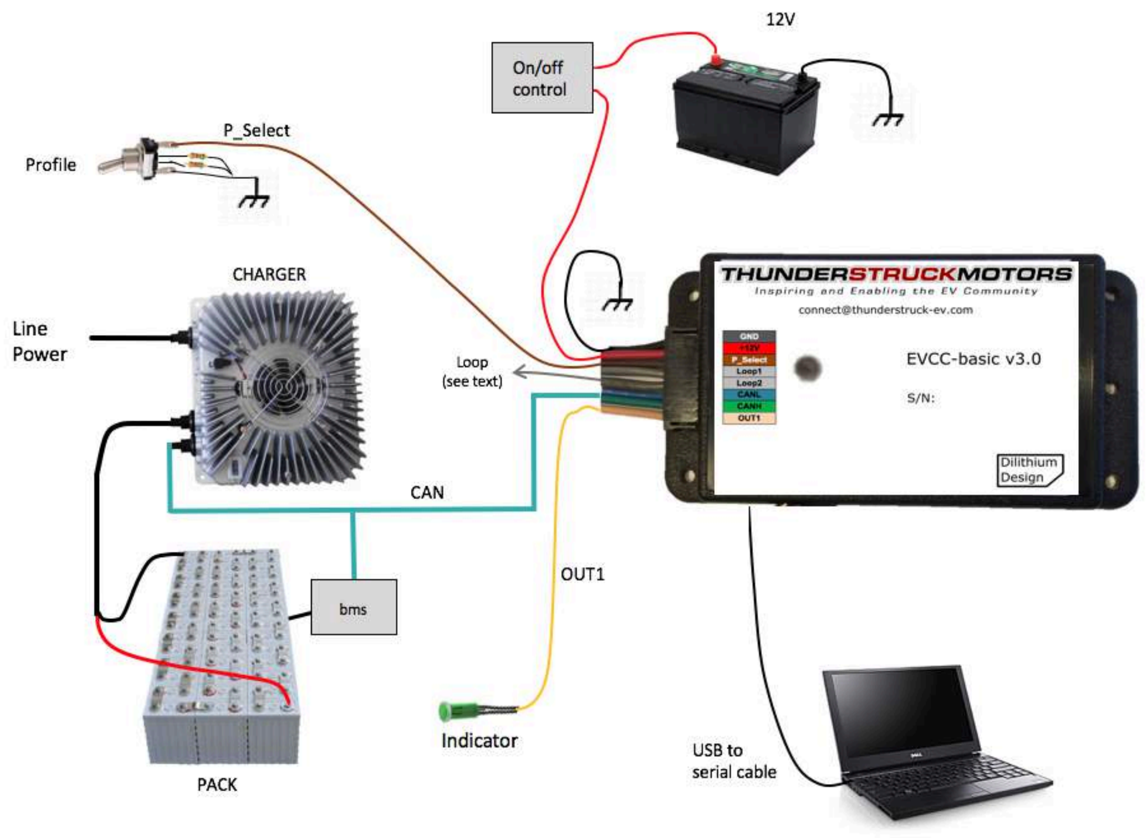
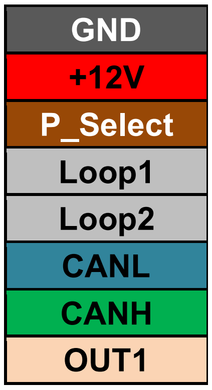


Figure 1 – EVCC-basic System Diagram



GND
+12V
P_Select
Loop1
Loop2
CANL
CANH
OUT1

Figure 2 – EVCC-basic Connections

- **GND** connects to Ground.
- **+12V** connects to switched 12V power.
- **P_Select** is a "charge profile" selection input which can be used to select charging parameters.
- **Loop1** and **Loop2** are used for the cell loop supervision circuit
- **CANH, CANL** connects to the CAN network
- **OUT1** is a mappable output.

Standard EVCC

The figures below show the system diagram and connections for the standard EVCC:

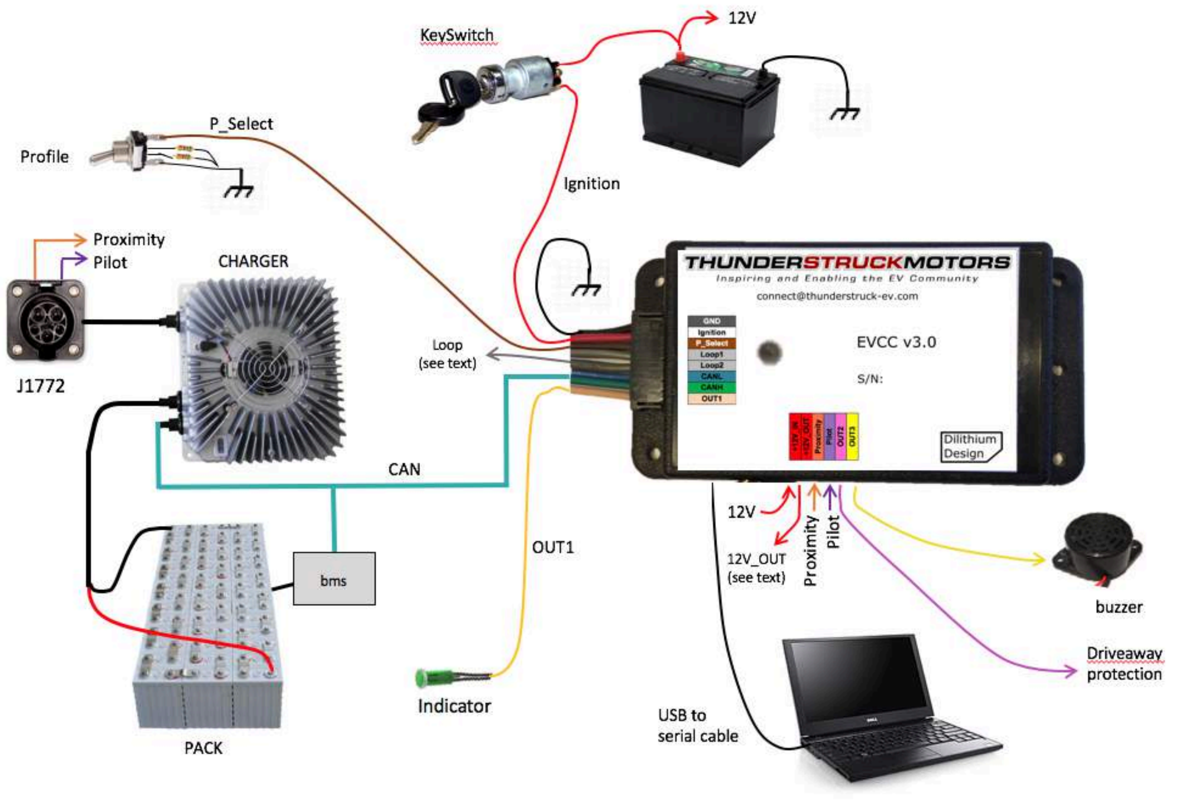
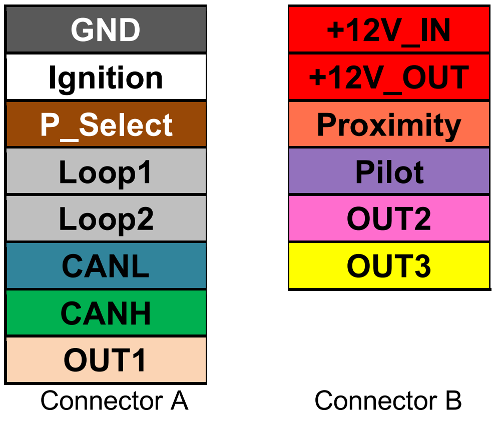


Figure 3 – Standard EVCC System Diagram



Connector A	Connector B
GND	+12V_IN
Ignition	+12V_OUT
P_Select	Proximity
Loop1	Pilot
Loop2	OUT2
CANL	OUT3
CANH	
OUT1	

Figure 4 – Standard EVCC Connections

The connections for Connector A are the same for both the EVCC-basic and standard EVCC: the only difference is that pin 2 of Connector A has been renamed from **+12V** to **Ignition**. In both cases, this signal is connected to switched 12V power such as an EV keyswitch.

Connector B has the following additional connections:

- **+12V_IN** connects to "always on" 12V power
- **+12V_OUT** is a source of switched 12V power
- **Proximity** connects to the J1772 Proximity signal
- **Pilot** connects to the J1772 Pilot signal
- **OUT2** and **OUT3** are mappable outputs.

Installation and Theory of Operation

Mechanical

The EVCC enclosure is a Serpac WM010I, a 4.61 x 2.32 x 0.6 plastic enclosure with mounting flanges. The datasheet for the enclosure can be found at http://www.serpac.com/userprints/wm010i_up_reva.pdf.

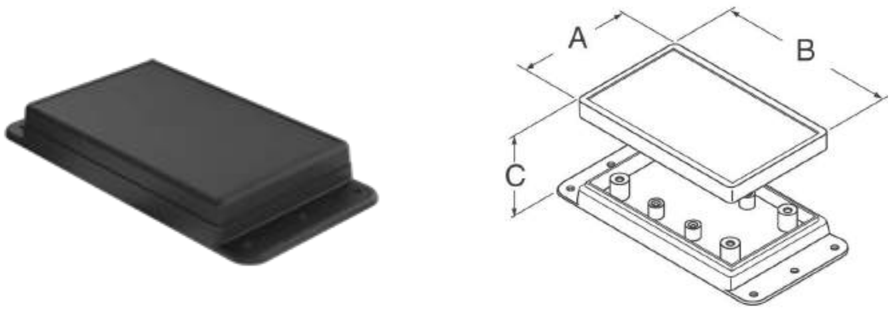


Figure 5 – EVCC Enclosure

Connector A and Connector B are "push-in" connectors. These connectors accept 20-24 gauge stranded or solid wire; stranded 20 gauge wire is recommended. To make a connection, strip the wire back 1/4". Twist the wire end and insert. Be sure that all strands of wire get correctly inserted to prevent shorting between adjacent wires.

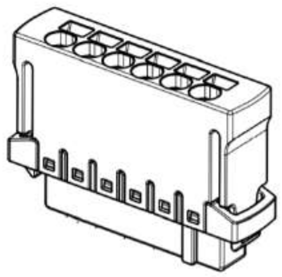


Figure 6 – Push-In Connector

Removing the wire from the connector requires a removal tool, supplied. To remove a wire, insert the tool into the associated slot above the wire and wiggle it in. This will collapse the spring holding the wire and the wire can be removed.

The connector part numbers are

• 8p Connector A	Harting 14310813101000
• 6p Connector B	Harting 14310613101000

Powering – EVCC-basic

The EVCC-basic will power up when 12V is applied to the **+12V** input and remains powered up until +12V is removed. When first powered on, the EVCC-basic will attempt a new charge cycle. If a charge completes, either normally or due to an error, then the EVCC-basic enters a STANDBY state and generally does not attempt to charge again until power is cycled[1].

The EVCC-basic draws no current when off, and approximately 35ma when on.

Powering – Standard EVCC

The standard EVCC may be powered up by providing 12V to the **Ignition** input. The standard EVCC also has an "autostart" feature which powers up the EVCC when the J1772 charge plug is inserted. To support this feature a source of "always on" 12V power is needed at the **+12V_IN** connection.

When the standard EVCC powers on, it will attempt a new charge cycle if the J1772 cable is connected and locked. If a charge completes, either normally or due to an error, the standard EVCC will attempt to power itself off. If the EVCC **Ignition** input is enabled, the standard EVCC will enter a DRIVE state. In order to reattempt charging, the J1772 charge cable must be removed and then re-inserted.

The standard EVCC provides a **+12V_OUT** output which is enabled when the EVCC is powered up. This output may be used to power downstream devices such as a BMS, instrumentation, or a relay. This output is rated at 500ma.

The standard EVCC draws approximately 3ma when off, and approximately 35ma when on.

Profile Selection (P_Select)

Up to four charging profiles may be defined with the EVCC, numbered from 1 to 4. Each profile contains a complete copy of all charging parameters. By default, Profile 1 is created and is used by default. A new profile may be defined using the "**set profile**" command and then editing the charge parameters for that profile.

When charging, the **P_Select** input is used to select the profile. The EVCC measures the resistance to GND at this input and determines four possible selections: "inf", 20K, 5K, and "0". If the **P_Select** input is left unconnected, it will read "open" (or "infinite" resistance), and maps to "inf". If the **P_Select** input is shorted to ground it will measure "0". If a resistor is connected between the **P_Select** input and ground, the remaining two choices "20K" and "5K" can be selected. The EV designer may decide to leave this feature unused, connect the input to a switch to GND to enable two profiles, or connect it to a multi-position switch and a resistor network and enable up to four profiles.

Note that the four input values represent a "switch setting" and not a "profile number". The mapping from input value to profile number is done using the command "**set map**".

Cell Loop (Loop1, Loop2)

The EVCC is intended to be used with a Battery Management System that monitors per-cell over voltage conditions when charging and per-cell undervoltage when driving. A simple BMS interface may be provided

[1] There is one exception to this rule. If the EVCC stops receiving CAN messages from the charger it will terminate charging. If however, CAN messages resume, then the EVCC-basic will assume that the charger had been powered down and has just been power cycled. In this case, the EVCC-basic will attempt a new charge cycle.

using the Cell Loop "go/no-go" contact closure. The EVCC can monitor pack health using the CAN interface, instead of, or in addition to, using the Cell Loop. See the command "**set bms**".

> WARNING: It is strongly recommended that per-cell monitoring be performed on the pack so that charging can be stopped if any cell exceeds a high voltage or low voltage cutoff. Lithium batteries can be dangerous if overcharged or undercharged.

The Cell Loop may be configured in one of two ways.

By default, the Cell Loop circuit measures the resistance between **Loop1** and **Loop2**, by applying +5v to Cell Loop1, and checking for continuity. If there is continuity, then the loop is "good"; if the loop is open, it is "bad". It is expected that the Cell Loop be implemented using solid state relays or optoisolators. Connecting the cell loop to the contacts of a mechanical relay is not recommended, as the cell loop current is limited to about 2ma. Running low current through the contacts of a mechanical relay may result in erratic operation.

Alternately, the Cell Loop may be configured to use an open collector input, when the **loopground** option is enabled. With this option, Loop1 is left unconnected. The loop is "good" if Loop2 is grounded, and "bad" if Loop2 is disconnected or "high impedance". Note that the Orion BMS supports this type of output.

A cell loop error will result in the **BUZZER** indication, which may be directed to any of the mappable outputs.

CAN (CANH, CANL)

CAN is a robust communications protocol designed for automotive applications. CAN uses a two-wire interface; the signals are designated CANH ("CAN high") and CANL ("CAN low"). Not shown, but necessary, is that each node on the CAN network must share a common ground (e.g., chassis ground). A CAN network is a daisy-chain, multi-station network that must be terminated on both ends of the string by 120ohm termination resistors. The CAN protocol has sophisticated error detection and recovery mechanisms that allow for automatic retry and recovery as well as ways of detecting and isolating misbehaving nodes. See below for a simple network diagram.

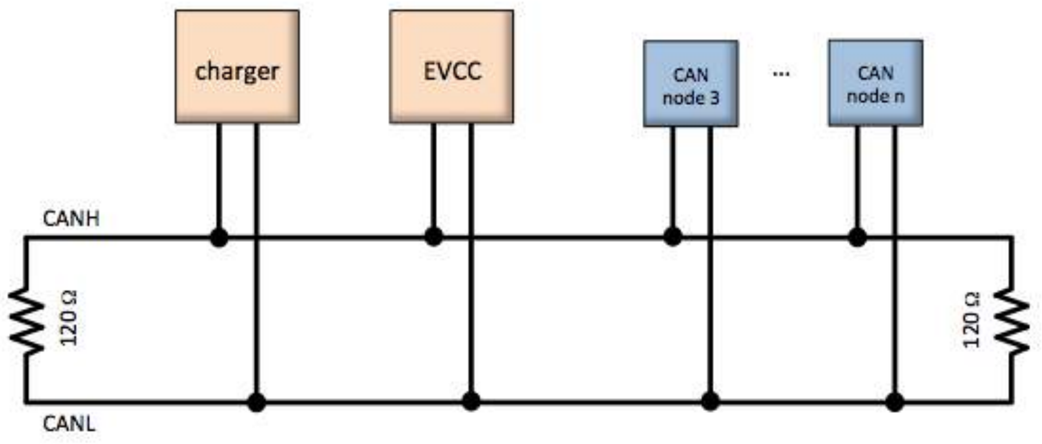


Figure 7 – CAN Network Diagram

CAN wiring should be kept short and the conductors should be twisted. Wiring stubs between the CAN network and the node should be kept as short as possible, ideally less than a few inches. Network wiring should be placed away from EMI (Electro-Magnetic Interference) such as the motor and controller, and parallel runs next to EV traction cabling should be avoided.

In a simple installation, there will be only two nodes on the CAN network: the charger and the EVCC, with a short and direct connection between the two. In this simple case, a short run of hand-twisted wire normally works fine. For longer runs, more nodes, or cases where EMI may be an issue, shielded cable may be used. In that case, the shield should be connected to chassis ground at a single place.

To simplify CAN network wiring, the EVCC contains an internal, configurable, CAN termination resistor. By default, this termination is enabled. (When the EVCC is used as an intermediate node, the termination resistor may be disabled by using the CLI command "**disable canterm**").

Some CAN devices contain an internal termination resistor and must be installed at the end of the CAN string. In order to verify the proper CAN terminations, make all connections and measure the resistance between CANH and CANL. The resistance between CANH and CANL should be 60 ohms, which indicates the presence of two 120 ohm resistors in parallel.

Note that the internal EVCC termination resistor is not bridged onto the CAN network unless the EVCC is powered up. This design choice covers the vast majority of installation scenarios[2].

CAN Protocol

The EVCC CAN interface supports 125kbps, 250kbps and 500kbp data rates and can support both Standard (11 bit) and Extended (29 bit) CAN Identifiers. The EVCC automatically sets the CAN data rate based on which charger is configured. Both the TSM2500 and ELCON chargers require 250kbps. Support is also available for the Lear charger which, if enabled, automatically reconfigures the CAN data rate to 500kbps. It is also possible to manually set the data rate (using the command "**set canbr**").

Usually two types of CAN messages are used to control a CAN enabled charger. The first, sent to Charger, provides the Charger with the desired values of charge voltage and charge current, and the second message, sent from Charger reports actual Charging Voltage and Current, as well as possibly additional charger status.

EVCC/Charger CAN messages are sent approximately twice a second, both from EVCC to Charger and from Charger to EVCC. If the EVCC stops receiving periodic messages from the Charger, charging will terminate with a termination reason of "Charger Rx Timeout".

The EVCC can also interpret BMS status messages that communicate BMS and pack status. BMS status can be used as an alternate to (or in addition with) the cell loop to indicate pack fault conditions. See *Integration with a CAN enabled BMS*, below, for more details.

CAN and Charger Message Trace

CAN messages may be lost or corrupted as the result of EMI, stubs that are too long, or improperly terminated cables. In order to verify correct operation there is both low level tracing tracing ("**trace can**") as well as high level tracing ("**trace charger**") to show CAN message traffic.

[2] The one exception is if the EVCC is a terminal node and the network needs to be active when the EVCC is powered down. In this case, the EVCC CAN termination resistor would not be enabled and network quality may suffer. In this case, the recommendation is to disable the termination resistor in the EVCC and connect a physical resistor instead.

J1772 (Proximity and Pilot)

SAE J1772 defines the physical connector and protocols used between the charging station (the "Electric Vehicle Service Equipment"), and the Electric Vehicle. The J1772 Proximity signal indicates if the charger plug is "inserted" or "locked". The J1772 Pilot signal is used to determine available charging current and allows the EV to enable the EVSE to start or stop charging.

The figure below shows the J1772 EV side connector and locations of the J1772 Proximity and J1772 Pilot signals. These are connected directly to corresponding signals at the EVCC.

> IMPORTANT: During installation, ensure that there is a good ground connection between the EV Chassis, J1772 Ground, and EVCC GND. A poor ground connection can result in nonoperation or erratic operation of the J1772 circuitry.

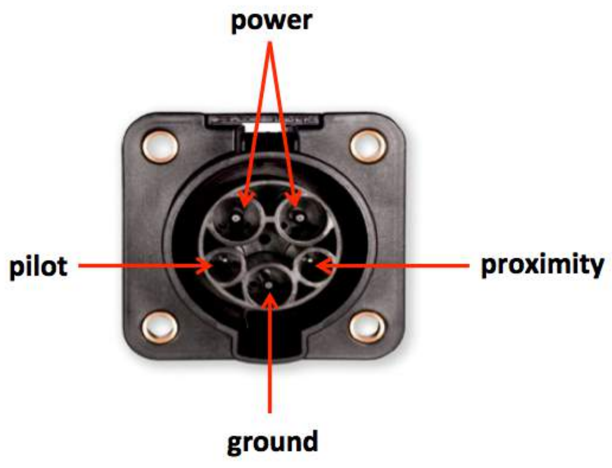


Figure 8 – Face of J1772 Socket

The J1772 charge plug is a latching plug. When it is fully inserted and latched, it is "locked". When the charger release button is pressed (by thumb on the charger plug), the charge plug becomes "unlocked", or simply "connected". The **Proximity** signal allows the EV and the EVSE to determine whether the J1772 charge plug is "disconnected", "connected" or "locked". Should the plug become "unlocked" or "disconnected" while charging, charging will immediately stop.

The **Proximity** signal allows the EV to implement "driveaway protection", which prevents the EV from being driven if the charge cable is still plugged in. If the charger plug is disconnected, the **J1772_DISCONNECTED** indication will be set. This indication can be mapped to any of the mappable outputs, which in turn, can be used to enable or disable the driving of the EV.

The **Pilot** signal allows the EVSE to advertise how much power is available for charging. The EVCC determines how much current is available from the line by measuring the duty cycle of the J1772 Pilot square wave. This can be used along with other configured information in the EVCC to determine the charging current. The **Pilot** signal also allows the EV to control when to enable charging. The EVCC controls when to start and stop charging by switching an appropriate diode and resistor between the J1772 **Pilot** signal to GND. The EVSE monitors J1772 **Pilot** and only connects mains power if requested to do so.

For more information on J1772 see http://en.wikipedia.org/wiki/SAE_J1772 and https://code.google.com/p/open-evse/wiki/J1772Basics).

Mappable Outputs (OUT1, OUT2, OUT3)

The EVCC-basic supports one mappable output (OUT1), and the standard EVCC supports three mappable outputs (OUT1, OUT2 and OUT3).

When OFF, OUT1, OUT2, and OUT3 are all disconnected and high impedance. When ON,

- OUT1 is switched to 12V. This output is rated at 200ma.
- OUT2 is switched to ground. This output is rated to 200ma.
- OUT3 is switched to ground. This output is rated to 500ma.

The EVCC firmware there are several indications that can be flexibly mapped to these outputs. These are:

Indication	**Default**	**Function**
LED	OUT1	tracks the state of the EVCC LED[3]
CHARGE		ON when charging
BUZZER	OUT3	ON when the Pack is in HVC or LVC or the cell loop is not good
J1772_DISCONNECTED	OUT2	ON when the J1772 cable is disconnected[4]
HVC		ON when the Pack is in HVC (only available when using a CAN BMS)
LVC		ON when the Pack is in LVC (only available when using a CAN BMS)

See "**set out<n>**" below for examples on how to change this mapping.

LED Operation

The EVCC LED has the following blink patterns:

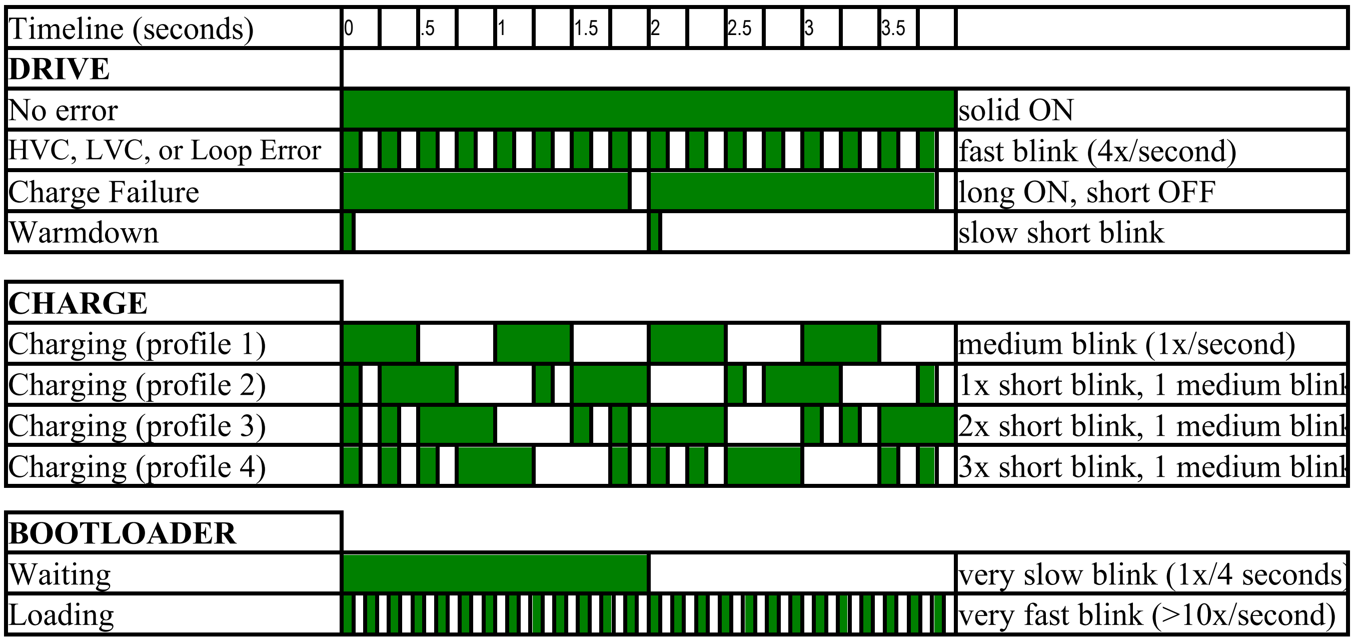


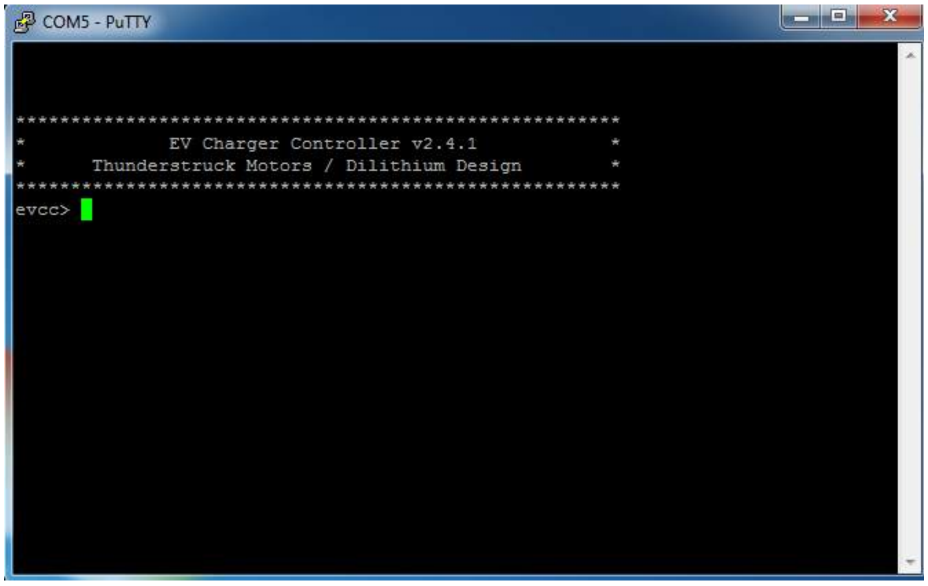
Figure 9 – EVCC Blink Patterns

[3] In EVCC 2.4 hardware, this was the EXTIND ("external indicator") output.
[4] In EVCC 2.4 hardware, this was the "OkToDrive" or "driveaway protection" output.

Serial Port

EVCC configuration is performed using a laptop and a USB to serial cable which connects to the EVCC using a 3.5mm jack. Before using the serial port, device drivers and a terminal application must be installed in the host computer. See the document *DD_SerialPortUtilities* for details on loading the drivers and on the firmware upgrade process.

Connect the serial cable to the EVCC. Apply power to the EVCC by providing a 12V supply to **12V** and **GND**. Connect +12V to **Ignition**. The EVCC **LED** should start blinking (assuming the cell loop has not been hooked up yet). If putty is the terminal program used on the host computer, the following banner should be displayed:



```
**********************************************************
*                EV Charger Controller v2.4.1            *
*         Thunderstruck Motors / Dilithium Design        *
**********************************************************
evcc>
```

At this point, the EVCC may be configured. Configuration is stored in non-volatile memory and retained across a power cycle. See below, *Command Line Interface*, for details on the commands and their syntax.

n addition to configuration, the EVCC firmware supports diagnostic commands to verify proper wiring, to trace CAN messages, and to retrieve charging history. The serial interface is used for configuration and debugging, but is not required for normal operation. Firmware upgrade is supported using the serial port, should that be necessary.

A bringup checklist is provided below. The EVCC also has several diagnostic commands that can be used to verify proper wiring (“**measure**”), to trace messages (“**trace can**”), to trace EVCC internal state changes (“**trace state**”) and to trace charger operation (“**trace charger**”).

Charging

The EVCC is supplied with defaults, but at the very minimum, it will be necessary to specify the Maximum Charging Voltage and Maximum Charging Current.

> WARNING: Lithium batteries can be dangerous if overcharged and it is strongly recommended that the user check with their battery supplier to determine appropriate charging parameters.

The maximum charging voltage (maxv) can be set using the “**set maxv**”.

The maximum charging current can be done in several ways, described below:

- Set maximum charging current, **maxc,** directly**.**

  ChargeCurrent = **maxc**

- Set the line voltage **linev** and maximum line current **linec,** do not configure **maxc**:
  In this case, the user specifies available line voltage and line current from the service connection. In this case, the EVCC will perform a power calculation and calculate the maximum charging current. The EVCC derates line power by a nominal 90% charger efficiency, and then computes an appropriate value for ChargeCurrent.

  ChargeCurrent = (**linev** * **linec** * .9)/ **maxv**

- Set **linev, linec** and **maxc**
  In this case the EVCC performs a power calculation and uses the minimum of power available and **maxc**.

  ChargeCurrent = min (**maxc**, (**linev** * **linec** * .9)/ **maxv)**

- Set **linec** to **"J1772"**, do not configure **linev** or **maxc**.
  In this case, the EVCC uses the J1772 Pilot signal duty cycle and converts it to available charge current (per J1772 this is given as 6A of charge current per 10% duty cycle). Since line voltage is not known, the EVCC uses the following rule. If the duty cycle is > 25% then the line voltage is assumed to be 220V, and if the duty cycle is less than 25%, then the line voltage is assumed to be 110V.

  ChargeCurrent = (220 * (DutyCycle * 6) * .9) / **maxv** ; if DutyCycle > 25
  = (110 * (DutyCycle * 6) * .9) / **maxv** ; if DutyCycle <= 25

  These design assumptions are driven by what is currently available in the market today in North America. EVSE equipment typically of two types: one that connects at 220V, rated at 30A, with a 50% duty cycle. The second type of equipment connects at 110V, rated at 12A, with a 20% duty cycle.

- Set **linec** to **"J1772"**, set **linev**, do not configure **maxc**
  In this case, the user specifies available line current to be **"J1772"**, and specifies the line voltage. This case is similar to the previous, however line voltage is now known.

  ChargeCurrent = (**linev** * (DutyCycle * 6) * .9) / **maxv**

Note that these calculations determine requested charging current delivered to the pack. The power capability of the charger is not configured in the EVCC and the EVCC may request more than the charger can deliver. In practice, if the requested power exceeds what the charger can deliver, charger firmware will reduce the delivered current automatically in order to stay within its power limits.

Examples of Using Different Charge Profiles

Suppose a user wants to use different EVSE charging stations: a primary EVSE at home capable of 220V at 30A, a secondary EVSE capable of 110V at 15A, and also wants the ability to do J1772 opportunity charging in general.

This could be configured in the EVSE with three charge profiles:

1) Primary EVSE. Define **maxv, linev=220, linec=30**
2) Secondary EVSE. Define **maxv, linev=110, linec=15**
3) Opportunity Charging. Define **maxv, linec=J1772**

The section *Charging Lead Acid Batteries*, below, describes the EVCC support for Sealed Lead Acid batteries. For SLA batteries, the EVCC supports a three-phase charging algorithm. The first phase (the "bulk" phase) is used by both Lithium and SLA batteries. The remaining phases, "finishing" and "float" have target voltage and current limits (fin_maxv, fin_maxc, flt_maxv, flt_maxc). The discussion above about voltage and current calculation applies equally to these two additional SLA charging phases, simply replace "xxx_maxv" and "xxx_maxc" for "maxv" and "maxc".

Charger Support

This section describes charger support by the EVCC. In EVCC terminology a "charger type" refers to both the charger model number and its unique CAN address.

TSM2500

See *TSM2500 Series High Efficiency Intelligent Charger, ThunderStruck User Manual Ver 1.0.5*. http://www.thunderstruck-ev.com/images/ThunderStruck%20TSM2500%20ManualV1.05.pdf.

The CAN connections are found on the four pin connector J3. CANL is pin #8 (wired with a blue wire) and CANH is pin #9 (wired with a green wire). No other connections are required on J3.

The TSM2500 charger does not have an integrated termination resistor (however a termination resistor may have been provided by Thunderstruck in the supplied harness). The charger is configured with a default CAN address. However, when using multiple chargers in an installation, the CAN address can be reprogrammed. The procedure to program the addresses is described below, *Setting the CAN address of a TSM2500 Charger.* Note that address programming for multiple chargers may have been done at Thunderstruck as part of the order.

The EVCC defines the following TSM2500 charger types:

- `tsm2500` - default
- `tsm2500_41`
- `tsm2500_42`
- `tsm2500_43`

The default value for tsm2500 chargers is "40". (Which is to say, the EVCC uses the CAN address 0x18e5**40**24 for messages TO the charger and 0x18eb24**40** FROM the charger to the EVCC).

The TSM2500 can report the following errors reported in the "trace charger" output:

- `rxerr`
- `hwfail`
- `overtemp`
- `not charging`
- `input voltage err`

- `pack voltage err`

ELCON PFC

In order to support a CAN interface, ELCON PFC chargers must programmed with the CAN option and a external ELCON-provided CAN module is needed. The CAN module has two pins are provided for the CAN connection: CANH and CANL and does not contain an integrated termination resistor.

The CAN addresses of the ELCON chargers are determined by the outboard serial to CAN converter: in order to change the CAN address, a different serial to CAN module is needed.

The EVCC supports the following ELCON charger types:

- `elcon` - default
- `elcon_e7`
- `elcon_e8`
- `elcon_e9`

The default value for ELCON chargers is "E5". (Which is to say, the EVCC uses the CAN address 1806**e5**f4 for messages TO the charger and 18ff50**e5** FROM the charger to the EVCC).

The ELCON charger can report the following errors reported in the "trace charger" output:

- `rxerr`
- `hwfail`
- `overtemp`
- `input voltage err`
- `pack voltage err`

LEAR

The EVCC supports some Lear chargers. This support is limited to the "control message" to the charger with CAN ID "0x00000050" and the status message from the charger with CAN ID "0x00000617". It has been found that not all Lear chargers use these messages due to different firmware and may not support this message set.

The EVCC only defines a single lear charger type, named "`lear`". When a Lear charger is configured, the CAN datarate is changed from the default of 250kbps to the Lear required 500kbps.

Only one Lear charger may be configured. Having more than one Lear charger is not supported. Having one Lear charger and another non-Lear charger is not supported. Since the Lear charger runs at 500kbps, all CAN devices on that network must be set to the 500kbps datarate.

Determining the CAN addresses of a Charger

If it is necessary to determine the CAN ID of a charger, then power up the chargers individually and use the debugging command **trace can** messages to determine what IDs are being used. The chargers will transmit these messages spontaneously, and it is not necessary to configure the charger in the EVCC to perform this test.

Bringup Checklist and Troubleshooting Hints

EV Installation

1) Install serial port drivers and terminal program on a host computer. Connect the USB to serial cable.
2) Connect GND. Connect battery to +12V (EVCC-basic) or Ignition (standard EVCC).
3) This should power up the EVCC and it will start to blink its LED. Verify that the serial port “comes up” and that a startup banner is printed.

Verify Loop (if using the cell loop)

1) With no connections to the cell loop, type “**measure loop**” and verify that the reading is < 0.2V
2) Connect the cell loop, type “measure loop” to verify that the reading is >4.8V

Verify P_Select (if using the Profile Select)

1) Type “**measure pselect**” with different switch settings to verify that the hardware is correctly reading the inputs.

Verify J1772 (if using the standard EVCC)

1) Remove +12V from Ignition / power down the EVCC
2) Connect +12V_IN
3) Connect Proximity, Pilot, GND to the charge plug
4) Plug in J1772 Plug, verify that the EVCC autostarts. The LED should start blinking and the EVCC banner should print on the serial port.
5) Assuming the CAN bus is not connected to the charger, the charge cycle should stop after 10-15 seconds and the EVCC will power itself down.

Verify Charger and CAN

1) Connect CAN between Charger and EVCC. Verify proper installation of the CAN termination resistors.
2) Power up the EVCC and configure initial charging parameters. Set **maxv** and **maxc**. See text.
3) Connect line power to the charger. If J1772 is used, this will come from the J1772 plug.
4) Power up the EVCC either by autostarting the EVCC (plug in the J1772 plug) or by applying +12V.
5) Now verify that a charge cycle is started and that messages are exchanged between EVCC and Charger. (Use “**trace charger**” or “**trace can**” to view the messages).
6) If the pack is not yet connected to the Charger, the charge cycle will stop after a minute.

BMS Integration

1) Connect +12V_OUT to power the BMS, if used.
2) Enable the bms in the EVCC (“set bms”).
3) Verify proper bms / EVCC CAN communication (“trace can”).

Systems Test

4) Configure profiles, charge parameters, and verify proper operation.
5) If J1772 is being used, verify that releasing the plug latch stops charging
6) Run a charge cycle. Check results using “**trace charger**” while charging and “**show history**” when complete.

Firmware Upgrade

The EVCC firmware may be upgraded by the customer using the serial port. See the document *DD_SerialPortUtilities* for details on loading the drivers and on the firmware upgrade process. Contact Thunderstruck if a firmware upgrade is needed.

Command Line Interface

Startup Banner

When the EVCC is powered up, it will print the following:

```
*******************************************************
*              EV Charger Controller v3.0.3           *
*       Thunderstruck Motors / Dilithium Design       *
*******************************************************
evcc>
```

help

The **help** command prints out command help.

```
evcc> help
  SHow [<>|Version|Config|History]
        <>         - status
        version    - firmware version
        config     - configuration
        history    - charge history
  SEt [ <>
        |BMS|CANBR|OUT1|OUT2|OUT3
        |CHARGER|CHARGER2|CHARGER3|CHARGER4
        |PROfile|MAP
        |LINEV|LINEC
        |MAXV|MAXC|MAXBC|TERMC|TERMT
        |FIN_MAXV|FIN_MAXC|FIN_TERMT
        |FLT_MAXV|FLT_MAXC|FLT_TERMT
       ]
  REset [History|PROFILE]
        history       - reset charge history
        profile <n>   - deletes a charge profile
  ENable [CANTERM}TOPBALANCE|LOOPGROUND]
        canterm    - enable CAN termination resistor
        topbalance- cell HVC stops current but not charging
        loopground- loop OK if loop2 grounded
  DISable [CANTERM|TOPBALANCE|LOOPGROUND]
  TRace [CHarger|CANbus|STate|OFF]
        <>         - trace toggle ON/OFF
        charger    - trace charger messages
        canbus     - trace canbus messages
        state      - trace EVCC state changes
        off        - disable all tracing
  MEasure [<>|LOOP|PROXimity|PSELect]
        <>         - 'measure' help
        loop       - measure Cell Loop A/D
        proximity - measure J1772 Proximity A/D
        pselect    - measure Profile Selection A/D
  UPGRADE          - performs a firmware upgrade
```

In many cases, either a full version or an abbreviated version of a command (or command parameter) can be used. This is shown in the "help" with the use of uppercase and lowercase letters. For example, the abbreviation for **show** is **sh**, and the abbreviation for **show config** is **sh c**.

show

The **show** command displays configured parameters or status. If “show” is entered without parameters, current status will be displayed.

In the Drive state, the EVCC monitors the cell loop and sets the BUZZER indication when the cell loop opens.

```
evcc> show
  state    : DRIVE
  cell loop: OK
  proximity: EVSE not connected
  OUT1     : OFF - LED
  OUT2     : ON  - EVSEDISC
  OUT3     : OFF - BUZZER
  uptime   : 0 hour(s), 0 minute(s), 33 second(s)
```

If, instead, the bms configuration is set to “**bmsc**” instead of “**loop**”, the output would be the following:

```
evcc> set bms bmsc
evcc> show
  state    : DRIVE
  bmsc     : OK
  proximity: EVSE not connected
  OUT1     : OFF - LED
  OUT2     : ON  - EVSEDISC
  OUT3     : OFF - BUZZER
  uptime   : 0 hour(s), 0 minute(s), 33 second(s)
```

In the CHARGE state, the EV is charging.

```
evcc> show
  state    : CHARGE
  cell loop: OK
  proximity: EVSE Connected and locked
  buzzer   : OFF
  J1772    : duty cycle= 50%, line current available= 30.0A
  charger  : tsm2500
    status : 14 msgs sent; 11 msgs received
    voltage:   53.4V
    current:    1.9A
    charge : 0.12Wh
  uptime   : 0 hour(s), 2 minute(s), 0 second(s)
```

show version

The **version** command displays firmware version number and build date.

```
evcc> show version
  version  :    v3.0.5; Aug 12 2018 06:36:40
evcc>
```

show config

The **show config** command displays configuration parameters. At its simplest, the output of **show config** is the following:

```
evcc> show config
  bms      : bmsc
  canbr    : 250kbps
  OUT1     : LED
  OUT2     : EVSEDISC
  OUT3     : BUZZER
  charger  : tsm2500
  maxv     :  20.0V
  maxc     :   2.0A
  termc    :   0.2A
  termt    : 720.0hr
  options  : canterm (CAN termination resistor enabled)
```

The output of **show config** becomes progressively more complex as more features are enabled. If only one charge profile is defined, the full set of configured parameters is given below:

- `bms` - the bms type(s) (`none` or one or more of `loop, bmsc, bmsc2, bmsc3, bmsc4`)
- `canbr` - configured can baud rate
- `OUT1` - mappings for OUT1 (`none` or `LED, HVC, LVC, CHARGE, BUZZER, EVSEDISC`)
- `OUT2` - mapping for OUT2 (standard EVCC only)
- `OUT3` - mapping for OUT3 (standard EVCC only)
- `charger` - the configured charger type
- `charger2` - (if configured) types of chargers2
- `charger3` - (if configured) type of chargers3
- `charger4` - (if configured) type of chargers4
- `linev` - (if configured) line voltage of service connection
- `linec` - (if configured) line current of service connection
- `maxv` - maximum charging voltage (in Volts).
- `maxc` - maximum charging current (in Amps).
- `maxbc` - (if configured) maximum balance current
- `termc` - terminating charging current (in Amps).
- `termt` - maximum charging time (in minutes).
- `fin_maxv` - (if configured) finishing charge voltage (for SLA charging)
- `fin_maxc` - (if configured) finishing charge current (for SLA charging)
- `fin_termt` - (if configured) finishing charge current (for SLA charging)
- `flt_maxv` - (if configured) float charge voltage (for SLA charging)
- `flt_maxc` - (if configured) float charge current (for SLA charging)
- `flt_termt` - (if configured) float charge current (for SLA charging)
- `options` - `canterm` (if enabled) can termination resistor enabled
  - `topbalance` (if enabled) enable 'top balance' feature
  - `loopground` (if enabled) enables open collector cell loop surveillance

An example of a full output with all options is shown below:

```
evcc> show c
  bms      : loop
  charger  : tsm2500
  charger2 : tsm2500_42
  charger3 : tsm2500_43
  charger4 : elcon
  linev    : 220.0V
  linec    :  30.0A
  maxv     : 155.0V
  maxc     :  15.0A
  maxbc    :   1.2A
  termc    :   0.2A
  termt    :   6.0hr
  fin_maxv : 160.0V
  fin_maxc :   2.0A
  fin_termt:   4.0hr
  flt_maxv : 152.0V
  flt_maxc :   0.5A
  flt_termt:   0.0hr
  options  : cantermdis (CAN termination resistor disabled}
           : topbalance (see documentation)
           : loopground (ignore loop1; apply ground to loop2 if OK)
```

If more than one charge profile is defined, **show config** will display the charge profiles in “tabular form”. The charge profile selected for editing is indicated with a “*”. Also, the profile map is shown.

Example output with multiple charge profiles is shown below:

```
evcc> show c
  bms      : loop
  charger  : tsm2500
  profiles   :    1         2*        3         4
    linev    : 220.0V
    linec    :  30.0A     J1772
    maxv     : 155.0V    152.0V
    maxc     :  15.0A      2.0A
    maxbc    :   1.2A
    termc    :   0.2A      0.2A
    termt    :   6.0hr     8.0hr
    fin_maxv : 160.0V
    fin_maxc :   2.0A
    fin_termt:   4.0hr
    flt_maxv : 152.0V
    flt_maxc :   0.5A
    flt_termt:   0.0hr
  profile map:
    inf      :    x
    20K      :    x
    5K       :    x
    0        :              x
```

show history

The **show history** command displays data about the last sixteen charge cycles. See also **reset history**, below.

In the first example, the system has no charge history.

```
evcc> show history
no charge history
```

The next example shows charge history, with different “termination reasons”. The termination reason contains the reason that the charge cycle stopped. In this example, in the most recent charge attempt, the user disconnected the J1772 plug one minute after charging started. (EVSE disc, 1 mins). The previous attempt

(“-1”) shows a normal charge completion with a charge time of 214 minutes and includes the number of watt hours delivered.

```
evcc> show history

      |    term    |  charge |            |   watt | maximum| maximum|  ending|
  num |  reason    |   time  |  charger   |  hours | voltage| current| current|
---------------------------------------------------------------------------
 last | EVSE disc|   1 mins|tsm2500    |     7Wh| 148.9V |   7.9A |   7.9A |
  - 1 |   normal | 214 mins|tsm2500    |  3249Wh| 152.9V |   7.9A |   0.5A |
  - 2 | EVSE disc|   1 mins|tsm2500    |     0Wh| 144.8V |   0.0A |   0.0A |
  - 3 | comm err |   0 mins|tsm2500    |     0Wh|   0.0V |   0.0A |   0.0A |
evcc>
```

The full set of “term reason” codes is:

• `EVSE disc`	- J1772 charge plug became unlocked while charging
• `cell loop`	- a cell loop error was detected
• `comm err`	- communications error with the charger
• `pack disc`	- no pack was detected
• `timeout`	- the maximum charge time was reached
• `normal`	- normal completion (charge current is less than terminating charging current)
• `fintimeout`	- finishing charge timeout
• `fin normal`	- normal termination of finishing charge
• `flttimeout`	- float charge timeout
• `HVC`	- BMS reported a cell HVC error
• `BMS fault`	- BMS reported fault (not locked, cell census, thermistor fault, or thermistor census)

When multiple chargers are configured, the format of the charge history is modified to show the contribution of each charger.

```
evcc> show history

      |    term    |  charge |            |   watt | maximum| maximum|  ending|
  num |  reason    |   time  | charger    |  hours | voltage| current| current|
--------------------------------------------------------------------------
 last | EVSE disc|   2 mins|tsm2500    |     6Wh| 127.8V |   2.2A |   0.0A |
                           |tsm2500_42|     6Wh| 127.5V |   2.0A |   0.0A |
                           |TOTAL      |    12Wh| 127.8V |   4.2A |   0.0A |
```

set

This command sets the configurable parameters. For voltage and current, whole numbers (145) or decimal numbers (145.2) can be entered. The EVCC supports one decimal digit of precision.

set <>

Using the **set** with no parameters will print help for the "set" command. The following example shows this output for the EVCC standard.

```
evcc> set
  SEt [ <>
        |BMS|CANBR|OUT1|OUT2|OUT3
        |CHARGER|CHARGER2|CHARGER3|CHARGER4
        |PROfile|MAP
        |LINEV|LINEC
        |MAXV|MAXC|MAXBC|TERMC|TERMT
        |FIN_MAXV|FIN_MAXC|FIN_TERMT
        |FLT_MAXV|FLT_MAXC|FLT_TERMT
       ]
      <>                   - 'set' help
      bms configuration
        set bms [NONE|<bmsn>[,<bmsn>[,<bmsn>[,<bmsn>]]]
            <bmsn>         - [LOOP|BMSC|BMSC2|BMSC3|BMSC4]
      CAN baud rate
        set canbr [125|250|500]
      OUT<n> mapping
        set <outn> [NONE|LED|HVC|CHARGE|BUZZER|EVSEDISC]
      charger configuration
        set <charger> <type> - defines <chargern>
            <chargern>     - [CHARGER|CHARGER2|CHARGER3|CHARGER4]
            <ch>           - [TSM2500|TSM2500_41|TSM2500_42|TSM2500_43
                             |ELCON|ELCON_E7|ELCON_E8|ELCON_E9|LEAR
                             |NONE]
        set <chargern> <ch> PROGRAM - programs tsm2500 CAN IDs
      profile editing
        set profile <n>     - set profile for editing
        set map [inf|20K|5K|0|all] <n>
                            - set profile mapping
      Service parameters
        set linev <v>       - available line voltage
        set linec [<a>|J1772] - available line current
      BULK charge parameters
        set maxv <v>        - maximum charge voltage
        set maxc <a>        - maximum charge current
        set maxbc <a>       - maximum balancing current
        set termc <a>       - charge termination current
        set termt <h>       - charge termination timeout
      FINISH charge parameters
        set fin_maxv <v>   - finishing charge voltage
        set fin_maxc <a>   - finishing charge current
        set fin_termt <h> - finishing charge termination timeout
      FLOAT charge parameters
        set flt_maxv <v>   - float charge voltage
        set flt_maxc <a>   - float charge current
        set flt_termt <h> - float charge termination timeout (0=no timeout)
```

set bms

This command sets the BMS type. The EVCC can use a cell loop and/or up to four CAN BMSs.

The following example just sets the bms type to be the cell loop.
```
evcc> set bms loop
```

The next example sets the bms to use CAN messaging.
```
evcc> set bms bmsc
```

The next example sets the bms to use both cell loop and CAN messaging.
```
evcc> set bms loop, bmsc
```

set canbr

This command sets the can baud rate. The EVCC supports 125kbps, 250kbps, and 500kbps. Note that the can baud rate is automatically set based on the charger type, however, this may be overridden by this command if necessary.

```
evcc> set canbr 500
```

set out1 (set out2, set out3)

This command sets the output mapping. Note that the EVCC-basic does not support OUT2 and OUT3, nor does it support the mappable functions of CHARGE, BUZZER or EVSEDISC. Note that the OUT1, OUT2, and OUT3 have different hardware capabilities and must be wired appropriately. See text above.

```
evcc> set out2 EVSEDISC
```

set charger<n>

This sets the charger type. The first charger is named "charger". Chargers 2 through 4 are named "charger2", "charger3", "charger4".

The following command sets a single charger

```
evcc> set charger tsm2500
```

The following command sets a second charger

```
evcc> set charger2 tsm2500_42
```

set profile <n>

This command selects a profile for editing. There are four possible profiles: 1-4. Profile 1 is automatically created. It is the default profile and cannot be deleted. If the user types "**set profile <n>**", then this will both select a profile for editing and create the profile if it does not already exist. Once a profile is selected, then subsequent editing commands (e.g., **set maxv**, etc.) apply to the parameters associated with the profile. Profiles 2-4 may be deleted using the command "**reset profile <n>**".

Examples of creating and editing profiles:

This is the default configuration:

```
evcc> show config
  bms      : loop
  charger  : tsm2500
  maxv     :  20.0V
  maxc     :   2.0A
  termc    :   0.2A
  termt    : 720.0hr
evcc>
```

This command creates Profile 2 with default configuration:

```
evcc> set profile 2
evcc> show c
  bms      : loop
  charger  : tsm2500
  profiles   :    1         2*        3         4
    maxv     :  20.0V     20.0V
    maxc     :   2.0A      2.0A
    termc    :   0.2A      0.2A
    termt    : 720.0hr   720.0hr
profile map:
    inf      :    x
    20K      :    x
    5K       :    x
    0        :    x
```

Now set some parameters in Profile 2:

```
evcc> set maxv 150
evcc> set maxc 12
evcc> show config
  bms      : loop
  charger  : tsm2500
  profiles   :    1         2*        3         4
    maxv     :  20.0V    150.0V
    maxc     :   2.0A     12.0A
    termc    :   0.2A      0.2A
    termt    : 720.0hr   720.0hr
profile map:
    inf      :    x
    20K      :    x
    5K       :    x
    0        :    x
```

Now return to Profile 1 and set some parameters in Profile 1:

```
evcc> set profile 1
evcc> set maxv 160
evcc> set maxc 15
evcc> set linec j1772
evcc> show config
  bms      : loop
  charger  : tsm2500
  profiles   :    1*        2         3         4
    linec    :  J1772
    maxv     : 160.0V    150.0V
    maxc     :  15.0A     12.0A
    termc    :   0.2A      0.2A
    termt    : 720.0hr   720.0hr
profile map:
    inf      :    x
    20K      :    x
    5K       :    x
    0        :    x
```

Finally, delete Profile 2:

```
evcc> reset profile 2
evcc> show config
  bms       : loop
  charger   : tsm2500
  linec     :  J1772
  maxv      : 160.0V
  maxc      :  15.0A
  termc     :   0.2A
  termt     : 720.0hr
```

set map

This command sets the profile map. The EVCC measures resistance to ground at the P_Select input and from the measurement determines four possible choices, as follows:

R >= 30K	the result is “inf”
30K > R >= 10K	the result is “20K”
10K > R >= 2K	the result is “5K”
20K > R	the result is “0”

Each P_Select range is mapped to a profile using the profile map. Initially everything is mapped to the default profile, Profile 1. If the user wants to use two profiles, a switch to ground may be connected at the P_Select input. If the switch is open then Profile 1 is used and if the switch is closed, then Profile 2 is used. Once Profile 2 is defined, this may be configured as follows:

```
evcc> set map 0 2
evcc> show config
  bms       : loop
  charger   : tsm2500
  profiles     :    1         2*        3         4
    linec      :  J1772
    maxv       : 160.0V    150.0V
    maxc       :  15.0A     12.0A
    termc      :   0.2A      0.2A
    termt      : 720.0hr   720.0hr
profile map:
    inf      :    x
    20K      :    x
    5K       :    x
    0        :              x
```

If a third profile were to be defined, a switchable resistor would be required at the P_Select input. The CLI command to enable the mapping from a 5K resistor to Profile 3 would be:

```
evcc> set map 5K 3
```

set linev, set linec

This sets the maximum line voltage and line current available.

```
evcc> set linev 110
evcc> set linec 12.5
```

Note that the set linec command has “j1772” as an option. In this case, the J1772 duty cycle will be used to determine available line current.

```
evcc> set linec j1772
```

set maxv, set maxc

The command **set maxv** sets the maximum charging voltage, in Volts.
The command **set maxc** sets the maximum charging current, in Amps.

```
evcc> set maxv 155.0
evcc> set maxc 8.5
```

set maxbc

This sets the maximum balancing charging current, in Amps. This option is only possible if a CAN BMS is used and it sends a “BVC threshold exceeded” indication to the EVCC.

```
evcc> set maxbc .7
```

set termc

This sets the termination charging current, in Amps. If the current drops below this setpoint then the charging stops.

```
evcc> set termc .5
```

set termt

This sets the maximum charging time, in hours.

```
evcc> set termt 6.5
```

set fin_maxv, set fin_maxc, set fin_termt

These commands are used to define the “finishing charge” phase voltage, current, and charge time for Sealed Lead Acid battery charging. See below, *Charging Lead Acid Batteries*, for examples of use.

set flt_maxv, set flt_maxc, set flt_termt

These commands are used to define the “float charge” phase voltage, current, and charge time for Sealed Lead Acid battery charging. See below, *Charging Lead Acid Batteries*, for examples of use.

reset

reset history

The **reset history** command resets the charge history.

```
evcc> reset history
charge history has been reset
evcc>
```

reset profile <n>

The reset profile command can be used to delete Profiles 2-4. It is not possible to delete Profile 1.

```
evcc> reset profile 3
```

enable | disable

The EVCC supports several options that may be "enabled" and "disabled". Options that are "enabled" are shown in the **show config** output.

enable | disable canterm

The EVCC contains an integrated, and programmable CAN termination resistor. By default, this termination resistor is enabled. In order to enable this termination resistor, use the command:

```
evcc> enable canterm
```

In order to disable this option, use the **disable canterm** command.

enable | disable topbalance

Normally, the charging cycle terminates when the cell loop opens or when the BMS indicates a pack HVC condition. Some customers may not want to completely stop charging at this stage: instead they may want to suspend charging, allow the pack to "rest" and for the HVC condition to clear, and to resume charging. The theory is that lower charged cells may be topped up with this process. In this case, the cell loop / HVC condition does not stop charging.

```
evcc> enable topbalance
```

In order to disable this option, and return the EVCC to default behavior, use the **disable topbalance** command.

enable | disable loopground

Setting this option will change the method of loop supervision. With the "loopground" option the Cell Loop is considered good if there is a ground on Loop2. (High Impedance means that the cell loop is not good). With this option, Loop1 must be left unconnected.

```
evcc> enable loopground
```

In order to disable this option, and return the EVCC to default behavior, use the **disable loopground** command.

trace

The **trace** command enables various forms of message or state tracing. These commands show a timestamp (uptime) and can be useful for logging or debugging. CHARGER, STATE, and CAN tracing may be independently enabled.

Trace configuration is stored in EEPROM and is present after reboot.

trace charger

The **trace charger** command displays messages from the charger. This trace also shows the current number of charging watts and the accumulated watt-hours of charge.

```
evcc> trace charger
charger tracing is now ON
evcc> 00:08:22.7  V=148.6, A= 7.9, W=1173, Wh= 0.96, TMP = 26C
00:08:23.1  V=148.6, A= 7.9, W=1173, Wh= 1.12, TMP = 26C
00:08:23.6  V=148.6, A= 7.9, W=1173, Wh= 1.28, TMP = 26C
00:08:24.1  V=148.6, A= 7.9, W=1173, Wh= 1.45, TMP = 26C
```

```
00:08:24.6  V=148.6, A= 7.9, W=1173, Wh= 1.61, TMP = 26C
00:08:25.1  V=148.6, A= 7.9, W=1173, Wh= 1.77, TMP = 26C
00:08:25.6  V=148.6, A= 7.9, W=1173, Wh= 1.93, TMP = 26C
00:08:26.1  V=148.6, A= 7.9, W=1173, Wh= 2.08, TMP = 26C
00:08:26.6  V=148.6, A= 7.9, W=1173, Wh= 2.25, TMP = 26C
00:08:27.1  V=148.6, A= 7.9, W=1173, Wh= 2.41, TMP = 26C
00:08:27.6  V=148.6, A= 7.9, W=1173, Wh= 2.57, TMP = 26C
00:08:28.0  V=148.6, A= 7.9, W=1173, Wh= 2.73, TMP = 26C
00:08:28.6  V=148.6, A= 7.9, W=1173, Wh= 2.89, TMP = 26C
00:08:29.0  V=148.6, A= 7.9, W=1173, Wh= 3.05, TMP = 26C
00:08:29.6  V=148.9, A= 7.9, W=1176, Wh= 3.22, TMP = 26C
```

trace can

The **trace canbus** command displays canbus messages to and from the charger. Each line gives a timestamp, the originator of the message (if known), the CAN ID and CAN message contents, in hexadecimal.

```
evcc> trace can
canbus tracing is now ON
evcc> 00:02:20.9        evcc: 18e54024 fc c8 00 6c 0c 01 ff ff
00:02:21.4        evcc: 18e54024 fc c8 00 6c 0c 01 ff ff
00:02:21.9        evcc: 18e54024 fc c8 00 6c 0c 01 ff ff
00:02:22.4        evcc: 18e54024 fc c8 00 6c 0c 01 ff ff
00:02:22.5  tsm2500    : 18eb2440 42 f7 41 fd 00 fe 12 dd
00:02:22.9        evcc: 18e54024 fc c8 00 6c 0c 01 ff ff
00:02:22.9  tsm2500    : 18eb2440 04 fd 13 02 80 0c 3f ff
00:02:23.4        evcc: 18e54024 fc c8 00 6c 0c 01 ff ff
00:02:23.5  tsm2500    : 18eb2440 00 fc 13 02 80 0c 3f ff
00:02:23.9        evcc: 18e54024 fc c8 00 6c 0c 01 ff ff
00:02:23.9  tsm2500    : 18eb2440 00 fc 13 02 80 0c 3f ff
```

trace state

The **trace state** command displays internal EVCC state transitions. For the standard EVCC, there is also J1772 state.

Here is an example of state trace output that shows the charger plug being plugged in and unplugged, also showing the attempt to charge.

```
evcc> trace state
state tracing is now ON
evcc>
evcc> 00:00:31.7 j1772=LOCKED
00:00:31.7 old state=DRIVE, new state=WARMUP, term rsn=0
00:00:31.8 old state=WARMUP, new state=CHARGE, term rsn=0
00:00:52.2 j1772=CONNECTED
00:00:52.3 j1772=WAITING FOR DISC
00:00:52.4 old state=CHARGE, new state=STANDBY, term rsn=EVSE UNLOCKED
00:00:56.1 j1772=DISCONNECTED
00:00:57.4 old state=STANDBY, new state=DRIVE, term rsn=0
```

The list of EVCC state values is:

• WARMUP	state just after power up
• CHARGE	charging – bulk charge phase
• FINISH CHARGE	charging – finish charge phase
• FLOAT CHARGE	charging – float charge phase
• CHARGE/TOPBALANCE	topbalance – waiting for cells to drop below HVC

• STANDBY	standby state
• DRIVE	drive state – standard EVCC only

The list of J1772 state values is:

• DISCONNECTED	charge plug is not plugged in
• CONNECTED	charge plug is plugged in but not "locked"
• LOCKED	charge plug is plugged in and "locked"
• WAITING FOR DISC	charge plug was LOCKED, is now CONNECTED now waiting for it to be completely removed

trace off

The **trace off** command turns off all tracing.

```
evcc> tr off
all tracing is now OFF
```

measure

The **measure** command is used to verify the A/D inputs. When this command is issued, the EVCC will repeatedly measure and print the value of an analog input. The command will run for 30 seconds and then automatically turn itself off. Alternately, the user can stop the command by typing any character.

The **measure** command with no parameters will display the expected values of the A/D inputs.

```
evcc> measure
This command repeatedly shows an analog input for 30 seconds.
Press any key to stop display

  The following values are expected
    loop      - Cell Loop A/D
                V >= 2.5V - OK
    proximity - J1772 Proximity A/D
                V >= 4.0V - disconnected
          4.0 > V >= 1.2V - connected
          else            - locked
    pselect   - Profile Selection A/D
                R >= 30K    - inf
          30K > R >= 10K    - 20K
          10K > R >= 2K     - 5K
           2K > R           - 0
evcc>
```

measure loop

The **measure loop** command gives a real time measurement of the **cell loop**.

```
evcc> measure loop
evcc> Loop A/D= 4.97V
Loop A/D= 4.97V
Loop A/D= 4.97V
Loop A/D= 4.97V
Loop A/D= 4.97V
```

measure proximity

The **measure proximity** command gives a real time measurement of the **Proximity** input.

In the example given below, both the **measure proximity** and **trace state** commands are enabled. Initially the J1772 charge plug is disconnected (measurement is 4.67V), then it becomes locked (measurement is 0.86V), then connected and unlocked (1.97V), and then finally, disconnected.

```
evcc> trace state
state tracing is now ON
evcc> me prox
evcc> Proximity A/D= 4.67V
Proximity A/D= 4.67V
Proximity A/D= 4.67V
Proximity A/D= 4.67V
Proximity A/D= 4.67V
Proximity A/D= 4.67V
Proximity A/D= 4.67V
Proximity A/D= 4.67V
Proximity A/D= 0.86V
00:05:02.4 j1772=LOCKED
00:05:02.4 old state=DRIVE, new state=WARMUP, term rsn=0
00:05:02.5 old state=WARMUP, new state=CHARGE, term rsn=0
Proximity A/D= 0.86V
Proximity A/D= 0.86V
Proximity A/D= 0.86V
Proximity A/D= 0.86V
Proximity A/D= 0.86V
Proximity A/D= 0.86V
Proximity A/D= 0.86V
Proximity A/D= 0.86V
Proximity A/D= 0.86V
Proximity A/D= 0.86V
Proximity A/D= 0.85V
Proximity A/D= 0.86V
Proximity A/D= 1.97V
00:05:05.6 j1772=CONNECTED
00:05:05.7 j1772=WAITING FOR DISC
00:05:05.8 old state=CHARGE, new state=STANDBY, term rsn=EVSE UNLOCKED
Proximity A/D= 1.96V
Proximity A/D= 1.96V
Proximity A/D= 1.97V
Proximity A/D= 1.97V
Proximity A/D= 1.97V
Proximity A/D= 4.67V
00:05:07.9 j1772=DISCONNECTED
Proximity A/D= 4.68V
Proximity A/D= 4.67V
Proximity A/D= 4.67V
00:05:10.8 old state=STANDBY, new state=DRIVE, term rsn=0
```

measure pselect

The **measure pselect** command gives a real time measurement of the **P_Select** input. Note that this output is reported in resistance.

```
evcc> me pselect
evcc> Pselect A/D= inf
Pselect A/D= inf
Pselect A/D= inf
Pselect A/D= inf
Pselect A/D= inf
```

upgrade

The **upgrade** command is used to perform a firmware upgrade. This command will place the EVCC into the serial bootloader state, waiting for the load to begin. The EVCC must be power cycled in order to leave this state. See *Firmware Upgrade*, for a description of how to perform a firmware upgrade.

```
evcc> upgrade


 ***
 ***                   Starting EVCC Upgrade                         ***
 *** 1) Exit from the terminal application                           ***
 *** 2) Start the bootloader and download a new .hex file            ***
 *** 3) Restart the EVCC                                             ***
```

Configuring the EVCC with Multiple Chargers

Up to four chargers can be used in parallel for faster charging. A logical picture is shown in the diagram below.

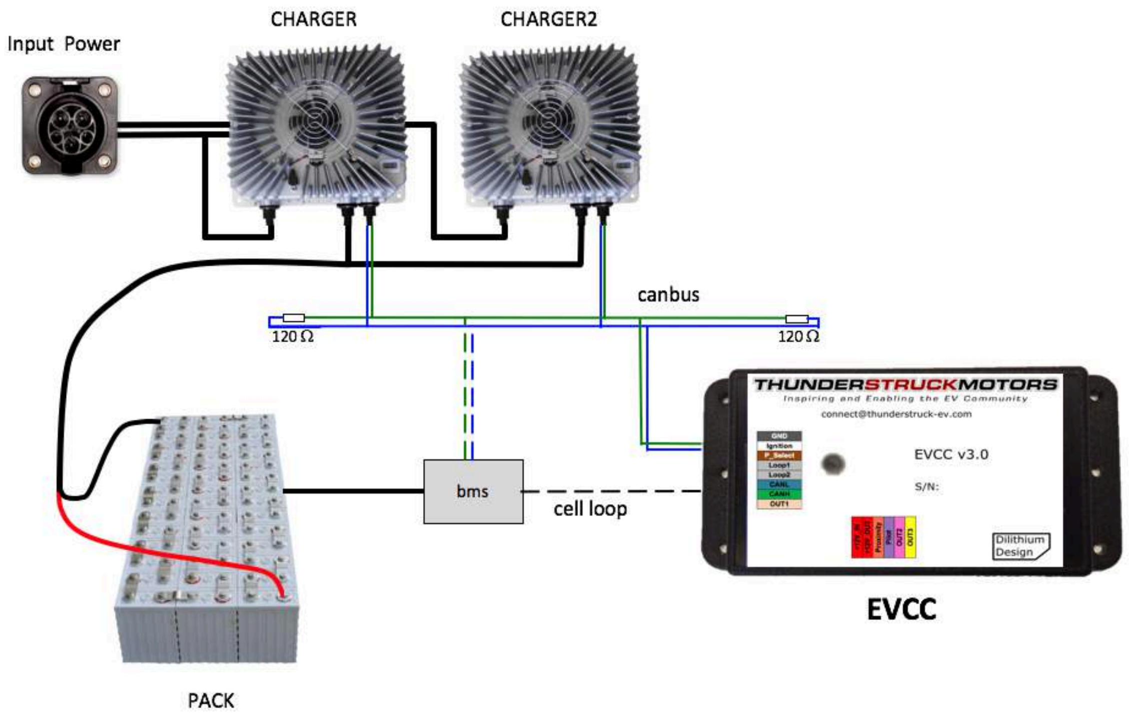


Figure 10 – Multiple Chargers - System Diagram

If J1772 is used, there is a single J1772 interface for line power which feeds all chargers. The chargers are in parallel and they charge a single pack. All chargers are placed on the CANBUS. There is a single EVCC and it communicates with the chargers independently. (Also shown on the CANBUS is a CAN enabled BMS, optionally present).

There are several design considerations when installing multiple chargers. Two chargers require more power than a single charger. One must verify that adequate line power is available. With more CAN nodes, the CAN wiring is no longer simply point to point and installation must be done with care. Each charger requires a unique CAN ID. Each charger must be explicitly configured.

Line Power

The EVCC assumes that the service can provide 220V at 30A. Power calculations are needed to make sure that there is sufficient power available to power all chargers. A 220V / 30A circuit has 6600Watts available. Two 2.5Kw chargers running at full power can be placed on the line, but three chargers cannot. (In contrast, a 110V / 15A circuit only has 1650Watts available).

CAN Wiring and Addressing

See the section on CANBUS, above, for general guidelines. When installing multiple chargers, care must be taken that termination resistors are properly placed. Keep in mind that some chargers have a termination resistor installed in the charger, and so that charger must be at the end of the CAN string.

Each charger must have a unique CAN address. See the section *Charger Support* for information on how to determine the charger CAN address and change it if necessary.

EVCC Configuration

The EVCC supports up to four chargers (named: `charger, charger2, charger3, and charger4).` Chargers are defined in the EVCC using the **set charger** command. When a charger is configured, it is set to a "charger type", which indicates both the manufacturer and its CAN address. It is possible to have chargers from multiple manufacturers (e.g., one ELCON and one TSM2500) at the same time.

The following example defines a single charger and sets its type to tsm2500:

```
evcc> set charger tsm2500
evcc> show config
  bms      : loop
  charger  : tsm2500
  maxv     : 158.0V
  maxc     :  12.0A
  termc    :   0.5A
  termt    : 720.0hr
evcc>
```

This example defines a second charger, and sets its type to tsm2500_42.

```
evcc> set charger2 tsm2500_42
evcc> show config
  bms      : loop
  charger  : tsm2500
  charger2 : tsm2500_42
  maxv     : 158.0V
  maxc     :  12.0A
  termc    :   0.5A
  termt    : 720.0hr
evcc>
```

A charger can be deleted by setting the type to "none".

```
evcc> set charger2 none
```

Charging

When charging with multiple chargers, maxc is divided by the number of chargers and given to each charger. So here is an example of charger tracing when maxc is set to 12A. Note that 6A goes to both

TSM2500 and TSM2500_42. Note that "trace charger" reports the status of the charger … and that voltage, current, watts, and watt hours may be slightly different.

```
evcc> trace charger
charger tracing is now ON
00:10:28.8  tsm2500_42: V=126.0, A= 5.8, W=730, Wh= 0.10, TMP = 26C
00:10:28.9  tsm2500   : V=126.3, A= 5.9, W=745, Wh= 0.09, TMP = 26C
00:10:29.3  tms2500_42: V=126.6, A= 5.7, W=721, Wh= 0.19, TMP = 26C
00:10:29.3  tsm2500   : V=126.6, A= 5.8, W=734, Wh= 0.19, TMP = 26C
00:10:29.8  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.30, TMP = 26C
00:10:29.9  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.31, TMP = 26C
00:10:29.9  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.33, TMP = 26C
00:10:30.0  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.34, TMP = 26C
00:10:30.0  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.36, TMP = 26C
00:10:30.1  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.37, TMP = 26C
```

Setting the CAN address of a TSM2500 Charger

This section describes how to set the CAN addresses of a tsm2500 charger.

For this procedure, the charger can either be directly connected to mains power, or can be installed in the vehicle and the J1772 charge plug can be used to supply line power. When doing this procedure, insure that only one charger is powered.

In this example, we want to define a second charger as type tsm2500_42. In order to program the charger, it is necessary to use the **program** keyword.

To do this, power up the EVCC by keyswitch. Then type the following command:

```
      evcc> set charger2 tsm2500_42 program
```

The EVCC will then print

```
       ***
       ***                      tsm2500 PROGRAMMING                      ***
       *** WARNING: This command changes the CAN IDs of a tsm2500 charger ***
       *** ONLY ONE tsm2500 charger should be powered up at this time     ***
       ***

      Proceed [Y/N] ?
```

If you type "y", the EVCC then prints

```
      Programming the charger ...
```

and then 5-10 seconds later it prints

```
      Programming the charger ... done.
      The charger must now be power cycled.
      evcc>
```

At that point the new charger will be programmed to tsm2500_42 and it will be configured in the evcc as "charger2".

Integration with CAN Enabled BMS

The EVCC can be used with a CAN enabled Battery Management System. The following functions are supported:

- **High Voltage Cutoff (HVC) Detection**. In this case, the BMS detects that at least one cell has exceeded its programmed High Voltage Cutoff limit. If this occurs, the BMS sends a message to the EVCC which causes the EVCC to stop charging.
- **Balance Voltage Cutoff (BVC) Detection**. In this case, the BMS detects that at least one cell has exceeded its programmed Balance Cutoff limit. If this occurs, the BMS sends a message to the EVCC that it should reduce its charging current to the maximum balancing current (**maxbc**). Lowering the charging current allows current cell balancers to prevent additional charging of the highest cells in the pack.
- **Low Voltage Cutoff (LVC) Detection**. In this case, the BMS detects that at least one cell voltage is less than its programmed Low Voltage Cutoff limit. If this occurs, the BMS sends a message to the EVCC which causes the EVCC to operate the buzzer.
- **Thermistor Overtemp Detection**. In this case, the BMS detects that at least one thermistor is overtemperature. The EVCC will stop charging.

BMS Operation

The programming of the actual HVC, BVC, and LVC are done in the BMS. The BMS must determine if any cell in the pack meets these conditions and if so, it sets a bit associated with each of these conditions. This information is sent in a message from the BMS to the EVCC; the message must be periodically sent at least once a second.

```
/*
 * The EVCC supports 250kbps CAN data rate and 29 bit identifiers
 */

#define uint8     unsigned char
/*
 * BMS->EVCC message Identifier
 */
#define BMS_EVCC_STATUS_IND         0x01dd0001

/* bBMSStatusFlag bits */
#define BMS_STATUS_CELL_HVC_FLAG          0x01  /* set if a cell is > HVC */
#define BMS_STATUS_CELL_LVC_FLAG          0x02  /* set if a cell is < LVC */
#define BMS_STATUS_CELL_BVC_FLAG          0x04  /* set if a cell is > BVC */

/* bBMSFault bits */
#define BMS_FAULT_OVERTEMP_FLAG           0x04  /* set for thermistor overtemp */

/*
 * BMS->EVCC message body
 */
typedef struct tBMS_EVCC_StatusInd
  {
    uint8 bBMSStatusFlags;   /* see bit definitions above */
    uint8 bBmsId;            /* reserved, set to 0        */
    uint8 bBMSFault;
    uint8 bReserved2;        /* reserved, set to 0        */
    uint8 bReserved3;        /* reserved, set to 0        */
} tBMS_EVCC_StatusInd;
```

Note that the CAN message only has a 5 byte message body, up to 8 bytes may be sent to the EVCC. These bytes should be set to 0 if so. The EVCC will ignore additional message bytes.

EVCC Operation

In order to use the CAN interface with the BMS, it must be configured in the EVCC, using the **set bms** command. It is possible to configure the EVCC to only use **loop**, only use **can**, or use both **loop** and **can**.

If the EVCC is configured to only use **loop**, then if the loop circuit is closed then the pack is error free; if the loop circuit is open, HVC is assumed if in CHARGE state and LVC is assumed if in DRIVE state.

If the EVCC is configured to use only **can**, then pack status is taken from the `BMS_EVCC_STATUS_IND` message. Note that the message also supports the BVC condition (which the loop does not). If that is reported, then the EVCC will drop back into balance cutback. If there is a message timeout and `BMS_EVCC_STATUS_IND` does not arrive, then this is treated as a pack error (e.g., HVC and LVC are assumed).

If both **loop** and **can** are configured then an error results if either input reports an error. So, in this case, charging will stop if the loop opens, the CAN message indicates HVC, or there is a CAN message timeout.

Charging Lead Acid Batteries

Lead Acid Batteries require a multi-stage charging algorithm. The terminology to describe the algorithms varies in the industry and between manufacturers. Here we follow the documentation and requirements from Trojan. See https://www.trojanbattery.com/pdf/TrojanBattery_UsersGuide.pdf.

As an example consider a EV pack that consists of 12 Trojan 30XHS deep cycle flooded batteries, charging at 25°C (77°F). See the following from http://www.trojanbattery.com/pdf/TRJN0111_ProdSpecGuide.pdf. For reference, the C_{20} rating of 30XHS batteries is 130AH (this number comes in handy below).

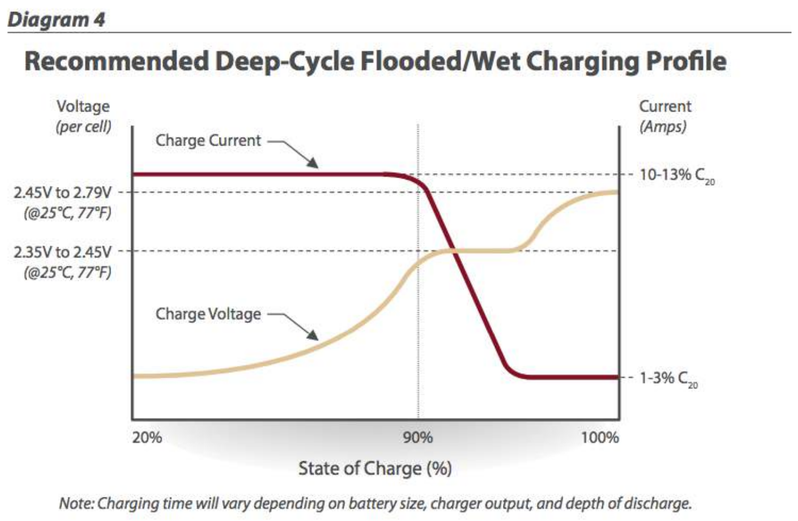


Figure 11 – Flooded Lead Acid Charging Profile (Trojan)

Bulk Charge

The first phase of charging is the Bulk Charge phase. (Note that the Bulk Charge phase is sometimes thought of as two phases: a constant current phase and a constant voltage phase). The EVCC supports this phase by the parameters **maxv** and **maxc**. This phase is used by both Lithium and Lead Acid chemistries (including flooded, AGM, and Gel).

See Figure 10, above. For flooded cells, the Bulk Charge phase brings the cells to over 90% state of charge. For its cells, Trojan recommends a maximum voltage of 2.35 to 2.45v per cell, and a current of 10-13% C_{20}. The bulk charge phase completes when the charging current drops to 1-3% of C_{20}.

In the example of twelve 30XHS cells, here are suggested EVCC settings:

- **maxv=172.8**: The charging voltage would be 2.4v * 6 cells * 12 batteries = 172.8v.
- **maxc=13**: Since 30XHS cells have a C_{20} rating of 130AH, the charging current would be 13A.
- **termc=2.6**: The guidelines are 1-3% of C_{20}. 2.6A is 2% of the C_{20} rating of 30XHS.
- **termt=480**: (10 hours). This parameter is a failsafe; the actual time of charge will depend on depth of discharge. In 10 hours, this would allow 13A*10H =130 AH to be delivered to the batteries.

Finishing Charge

For Lead Acid batteries, the second phase of charging is the "finishing charge" or "absorption charge" phase. The EVCC will only enter the finishing charge phase if the bulk charging phase completes successfully, if termc is reached. (In particular, if the bulk charge phase terminates because of a charging timeout [termt], then this is considered an abnormal termination).

For its cells, Trojan recommends a maximum voltage of 2.45 to 2.79v per cell, and a current limit of 1-3% of C_{20}. This phase completes when the charging voltage rises to the target finishing voltage.

In the example of twelve 30XHS cells, here are suggested EVCC settings:

- **fin_maxv=187.2**: The finishing voltage would be 2.6v * 6 cells * 12 batteries = 187.2v.
- **fin_maxc=2.6**: Note that this is the same as the termc setting above.
- **fin_termt=480**: (2 hours). fin_termt may be set to "0" (or "forever") but if it is set to a finite time, is a failsafe on this charging phase.

Float Charge

Once Lead Acid batteries are charged, they may be kept on a "float charge" or "trickle charge". Lead Acid batteries have a relatively high self-discharge rate and this phase keeps them topped up if the EV sits for an extended period of nonuse.

For its cells, Trojan recommends a float voltage of 2.2v per cell. A current limit is not explicitly specified.

In the example of twelve 30XHS cells, here are suggested EVCC settings:

- **flt_maxv=158.4**: The float voltage would be 2.2v * 6 cells * 12 batteries = 158.4v.
- **flt_maxc=2.6**: Note that this is the same as the termc setting, above.
- **flt_termt=0**: No timeout

Limitations

The EVCC does not support "equalization charge". This type of charging purposely overchargers the batteries in order to balance the cells. Higher charge cells bubble off excess charge as hydrogen gas, and lower charged cells "catch up".

Temperature sensors are not supported in the EVCC, so the EVCC does not perform temperature compensated charging. The examples assumes charging at a constant 25°C in a well ventilated area.

> DISCLAIMER: This is an example only. These instructions do not cover all details or variations in the equipment and do not claim to provide for every possible contingency met in connection with installation, operation, or maintenance. It is strongly recommended that the user check with their battery supplier to determine appropriate charging parameters.

Differences between EVCC 2.4 and EVCC 3.0

EVCC-basic Changes

The EVCC-basic is new for EVCC 3.0. EVCC-basic uses the STANDBY state and does not use the DRIVE state.

Mappable Outputs

The EVCC 2.4 had three fixed outputs:

• Ext_Ind	- 12V, 500ma output.	Tracked the EVCC LED
• PackFault	- 12V, 500ma output.	Enabled if Cell Loop was open or BMS HVC or LVC
• OKToDrive	- Open collector, 500ma.	Grounded when J1772 plug is not connected.

The EVCC3.0 has three mappable outputs:

• OUT1	- 12V, 500ma output.	Default is LED (same as EXT_IND)
• OUT2	- Open collector, 200ma.	Default is J1772_DISCONNECTED (same as OKToDrive)
• OUT3	- Open collector, 500ma.	Default is BUZZER (same as PackFault)

Command Line Changes

The EVCC 2.4 syntax uses "**set**" and "**reset**" to set the options (**extindcharge**, **canterm**, **topbalance**, **loopground**).

The EVCC 3.0 syntax uses "**enable**" and "**disable**" to set the options (**canterm**, **topbalance**, **loopground**). The option extindcharge has been removed from EVCC 3.0 as it can be enabled using the mappable outputs.

The EVCC 2.4 syntax to set the CAN BMS is "**set bms [can | can2 | can3 | can4]**".
The EVCC 3.0 syntax to set the CAN BMS is "**set bms [bmsc | bmsc2 | bmsc3 | bmsc4]**".

Warrantee and Support

The Thunderstruck return policy is available at http://www.thunderstruck-ev.com/return-policy.html.

The EV Charger Controller is warranted to be free from defects in components and workmanship under normal use and service for a period of 1 year.

When failing to perform as specified during the warranty period we will undertake to repair, or at our option, replace this product at no charge to its owner, provided the unit is returned undamaged and shipping prepaid, to Thunderstruck motors.

The product is intended for non-commercial use by hobbyists. The warranty does not apply to defects arising from miswiring, abuse or negligence, accidents, opening the enclosure, or reverse engineering. Thunderstruck Motors and Dilithium Design shall not be responsible for any incidental or consequential damages.

Thunderstruck Motors and Dilithium Design reserve the right to make changes or improvements in design or manufacturing without assuming any obligation to change or improve products previously manufactured and / or sold.

For general support and warrantee issues, contact
connect@thunderstruck-ev.com

For errors in this document, or comments about the product, contact
djmdilithium@gmail.com

Document History

Rev 3.0	Aug 2018	- Initial Version
Rev 3.0.1	Sept 2018	- Minor updates
Rev 3.0.2	Jan 2019	- clarify operation of set <outn>