Thunderstruck Motors
EV Charger Controller v2.3

Contents

Figures

Overview

The Electric Vehicle Charger Controller (EVCC) integrates charger CANBUS control and J1772 functions in a simple to use, cost effective, and environmentally robust enclosure. Charge parameters such as maximum voltage, maximum current, and total charge time are configured, saved in nonvolatile memory, and used when charging to control a CAN enabled charger. The EVCC connects to analog “cell loop” Battery Management Systems (BMSs) as well as CAN enabled BMSs.

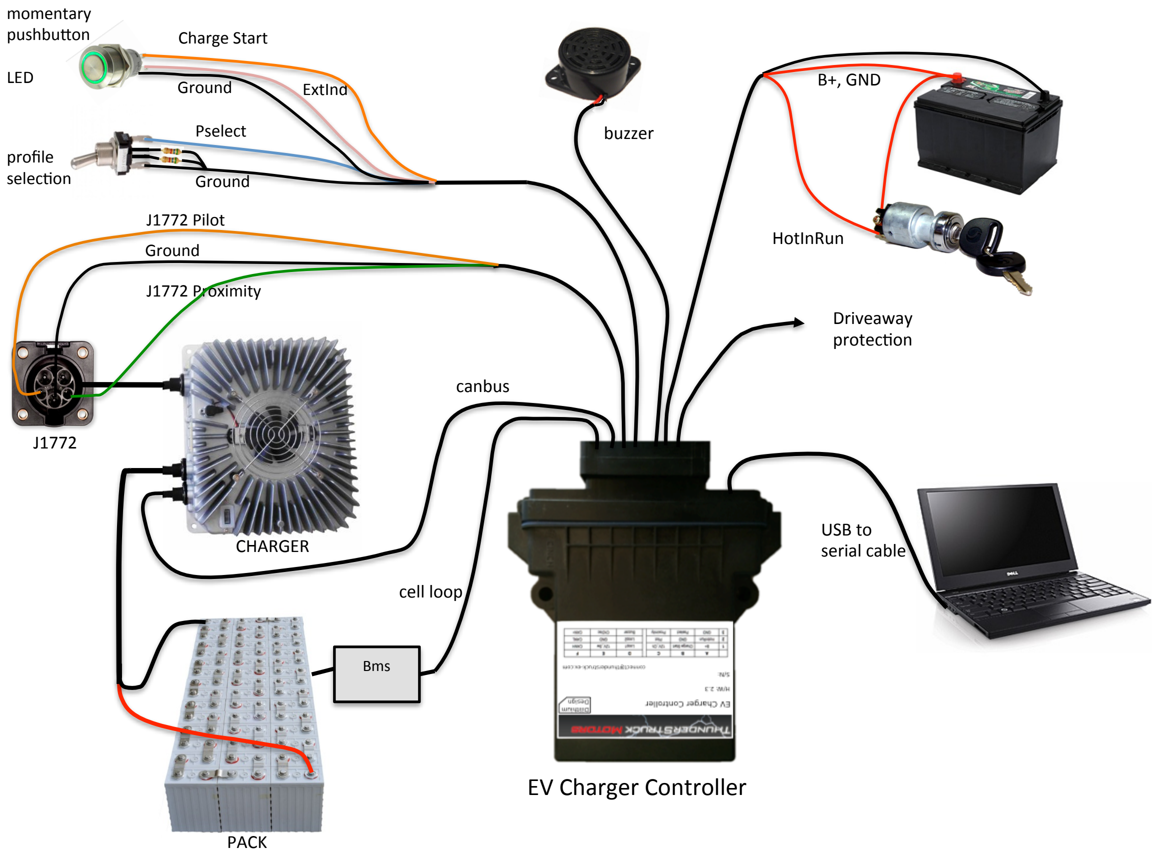


Figure 1 – EVCC System Diagram

The EVCC draws negligible current (less than 0.1 mA) when off. When charging, the EVCC is started by a momentary pushbutton and turns itself off when the charge cycle is completed.

The EVCC is configured using a simple serial interface. This interface is used for configuration and debugging, but is not required for normal operation. Diagnostic commands are supported to verify proper wiring, to trace CANBUS messages, and to retrieve charging history.

The EVCC supports the SAE J1772 standard. J1772 defines the physical connector and protocols used between the charging station (known as the “Electric Vehicle Service Equipment”), and the Electric Vehicle. The J1772 Proximity signal is used to determine if the charger plug is present. The J1772 Pilot signal is used to start and stop charging (by enabling and disabling the contactor in the EVSE). The J1772 Pilot duty cycle is measured and can be used to limit charging current. “Driveaway protection” is supported so that the EV cannot be driven if the charge cable is still plugged in.

The EVCC supports TSM2500 and ELCON CAN-enabled chargers. Charge voltage, charge current and overall charge time are controlled completely by the EVCC over 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 will stop charging if the J1772 plug becomes unplugged, a cell overvoltage error occurs, there is loss of communication between the EVCC and Charger, or the maximum configured charge time is reached. Charging also stops at the end of a normal charge cycle, which, for Lithium batteries, occurs when the charging current drops below the minimum configured charge current.

Determining cell overvoltage errors and cell undervoltage error detection is the function of an EV 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 handle “balance cutoff”, which lowers the charging current when a cell exceeds a “balancing threshold”.

Charging history is provided for the last sixteen charge cycles and includes: the reason that charging stopped, total charge time, maximum voltage, maximum current, final current, and watt hours delivered.

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

When driving, the EVCC is started by the keyswitch, and can be used as a simple “BMS Master” - it can sound a buzzer when a cell undervoltage error is detected.

EVCC features work largely independently and it is not necessary to wire up or use all features. Installation may be customized per customer requirements.

The EVCC is housed in a 4.55” x 5.13” x 1.67” automotive grade water-resistant enclosure. Connections are made with a single 18 pin connector. The EVCC is shipped with a pre-wired harness and with a USB to serial port cable.

Installation and Theory of Operation

Mechanical

The enclosure outline is shown below. The EVCC can be mounted in any convenient location, however would ideally be located physically close to both the charger and the J1772 charge port.

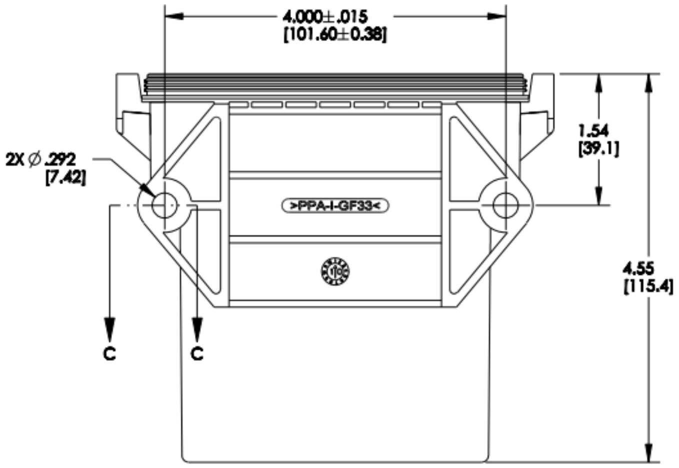
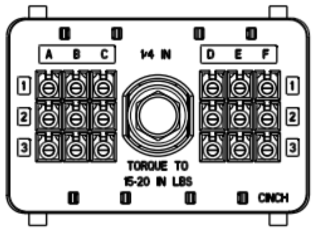


Figure 2 – EVCC Enclosure

The figure below shows the 18 pin connector and wiring harness. Note the LED to the right of the connector and the serial port jack to the left of the connector.



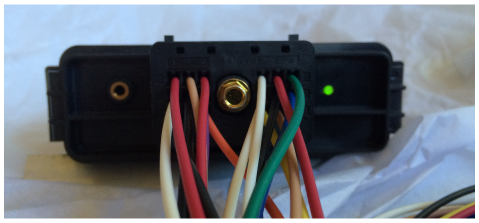


Figure 3 – EVCC Connector and Front Panel

The figure below shows the EVCC pinout.

	A	B	C		D	E	F
1	B+	ChargeStart	ExtInd		CellLoop1	12V_Sw	CANH
2	HotInRun	GND	Pilot		CellLoop2	GND	CANL
3	GND	PSelect	Proximity		Buzzer	EVSEDisc	reserved

Figure 4 – EVCC Pinout

Power

B+ and **GND** (A1, A3) are Power Inputs and should be connected to the EV 12V accessory battery.

HotInRun (A2) is connected to the Ignition swich. Supplying +12V to **HotInRun** will turn the EVCC on.

ChargeStart (B1) is used to start charging. By momentarily grounding this input (e.g., by a pushbutton switch), the EVCC will power up and latch power on. The EVCC automatically turns itself off when charging is complete.

12V_Sw (E1) is a switched output that can provide up to 200ma of current to downstream equipment when the EVCC is on.

ExtInd (C1) is a 12V output that can drive an external indicator light or a relay. By default, this output tracks the EVCC LED. However, this output may be configured to be ON whenever a charge cycle is in progress (see "**set extindcharge**"). The ExtInd output is protected by a 200ma resettable fuse.

The figure below shows the Power connections.

	A	B	C	D	E	F
1	B+	ChargeStart	ExtInd	CellLoop1	12V_Sw	CANH
2	HotInRun	GND	Pilot	CellLoop2	GND	CANL
3	GND	PSelect	Proximity	Buzzer	EVSEDisc	reserved

Figure 5 - Power Connections

The **ChargeStart** and **ExtInd** connections are intended to be connected to a momentary pushbutton and integrated 12V LED indicator near the J1772 charge port. Charging is begun by plugging in the charger plug, pushing the button, and observing the light come on.

One implementation is to use a readily available pushbutton with integral 12v indicator. In this example, the switch has five terminals. To the left and right are the connections for the indicator light (left is +, right is -). And from top to bottom are the switch contacts: NC, NO, C. Wire GND (black) to Light- and C. Wire ChargeStart (TAN) to NO. And Wire ExtInd (PINK) to Light+.

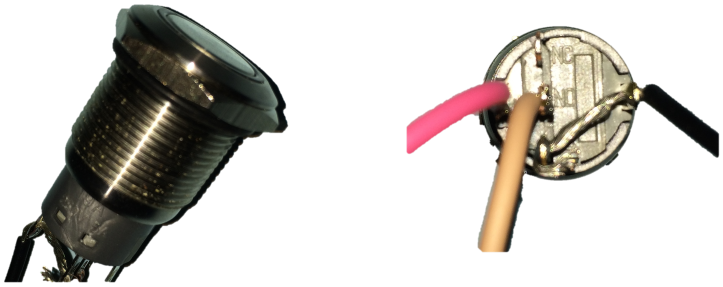


Figure 6 - Charge Start and ExtInd Wiring

J1772

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.

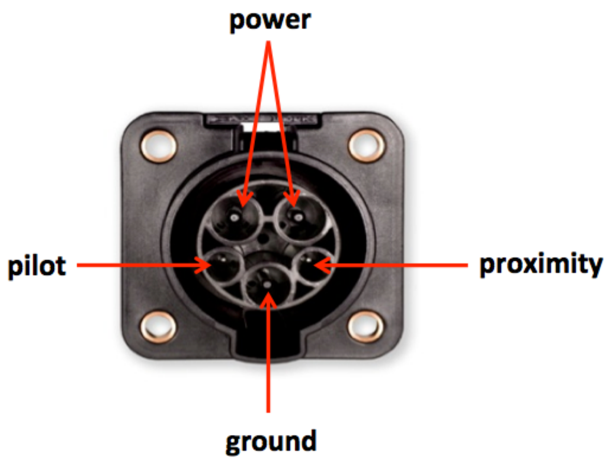


Figure 7 – Face of J1772 Socket

The **Proximity** signal (C3) allows the EV and the EVSE to determine whether the J1772 charge plug is "disconnected", "connected" or "locked". When the J1772 charge plug is fully inserted, it is "locked". When the charger release button is pressed (by thumb on the charger plug), the charge plug becomes "unlocked", or simply "connected". Should the plug become "unlocked" while charging, charging will immediately stop.

The **Pilot** signal (C2) allows the EVSE to advertise how much power is available for charging, and it allows the EV to control when to start and stop 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 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.

The figure below shows the J1772 connections.

	A	B	C	D	E	F
1	B+	ChargeStart	ExtInd	CellLoop1	12V_Sw	CANH
2	HotInRun	GND	Pilot	CellLoop2	GND	CANL
3	GND	PSelect	Proximity	Buzzer	EVSEDisc	reserved

Figure 8 – J1772 Connections

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

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

The following schematic shows the connections required for J1772.

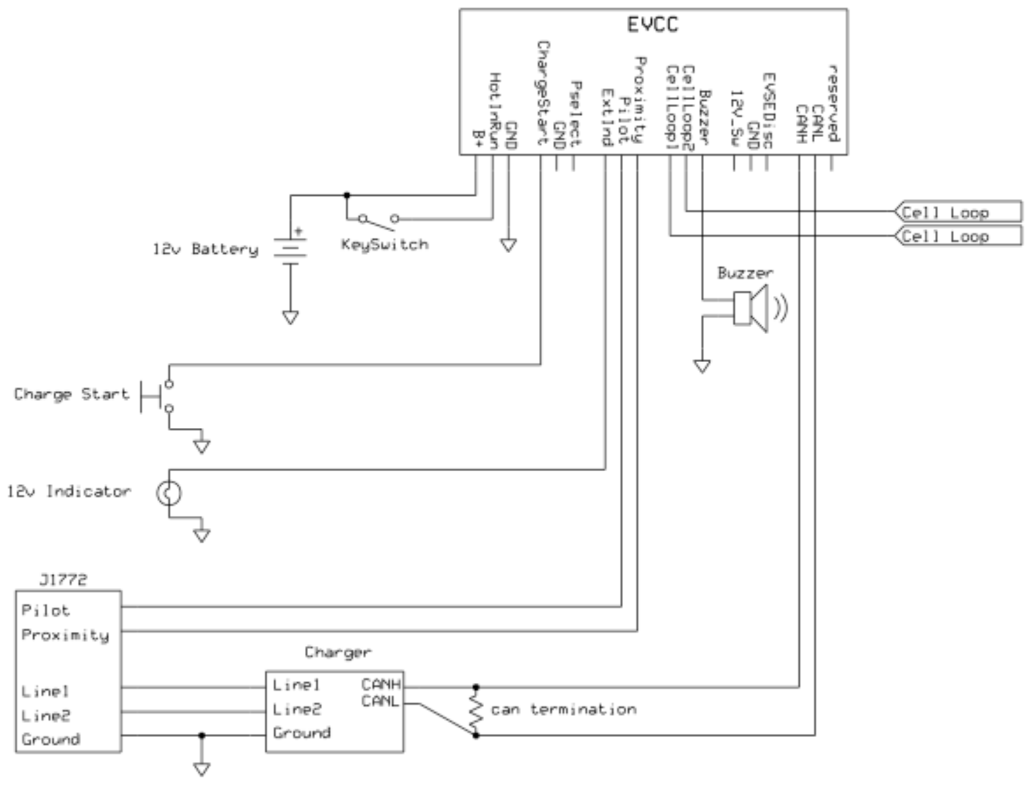


Figure 9 - Wiring With J1772

Wiring Without J1772

Although J1772 is recommended, its use is optional. When using J1772, the EVCC **Proximity** signal is connected to ground through a 150 ohm resistor built into the J1772 charge plug. This indicates that the charge plug has been plugged in and is "locked". When J1772 is not being used, the EVCC **Proximity** must either be connected directly (or through a switch) to GND to allow charging.

Here are two wiring options that do not use J1772:

Option 1 retains most EVCC functionality.

- Wire **Proximity** to **GND** through a switch (a "charger present" switch). This is an additional, latching, SPST switch such as a toggle switch. To use this feature, plug in the charger, close the "charger present" switch, and press **ChargeStart**. Charging operates as designed and the EVCC turns itself off when complete. The EVCC Drive mode operates as designed (**HotInRun** enables the EVCC, the cell loop operates the buzzer). If driveaway protection is implemented, the "charger present" switch must be turned OFF in order to operate the EV.

Option 2 is used when the EVCC is only used for charging.

- Wire **Proximity** directly to **GND**. Do not wire **Charge Start**. To charge, plug in the charger, and apply 12V to **HotInRun** and **B+**. The EVCC will power up and begin charging. When the EVCC completes charging, it will stop sending CAN messages to the charger, but will remain powered ON until power is removed from **HotInRun**. To start charging again, it is necessary to cycle power to the EVCC.

The following schematic shows how to wire up the EVCC without J1772 (option 1):

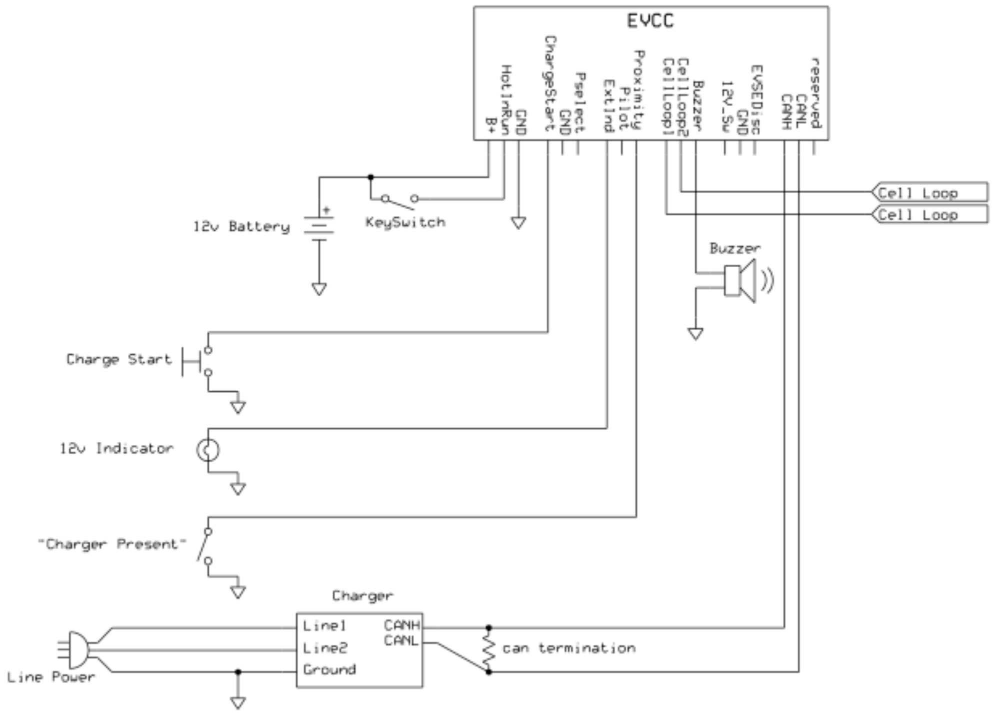


Figure 10 - Wiring Without J1772 (With Charger Present Switch)

The following schematic shows how to wire up the EVCC without J1772 (option 2):

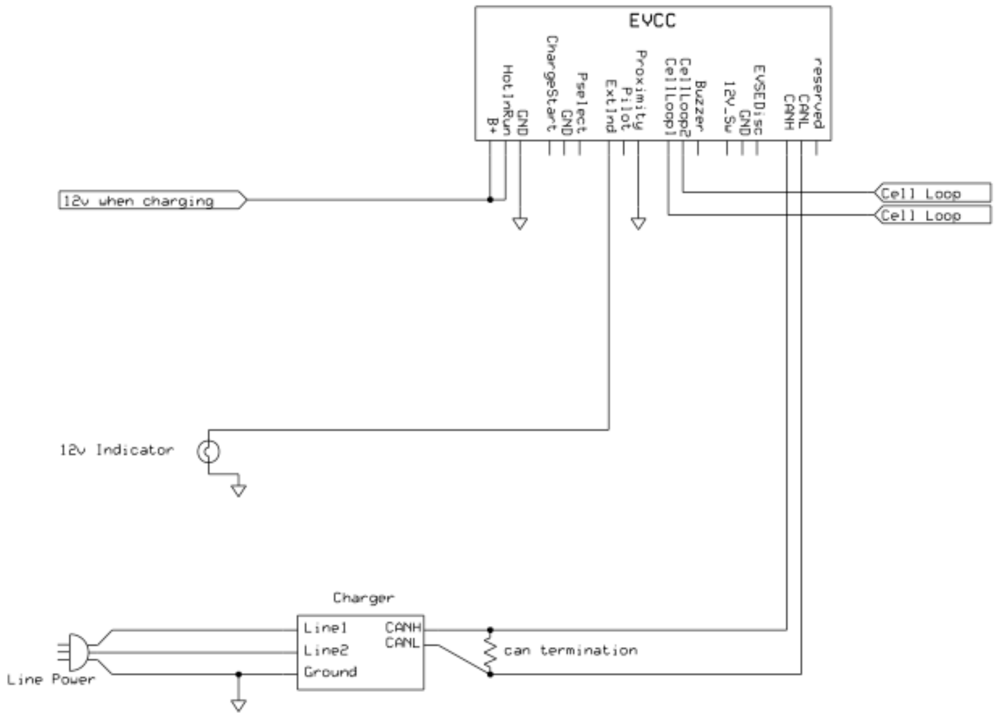


Figure 11 - Wiring Without J1772 (Charging only)

Cell Loop and Buzzer

The EVCC is intended to be installed with a Battery Management System that monitors per-cell over voltage conditions when charging and per-cell undervoltage when driving.

The EVCC Cell Loop surveillance circuit measures the resistance of the circuit between **CellLoop1** (D1) and **CellLoop2** (D2). The circuit applies +5v to Cell Loop1 and limits the current to about 2ma. It is expected that the Cell Loop be implemented using a solid state relays or optoisolators. Connecting the cell loop to the contacts of a mechanical relay is not recommeneded, as the cell loop may not provide enough "wetting current" for the relay contacts.

The **Buzzer** (D3) output provides up to 200ma at 12V that can be used to sound a buzzer when there is a pack fault error. The operation of the buzzer depends on how cell surveillance is configured in the EVCC. By default, cell surveillance is performed by the cell loop. However, cell surveillance can also be disabled, performed by a CAN BMS, or performed by both the cell loop and the CAN BMS. (See the command **set bms**, below).

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 figure below shows the Cell Loop and Buzzer connections.

	A	B	C	D	E	F
1	B+	ChargeStart	ExtInd	CellLoop1	12V_Sw	CANH
2	HotInRun	GND	Pilot	CellLoop2	GND	CANL
3	GND	PSelect	Proximity	Buzzer	EVSEDisc	reserved

Figure 12 – Cell Loop and Buzzer Connections

Driveaway Protection

Driveaway Protection is a failsafe mechanism that prevents the EV being driven if the charger plug is connected. This feature is implemented by signal **EVSEDisc** (E3). This is an "open collector" signal to ground, fused to 200ma.

The signal is open if the J1772 cable is plugged in (or if the EVCC is not powered). Conversely, the EVSEDisc input is switched to ground if the EVCC is powered up and the cable is not plugged in.

How to disable the EV from driving is up to the discretion EV designer. These contacts could be wired into the control logic of the primary contactor.

Note: The EVSEDisc contacts may not be suitable for directly control of a primary contactor. A typical primary contactor requires 1A or more of holding current which is well above the 200ma fused limit.

The figure below shows the connections used for Driveaway Protection.

	A	B	C	D	E	F
1	B+	ChargeStart	ExtInd	CellLoop1	12V_Sw	CANH
2	HotInRun	GND	Pilot	CellLoop2	GND	CANL
3	GND	PSelect	Proximity	Buzzer	EVSEDisc	reserved

Figure 13 – Driveaway Protection Connections

Profile Selection

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 such as (maximum charging voltage and maximum charging current) associated with that profile.

When charging, a profile is selected using the **Pselect** input (B3). The EVCC measures the resistance to GND at this input and deduces four possible selections: "inf", 20K, 5K, and "0". If the PSelect input is left unconnected, it will read "open" (or "infinite" resistance) to ground, and would map to "inf". If the Pselect input is shorted to ground it will measure "0". Finally, if a resistor is connected between the Pselect 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.

In operation, these four inputs represent a "switch setting" and not a "profile number". The mapping from setting to profile number is done using the profile map. (See the command "**set map**"). So, for example, "inf" might be mapped to "Profile 1", and "0" might be mapped to "Profile 2".

	A	B	C	D	E	F
1	B+	ChargeStart	ExtInd	CellLoop1	12V_Sw	CANH
2	HotInRun	GND	Pilot	CellLoop2	GND	CANL
3	GND	PSelect	Proximity	Buzzer	EVSEDisc	reserved

Figure 14 – Profile Selection Connections

CANBUS

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, multistation network that must be terminated on both ends of the string by 120ohm termination resistors. See below for a simple network diagram.

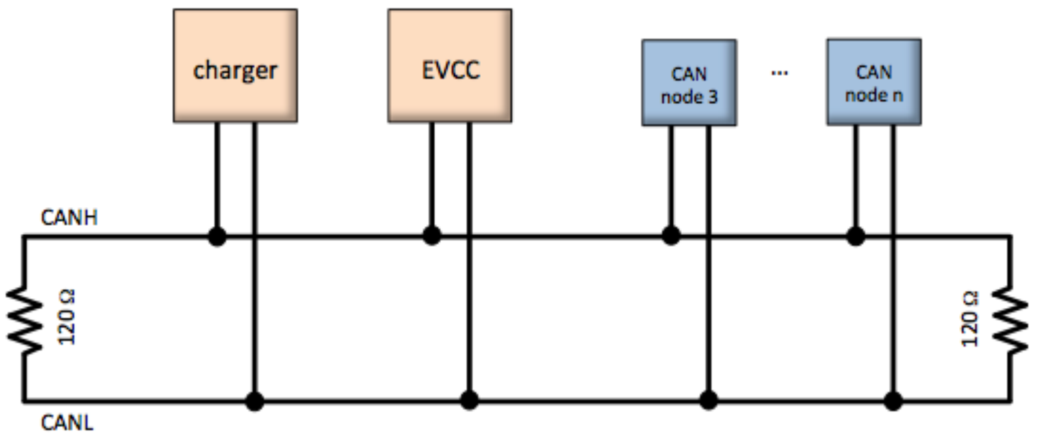


Figure 15 – CAN Network Diagram

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 (ElectroMagnetic 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 wiring should work fine.

For longer runs, more nodes, or cases where EMI may be an issue, shielded cable may be needed. If a shielded cable is used, the shield should be connected to chassis ground at a single place.

The figure below shows the connections used for CAN.

	A	B	C	D	E	F
1	B+	ChargeStart	ExtInd	CellLoop1	12V_Sw	CANH
2	HotInRun	GND	Pilot	CellLoop2	GND	CANL
3	GND	PSelect	Proximity	Buzzer	EVSEDisc	reserved

Figure 16 – CAN Connections

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 "**set cantermdis**").

When installing the CAN network, keep in mind that 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 will cover the vast majority of installation scenarios. However, there is one case where this may not be acceptable, namely when (1) the EVCC is a terminal node and (2) the CAN network is active by other devices and the EVCC is powered down. In this (rare) case, the recommendation is to disable the termination resistor in the EVCC and connect a physical resistor instead.

CAN Protocol

The EVCC supports a single CAN interface, which runs at 250Kbs and uses 29-bit Extended Identifiers. These parameters are not software configurable, however, both the TSM2500 and ELCON chargers require this rate.

The EVCC uses two types of messages to control a CAN enabled charger. The first, from EVCC to Charger, provides the Charger with the desired values of charge voltage and charge current, and the second message, from Charger to EVCC that reports the actual Charging Voltage and Current. This message may also report additional charger status.

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

EVCC/BMS CAN messages communicate pack status such as Cell Undervoltage or Cell Overvoltage conditions. These messages are used as an alternate to (or conjunction with) the cell loop to indicate pack fault conditions. See

Integration with CAN Enabled BMS, below, for supported message definitions.

CAN Debugging

CAN messages may be lost or corrupted as the result of EMI, stubs that are too long, or improperly terminated cables. 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.

In order to verify correct operation there are both high level tracing (“**trace charger**”) and a low level tracing (“**trace can**”) facilities to show CAN message traffic.

LED Operation

The EVCC LED has the following blink patterns:

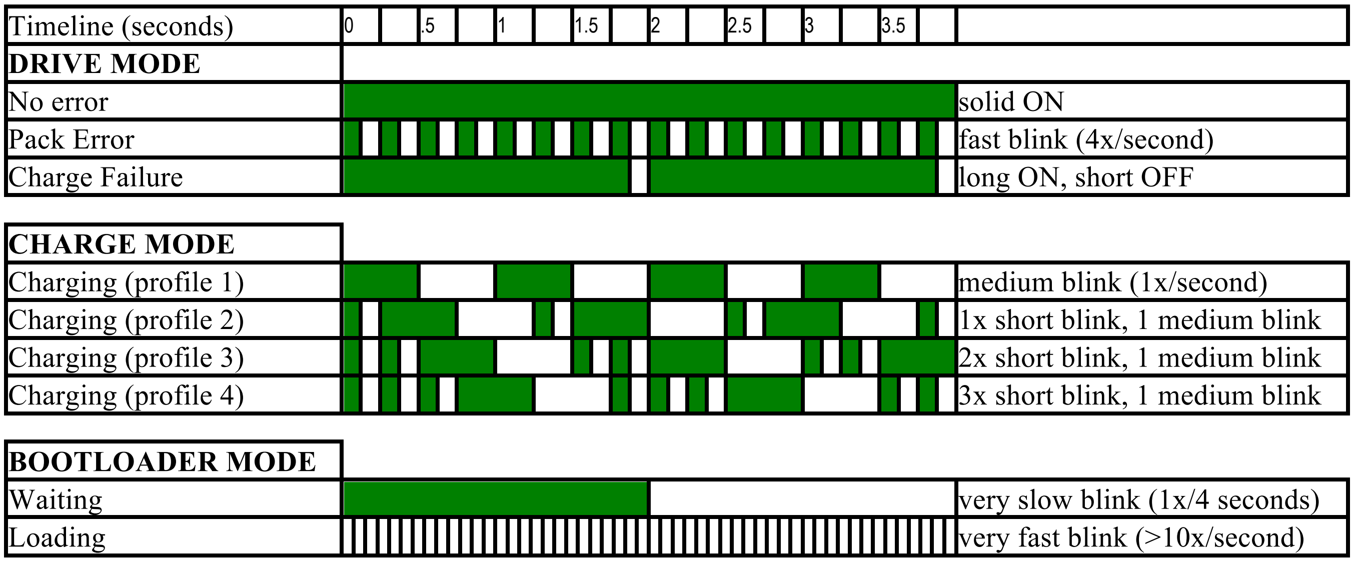


Figure 17 – EVCC Blink Patterns

Charging

This section describes the charging process in more detail.

Determining Charge Voltage and Charge Current

The maximum charge voltage is specified by the EVCC parameter “**maxv**”. This parameter must be configured by the user.

The maximum charge current can be determined by several methods, which depend on how much information that the user configures. These are described in more detail below.

Using maxc only: The user can specify the maximum charge current explicitly (**maxc**) and set no other parameters. In this case, the EVCC will use **maxc** for the charge current.

ChargeCurrent = **maxc**

Set linev and linec, do not configure maxc: In this case, the user specifies available line voltage and line current from the service connection and does not configure maxc. This approach will perform a power calculation as a convenience for the user. When the EVCC computes the maximum power available from the service it derates it 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 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 user specifies available line current to be "J1772". In this case, the EVCC measures the J1772 Pilot signal duty cycle in percent (J1772DutyCycle) and converts it to available charge current (per J1772 this is given as 6A of charge current per 10% duty cycle). Line voltage is not known in this case, and so 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 available line voltage is 110V.

```
ChargeCurrent  = (220 * (J1772DutyCycle * 6) * .9) / maxv      ; if J1772DutyCycle > 25
               = (110 * (J1772DutyCycle * 6) * .9) / maxv      ; if J1772DutyCycle <= 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 the line voltage is now known.

ChargeCurrent = (**linev** * (J1772DutyCycle * 6) * .9) / **maxv**

As a final comment, note that the power capability of the charger is not configured in the EVCC. In practice, if the requested power is more than what the charger can deliver, charger firmware will reduce the delivered current in order to stay within its power limits.

Examples

Let's 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**. In this case, the EVCC will compute the charge current. (Alternately the user could define **maxv** and **maxc** directly, or could use profile 3, below).
2) Secondary EVSE. Define **maxv, linev=110, linec=15**
3) Opportunity Charging. Define **maxv, linec=J1772**.

Determining Charge Parameters for Sealed Lead Acid Batteries

The EVCC supports a three phase charging algorithm to charge SLA batteries. 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".

Configuration

Serial Port

This section describes how to install the serial port drivers and establish serial communications from a host computer and the EVCC. To use the serial cable, a Virtual Comm Port driver (VCP driver) and a terminal application (or "telnet client") is required.

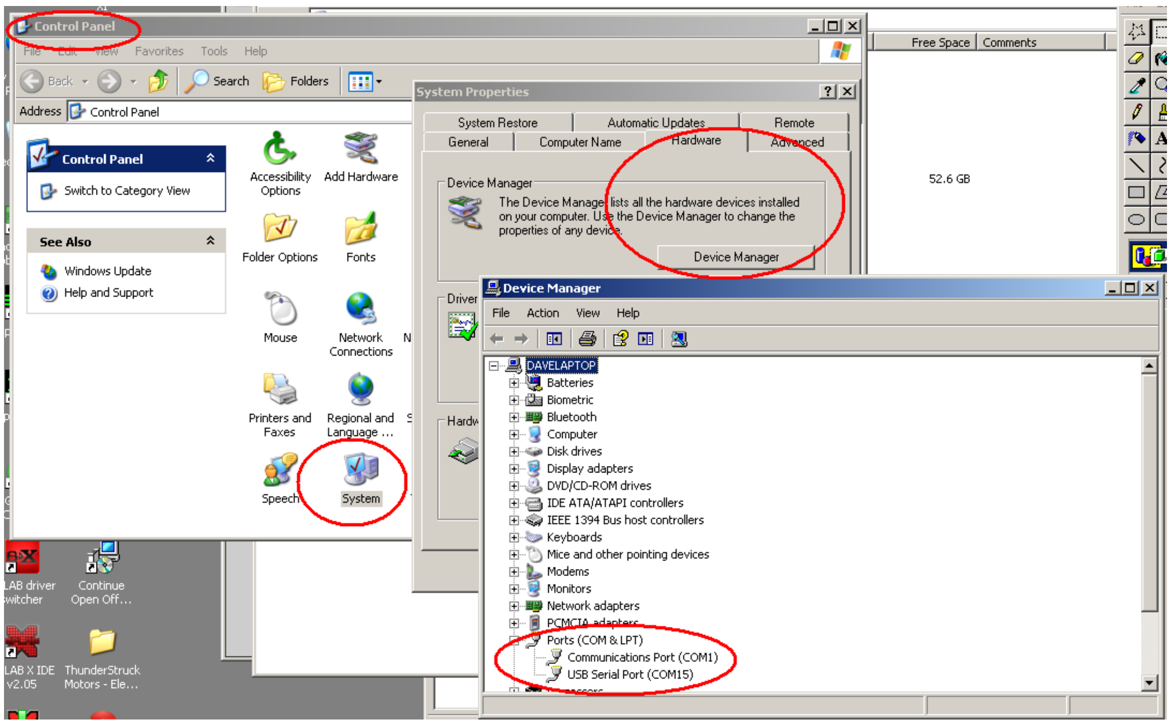
Using a USB to serial bridge is a generic and popular way to connect a host computer to a microcontroller, and the steps are basically the same regardless of the host computer and operating system. Installation instructions are given below for Windows XP. See Mac OSX Support, below, for instructions on how to enable the serial port on a MAC OSX machine. Note that there are good tutorials on how to install the necessary drivers and application software available on the Internet (for other versions of Windows, MAC, Linux, etc). (Search for "ftdi installation", "putty installation", etc).

Step 1: Install the Virtual Comm Port (VCP) driver on the host computer. The VCP driver is software on the host computer that emulates a serial port "on top of" a USB connection.

- VCP drivers are available at: www.ftdichip.com/Drivers/VCP.htm.
- Installation documentation is available at www.ftdichip.com/Support/Documents/InstallGuides.htm.

Step 2: Plug in the USB to serial port cable. If the drivers are correctly installed, the host computer will recognize the new virtual serial port device.; to use this device, is necessary to determine the virtual serial port device name.

- The virtual serial port device name is of the form "COM<n>", where n is a small number. This number can be determined by looking at "Control Panel -> System -> Device Manager -> Ports". In the example below, it is "COM15".



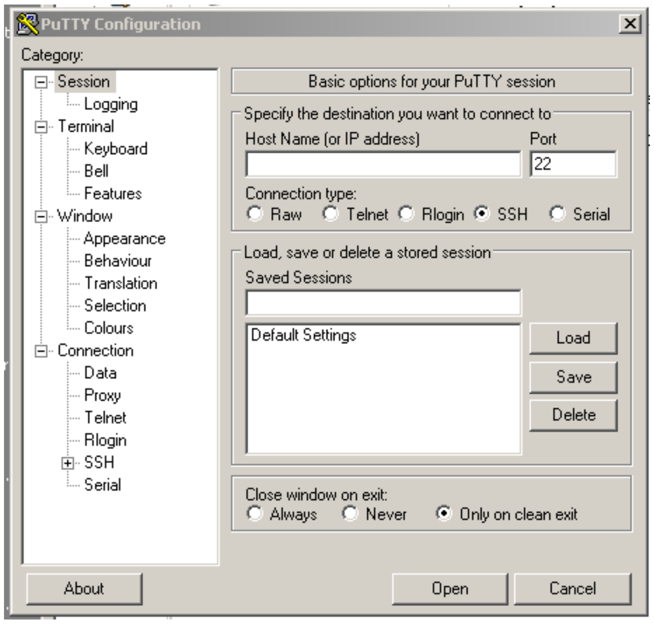
The Device Manager lists all the hardware devices installed on your computer. Use the Device Manager to change the properties of any device.

Step 3: Install a terminal console program (e.g., a "telnet client") on the host computer.

There are many suitable telnet clients that may be used. For Windows (and linux), one popular choice is PuTTY, available for download at http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

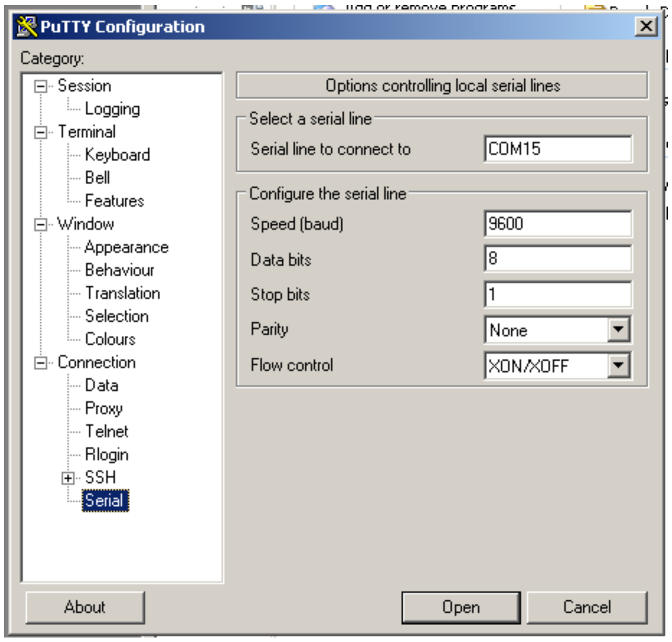
Step 4: Configure the telnet client for use.

The first time PuTTY is opened, it will present the following:



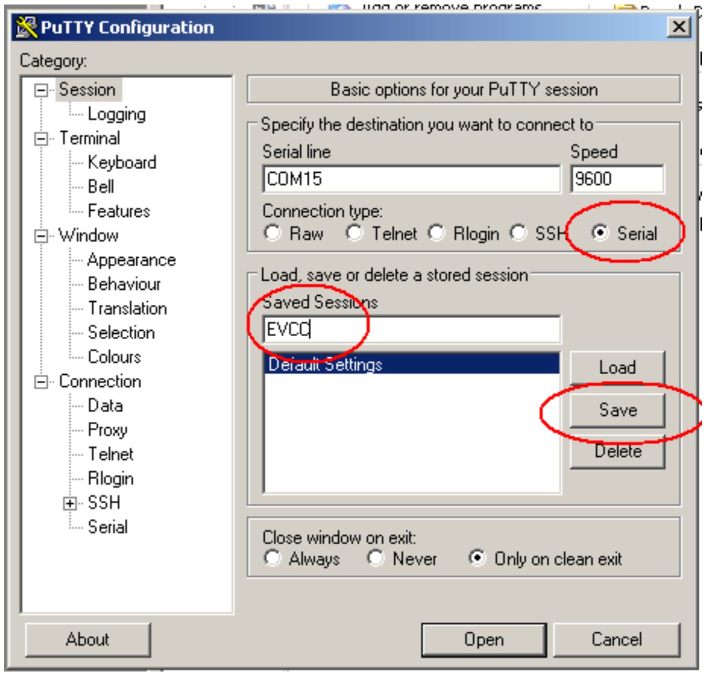
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address):
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions:
Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

Click on "Serial" in the Category column. Verify that the Speed is 9600, 8 data bits, 1 stop bit. Enter the Serial Line to connect to (in this case, "COM15").



Options controlling local serial lines
Select a serial line
Serial line to connect to: COM15
Configure the serial line
Speed (baud): 9600
Data bits: 8
Stop bits: 1
Parity: None
Flow control: XON/XOFF

Do not hit "Open" just yet. Go back to "Session" by clicking the word "Session" in the Category window.



Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM15 Speed: 9600

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Load, save or delete a stored session

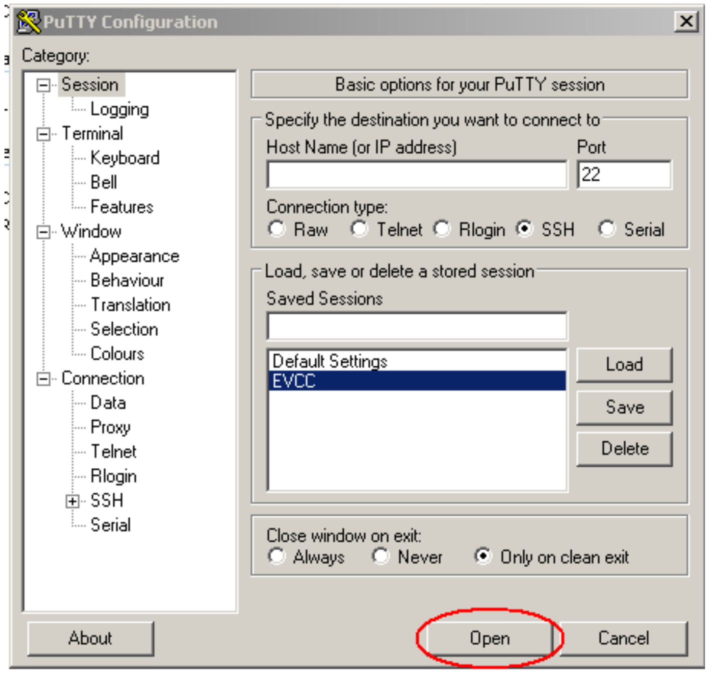
Saved Sessions: EVCC

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

Set the Connection type to Serial. Give the new session a name (in this case “EVCC” in the Saved Sessions window) and press “Save” to save the session. PuTTY is now configured.

Step 4: Open the comm port. Select the saved session “EVCC” and click Open.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

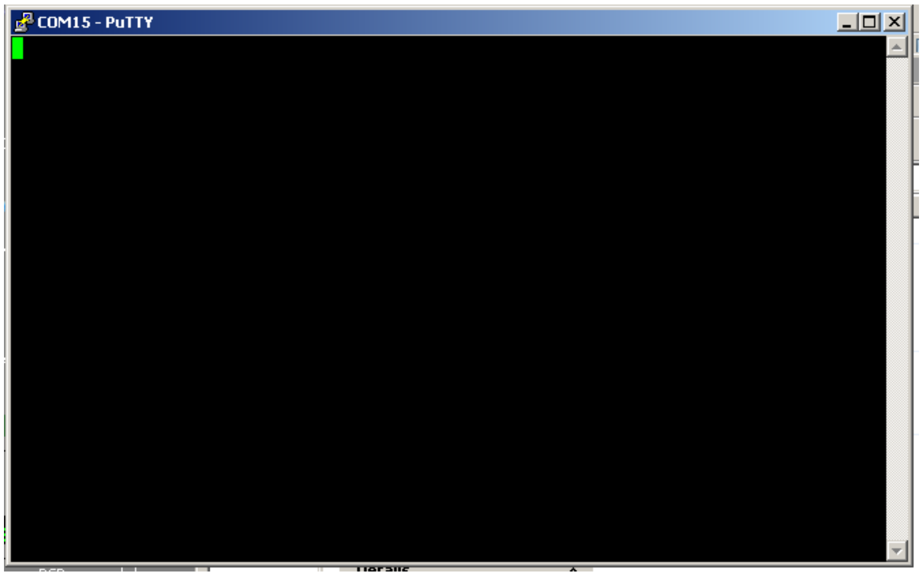
Load, save or delete a stored session

Saved Sessions:

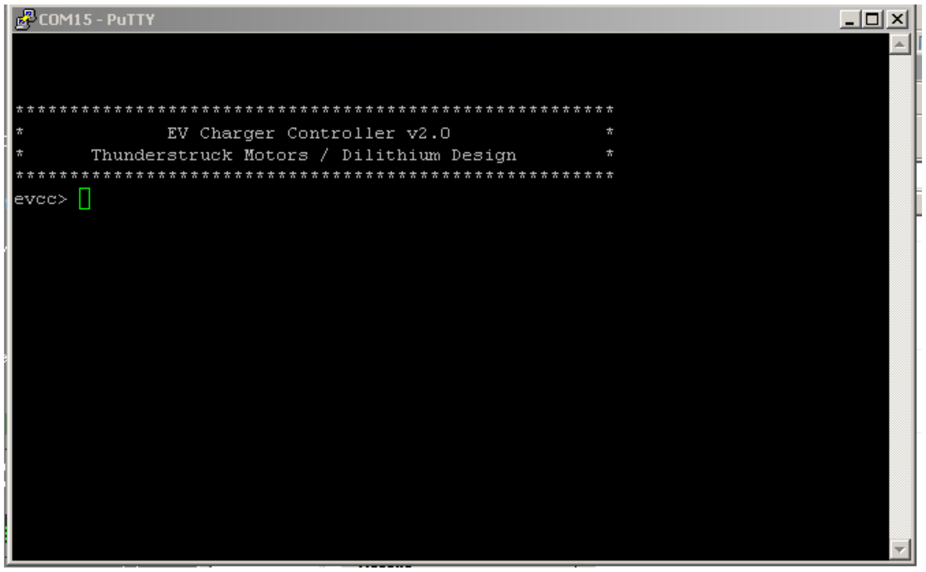
Default Settings
EVCC

Close window on exit: Always Never [✓ Only on clean exit]

A screen like the following should appear:



Step 5: Connect the serial cable to the EVCC. Apply power to the EVCC by providing a 12V supply to **B+** and **GND**. Connect +12V to **HotInRun**. The EVCC **LED** should start blinking (assuming the cell loop has not been hooked up yet), and the following banner should be displayed:



```
*************************************************************
*              EV Charger Controller v2.0                   *
*       Thunderstruck Motors / Dilithium Design             *
*************************************************************
evcc>
```

Step 6: 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 what commands are supported and their syntax.

The EVCC is supplied with defaults, but at the very minimum, it will be necessary to set the Maximum Charging Voltage (using the command "**set maxv**") and Maximum Charging Current (using the command "**set maxc**").

> 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.

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

Charger Support

This section gives details on which charger types are supported by the EVCC. Each charger in an installation requires a unique CAN address. In EVCC terminology a "charger type" refers to both the manufacturer 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. It is configured with a default CAN address however 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 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:

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

These strings would be printed in "trace charger" output.

ELCON

ELCON chargers must programmed with the CAN option. In addition, an external ELCON-provided CAN module is needed that terminates the CAN and communicates to the charger over a serial interface. Only two pins are provided for the CAN connection: CANH and CANL. The ELCON CAN module 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 generate the following errors:

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

These strings would be printed in "trace charger" output.

LEAR

EVCC firmware v2.3.5 and later supports beta code for 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.

Lear charger support has a number of limitations:
.
- 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.
- If a CAN enabled BMS is being used with the EVCC, it must be configured for a 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) Connect B+, GND, HotInRun
2) Connect Proximity, Pilot, GND
3) Connect PSelect to a selector switch (and resistor array), if used

Verify Analog Inputs

1) Type "**measure"** with no parameters to get the expected readings for each analog input. Note that if there is not a good ground connection between J1772 ground and EV chassis ground that the J1772 readings will be erratic.
2) Verify Cell Loop, using "**measure loop**"
   a. Disconnect J1772 plug if connected
   b. Verify readings with cell loop open and closed.
3) Verify Proximity, using "**measure proximity**"
   a. Disconnect cell loop, if connected
   b. Verify readings with charger plug disconnected, connected, and unlocked.
4) Verify Cutback, if used, using "**measure pselect**".

a. Verify readings with different pselect switch settings.

Verify Charge Start and J1772

1) Connect Cell Loop to CellLoop1 and CellLoop2.
2) Plug in J1772 Plug
3) Apply 12V to HotInRun. The EVCC should start charging (LED blinks once per second), and the relay in the EVSE should operate after a short delay.
4) Assuming the CAN bus is not connected to the charger yet, the charge cycle should stop after 10-15 seconds.
5) Remove 12V from HotInRun, the EVCC should lose power (LED goes off).
6) Ground ChargeStart. The EVCC should power up and go into Charge state.
7) For debugging, use "**trace state**" to verify that the EVCC attempts to start charging if the J1772 plug is in and the user powers up the EVCC.

Verify Charger and CAN

1) Connect Charger to J1772, connect CAN between Charger and EVCC.
2) Verify proper installation of the CAN termination resistors. Measure between CANH and CANL to verify that the resistance of 60ohms. Remember that if the internal EVCC termination resistor is used, that it must be powered up in order to make this measurement.
3) Now verify that when a charge cycle is started, that messages are exchanged between EVCC and Charger. (Use "**trace charger**" or "**trace can**" to log the messages).
4) If the pack is not yet connected to the Charger, the charge cycle will stop after a minute.

Systems Test

1) Verify all systems functions.

Firmware Upgrade

The EVCC firmware may be upgraded by the customer using the serial port. Firmware upgrade is supported in EVCC version v2.3.7 and later. EVCC versions earlier than v2.3.7 must be returned for reprogramming.

Bootloader Application

The opensource ds30 PIC bootloader is used, distributed under the GNU GPL license. This bootloader is free and the application runs on Windows, MAC and Linux servers. Documentation, screenshots and downloads are available at:

http://picbootloader.com/

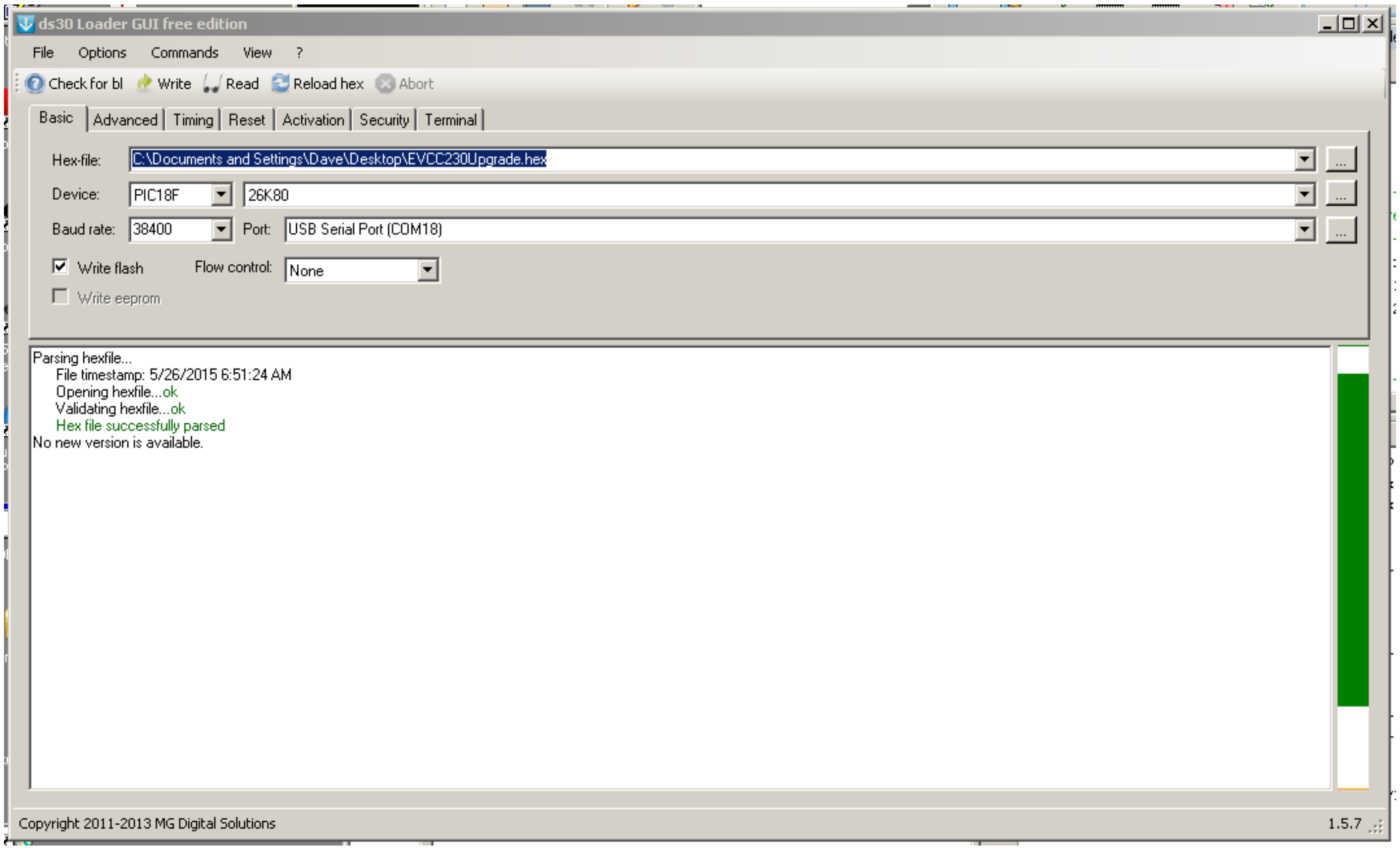
When starting the bootloader utility, set the following options.

Device = **PIC18F26K80**
Baud Rate = **38400**

Use the pulldown on the Port option to find the COM port associated with the USB to serial cable to the EVCC. Select the correct COM port.

Tick the "Write flash" box.

Select the hex file to download. Thunderstruck will provide firmware upgrades by its website as needed. An upgrade file will be named by the convention "evcc231upgrade.hex".



Hex-file: C:\Documents and Settings\Dave\Desktop\EVCC230Upgrade.hex
Device: PIC18F 26K80
Baud rate: 38400 Port: USB Serial Port (COM18)
☑ Write flash Flow control: None
☐ Write eeprom

```
Parsing hexfile...
    File timestamp: 5/26/2015 6:51:24 AM
    Opening hexfile...ok
    Validating hexfile...ok
    Hex file successfully parsed
No new version is available.
```

Copyright 2011-2013 MG Digital Solutions 1.5.7

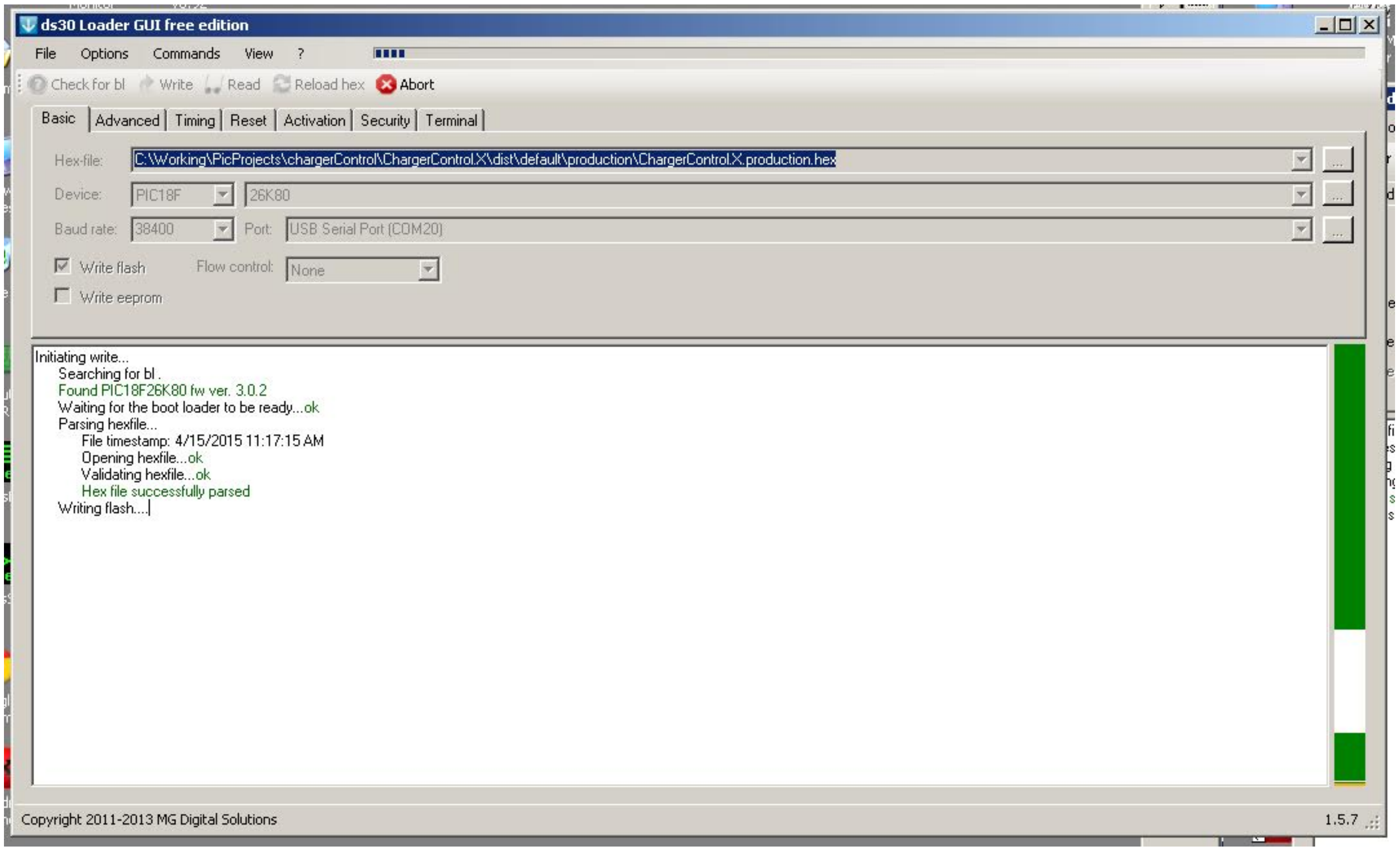
Performing an Upgrade

In order to start the upgrade, must first bring up the EVCC CLI and type:

```
evcc> upgrade
```

The EVCC will the print some informational text and the EVCC will enter the serial bootloader. At this point the EVCC is waiting for an upgrade image. The EVCC LED will blink at a very slow blink (once every 2 seconds). It is then necessary to exit from the the terminal application (e.g., puTTY) and start the ds0bootloader.

Once the correct parameters are set (described above), click on the "Write" tab and the upgrad should start. The ds0bootloader will show a progress bar, and the EVCC LED will blink at a rapid rate to show that the loading has commenced.


ds30 Loader GUI free edition
File Options Commands View ?
Check for bl Write Read Reload hex Abort
Basic Advanced Timing Reset Activation Security Terminal
Copyright 2011-2013 MG Digital Solutions
1.5.7

Hex-file: C:\Working\PicProjects\chargerControl\ChargerControl.X\dist\default\production\ChargerControl.X.production.hex
Device: PIC18F 26K80
Baud rate: 38400 Port: USB Serial Port (COM20)
[✓] Write flash Flow control: None
[ ] Write eeprom

```
Initiating write...
    Searching for bl .
    Found PIC18F26K80 fw ver. 3.0.2
    Waiting for the boot loader to be ready...ok
    Parsing hexfile...
        File timestamp: 4/15/2015 11:17:15 AM
        Opening hexfile...ok
        Validating hexfile...ok
        Hex file successfully parsed
    Writing flash....
```

Once the upgrade has completed, it is necessary to exit from the ds0bootloader application, cycle power to the EVCC, and start the terminal application.

One can use the **`show version`** command to verify that the proper version is running.

Command Line Interface

Startup Banner

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

```
*******************************************************
*             EV Charger Controller v2.3.0            *
*      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
        |CHARGER|CHARGER2|CHARGER3|CHARGER4
        |PROfile|MAP
        |EXTINDCHARGE|CANTERMDIS|TOPBALANCE
        |LINEV|LINEC
        |MAXV|MAXC|MAXBC|TERMC|TERMT
        |FIN_MAXV|FIN_MAXC|FIN_TERMT
        |FLT_MAXV|FLT_MAXC|FLT_TERMT
       ]
  REset [History|PROFILE|EXTINDCHARGE|CANTERDIS|TOPBALANCE]
        history       - reset charge history
        profile <n>   - deletes a charge profile
        extindcharge - reset EXTIND to default (EVCC LED)
        cantermdis    - resets CAN termination to default (enabled)
        topbalance    - resets 'top balance' feature (see documentation)
  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 PSelect A/D
  UPGRADE          - performs a firmware upgrade
```

In most 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 mode, the EVCC monitors the cell loop and operates the buzzer when the cell loop indicates a pack fault.

```
evcc> show
  state    : DRIVE
  cell loop: OK
  proximity: EVSE not connected
  buzzer   : OFF
  uptime   : 0 hour(s), 0 minute(s), 33 second(s)
```

If, instead, the bms configuration is set to "**can**" instead of "**loop**", the output would be the following:

```
evcc> set bms can
evcc> show
  state    : DRIVE
  can bms  : OK
  proximity: EVSE not connected
  buzzer   : ON
  uptime   : 0 hour(s), 0 minute(s), 33 second(s)
```

In the CHARGE mode, 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  :    v2.3.0; Apr 28 2015 14:25:17
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      : loop
  charger  : tsm2500
  maxv     :  20.0V
  maxc     :   2.0A
  termc    :   0.2A
  termt    : 720.0hr
evcc>
```

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 (loop, can, or both)
• `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)
• `fln_maxv`	- (if configured) float charge voltage (for SLA charging)
• `fln_maxc`	- (if configured) float charge current (for SLA charging)
• `fln_termt`	- (if configured) float charge current (for SLA charging)
• `options`	- `extindcharge` (if configured) – ExtInd tracks CHARGE state
	- `cantermdis` (if configured) – can termination resistor disabled
	- `topbalance` (if configured) – enable ‘top balance’ feature

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  : extindcharge (ExtInd is ON when 'charging')
           : cantermdis (CAN termination resistor disabled}
           : topbalance (see documentation)
```

If more than one charge profile is defined, **show config** will display the four 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 yet.

```
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.

Note that the voltage and current measurements are provided by the charger in the CAN message to the EVCC. The EVCC does not measure pack voltage or current.

```
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
- `pack flt` - a cell loop fault or HVC condition 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

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 option will print help for the “set” command.

```
evcc> set
  SEt [ <>
        |BMS
        |CHARGER|CHARGER2|CHARGER3|CHARGER4
        |PROfile|MAP
        |EXTINDCHARGE|CANTERMDIS|TOPBALANCE
        |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|LOOP|CAN|LOOP,CAN]
      charger configuration
        <chargern>         - [CHARGER|CHARGER2|CHARGER3|CHARGER4]
        <type>             - [ TSM2500|TSM2500_41|TSM2500_42|TSM2500_43
                              |ELCON |ELCON_E7 |ELCON_E8 |ELCON_E9
                              | LEAR ]
        set <chargern> <type> - defines <chargern>
        set <chargern> NONE   - deletes <chargern>
        set <chargern> <type> 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
      options
        set extindcharge      - ExtInd is ON when 'charging'
        set cantermdis        - disables the CAN termination resistor
        set topbalance        - enables top balancing (see documentation)
      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 <m>     - charge termination timeout
      SLA charge parameters
        set fin_maxv <v>  - finishing charge voltage
        set fin_maxc <a>  - finishing charge current
        set fin_termt <m> - finishing charge termination timeout
        set flt_maxv <v>  - float charge voltage
        set flt_maxc <a>  - float charge current
        set flt_termt <m> - float charge termination timeout (0=no timeout)
```

set bms
This sets the BMS type. The EVCC can use a cell loop and/or up to four CAN BMSs. The BMS determines whether a cell in the pack has exceeded the High Voltage Cutoff, Low Voltage Cutoff, or Balance Voltage Cutoff. Multiple BMSs can be defined.

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 can
```

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

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. Initially, only profile 1 is defined: 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 selcted, 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 PSelect 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 result is mapped to a profile using the profile map. Initially all four results select the default profile, Profile 1. If the user wants to use two profiles, a switch to ground may be connected at the PSelect 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 PSelect input. The CLI command to enable the mapping from a 5K resistor to Profile 3 would be:

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

set extindcharge

The ExtInd output follows the state of the EVCC LED and provides a remote LED indicator. The intention is that a 12V LED or bulb can provide a remote indication of the EVCC state by interpreting the EVCC blink patterns.

However, the operation of the ExtInd output may be redefined to be +12V whenever the EVCC is charging. This might be used, for example, to drive a relay or some other equipment in the EV.

To enable this option use the command:

```
evcc> set extindcharge
```

If this option is set, this will be indicated in the **show config** output.

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

set cantermdis

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

```
evcc> set cantermdis
```

If this option is set, this will be indicated in the **show config** output.

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

set 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. Enabling this option allows this behavior. In this case, the cell loop / HVC condition does not stop charging, but all other termination reasons (overall time, charge plug disconnected, message timeout, etc) will still apply.

This feature has not been extensively tested, and is in beta.

```
evcc> set topbalance
```

If this option is set, this will be indicated in the **show config** output.

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

set linev_cb, set linec_cb

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, Finishing Charge 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, Float Charge 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
```

reset extindcharge

The **reset extindcharge** command sets the ExtInd output back to default behavior (e.g., ExtInd tracks the EVCC LED).

reset cantermdis

The **reset cantermdis** command sets the CAN termination resistor back to default behavior (e.g., connected).

reset topbalance

The **reset topbalance** command sets the charging behavior back to default (e.g., terminate a charging cycle when there is a cell loop or HVC condition).

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 CANBUS tracing may be independently enabled.

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

trace <>

Trace with no parameters toggles state trace on and off.

trace charger

The **trace charger** command displays messages from the charger. This trace also shows the current number of charging watts and the accumulated WattHours 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
00:08:23.1  V=148.6, A= 7.9, W=1173, Wh= 1.12
00:08:23.6  V=148.6, A= 7.9, W=1173, Wh= 1.28
00:08:24.1  V=148.6, A= 7.9, W=1173, Wh= 1.45
00:08:24.6  V=148.6, A= 7.9, W=1173, Wh= 1.61
00:08:25.1  V=148.6, A= 7.9, W=1173, Wh= 1.77
00:08:25.6  V=148.6, A= 7.9, W=1173, Wh= 1.93
00:08:26.1  V=148.6, A= 7.9, W=1173, Wh= 2.08
00:08:26.6  V=148.6, A= 7.9, W=1173, Wh= 2.25
00:08:27.1  V=148.6, A= 7.9, W=1173, Wh= 2.41
00:08:27.6  V=148.6, A= 7.9, W=1173, Wh= 2.57
00:08:28.0  V=148.6, A= 7.9, W=1173, Wh= 2.73
00:08:28.6  V=148.6, A= 7.9, W=1173, Wh= 2.89
00:08:29.0  V=148.6, A= 7.9, W=1173, Wh= 3.05
00:08:29.6  V=148.9, A= 7.9, W=1176, Wh= 3.22
```

trace canbus

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. It shows whether the EVCC is in DRIVE, CHARGE, or CHARGE/WARMDOWN, as well as the state of the J1772 charge plug.

Here is an example of state trace output that shows the charger plug being plugged in and unplugged.

```
evcc> trace state
state tracing is now ON
evcc> 00:06:53.4  old state=DRIVE, new state=CHARGE, j1772=LOCKED, term rsn=0
00:07:16.9  old state=CHARGE, new state=CHARGE/WARMDOWN, j1772=WAITING FOR DISC, term
rsn=EVSE UNLOCKED
00:07:17.2  old state=CHARGE/WARMDOWN, new state=CHARGE/WARMDOWN, j1772=DISCONNECTED,
term rsn=0
00:07:28.9  old state=CHARGE/WARMDOWN, new state=DRIVE, j1772=DISCONNECTED, term rsn=0
```

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
                  V > 2.5V - 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 connected, then it becomes unlocked, and then finally, removed.

```
evcc> me prox
evcc> Proximity A/D= 1.50V
Proximity A/D= 1.50V
Proximity A/D= 1.50V
00:06:07.5  old state=CHARGING, new state=WARMDOWN, j1772=WAITING FOR DISC, term
rsn=EVSE UNLOCKED
Proximity A/D= 2.76V
Proximity A/D= 2.76V
Proximity A/D= 4.45V
00:06:12.0  old state=WARMDOWN, new state=WARMDOWN, j1772=DISCONNECTED, term rsn=0
Proximity A/D= 4.45V
Proximity A/D= 4.45V
```

measure pselect

The **measure pselect** command gives a real time measurement of the **pselect** 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 mode, waiting for the load to begin. The EVCC must be power cycled in order to leave this mode. See above, Firmware Upgrade, for a description of how to perform a firmware upgrade.

```
evcc> upgrade
```

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.

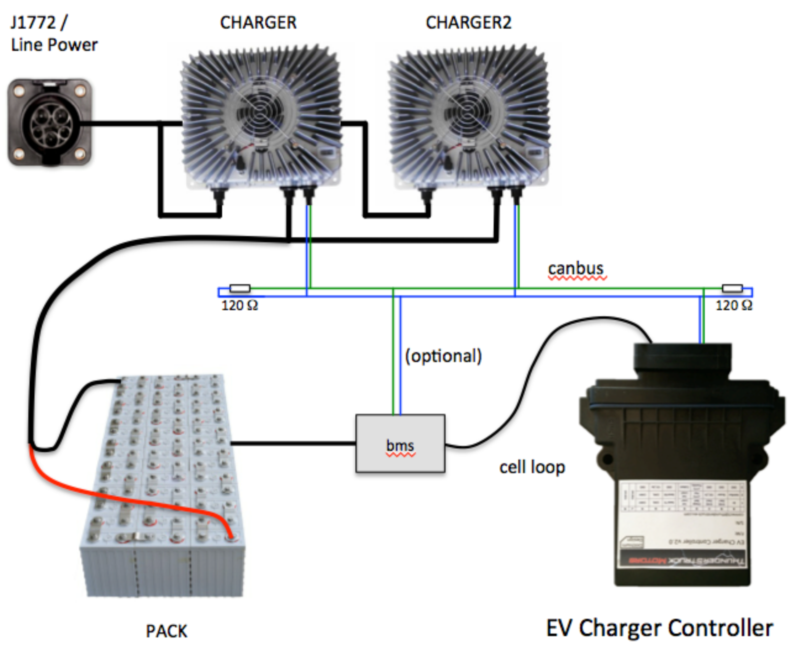


Figure 18 – Multiple Chargers - System Diagram

Note that 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.

- **Line power**. Two chargers require more power than a single charger. One must verify that adequate line power is available.
- **CAN wiring and addressing**. 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.
- **EVCC configuration**. Each charger must be explicitly configured in the EVCC.

Line Power

The EVCC assumes that the service can provide 220V at 30A. Note that the cutback feature, if enabled, will limit line voltage and current to configured limits.

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 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
```

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 can receive line power.

In this example, we want to define a second charger as type tsm2500_42. If the charger is already programmed as type tsm2500_42, then it would only be necessary to use the command **set charger2 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".

Charging with Multiple Chargers

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
00:10:28.9  tsm2500   : V=126.3, A= 5.9, W=745, Wh= 0.09
00:10:29.3  tms2500_42: V=126.6, A= 5.7, W=721, Wh= 0.19
00:10:29.3  tsm2500   : V=126.6, A= 5.8, W=734, Wh= 0.19
00:10:29.8  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.30
00:10:29.9  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.31
00:10:29.9  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.33
00:10:30.0  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.34
00:10:30.0  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.36
00:10:30.1  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.37
00:10:30.2  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.39
00:10:30.3  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.40
00:10:30.3  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.42
00:10:30.4  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.43
00:10:30.5  tsm2500_42: V=127.2, A= 5.9, W=750, Wh= 0.44
```

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.

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
#define BMS_EVCC_CELL_HVC_FLAG            0x01  /* set if a cell is > HVC */
#define BMS_EVCC_CELL_LVC_FLAG            0x02  /* set if a cell is < LVC */
#define BMS_EVCC_CELL_BVC_FLAG            0x04  /* set if a cell is > BVC */

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

Note that although the CAN message only has a 2 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 mode and LVC is assumed if in DRIVE mode.

If the EVCC is configured to use only **can**, then the 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 http://www.trojanbattery.com/pdf/TRJN0109_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).

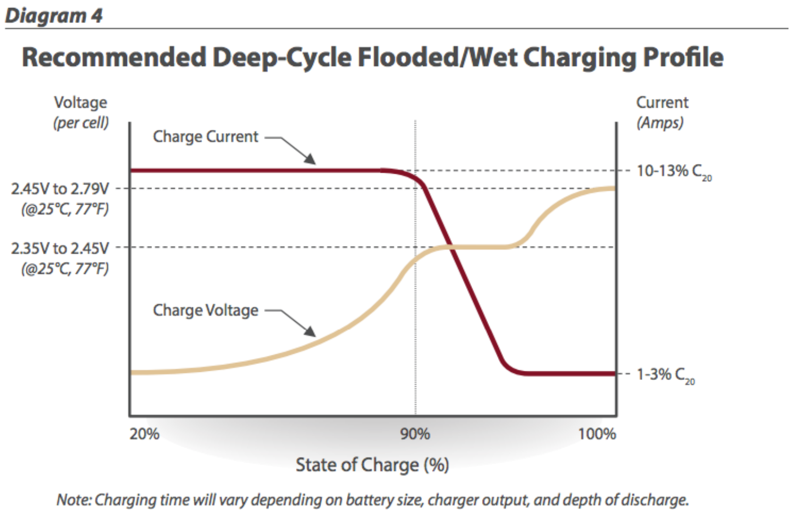


Figure 19 – Flooded Lead Acid Charging Profile (Trojan)

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 14, 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.

EVCC Change Log

This section describes the main functional differences between EVCC v2.3 hardware and firmware releases.

Hardware v2.3

There are two versions of EVCC hardware: the original v2.1 hardware and v2.3 hardware. Hardware version v2.3 has the following major changes:

- 18 pin connector (rather than a 30 pin connector in hardware version v2.1)
- programmable CAN termination resistor (see the CANTERMDIS option)
- J1772 pilot duty cycle support.

The remaining hardware differences between v2.1 and v2.3 are small. It is possible to upgrade v2.1 hardware to the latest v2.3 firmware.

Firmware v2.3.x

The first release of firmware for v2.3 hardware was v2.3.0. This release introduced many new features including:

- CANTERMDIS option to enable and disable a programmable CAN termination resistor
- EXTINDCHARGE option to change the behaviour of the external LED
- adds MAXBC support, which allows a CAN BMS to lower the charging current (for "balancing mode")
- adds J1772 pilot duty cycle support
- support for the TSM2500 LED
- additional TSM2500 diagnostics
- adds different blink patterns depending on the profile selected
- adds a blink pattern to indicate a previously failed charge
- fixes a number of profile editing bugs

v2.3.9

- added TOPBALANCE option (beta only)

v2.3.8

- change the maximum allowed value of **fin_termt** and **flt_termt** from 33.6 hours to 336 hours.
- added support for flt_termt value of 0 (meaning "forever").

v2.3.7

- upgrade support operation

v2.3.6

- Fixed a problem in lead acid support for finishing charge termination. (Add 1V to maxv requested so that the finishing voltage is actually achieved).
- Fixed a bug in TSM2500 status flag error messages

v2.3.5

- Fixed a bug in termt calculations that was introduced in v2.3.0. The calculations were too low by a factor of 6 as a result of changing the units of time from minutes (pre v2.3.0) to tenths of an hour in v2.3.0.
- Maximum values for charge voltage (**maxv**), finishing voltage (**fin_maxv**), and float voltage (**flt_maxv**) changed from 400V to 450V.
- Maximum value of termination current (**termc**) changed from 5A to 10A; default for termination current (**termc**) changed from 0.2A to 2A.
- Maximum values of termination time (**termt**), finishing charge termination time (**fin_termt**) and float charge termination time (**flt_termt**) changed from 72 hours to 2 weeks.
- beta support for Lear charger

v2.3.2

- maximum allowable float current (**flt_maxc**) changed from 1A to 10A
- beta support for upgrade

v2.3.1

- Charger rampup added. When a charge cycle is started, the requested current is ramped from 0 to maxc over a period of 16 seconds. (Results in better startup behaviour for some chargers).

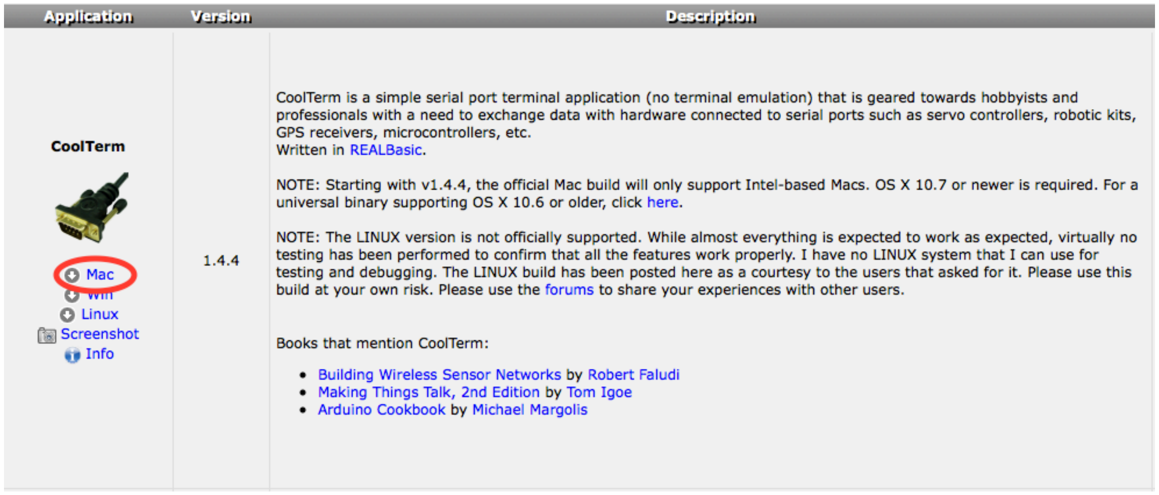
Mac OSX Driver Installation

Before starting the procedure below, ensure the 12V power is hooked up to EVCC B+ and GND, and that 12V is connected to HotInRun. Finally, insure that the USB to serial cable is plugged into the computer.

For MAC OS X, the virtual serial port device name is of the form "usbserial-<sn> where <sn> is the serial number of the USB to serial device. An example of what the name of the EVCC would look like is the following: usbserial-FTGDTR8M.

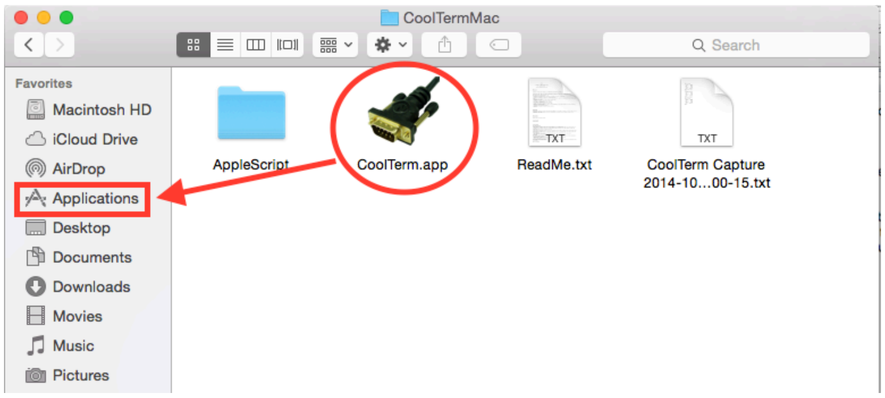
The MAC OSX distribution includes the applications "terminal" and "screen", which may be used. However, we have found that CoolTerm is simpler to install and use.

CoolTerm is a program that allows the user to easily access and program the EVCC via OS X.
1. Go to http://freeware.the-meiers.org
2. Click download for mac

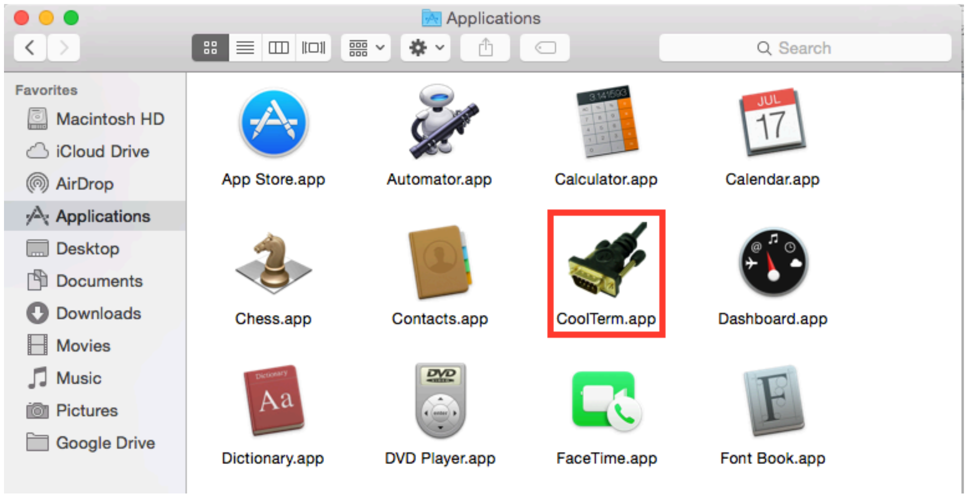


Application	Version	Description
CoolTerm Mac Win Linux Screenshot Info	1.4.4	CoolTerm is a simple serial port terminal application (no terminal emulation) that is geared towards hobbyists and professionals with a need to exchange data with hardware connected to serial ports such as servo controllers, robotic kits, GPS receivers, microcontrollers, etc. Written in REALBasic. NOTE: Starting with v1.4.4, the official Mac build will only support Intel-based Macs. OS X 10.7 or newer is required. For a universal binary supporting OS X 10.6 or older, click here. NOTE: The LINUX version is not officially supported. While almost everything is expected to work as expected, virtually no testing has been performed to confirm that all the features work properly. I have no LINUX system that I can use for testing and debugging. The LINUX build has been posted here as a courtesy to the users that asked for it. Please use this build at your own risk. Please use the forums to share your experiences with other users. Books that mention CoolTerm: • Building Wireless Sensor Networks by Robert Faludi • Making Things Talk, 2nd Edition by Tom Igoe • Arduino Cookbook by Michael Margolis

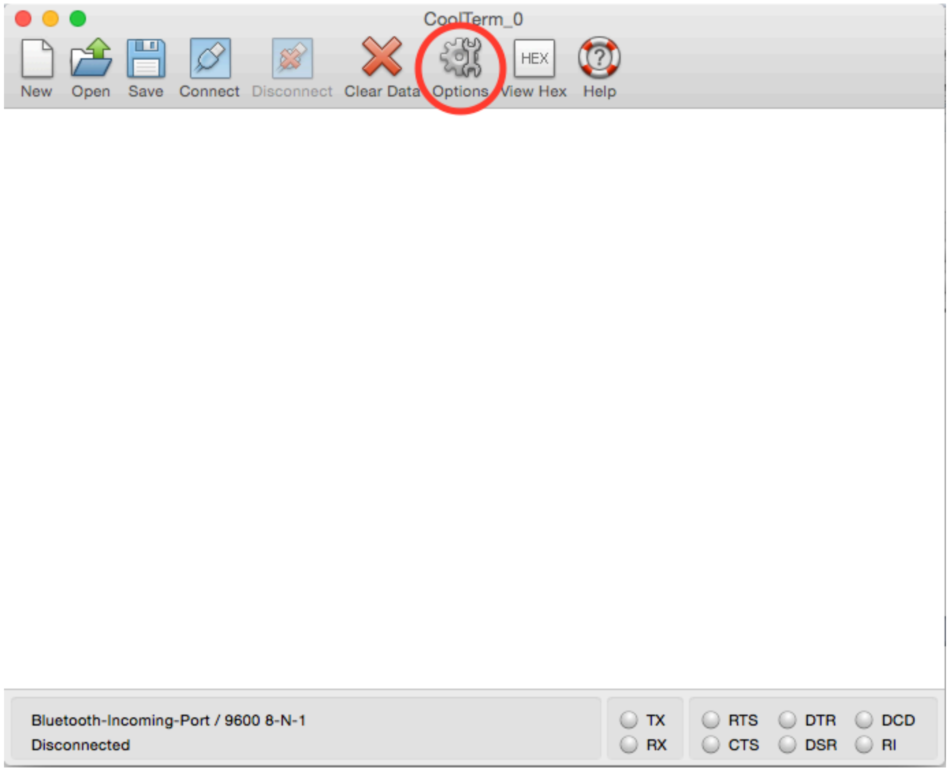
3. Extract the .zip file, open the CoolTermMac folder and drag the CoolTerm app into the applications folder.



4. Open the applications folder and double click CoolTerm.app

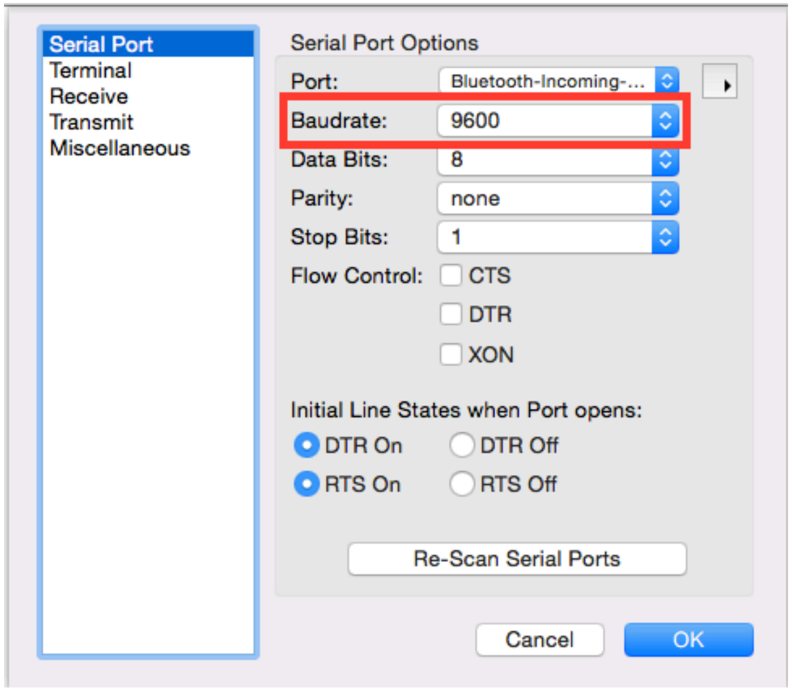


5. Click "Options"



Bluetooth-Incoming-Port / 9600 8-N-1
Disconnected

6. Ensure the "baudrate" is set to 9600 (which should already be set by default).



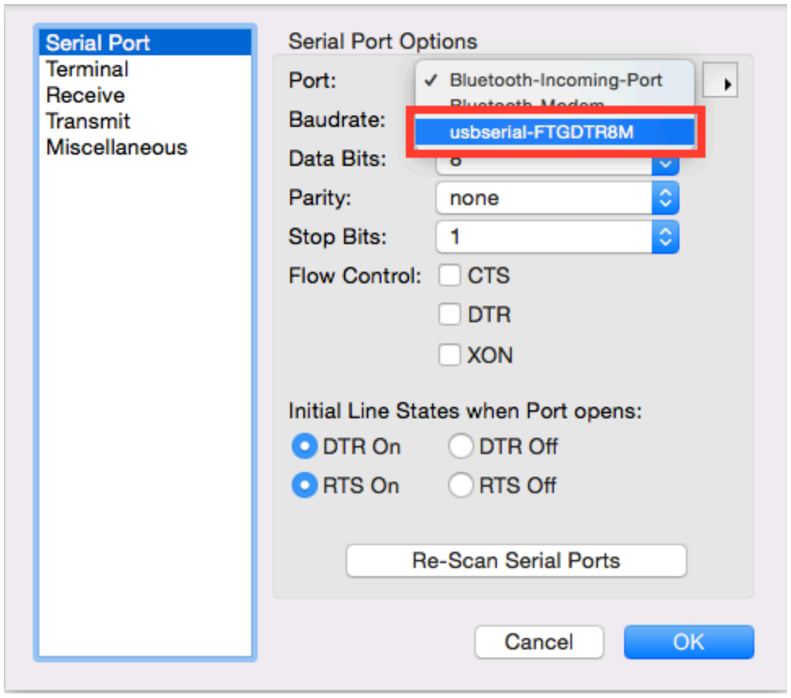
Serial Port Options

Port: Bluetooth-Incoming-...
Baudrate: 9600
Data Bits: 8
Parity: none
Stop Bits: 1
Flow Control: [ ] CTS [ ] DTR [ ] XON

Initial Line States when Port opens:
[✓ DTR On] DTR Off
[✓ RTS On] RTS Off

7. Click the drop down menu and select "usbserial-<sn>" where <sn> is the specific serial number of the EVCC as discussed earlier.

➢ **Note**: The usbserial-<sn> will not show up in the drop down menu if the USB is not plugged in prior to starting the program. If this occurs, exit CoolTerm, plug in the USB cable and restart CoolTerm.

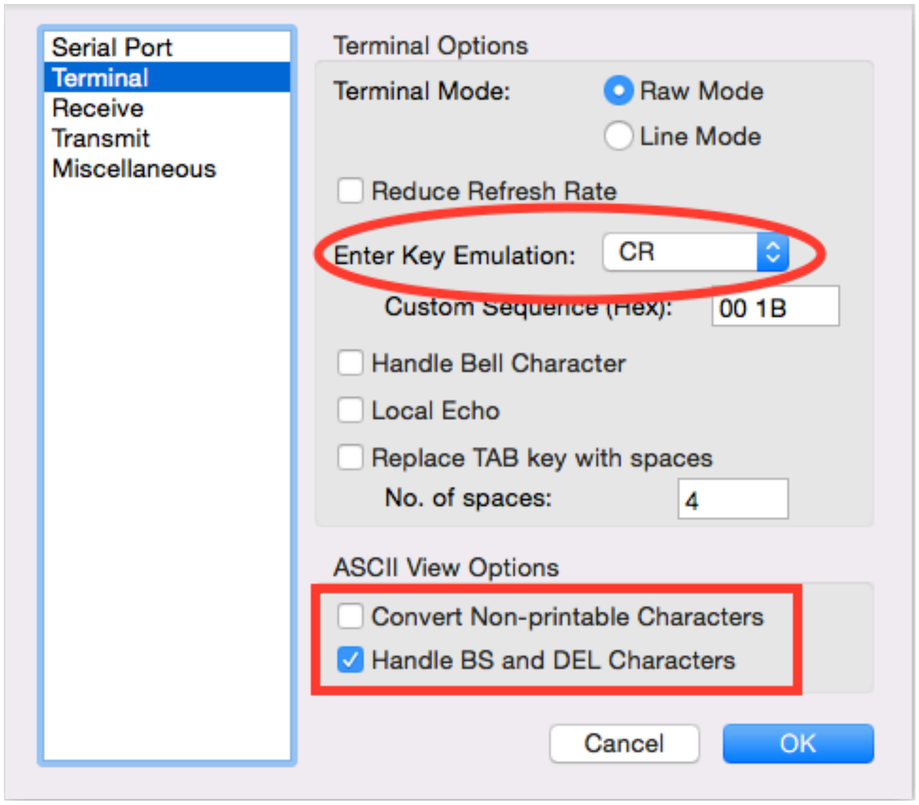


Serial Port Options

Port: usbserial-FTGDTR8M
Parity: none
Stop Bits: 1
Flow Control: [ ] CTS [ ] DTR [ ] XON

Initial Line States when Port opens:
[✓ DTR On] DTR Off
[✓ RTS On] RTS Off

8. Still in "Options" go to the left hand column and click "terminal." Then change the window to match the settings below.



Terminal Options

Terminal Mode: [✓ Raw Mode] Line Mode

☐ Reduce Refresh Rate

Enter Key Emulation: CR

Custom Sequence (Hex): 00 1B

☐ Handle Bell Character

☐ Local Echo

☐ Replace TAB key with spaces

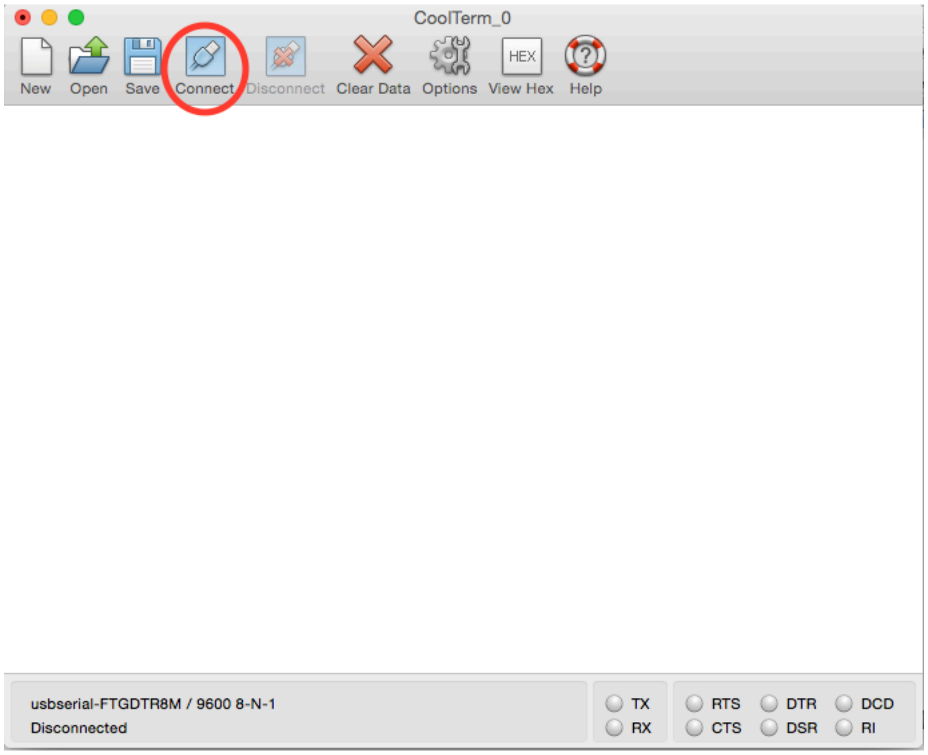
No. of spaces: 4

ASCII View Options

☐ Convert Non-printable Characters

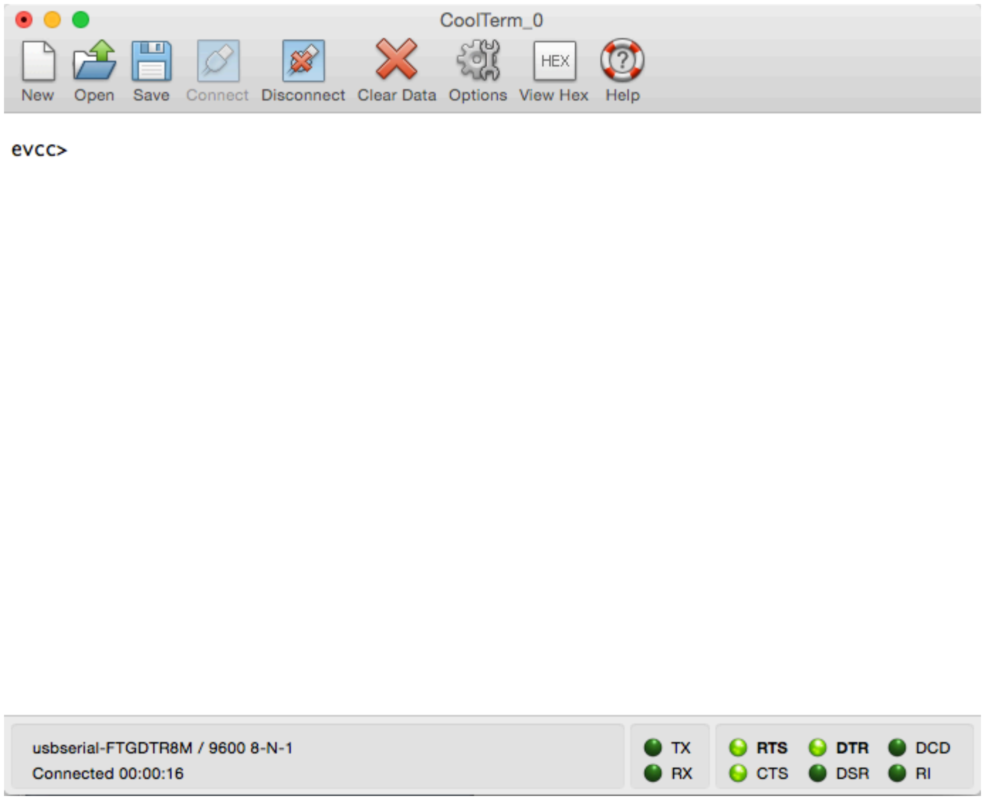
☑ Handle BS and DEL Characters

9. Click “Connect”



usbserial-FTGDTR8M / 9600 8-N-1
Disconnected

10. Press the “return” key, the EVCC command prompt should come up.



```
evcc>
```

usbserial-FTGDTR8M / 9600 8-N-1
Connected 00:00:16

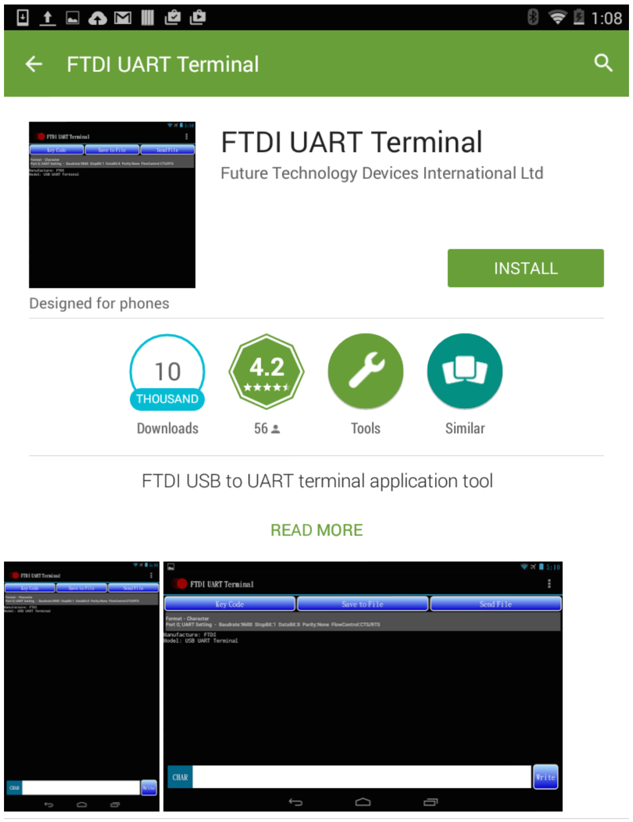
- **Note**: Although the operation of the serial port is very similar to the Windows examples, above, there is one important difference. Windows keyboards generate an ASCII “DEL” character when a “delete” is pressed. MAC keyboards generate an ASCII “BS” character. Current EVCC firmware only interprets the DEL key and the MAC “delete” key may not work as expected. However, the ASCII “DEL” character can usually be generated by MAC keyboards (look for another “delete” key with an “x” or try pressing FN-DEL).

Android Driver Installation

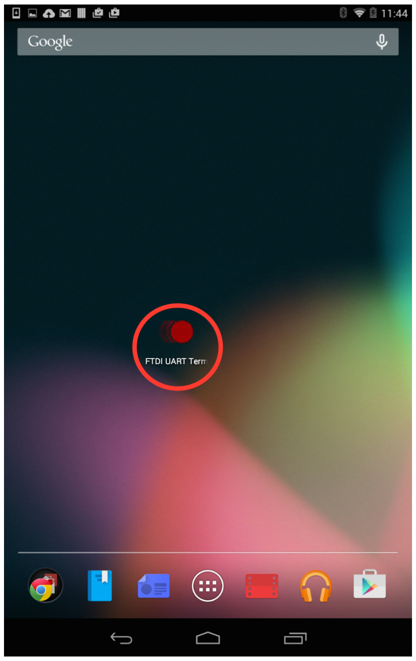
Before starting the procedure below, ensure the 12V power is hooked up to EVCC B+ and GND, and that 12V is connected to HotInRun. Finally, insure that the USB to serial cable is plugged into the OTG cable which is then plugged into the Android device. You will need an "On The Go" (OTG) USB female to USB micro adapter found here:
http://www.amazon.com/IVSO%C2%AE-Nexus-Tablet-Micro-Cable/dp/B00932N46S
In this guide, we will be using an Asus Nexus 7 Tablet. The same (or very similar) process should follow for smart phones and others.

1. Ensure your Android is running 3.2 or later.

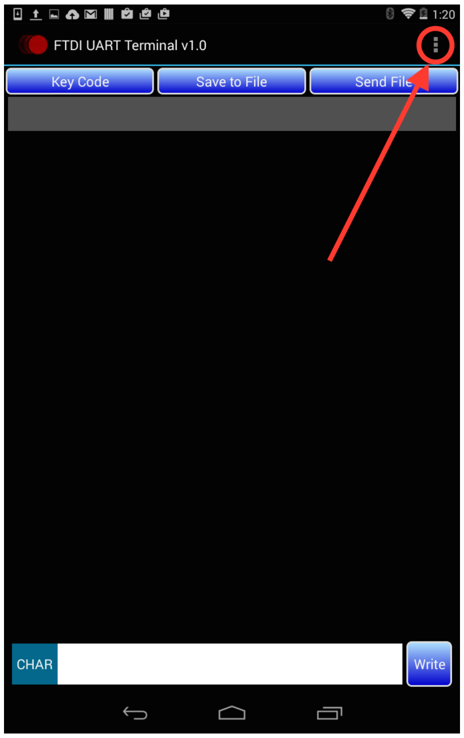
2. Download and install the FTDI UART Terminal app from the app store.



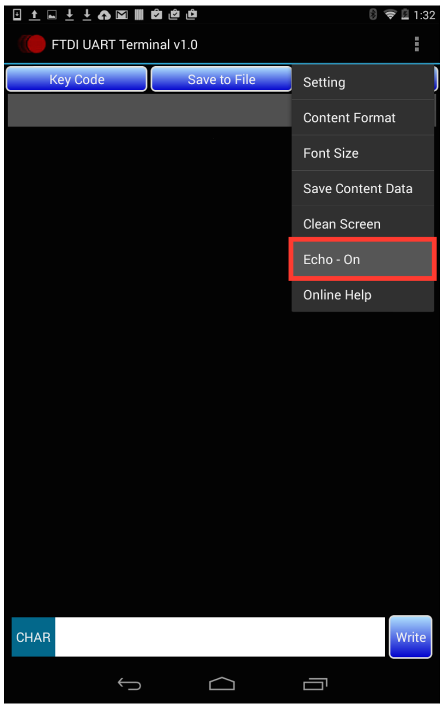
3. It should appear on the homescreen and look like this.

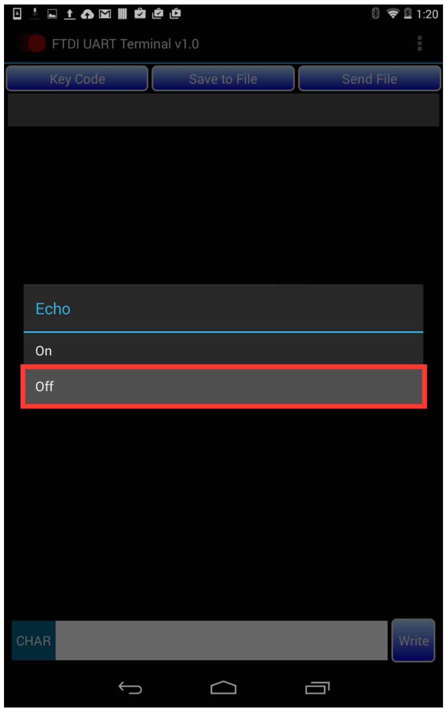


4. You can open the app, or plug in the EVCC and it should auto-detect and prompt you to open it with a program from the list. If this is what you choose to do, select FTDI UART Terminal from the list. (you can also check the box to always open with selected program). Once it is open, push the menu button (this differs from android to android).

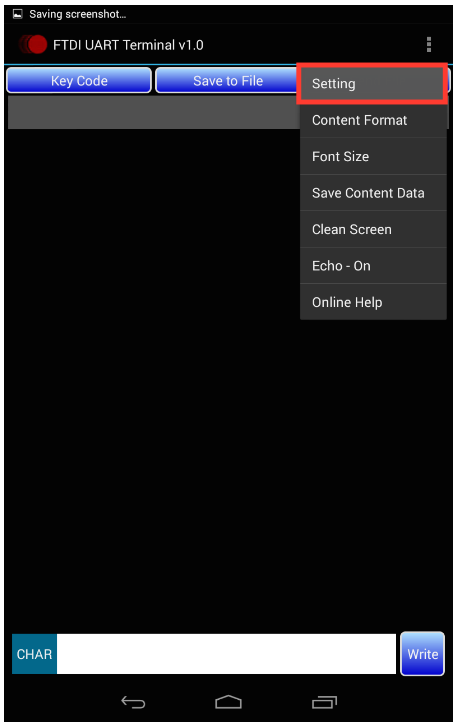


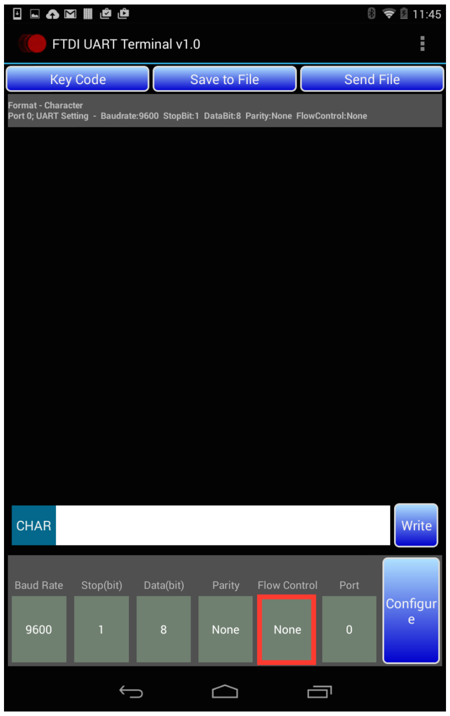
5. Turn Echo Off





6. Go to the menu again and this time go to settings. Change "Flow Control" to "None" then press configure. **Note:** It will not allow the settings to be configured if the EVCC is not connected first.





```
Format - Character
Port 0; UART Setting  -  Baudrate:9600  StopBit:1  DataBit:8  Parity:None  FlowControl:None
```

Baud Rate	Stop(bit)	Data(bit)	Parity	Flow Control	Port
9600	1	8	None	None	0

7. The Android Device should now be set up to program the EVCC.

Please Note: because of the way the text data is sent to the device, it is nessesary to place a "return" after each command before writing the command. This does what "enter" would do on the PC.

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 2.0.0	Sept 22, 2014	In review
Rev 2.0.1	Sept 30, 2014	Production Version
Rev 2.0.2	Nov 10, 2014	Added Mac OSX serial support
Rev 2.1.0	Feb 25, 2015	Added support for v2.1 features, including: multiple chargers, Lead Acid, CAN BMS integration. Cutback configuration was changed.
Rev 2.3.0	April 25, 2015	Updates for EVCC 2.3 Hardware. Preliminary version.
Rev 2.3.1	May 29, 2015	EVCC v2.3 Production
Rev 2.3.2	Jan 13, 2016	Added a changelog, description of TOPBALANCE and Lear charger