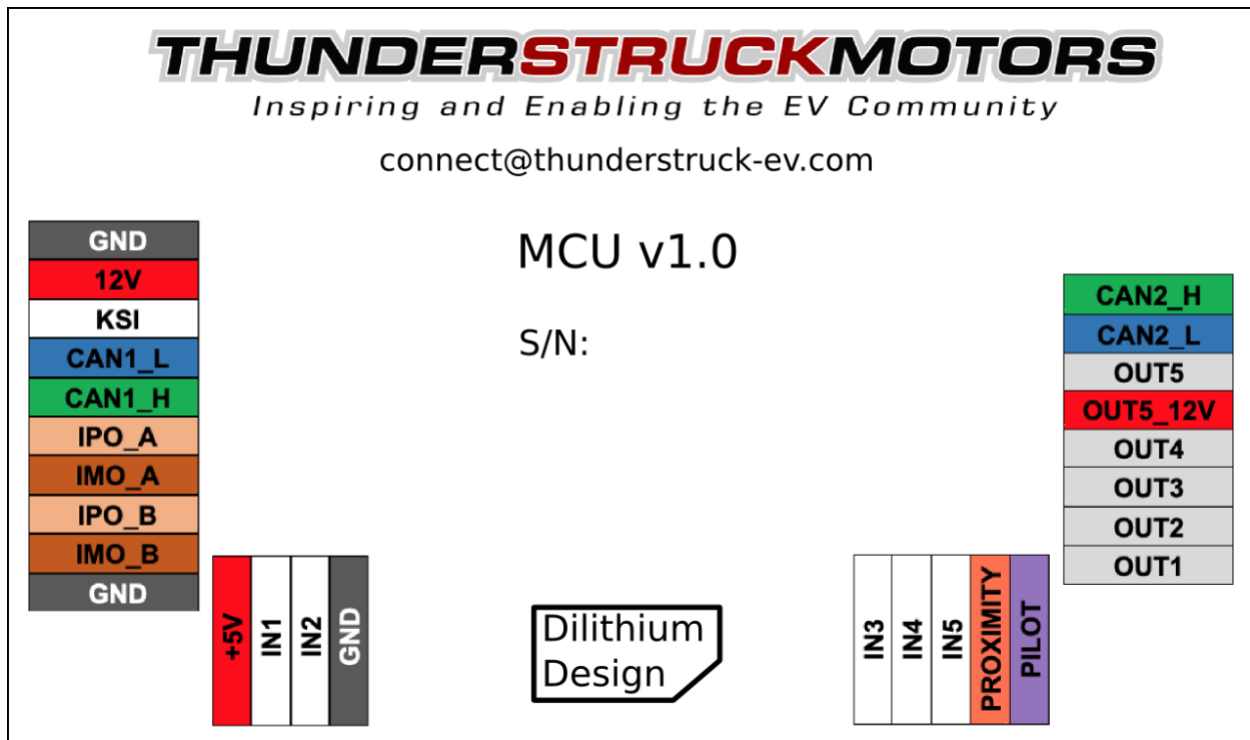


Thunderstruck Motors

EV Master Control Unit v1.0



Doc Rev 1.5, December 2023

© 2021, 2022, 2023 Dilithium Design

Contents

OVERVIEW	6
INSTALLATION	7
Power	8
Inputs	9
Outputs	9
CAN	10
IsoSPI	11
Conditions	12
Serial Port	13
Firmware Upgrade	14
SYSTEM	15
Input Functions	15
Output Functions	16
CAN Services	19
BATTERY MANAGEMENT	20
BMSS24	21
BMSS18	22
Harness Verification	23
A123	24
LTC Cell Group Wiring	26
Probe and Diagnostics	27
Pack, Group, and Cell Numbering	29
Cell Measurement	33
Cell Balancing	37
Thermistors	37
ONBOARD CHARGING	39
Charge Plugs and Line Powering	41
Charging	47
Charge Configuration	51
Charger Support	54
Debugging	57
DC FAST CHARGING	59
ChargeMo	59
ZEV CCS Charging	63

INSTRUMENTATION	66
State of Charge Measurement	66
Sensor Configuration.....	69
MCU EV Display	70
Analog Fuel Gauge.....	75
Analog Temperature Gauge	76
Speedhut Gauges	76
Curtis 3100r.....	77
Torque	78
DC/DC Converter Support	79
CAN SUPPORT SUMMARY	80
SID Support.....	81
EID Support.....	89
Thermistor Retrieval.....	94
SERIAL PORT MESSAGES	96
Startup Banner.....	96
LTC Discovery Messages	96
Cell HVC and LVC Messages.....	97
Thermistor Temperature Messages	97
IsoSPI Error Messages	97
SERIAL PORT COMMANDS.....	97
help	98
show.....	101
set.....	110
enable disable	113
edit delete.....	115
reset	116
measure.....	117
trace	117
lock	118
upgrade	119
Warranty and Support.....	120
Document Revision	120

Figures

Figure 1 – MCU Enclosure	7
Figure 2 – Push-In Connector	7
Figure 3 – MCU Connectors.....	8
Figure 4 – CAN Network Diagram	11
Figure 5 – CAN Termination Solder Jumper	11
Figure 6 – MCU Conditions	12
Figure 7 – MCU Blink Patterns	13
Figure 8 – Firmware Upgrade Utility	14
Figure 9 – Input Functions.....	15
Figure 10 – Dedicated Output Functions.....	17
Figure 11 – BMSS24 System Connectors	21
Figure 12 – BMSS24 Cell Harness Connectors	21
Figure 13 – BMSS24 Thermistor Connectors	22
Figure 14 – BMSS18 System Connector.....	22
Figure 15 – BMSS18 Harness Connector.....	23
Figure 16 – Using the Cell Harness Tester.....	23
Figure 17 - A123 Module Connector Factory Wiring	25
Figure 18 – Cell Group Wiring.....	26
Figure 19 – Cell Group Wiring Spanning A Fuse	27
Figure 20 – 48 Cell Pack Wiring	28
Figure 21 – 44 Cell Pack, Cell Groups Connected Where Convenient.....	30
Figure 22 – Parallel Packs	32
Figure 23 – BMSS24 Thermistor Connector.....	38
Figure 24 – BMSS18 Thermistor Connectors	38
Figure 25 – J1772 Type 1 System Diagram	42
Figure 26 – J1772 Type 1 EV Connector	43
Figure 27 – J1772 Type 2 Connector	44
Figure 28 – J1772 Type 2 - One Charger Per Phase.....	45
Figure 29 – Onboard Charge Termination Reasons	50
Figure 30 - CHAdeMO Charge Port Connector.....	59
Figure 31 – ChargeMo Charge Port Connections.....	59
Figure 32 – ChargeMo Onboard Equipment Connections	60
Figure 33 – ChargeMo Charge Termination Reasons	62
Figure 34 - DHAB Sensor Support.....	66
Figure 35 – DHAB V1 Hall Sensor Installation	66
Figure 36 – DHAB V2 Hall Sensor Installation	67
Figure 37 – DHAB Hall Sensor Connections	67
Figure 38 – HTFS Sensor Support.....	67
Figure 39 – HTFS Sensor Installation Direction	68
Figure 40 – HTFS Hall Sensor Connections	68
Figure 41 - Isabellenhütte IVT-S Current Sensor.....	69
Figure 42 – MCU EV Display Hardware	70
Figure 43 – MCU Display Startup and Status Screens.....	71
Figure 44 – MCU Display Charging Progress Screen	75
Figure 45 – Analog Fuel Gauge Typical Connections	75
Figure 46 – 3100R Display	77
Figure 47 – 3100R Display Examples	77
Figure 48 – OBDII PID Support.....	79

Figure 49 – CAN Services.....80

OVERVIEW

The Thunderstruck EV Master Control Unit (MCU) performs Battery Management, Charge Control, and Instrumentation Control for Electric Vehicles.

The MCU supports three Battery Management architectures: LTC12, LTC18, and A123. The BMS architecture determines the measurement technology and the cell group size.

- LTC12: 12 cells per cell group, uses Dilithium Design BMSS24
- LTC18: 18 cells per cell group, uses Dilithium Design BMSS18
- A123: 13 cells per cell group, uses the 13-cell A123 MBB (Measurement and Balance Board)

The MCU supports two Analog Device IsoSPI interfaces to communicate with BMSS24 and BMSS18 measurement boards. Each IsoSPI interface can support up to 8 daisy-chained measurement devices. An LTC12 architecture can support up to 196 cells and a LTC18 architecture can support up to 288 cells.

The A123 MBBs use CAN to communicate. Up to 16 MBBs are supported, for a total of 208 cells.

Onboard Charge Control functions supported by the MCU are the J1772 Type 1 and Type 2 charge protocols. A variety of CAN enabled chargers are supported, including the Thunderstruck TSM2500 charger. The MCU can auto-start upon charge plug insertion. Multiple chargers and multiple charge profiles are supported.

The MCU supports ChargeMo and ZEVCCS DC Fast Charging (see *DC Fast Charging* for details).

The MCU uses an external Hall current sensor to perform State of Charge calculations. It can drive an analog OEM “fuel gauge,” as well as CAN based gauges such as the Curtis 3100r display, CAN-enabled Speedhut gauges, and the Dilithium Display.

The MCU supports OBDII and can operate with the Torque PRO application. The MCU supports limited CANopen allowing CAN based monitoring and configuration.

The MCU has five general purpose inputs and five general purpose outputs which may be mapped to a variety of functions, depending upon the application.

The MCU has a green/red LED to show overall system status, and can also indicate status using discrete outputs. MCU configuration is performed using a serial interface and a terminal emulation program such as PuTTY or Coolterm. The firmware supports diagnostic commands to verify proper hardware operation and wiring, to trace CAN messages, and for system reporting. MCU firmware can be upgraded using the serial interface should that be necessary.

INSTALLATION

The MCU is housed in a Serpac WM021I enclosure, a 4.1 x 2.65 x 1.1 plastic enclosure with mounting flanges. The enclosure and connector are not waterproof and should be appropriately protected from the elements.

The datasheet for the enclosure is at https://www.serpac.com/userprints/wm021i_up.pdf.

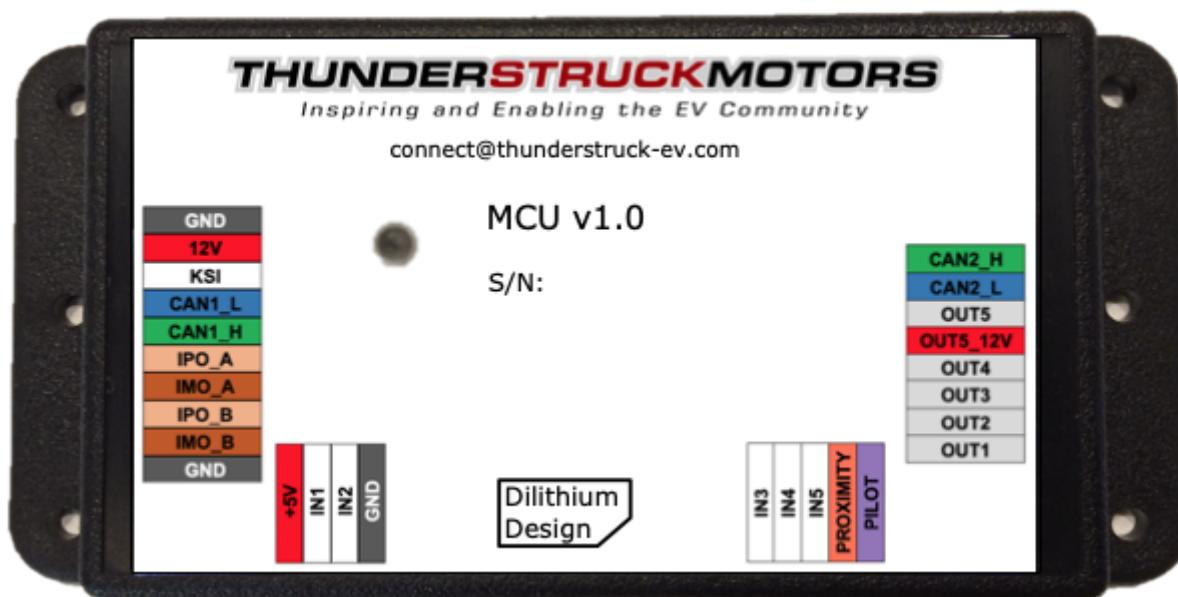


Figure 1 – MCU Enclosure

The four connectors (labelled A, B, C, and D) are “push-in” connectors. These connectors accept 20-24 gauge stranded or solid wire: stranded 20-gauge wire is recommended. To make a connection, strip the wire back 1/4”. Twist the wire end and insert. Be sure that all strands of wire get fully inserted to prevent shorting between adjacent wires.

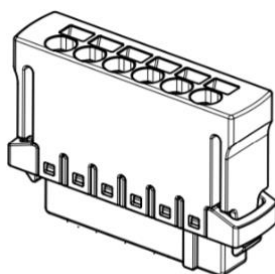


Figure 2 – Push-In Connector

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

The connector part numbers are:

- 10p Connector A Harting 14311013101000
- 4p Connector B Harting 14310413101000
- 5p Connector C Harting 14310513101000
- 8p Connector D Harting 14310813101000

The following figure lists the MCU connections:

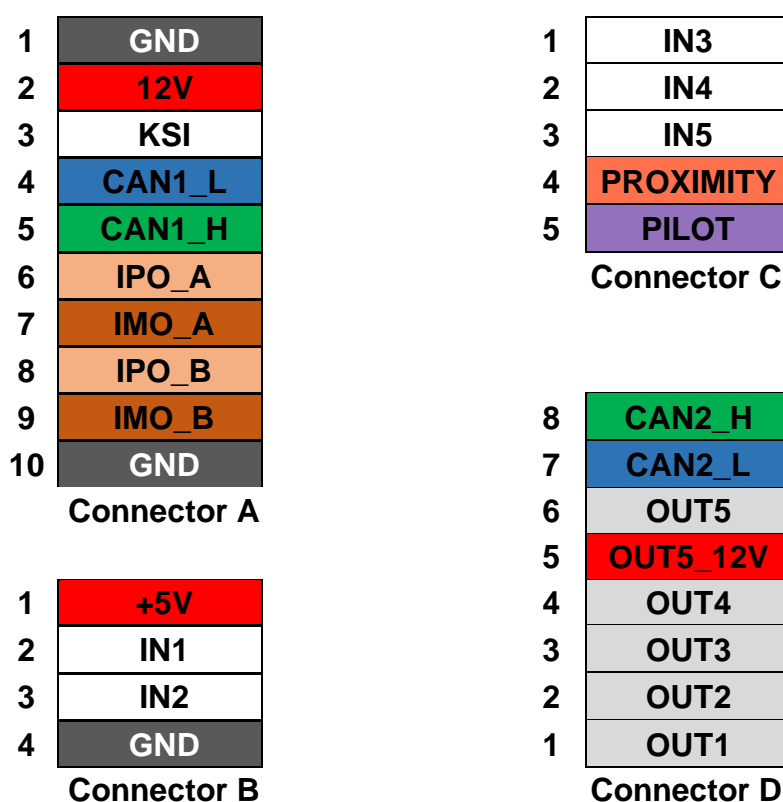


Figure 3 – MCU Connectors

Power

12V (input) and **GND** (Connector A, pins 1 and 2) provide “always on” power.

+5V (output) and **GND** is provided on Connector B, pins 1 and 4. Connector B is intended to connect directly to a dual range Current Sensor such as the LEM DHAB S137.

The pin **OUT5_12V** (Connector D, pin 5) is an input which provides power for the OUT5 output.

Inputs

IN1 to **IN5** are general purpose inputs.

- **IN1, IN2, IN3, IN4** measure voltage from 0 to 5V. These inputs are tolerant up to 14V.
- **IN5** contains an onboard pullup resistor and measures resistance to ground.

KSI, Proximity, Pilot, and are fixed function inputs.

- **KSI** (Connector A, pin 3) is the “Key Switch Input. This input is connected to +12V when the EV KeySwitch is in the “ON” position.
- **Proximity** (Connector C, pin 4) handles the J1772 PROXIMITY signal. It measures the proximity resistor and has autostart support.
- **Pilot** (Connector C, pin 5) handles the J1772 PILOT signal. As an input, it measures the duty cycle of the J1772 PILOT square wave.

General purpose inputs are assigned to an **Input Function** which determines how the input is interpreted by MCU firmware. The full list of Input Functions is described in the SYSTEM section, below.

The **measure** serial port command can be used to report the value of an input and aids hardware troubleshooting. The **measure** command takes one argument which can either an Input (such as **IN3**) or an Input Function.

Once **measure** is initiated, the output will repeatedly print until any key is pressed.

```
sys> measure in3
sys> in3 0.90v
in3 0.89v
in3 0.89v
in3 0.90v
in3 0.90v
```

Outputs

OUT1 to **OUT5** are general purpose outputs.

- **OUT1, OUT1, OUT3** and **OUT4** are “open collector” outputs rated to 200ma.
- **OUT1, OUT2,** and **OUT3** support a PWM output mode.
- **OUT5** is an output that can switch to 1.5A of +12V. When using this output, **OUT5_12V** must be connected to a 12V source which provides the current for this output.

“Open-collector” outputs are high impedance (e.g., not connected to anything) when disabled, and switched to ground when enabled.

OUT5 is a switched output. It is high impedance when disabled and switched to **OUT5_12V** when enabled.

General purpose outputs are assigned to an **Output Function** which determines how the output

is used by MCU firmware. The full list of Output Functions is described in the SYSTEM section, below.

Pilot is both an input and output. As an output, it can apply the J1772 PILOT resistor and diode as required by J1772.

It is possible to manually **enable** or **disable** general purpose outputs. The **show out** serial port command will print the current state of the outputs. Note that it is best to **enable** or **disable** an output that does not have an associated Output Function, otherwise running firmware may overwrite any “enable” or “disable” setting.

```
sys> enable out3
sys> enable out5
    out5  +12v
sys> sh out
    out1  ---
    out2  ---
    out3  GND
    out4  ---
    out5  +12v
    pilot          EVSE disabled
sys>
```

CAN

CAN is a communications protocol widely used in automotive applications. 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 devices.

The MCU supports two independent, non-isolated CAN interfaces, CAN1 and CAN2.

CAN1_L and **CAN1_H**, on Connector A, are the CANL and CANH signals for CAN1.

CAN2_L and **CAN2_H**, on Connector D, are the CANL and CANH signals for CAN2.

CAN baud rate programming and service configuration is described in the SYSTEM section, below.

A CAN network is a multi-station network that is terminated at both ends of the network by 120-ohm termination resistors. Each device on the CAN bus uses two connections: CANH (“CAN high”) and CANL (“CAN low”) to communicate on the bus. Although usually not shown, but important, is that each node on the CAN network share a common ground (GND).

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

A CAN network might only consist of two devices. In this case a short direct connection between the two using hand-twisted wire is sufficient. For longer runs, more devices, or cases

where EMI may be an issue, shielded cable may be used. If a shield is used, it should be connected to chassis ground at a single place.

The following generic diagram depicts a typical CAN network:

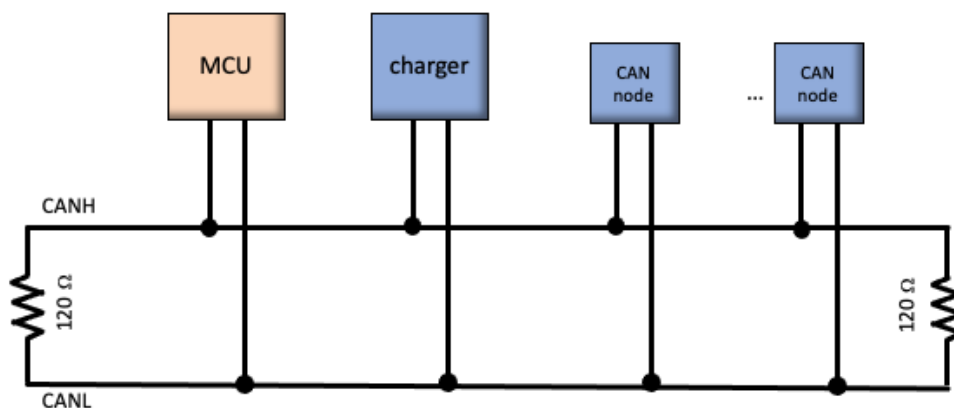


Figure 4 – CAN Network Diagram

CAN termination resistors are installed on both CAN interfaces in the MCU, which can obviate the need for an external termination resistor. The assumption, however, is that the MCU is placed as a terminal node in the CAN network(s).

In order to verify that the CAN network is properly terminated, make the connections to all CAN nodes, and measure the resistance between CANH and CANL: it should be 60 ohms, which indicates the presence of two 120-ohm resistors in parallel.

If necessary, the MCU termination resistors may be disabled: open the enclosure, locate JP1 or JP2 and cut the trace connecting the gold pads using an X-Acto knife or equivalent. (If, later, it is necessary to reinstate the jumper, a blob of solder may be placed on the jumper to bridge the termination)

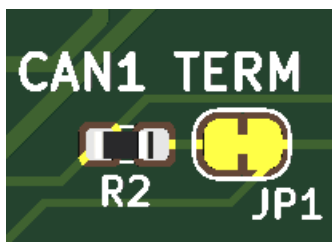


Figure 5 – CAN Termination Solder Jumper

IsoSPI

IsoSPI is a protocol defined by Analog Devices used to communicate with its battery cell measurement devices. IsoSPI provides galvanic isolation between nodes and is resistant to EMI. IsoSPI is a “daisy-chain” protocol: each device receives every message and forwards it

downstream. Replies are clocked in a serial fashion from all connected devices by the MCU. IsoSPI can operate at 1 Mbps over 100 meters of cable. The protocol supports a Packet Error Code to detect datalink errors.

The MCU supports two IsoSPI interfaces. These interfaces require no configuration and are automatically used by the MCU if the LTC12 or LTC18 BMS architecture is configured.

IPO_A and **IMO_A**, on Connector A, are the IPO and IMO signals for IsoSPI interface A. **IPO_B** and **IMO_B**, on Connector A, are the IPO and IMO signals for IsoSPI interface B.

The rules for wiring an IsoSPI network are similar to those for CAN. Wiring should be kept short and the conductors should be twisted. Wiring should be placed away from EMI (Electro-Magnetic Interference) such as the motor and controller, and parallel runs next to EV traction cabling should be avoided. There is, however, no need to add external termination resistors to the IsoSPI network as all Dilithium hardware has the necessary on-board terminations.

Conditions

An MCU *Condition* is MCU system status that is either TRUE or FALSE. For example, the **PLUGIN** condition is TRUE if the charge plug is inserted. Some conditions indicate a system fault; the term *Alert* is sometimes used which refers to a Condition that indicates a system fault.

Condition	Description	Reference
HWFault	BMS Hardware Fault	see <i>Probe and Diagnostics</i>
CCensus	BMS Cell Census Fault	see <i>Cell Measurement, Cell Census</i>
TCensus	BMS Thermistor Census Fault	see <i>Thermistors</i>
HVC	BMS High Voltage Cutoff	see <i>Cell Measurement, HVC and LVC Conditions</i>
LVC	BMS Low Voltage Cutoff	see <i>Cell Measurement, HVC and LVC Conditions</i>
HIPACK	BMS High Pack Voltage	see <i>Cell Measurement, HIPACK and LOWPACK Conditions</i>
LOWPACK	BMS Low Pack Voltage	see <i>Cell Measurement, HIPACK and LOWPACK Conditions</i>
HITEMP	BMS High Temperature	see <i>Thermistors, HITEMP and LOWTEMP Conditions</i>
LOWTEMP	BMS Low Temperature	see <i>Thermistors, HITEMP and LOWTEMP Conditions</i>
NOTLOCKED	BMS Configuration not “locked”	see <i>Cell Measurement, Cell Census</i>
KSI	Key Switch Input	see <i>Onboard Charging</i>
CHARGE	Charging in progress	see <i>Charging</i>
PLUGIN	Charge Plug is connected	see <i>Charging</i>

Figure 6 – MCU Conditions

Conditions are updated by running MCU firmware and may be used singly or in combination to create Output Functions.

Some conditions determine LED behavior. Because multiple conditions may be TRUE at the same time, the highest priority condition is used to determine which blink pattern is displayed by

the LED. The table below lists the conditions in order of priority, and the associated LED blink pattern. Each rectangle represents 125ms, so each “pattern” read right to left indicates, in all, two seconds of LED output.

The “ALL OK” row in the table does not represent a “condition” per se. It is the MCU blink pattern when no other blink-pattern-determining condition is present.

[illegible]

Figure 7 – MCU Blink Patterns

Serial Port

The MCU uses a serial port for configuration, diagnostic output, debugging, and for firmware upgrade. A 5V USB to 3.5mm TRS jack serial cable is required. The FTDI TTL-232R-5V-AJ is recommended. To use the cable, the host computer requires serial port drivers (also known as “Virtual Comm Port Drivers”), which may be downloaded from the FTDI website www.ftdichip.com/Drivers/VCP.htm.

A terminal application such as PuTTY (Windows) or Coolterm (Mac or Windows) is needed on the host computer. See the Serial Port Utilities document for installation and connection details:

www.thunderstruck-ev.com/images/companies/1/DD_SerialPortUtilities_v1.3.pdf

The serial interface may be tested with just a few connections to the MCU. Connect +12V to the **12V** and **KSI** inputs, the 12v ground to the **GND** input, and connect the serial cable to the MCU serial port next to **Connector A**. Configuration in the terminal application includes these settings: speed:115200 baud, 8 data bits, 1 stop bit.

When the MCU powers up, it prints a startup banner on the Serial Port which includes firmware version information, it may also print startup messages, including inventory messages (such as how many BMS measurement devices have been detected) and alert messages.

The MCU firmware is functionally organized into four areas: MCU, System, Battery Management, Charge Control, and Instrumentation. These are abbreviated: MCU, SYS, BMS, EVCC, and INST. Navigation between areas is done by typing the name of the subsystem. For example, to navigate to the BMS settings, type “bms” at any prompt.

Firmware Upgrade

The firmware upgrade procedure for the MCU uses a Firmware Upgrade Utility (which is available for Windows computers only). The utility is named MCU_FW_Upgrader.exe, and can be downloaded from the following Google Drive location:

<https://drive.google.com/file/d/1gjNkjwqQjk40tTrIRFiTmzgsw0W0C72v/view?usp=sharing>

Upgrading to a newer version will generally retain existing configuration and set any new configuration to default values.

To upgrade:

1. Power on the MCU and connect it to the computer with the provided serial cable.
2. Start the MCU_FW_Upgrader application. A window will open as shown below.
3. Use the pulldown to select the **Communications Port**, and the new **Firmware Upgrade File** provided by ThunderStruck.
4. Select **Update**. The upgrade should start, and the utility will show a progress bar. The MCU LED will blink to show that the upgrade has started.
5. When the upgrade completes, exit the MCU_Firmware_Upgrader, start a new terminal application session, and cycle all power to the MCU. **To cycle power, remove and reconnect Connector A.**
6. Enter **show version** to verify that the proper version is running.

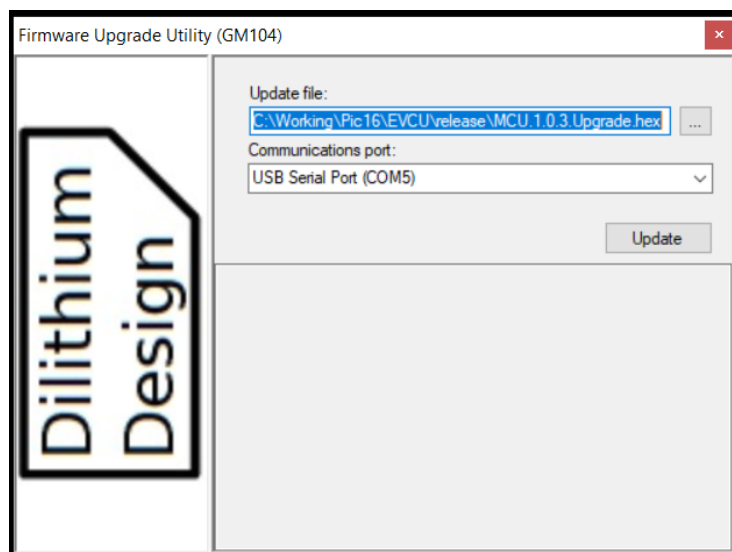


Figure 8 – Firmware Upgrade Utility

SYSTEM

System settings can be retrieved at the serial port by navigating to the “sys” prompt and typing the command **show config**. This will show Inputs, Outputs, CAN configuration, and CAN services.

```
mcu> sys
sys> show config
--inputs-----
  in1      : hall1
  in2      : hall2
--outputs-----
  out1     : fuelg
  out2     : hvc
--can-----
  can1br   : 250 Kbps
  can2br   : 500 Kbps
--services-----
  canopen  : enabled
  obcharge : can1
```

Input Functions

An **Input Function** defines the behavior of an Input. The following is a list of supported Input Functions:

Input Function	Description	IN1/2/3/4	IN5	Reference
hall1	Hall Sensor Low	X		see <i>Hall Sensor Support</i>
hall2	Hall Sensor High	X		see <i>Hall Sensor Support</i>
loop, -loop	Cell Loop	X	X	see <i>Charge Configuration</i>
socres, -socres	SOC Reset	X	X	see <i>Current Sensor Support</i>
pselect	Profile Selection	X	X	see <i>Charge Profiles</i>
cseq1	Charge SEQUENCE 1	X		see <i>ChargeMo</i>
cseq2	Charge SEQUENCE 2	X		see <i>ChargeMo</i>
plugin	Charge Cable Present	X		see <i>ZEV CCS Charging</i>

Figure 9 – Input Functions

The **pselect** Input Function is used to select a charge profile. For example, the input **IN5** may be assigned to the **pselect** Input Function by the following command:

```
sys> set in5 pselect
```

The assignment may be deleted by setting an Input to “none”.

```
sys> set in5 none
```

In many cases, the Input Function completely specifies both its hardware characteristics and its software interpretation. The Input Function **hall1**, for example, requires a voltage input from 0 to 5V and is used in the software to indicate a hall current sensor reading.

The **loop** input is used as a “go/no-go” indication of whether it is permissible to charge. The **socres** input is used to manually reset the State of Charge meter. These functions are meant to be provided by external hardware such as a relay or pushbutton. In order to provide design flexibility, the input functions **-loop** and **-socres** can be used to invert the sense of these inputs before firmware interpretation. So, for example, if **loop** is assigned to IN1, then if IN1 is “+5V” then the loop function will be considered “good” (and if it is 0V, loop would be considered “bad”). However, if instead **-loop** is assigned to IN1, then the logic is inverted: if IN1 is 0V, the loop function is considered good.

The command **show input** will show the Input Function mappings, and will display both the hardware value (e.g., the voltage or resistance reading at the Input) and the “logical value” (e.g., the interpretation of that reading, depending on the mapped Input Function).

In the following example, IN1, IN2 and IN5 are assigned to Input Functions:

```
sys> show input
in1  1.21v  hall1  DHABS137 H = -643.0A
in2  0.66v  hall2  DHABS137 L = -68.1A
in3  0.92v
in4  0.57v
in5  inf    pselect 1, profile=1
prox  2.7K  J1772 disconnected
pilot          not detected
ksi      1   detected
sys>
```

If the **measure** command is used on an Input with an associated Input function, both the physical and “logical” value of the input will be displayed.

Once initiated, the output will repeatedly print until any key is pressed.

```
sys> measure hall1
sys> in1  1.21v  hall1  DHABS137 H = -642.5A
in1  1.21v  hall1  DHABS137 H = -642.5A
in1  1.21v  hall1  DHABS137 H = -642.5A
in1  1.21v  hall1  DHABS137 H = -642.5A
sys>
```

Output Functions

An **Output Function** defines the behavior of an Output. There are two types of Output Functions: **Dedicated** Output Functions and **Condition** Output Functions.

Dedicated Output Functions

A Dedicated Output Function has predefined behavior. The following table lists the Dedicated Output Functions.

Output Function	Description	Type	Reference
power	Power	Dedicated	see <i>Charging / Output Functions</i>
cperm	Charge Permission	Dedicated	see <i>ChargeMo</i>
ccont	Contactor Control	Dedicated	see <i>ChargeMo</i>
fuelg	Analog Fuel Gauge	Dedicated / PWM	see <i>Analog Fuel Gauge</i>
tempg	Analog Temperature Gauge	Dedicated / PWM	see <i>Analog Fuel Gauge</i>

Figure 10 – Dedicated Output Functions

An Output is assigned to an Output Function using the **set** command. The **show** command lists the physical output as well as the logical interpretation of the Output Function.

As an example, the **fuelg** function is mapped to OUT1. In this case the State of Charge (soc) is currently 6%, the “102” is the PWM duty cycle that is being output from OUT1.

```
sys> set out1 fuelg
sys> show out
  out1    102  FUELG soc = 6%
  out2     ---
  out3     ---
  out4     ---
  out5     ---
  pilot           EVSE disabled
sys>
```

Condition Output Functions

A **Condition** Output Function is a logical combination of one or more MCU Conditions (see above).

The following command configures **OUT3** to be enabled when the **HVC** condition is TRUE and OUT4 to be enabled when the **LOWPACK** condition is TRUE.

```
sys> set out3 hvc
sys> set out4 lowpack
sys> sh out
  out1    102  FUELG soc = 6%
  out2     ---
  out3     ---  hvc
  out4    GND  LOWPACK
  out5     ---
  pilot           EVSE disabled
sys>
```

In this example, the **show** command shows that OUT3 is not enabled (as hvc is FALSE, and is shown in lowercase), and OUT4 is enabled as the **LOWPACK** condition is TRUE (and shown in UPPERCASE).

To configure **OUT3** to be enabled when **HVC** is FALSE, use a “-” before the condition, which will invert the polarity of the output.

```
sys> set out3 -hvc
sys> sh out
  out1    102  FUELG soc = 6%
  out2    ---
  out3     GND  -HVC
  out4     GND  LOWPACK
  out5    ---
  pilot           EVSE disabled
sys>
```

A “logical OR” of several Conditions can be used as an Output Function. To set the high pack threshold voltage (to 420V) and to configure **OUT3** to be enabled when either **HVC** or **HIPACK** are TRUE:

```
sys> set hipackv 420
sys> set out3 hvc hipack
sys> sh out
  out1    102  FUELG soc = 6%
  out2    ---
  out3     GND  HVC hipack
  out4    ---
  out5    ---
  pilot           EVSE disabled
sys>
```

The addition of the “- (“ and “)” characters around a Condition list negates the logic of the final result. So, if in the next example, **OUT3** will be disabled when either **HVC** or **HIPACK** are TRUE.

```
sys> set out3 -(hvc hipack)
sys> sh out
  out1    102  FUELG soc = 6%
  out2    ---
  out3    ---  -(HVC hipack)
  out4    ---
  out5    ---
  pilot           EVSE disabled
sys>
```

An Output Function can use many fault conditions:

```
sys> set out4 hwfault ccensus tcensus hvc lvc hipack lowpack hitemp lowtemp notlocked
```

It is also possible to invert individual conditions of a condition list:

```
sys> set out3 lvc lowpack -plugin
```

In the last example, **OUT3** is enabled if **LVC** is TRUE, **LOWPACK** is TRUE, or **PLUGIN** is FALSE. (Or, equivalently, **OUT3** is not enabled if **LVC** is FALSE, **LOWPACK** is FALSE, and **PLUGIN** is TRUE).

Finally, the polarity of individual elements as well as the entire expression can be inverted:

```
sys> set out3 -(lvc lowpack -plugin)
```

In this example, OUT3 is enabled if **LVC** is FALSE, **LOWPACK** is FALSE, and **PLUGIN** is TRUE. Such an Output Function could be used to control the charger in a solar power application (which turns on a charger when the pack drops below a setpoint).

Output Function Shortcuts

Two Condition Output “Shortcuts” have been defined: **FAULT**, and **CHARGEOK**.

The **FAULT** shortcut lists all MCU fault conditions, namely: **HWFAULT**, **CCENSUS**, **TCENSUS**, **NOTLOCKED**, **HVC**, **LVC**, **HIPACK**, **LOWPACK**, **HITEMP** and **LOWTEMP**. The resulting Output Function is TRUE if any fault condition is TRUE.

The **CHARGEOK** shortcut is TRUE when charging is permitted. It is assumed that any of **HWFAULT**, **CCENSUS**, **TCENSUS**, **HVC**, **HIPACK**, **HITEMP**, **LOWTEMP**, and **NOTLOCKED**, if TRUE, would be sufficient to stop charging. In particular, since **LVC** and **LOWPACK** are omitted, the presence of these conditions does not prevent charging.

CAN Services

The MCU supports two CAN interfaces, CAN1 and CAN2.

The can baud rate for the CAN1 or CAN2 can be changed using the commands:

```
sys> set can1br 500  
sys> set can2br 250
```

See below, *CAN Support Summary*, for a list of CAN Services that are supported by the MCU.

BATTERY MANAGEMENT

The MCU supports several BMS architectures: **LTC12**, **LTC18**, and **A123**. The architecture type determines the measurement technology and modularity of the BMS functions. The BMS uses the term **cell group** to refer to a series connected set of cells. Each cell group has associated measurement hardware that: monitors cells within the group, is electrically isolated from the MCU, and communicates over an isolated datalink. Cell groups can be flexibly mapped to “packs” and “groups” in order to support different pack sizes and configurations.

The architecture is determined by the **arch** parameter may be set to one of **ltc12**, **ltc18**, **a123** or **none** (which disables the BMS). The **arch** parameter is not affected by the **reset config** command and must be edited separately.

The **LTC12** architecture has a cell group size of 12 cells. This architecture works with Dilithium Design BMSS24 measurement boards which can monitor two cell groups or 24 cells.

The **LTC18** architecture has a cell group size of 18 cells. This architecture works with the Dilithium Design BMSS18 measurement board which can monitor up to 18 cells.

The underlying measurement technology used in the BMSS24 and BMSS18 are the Analog Devices ltc6804, ltc6811, and ltc6813 devices (LTCs). These devices share many features and specifications. Each LTC device can measure its cells in less than 2.5ms with a total measurement error of less than 1.2mV. Accuracy is stable over voltage, temperature, and time. LTCs support diagnostics to identify hardware and wiring problems and are tolerant of common wiring errors such as shorted or open connections. LTCs supports passive cell balancing.

LTCs use the Analog Devices IsoSPI protocol to communicate between devices and with the MCU. IsoSPI is a two-wire protocol which provides galvanic isolation between cell groups.

LTCs are powered from the pack connections. The LTC devices have been designed to minimize power consumption, especially during long-term storage where battery drain is unacceptable. When the MCU enters “sleep” mode it will stop sending IsoSPI messages to the LTCs, and the LTCs automatically enter a low power mode which draws minimal current (~4 μ A).¹

When an IsoSPI message arrives, the LTCs will power up. Current is used in short bursts to operate the IsoSPI datalink and for cell measurement. The same amount of current is drawn from each group, regardless of the number of cells connected which minimizes the possibility of unbalancing the pack due to LTC current needs.

The **A123** architecture works with the A123 13-cell modules that integrate 13 cells with a Measurement and Balance Board (MBB). The A123 MBB integrates the cell measurement and

¹ For datasheets, see <https://www.analog.com/en/products/ltc6811-1.html>, <https://www.analog.com/en/products/ltc6804-1.html>, <https://www.analog.com/en/products/ltc6813-1.html>

balance circuitry with the MBB module. Each module must have a unique pre-assigned unit id. Module Id assignment is typically performed as an offline step before installation. The modules communicate with the MCU using CAN.

When setting the architecture to **a123**, it is also necessary to define which CAN interface to use to communicate with the A123 MBBs. This is done with the commands

```
mcu> set arch a123
mcu> enable a123 can1
```

BMSS24

The Dilithium Design BMS Satellite (BMSS24) uses the Analog Devices ltc6811 or ltc6804 battery monitor chips and supports two cell groups of 12 cells each. A cell group can consist of fewer than 12 cells, but cells must be series connected and in total result in greater than 11V. Depending on nominal cell voltage, the minimum cell group size should be at least four (or five) cells.

The BMSS24 has an upstream IsoSPI interface (IPI and IMI) and a downstream IsoSPI interface (IPO and IMO) which connects to upstream or downstream BMSS24 or BMSS18 devices, if any. There is also an internal IsoSPI connection between the two onboard measurement devices.

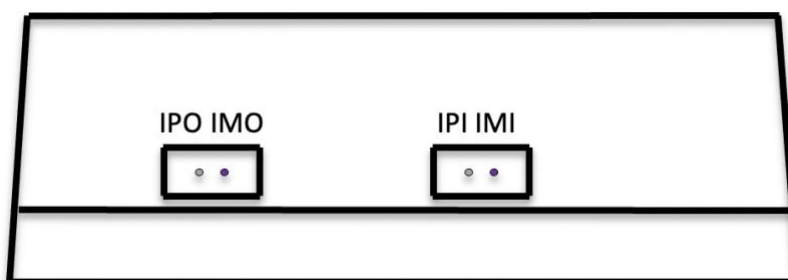


Figure 11 – BMSS24 System Connectors

A BMSS24 has two cell harness connectors, A and B. If only one cell group is used in the BMSS24, it must be connected to the “A” harness connection. The two LTCs in a BMSS24 are electrically isolated from each other and as a result, LTCs can be connected to Cell Groups anywhere in the pack and in any order, wherever convenient.

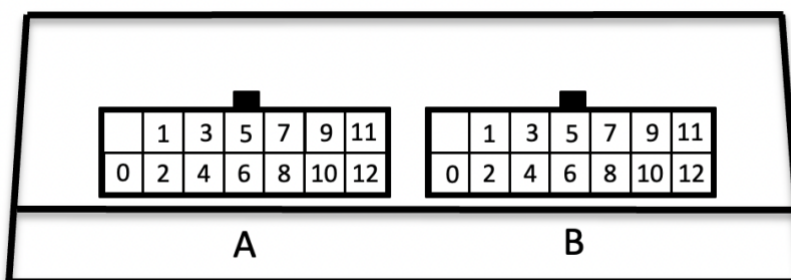


Figure 12 – BMSS24 Cell Harness Connectors

The BMSS24 supports two sets of thermistors; connectors are located on the sides of the enclosure.



Figure 13 – BMSS24 Thermistor Connectors

The BMSS24 and BMSS18 Thermistor Connection pinouts are identical. See below, *Thermistors*.

BMSS18

The 18 cell BMSS18 has one ltc6813 18-cell measurement device and must be used with the LTC18 architecture². These devices support a single cell group of 18 cells. A cell group can consist of fewer than 18 cells, but cells must be series connected and in total result in greater than 16V. Depending on nominal cell voltage, the minimum cell group size should be at least six (or more) cells.

The BMSS18 has one four-pin connector that connects to two IsoSPI interfaces: signals are labeled IPO, IMO, IPI, IMI. A BMSS18 has one cell harness connector.

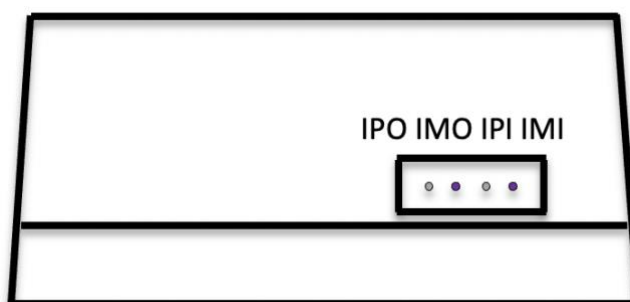


Figure 14 – BMSS18 System Connector

² When using a BMS architecture of LTC18, it is permissible to mix 12 cell LTC6811 BMSS24s with 18 cell BMSS18 devices.

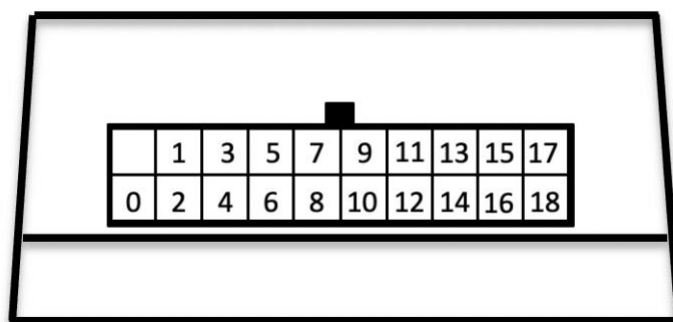


Figure 15 – BMSS18 Harness Connector

The BMSS18 has one thermistor connector and supports five thermistor connections, two pins for each thermistor. The BMSS24 and BMSS18 Thermistor Connection pinouts are identical. See below, *Thermistors*.

Harness Verification

Cell Harness testers are available for the BMSS24 and BMSS18 LTC measurement boards. The cell harness tester is used to verify that cells are connected in the proper order and with the proper polarity.

NOTE: If the cell harness is not wired correctly, the BMS measurement board may be damaged.

After wiring the harness to the pack, plug the connector into the harness tester and use a voltmeter to verify proper hookup. The voltage difference between adjacent cells should be the voltage of a single cell in the group. If fewer than 12 cells are used in the group then the top cells should all have identical voltage readings, indicating that they are correctly bridged together at the highest cell.

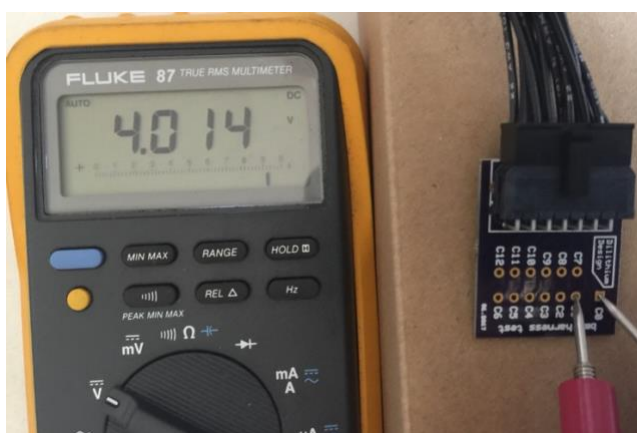


Figure 16 – Using the Cell Harness Tester

NOTE: The Harness Tester is connected directly to pack voltage and so there may be up to 60V present on the harness. When using the Harness Tester, be careful with metal tools and make sure the working surface is nonconductive.

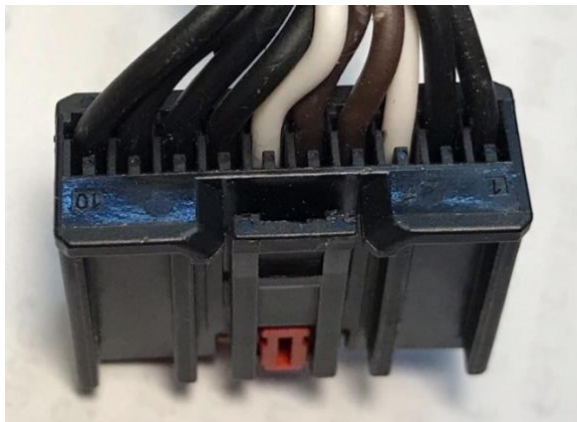
Once the cell harness is wired and the connections verified, it is safe to plug the harness into the BMSS24 or BMSS18. In general, Cell Harness connectors may be plugged in or unplugged in any order.

For more information, see:

<https://www.thunderstruck-ev.com/images/companies/1/BMS/BMSHarnessVerify-Safety1.pdf>

A123

The MCU supports A123 26s3p 60 AH Modules. These modules integrate two groups of 13 cells, each with an associated MBB which provides the cell management and balance circuitry. The MCU can support up to 8 modules or 16 MBBs. The MBBs communicate with the MCU using CAN. The A123 factory connectors are shown below:



MBB Connector



BMS Connector

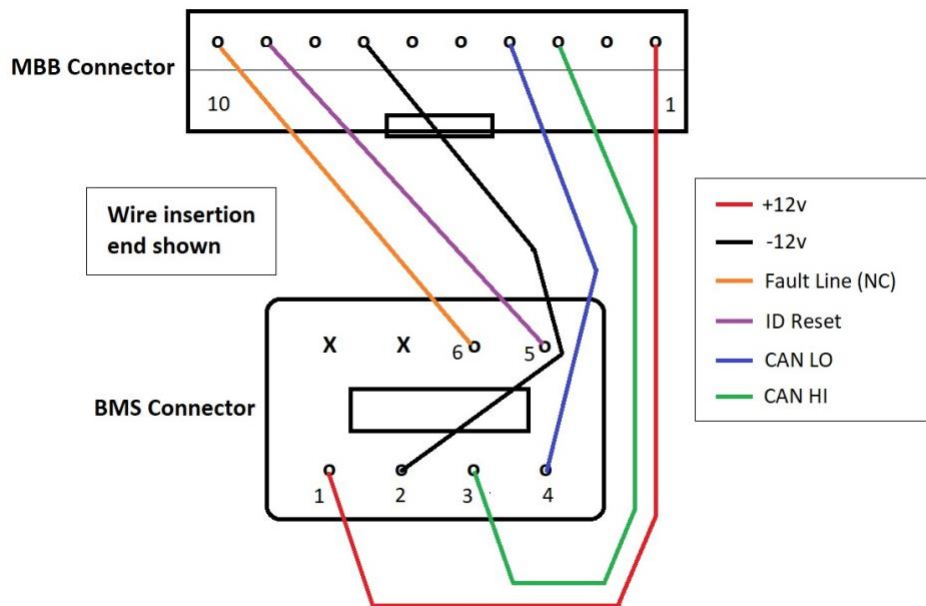


Figure 17 - A123 Module Connector Factory Wiring

MBB Wiring

The A123 OEM module wiring is designed so that it can be connected in serial to extend the power, CAN and control signals to make larger packs. A harness consists of a single BMS connector and (typically) several MBB connectors: one MBB connector for each MBB.

MBBs operate on 12V power, which is provided to the +12V and -12V (e.g., chassis ground) signals on the A123 BMS connector. It is recommended that the OUT5 output from the MCU be used to power the modules when the MCU is on. See *Outputs*, above.

The MBB CAN connections (CAN LO and CAN HI in the BMS connector) are connected to an MCU CAN network. For best performance it is recommended that MCU CAN1 be used and that no other devices are on CAN1. The MCU has an internal CAN termination resistor on both CAN interfaces. There may already be a CAN termination resistor in the original A123 harness from the factory. However, smaller packs or custom wired packs may require an additional 120-ohm termination resistor installed in the harness. (To verify that the harness is correctly terminating the CAN network, measure CANH to CANL at the BMS connector before connecting it to the MCU; the value should be 120-ohms, indicating the presence of the termination resistor).

A123 setup

Each MBB module must have a unique pre-assigned unit id; the ID is used to determine the CAN addresses that it uses. Assigning the unit id is typically performed as an offline step before installation and would typically be performed by ThunderStruck if it is providing the A123 Modules.

The following steps are used if an addressing sequence needs to be performed. The example is for a pack consisting of 4 modules or 8 MBBs. The MCU supports a pack with up to 16 MBBs.

1. Connect the harnesses to the MBBs and MCU as indicated above.
2. Connect the “ID Reset” wire from the BMS connector to chassis ground.
3. Power up the MCU and connect to the MCU serial port.
4. Enter **set mbbaddr 1 8** in the serial monitor (8 is the number of MBBs.)
5. Remove the ID Reset wire from ground.
6. Cycle power to the MCU and MBBs, reconnect to the MCU serial port.
7. Enter **show mbb** to ensure the expected MBBs are listed.
8. Enter **show cells** to verify cell connections.
9. Enter **lock** to initiate error reporting with the found modules and cells.

As with all packs, it is necessary to configure the appropriate charge parameters and BMS thresholds. Contact the vendor or ThunderStruck for advice.

LTC Cell Group Wiring

The figure below shows two LTCs, one connected to a Cell Group of 12 cells and a second LTC connected to a partially populated Cell Group with only 8 cells. Note that it requires 13 connections (or “wires”) to connect to the 12 cells in a Cell Group. The wires can be numbered 0 to 12. All cells in a Cell Group must be wired consecutively: the positive terminal of one cell must connect to the negative terminal of the next cell. If fewer than 12 cells are to be monitored, the unused cell inputs should be at the top, connected together and to the last positive cell connection in the group.

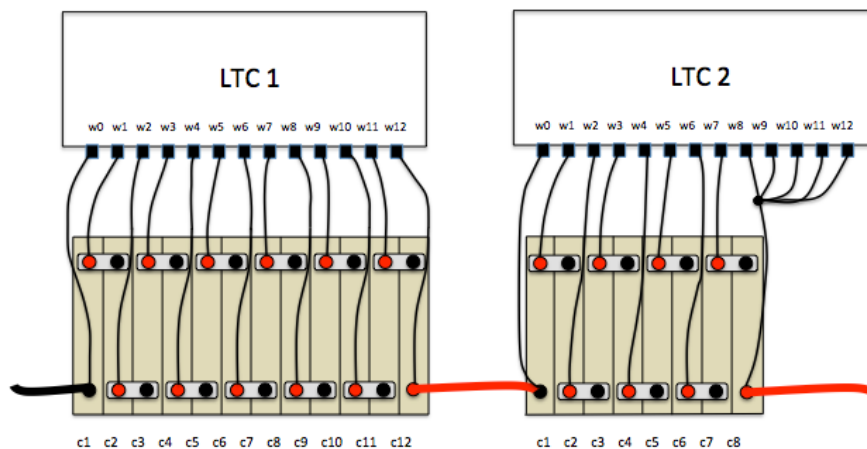


Figure 18 – Cell Group Wiring

A Cell Group should not “span” a circuit protection device (such as a contactor, fuse, or a circuit breaker). See the figure below for an example.

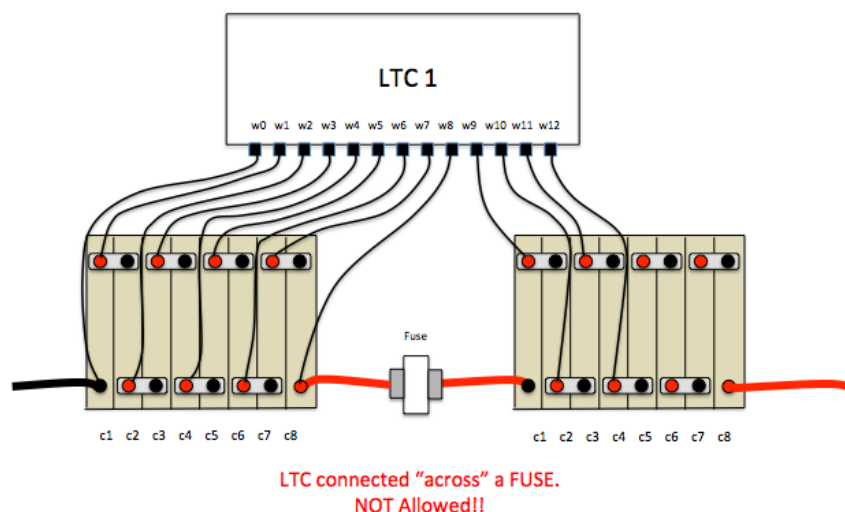


Figure 19 – Cell Group Wiring Spanning A Fuse

The reason for this guideline is that if the fuse blows (or contactor opens, or circuit breaker trips), then the LTC may be destroyed because the LTC will span the newly open circuit and could be damaged by full pack voltage.

A Cell Group should not span a traction pack jumper. If, despite these guidelines, this cannot be avoided, then the Cell Group must be disconnected at the BMS before the traction jumper is disconnected. A warning label is advised.

Probe and Diagnostics

Upon startup and periodically afterwards, the MCU performs a “probe” to determine how many measurement devices are present, and diagnostics are performed on all discovered devices. The results of discovery and diagnostics are printed at the serial port.

For the **LTC12** and **LTC18**, each LTC detected will be assigned an address from A1 to A8 or B1 to B8, depending upon which IsoSPI interface is used. The LTC that is wired directly to the MCU IsoSPI A interface will be numbered A1.

The following example shows a 48-cell pack wired as one series string of cells. The 48 cells are managed by two 24-cell BMSS24s; each BMSS24 can manage two cell groups as it has two 12-cell LTCs. MCU IsoSPI interface A is wired to the first BMSS24, which is then daisy-chained to the second BMSS24.

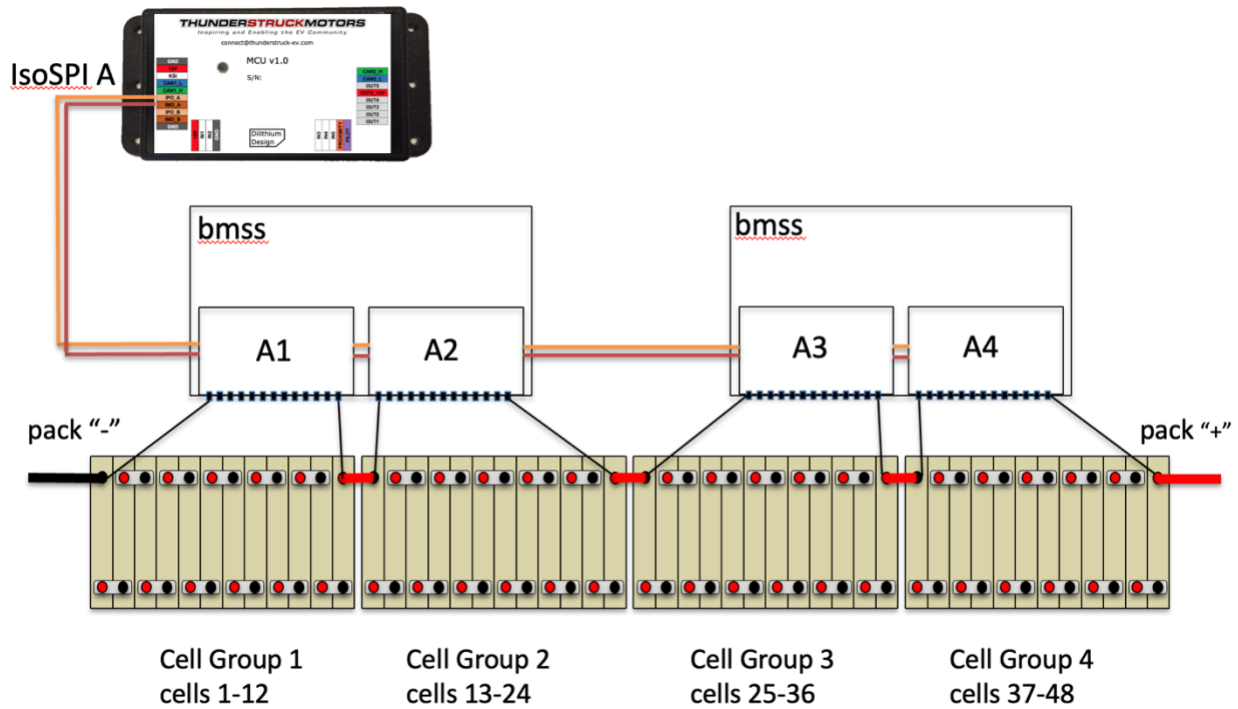


Figure 20 – 48 Cell Pack Wiring

In this case, four LTCs will be discovered by the MCU and assigned the addresses A1, A2, A3 and A4, in the order that the links are connected. The following output will show at the serial port upon powerup:

```
ltc A1 detected
ltc A2 detected
ltc A3 detected
ltc A4 detected
```

The **show ltc** command can be used to retrieve the results of diagnostics, the chip type and revision, values of internal voltages, and the total voltage measured by this device.

```
mcu> show ltc
ltc| status | rev | Van | Vdig | Vref | SOC | SUM | conf | disc
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
A1 | ok | 6811-6 | 5.092v | 3.205v | 3.006v | 38.16v | 38.15v | 0x00000 | 0x00fff
A2 | ok | 6811-6 | 5.127v | 3.204v | 3.003v | 38.17v | 38.16v | 0x00000 | 0x00fff
A3 | ok | 6811-6 | 5.043v | 3.214v | 3.004v | 38.21v | 38.20v | 0x00000 | 0x00fff
A4 | ok | 6811-6 | 5.089v | 3.208v | 3.003v | 38.22v | 38.21v | 0x00000 | 0x00fff
```

If there is a diagnostic failure, it will be reported in this initial discovery message. A hardware fault will print with a hex value that gives additional developer information.

```
ltc A1 detected
ltc A2 detected
ltc A3 detected, diagnostic failure=0x2840
ltc A4 detected
```

Any diagnostic failure will set the **HWFAULT** condition. The results will also be available in the **show ltc** command.

```
mcu> show ltc
ltc| status | rev | Van | Vdig | Vref | SOC | SUM | conf | disc
----|-----|-----|-----|-----|-----|-----|-----|-----|-----
A1 | ok | 6811-6 | 5.092v | 3.205v | 3.006v | 38.16v | 38.15v | 0x00000 | 0x00fff
A2 | ok | 6811-6 | 5.127v | 3.204v | 3.003v | 38.17v | 38.16v | 0x00000 | 0x00fff
A3 | 0x2840 | 6811-6 | 5.043v | 3.214v | 3.004v | 38.21v | 38.20v | 0x00000 | 0x00fff
A4 | ok | 6811-6 | 5.089v | 3.208v | 3.003v | 38.22v | 38.21v | 0x00000 | 0x00fff
```

The Probe and Diagnostic procedure also occur for the A123 MBBs, however the addressing is different and the diagnostics are not as extensive. Discovered MBBs are printed:

```
mbb1 detected
mbb2 detected
mbb3 detected
mbb5 detected
```

The **show mbb** command can be used to retrieve the MBBs that have been discovered. This listing shows the MBB address, the minimum, maximum, and mean cell voltage in the MBB, as well as additional debugging information.

```
mcu> show mbb
mbb| status | min | max | mean | flt_xx | conf | disc
----|-----|-----|-----|-----|-----|-----|-----
1 | OK | 3.119v | 3.142v | 3.133v | in | 0x00000 | 0x01fff
2 | OK | 3.110v | 3.144v | 3.132v | in | 0x00000 | 0x01fff
3 | OK | 3.126v | 3.189v | 3.177v | in | 0x00000 | 0x01fff
5 | OK | 3.126v | 3.189v | 3.177v | in | 0x00000 | 0x01fff
```

Pack, Group, and Cell Numbering

Group and Cell Numbering

By default, the MCU assumes that all cell groups are series connected and assigns cell numbers depending upon the LTC or MBB address. A pack with four 12-cell LTCs will number the cells 1 to 48 by default.

The numbering and the cell values can be shown by the commands **show cmap** (e.g., “show cell map”) and **show cells**.

```
bms> show cmap
ltc|pack|group| cells
----|-----|-----|-----
A1 | 1 | 1 | (c1 -c12) . . . . .
A2 | | 2 | (c13-c24) . . . . .
A3 | | 3 | (c25-c36) . . . . .
A4 | | 4 | (c37-c48) . . . . .
```

```

mcu> sh cells
c1  - 3.070v --      c13 - 3.077v +      c25 - 3.072v --      c37 - 3.077v ++
c2  - 3.073v -      c14 - 3.075v      c26 - 3.075v      c38 - 3.078v ++
c3  - 3.076v +      c15 - 3.077v +      c27 - 3.077v +      c39 - 3.076v +
c4  - 3.073v      c16 - 3.074v      c28 - 3.076v +      c40 - 3.075v
c5  - 3.073v      c17 - 3.076v +      c29 - 3.076v +      c41 - 3.073v
c6  - 3.076v +      c18 - 3.077v +      c30 - 3.074v      c42 - 3.074v
c7  - 3.076v +      c19 - 3.075v      c31 - 3.075v      c43 - 3.077v ++
c8  - 3.075v      c20 - 3.076v +      c32 - 3.074v      c44 - 3.076v
c9  - 3.076v +      c21 - 3.075v      c33 - 3.075v      c45 - 3.074v
c10 - 3.075v      c22 - 3.074v      c34 - 3.074v      c46 - 3.075v
c11 - 3.075v      c23 - 3.076v      c35 - 3.075v      c47 - 3.075v
c12 - 3.072v -      c24 - 3.072v -      c36 - 3.075v      c48 - 3.074v

```

The **show cells** command gives the current cell voltage.

The MCU computes the standard deviation (SD) of all cells in the pack. A cell that measures more than 1.5 SD is flagged with a “++”, more than 1 SD is flagged with a “+”, less than 1 SD is flagged with a “-” and less than 1.5 SD is flagged with a “--”.

It is possible to modify the default numbering. The LTC IsoSPI links may be connected in a different order than the connection order of the cell connections. The figure below shows an example. This is a 44-cell pack where the LTC IsoSPI links are connected to Cell Groups without regard to cell connection order. LTC A4, for example, monitors cells 1 to 12.

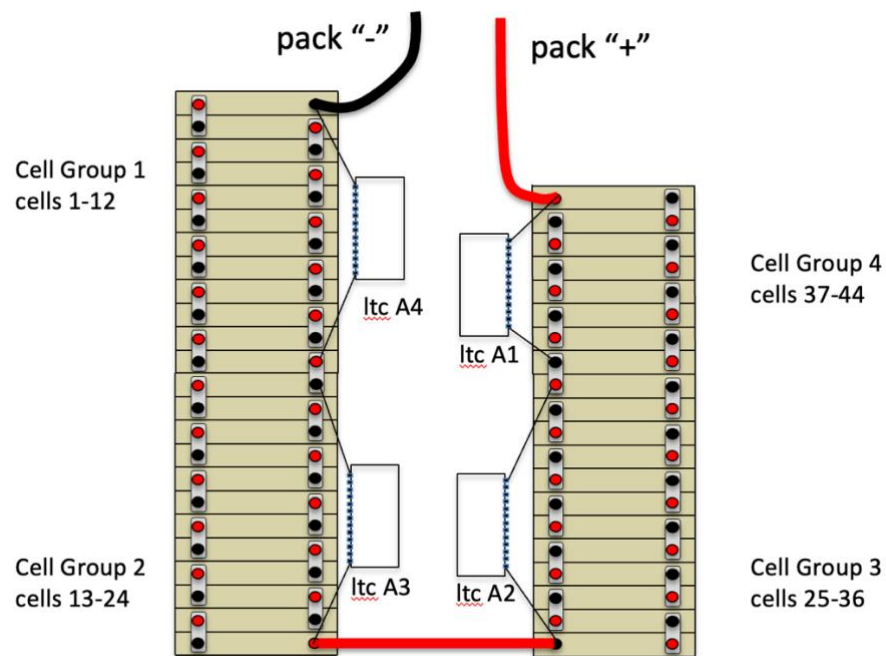


Figure 21 – 44 Cell Pack, Cell Groups Connected Where Convenient

The default numbering for an LTC can be changed using the **set cmap** command. This command takes three arguments: the LTC address, the pack, and the group. The mapping can be changed to renumber the cells with the following commands:

```

bms> set cmap A1 1 4
bms> set cmap A2 1 3
bms> set cmap A3 1 2
bms> set cmap A4 1 1

```

In this example, there is only one pack (pack 1). The results of this mapping can be verified by using the **show cmap** command.

```

bms> show cmap
  ltc|pack|group| cells
-----|-----|-----|-----
A4 | 1 | 1 | (c1 -c12) . . . . .
A3 |  | 2 | (c13-c24) . . . . .
A2 |  | 3 | (c25-c36) . . . . .
A1 |  | 4 | (c37-c48) . . . . .

```

Note that the cell group monitored by ltc A1 now consists of cells c37 to c48.

If the **A123** architecture is used, the default numbering for an MBB can also be changed using the **set cmap** command. By default, the cells are numbered 1 to 13 for mbb1, 14 to 26 in mbb2, and so on. In order to swap these two groups, the following commands can be used:

```

bms> set cmap 1 1 2
bms> set cmap 2 1 1

```

This maps mbb1 to pack 1, group 2 (cells 14 to 26), and mbb2 to pack 1 group 1 (cells 1 to 13).

Pack Numbering

The MCU supports configurations of multiple parallel strings of cells. In the MCU, each parallel string of cells is called a **pack**. This is shown in the figure below.

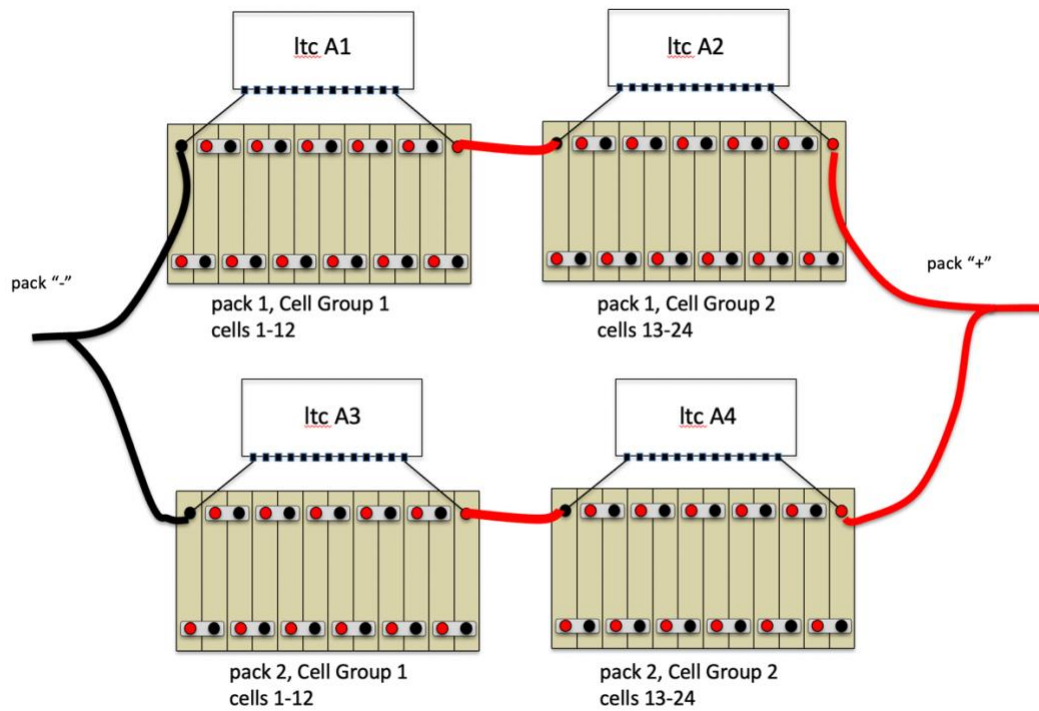


Figure 22 – Parallel Packs

Use the following commands to map LTC A3 to Pack 2, Cell Group 1 and LTC A4 to Pack 2, Cell Group 2:

```
bms> set cmap a3 2 1
bms> set cmap a4 2 2
```

Check the configuration with **show cmap**. Now there are now two packs.

```
bms> show cmap
ltc|pack|group| cells
-----|-----|-----|-----
  1 |  1 |  1 | (c1 -c12) . . . . .
  2 |  1 |  2 | (c13-c24) . . . . .
-----|-----|-----|-----
  3 |  2 |  1 | (c1 -c12) . . . . .
  4 |  2 |  2 | (c13-c24) . . . . .
```

Here is example output of **show cells** with two packs.

```
bms> show cells
pack 1 -----
c1 - 3.646v ++      c13- 3.630v ++
c2 - 3.477v +       c14- 3.476v +
c3 - 3.547v +       c15- 3.553v +
c4 - 3.478v +       c16- 3.468v
c5 - 3.534v +       c17- 3.540v +
c6 - 3.469v         c18- 3.467v
c7 - 3.352v         c19- 3.311v
c8 - 3.234v         c20- 3.223v
c9 - 3.209v         c21- 3.222v
```



```

c10- 3.206v      c22- 3.230v
c11- 3.216v      c23- 3.229v
c12- 3.222v      c24- 3.212v
pack 2 -----
c1 - 3.249v      c13- 3.143v -
c2 - 3.316v      c14- 3.082v -
c3 - 3.352v      c15- 3.311v
c4 - 3.480v +    c16- 3.321v
c5 - 3.052v --   c17- 3.366v
c6 - 3.278v      c18- 3.394v
c7 - 3.199v      c19- 3.424v
c8 - 3.325v      c20- 3.308v
c9 - 3.068v -    c21- 3.315v
c10- 3.370v      c22- 3.082v -
c11- 3.067v -    c23- 3.090v -
c12- 3.282v      c24- 3.075v -

```

Cell Measurement

In operation, all cells in the system are measured approximately 8 times a second. Cell High Voltage Cutoff (HVC) and Low Voltage Cutoff (LVC) condition states are reevaluated every cell scan and pack-based and cell-based statistics are collected.

A cell census is performed to ensure all cells that are supposed to be there (“configured”), really are there (“discovered”). Cells are discovered automatically; cells are configured using the **lock** command.

Cell Census

There are two conditions associated with the cell census: the **NOTLOCKED** condition and the **CCENSUS** condition.

The **show cmap** command gives a list of all cells. Each cell has four possible entries. The following is an example of **show cmap** before the **lock** command has been executed:

```

bms> sh cmap
ltc|pack|group| cells
----|----|-----|-----
A1 | 1 | 1 | (c1 -c12 ) . . . . .
A2 |   | 2 | (c13 -c24 ) . . . . .
A3 |   | 3 | (c25 -c36 ) . . . . .
A4 |   | 4 | (c37 -c48 ) . . . . .

```

The “dots” indicate that the cells are detected but not configured. The **lock** command marks all currently discovered cells as “configured” and clears the **NOTLOCKED** alert.

```

bms> lock
bms> sh cmap
ltc|pack|group| cells
----|----|-----|-----
A1 | 1 | 1 | (c1 -c12 ) X X X X X X X X X X X
A2 |   | 2 | (c13 -c24 ) X X X X X X X X X X X
A3 |   | 3 | (c25 -c36 ) X X X X X X X X X X X
A4 |   | 4 | (c37 -c48 ) X X X X X X X X

```

When the configuration is locked, cells will display with an “X”.

The MCU periodically probes all LTCs and monitors if LTCs or cells arrive or disappear. Suppose, for example, that the harness for LTC A4 is disconnected. The MCU will detect this condition and report the following:

```
ltc A1 detected
ltc A2 detected
ltc A3 detected
ltc A4 NOT DETECTED!
ltc A4 CELL CENSUS ALERT!
```

The MCU will assert the **CCENSUS** alert and the results of **show cmap** will indicate the missing cells.

```
bms> show cmap
ltc|pack|group| cells
-----|-----|-----|-----
A1 | 1 | 1 | (c1 -c12) X X X X X X X X X X X X
A2 |   | 2 | (c13-c24) X X X X X X X X X X X X
A3 |   | 3 | (c25-c36) X X X X X X X X X X X X
A4 |   | 4 | (c37-c48) ? ? ? ? ? ? ? ?
```

If the harness is then modified and two additional cells are connected, when the harness gets reconnected, the output could look like:

```
bms> show cmap
ltc|pack|group| cells
-----|-----|-----|-----
A1 | 1 | 1 | (c1 -c12) X X X X X X X X X X X X
A2 |   | 2 | (c13-c24) X X X X X X X X X X X X
A3 |   | 3 | (c25-c36) X X X X X X X X X X X X
A4 |   | 4 | (c37-c48) X X X X X X X X . .
```

Although the configuration has been locked already, it can be locked again to record this new configuration.

```
bms> lock
bms> show cmap
ltc|pack|group| cells
-----|-----|-----|-----
A1 | 1 | 1 | (c1 -c12) X X X X X X X X X X X X
A2 |   | 2 | (c13-c24) X X X X X X X X X X X X
A3 |   | 3 | (c25-c36) X X X X X X X X X X X X
A4 |   | 4 | (c37-c48) X X X X X X X X X X
```

HVC and LVC Conditions

The High Voltage Cutoff (HVC) and Low Voltage Cutoff (LVC) thresholds are configurable, using the parameters **hvc** and **lvc**. In general, when any cell voltage exceeds **hvc** or falls lower than **lvc**, an HVC or LVC alert is generated, and the associated Output Functions **hvc** and **lvc** are set TRUE. Alerts can be configured to operate an external relay or indicator to notify the driver or disable the EV by mapping the appropriate Output Function to an Output.

Reporting these alerts can be modified by the optional parameters **hvcdelay** and **lvcdelay**. These parameters, given in seconds, allow for a delay in reporting the HVC or LVC alert. The **lvcdelay** parameter can be used to minimize LVC alarms under a short period of high current draw.

The logic necessary to clear these conditions can be modified by the optional parameters **hvcc** and **lvcc**. Using these parameters, the HVC or LVC alerts are reported normally, but the alert is not cleared until the cell voltage drops below **hvcc** or rises above **lvcc**. The **hvcc** parameter would typically be a bit less than the **hvc** setting; the parameter allows hysteresis in reporting. (Example usage: In a solar application, the charger may always be connected to the pack and the HVC alert is used to stop the charger. Without hysteresis, the charger will cycle ON and OFF as a cell voltage exceeds **hvc** only to drop back quickly once charging current stops).

When an HVC condition is reported or cleared, a message will be sent to the serial port. This information includes the pack and cell number:

```
1/c25: in HVC !
1/c26: in HVC !
1/c25: OK
1/c26: OK
1/c12: in LVC !
1/c11: in LVC !
```

HVC and LVC conditions are available in the output to **show cells**. In this example, **hvc** is set to 4.1 and **lvcc** is set to 3.0.

```
bms> show cells
c1  - 4.059v      c13 - 4.060v      c25 - 4.066v      c37 - 4.074v
c2  - 4.062v      c14 - 4.062v      c26 - 4.071v      c38 - 4.077v
c3  - 4.063v      c15 - 4.064v      c27 - 4.070v      c39 - 4.079v
c4  - 4.058v      c16 - 4.062v      c28 - 4.071v      c40 - 4.072v
c5  - 4.169v >HVC c17 - 4.060v      c29 - 2.970v <LVC c41 - 4.074v
c6  - 4.055v      c18 - 4.056v      c30 - 4.067v      c42 - 4.075v
c7  - 4.061v      c19 - 4.064v      c31 - 4.070v      c43 - 4.075v
c8  - 4.058v      c20 - 4.059v      c32 - 4.203v >HVC c44 - 4.076v
c9  - 4.059v      c21 - 4.061v      c33 - 4.070v      c45 - 4.076v
c10 - 4.062v      c22 - 4.063v      c34 - 4.073v      c46 - 4.073v
c11 - 4.058v      c23 - 4.061v      c35 - 4.072v      c47 - 4.075v
c12 - 4.055v      c24 - 4.058v      c36 - 4.072v      c48 - 4.066v
```

The HVC and LVC alerts are available at the mcu **show** command, are visible as LED blinking, and they are available as Output Functions.

HIPACK and LOWPACK Conditions

The **hipack** condition is true if the total pack voltage exceeds the configured value of **hipackv**.

The **lowpack** condition is true if the total pack voltage is less than the configured value of **lowpackv**.

Pack Statistics

Mean and Standard Deviation are recalculated after every cell scan and are available at the **bms show** command.

```
bms> show
pack
  voltage : 228.72v
  cells   : 72
  mean    : 3.035v
  std dev : 0.364v
  alerts   : none
bms>
```

Cell Statistics

Standard Deviation is a measure of how well balanced the cells are ... the lower the better.

For each cell, the Standard Deviation Multiple (SDM) is calculated. If the SDM is 1, then the cell voltage is “one standard deviation from the mean”. SDM is stored as a number from -9.9 to +9.9.

Example:

```
if
    SD          = 0.020v    // value of standard deviation
    ave         = 3.800v    // the average cell voltage
    c8          = 3.820v    // current voltage of cell c8
    c9          = 3.790v    // current voltage of cell c9
then
    SDM(c8)     = 1.0       // cell c8 is ave + 1.0*SD
    SDM(c9)     = -0.5      // cell c9 is ave - 0.5*SD
```

Positive values of SDM mean the cell voltage is higher than average, and negative values of SDM means the cell voltage is lower than average. The **show cells** command flags cells that are ‘especially high’ (>1.5 SD) with “++”, ‘high’ (>1 SD) with “+”, ‘especially low’ (<1.5 SD) with “--“, and ‘low’ (< 1 SD) with “-“.

```
bms> show cells
c1 - 3.646v ++    c13- 3.630v ++
c2 - 3.477v +     c14- 3.476v +
c3 - 3.547v +     c15- 3.553v +
c4 - 3.478v +     c16- 3.468v
c5 - 3.534v +     c17- 3.540v +
c6 - 3.469v       c18- 3.467v
c7 - 3.352v       c19- 3.311v
c8 - 3.234v       c20- 3.223v
c9 - 3.209v       c21- 3.222v
c10- 3.206v       c22- 3.230v
c11- 3.216v       c23- 3.229v
c12- 3.222v       c24- 3.212v
```

The BMS records “low watermark” and “high watermark” cell values (e.g., the lowest and highest values for the cell), as well as the minimum and maximum values of SDM. The

difference between the maximum and minimum SDM (the “delta”) can give an idea of cell performance under load: the larger the “delta” the more nonlinear the cell.

Cell statistics are displayed with the command **show stats**. The statistics are cleared on power-up, however they are also cleared if the Probe discovers a change in how many cell groups are present and can be reset using the command **reset stats**.

```
bms> show stats
total|-mean cell voltage-----|-standard deviation-----
      | 3.414v                      | 0.261v
pack1|-voltage---min---max--|---deviation---min---max--delta-
c1   | 3.447v   3.392v 3.520v | 0.033v +0.1s +0.1s +0.1s 0.0s
c2   | 3.400v   3.345v 3.471v | -0.014v +0.0s +0.0s +0.0s 0.0s
c3   | 3.473v   3.417v 3.546v | 0.059v +0.2s +0.2s +0.2s 0.0s
c4   | 3.414v   3.359v 3.486v | 0.000v +0.0s +0.0s +0.0s 0.0s
c5   | 3.463v   3.407v 3.536v | 0.049v +0.1s +0.1s +0.1s 0.0s

(etc.)
```

Cell Balancing

Automatic cell balancing by discharging is enabled using the command **enable balance**. If enabled, cell discharging runs whenever the MCU is powered up, whether the EV is being driven or being charged. A minimum cell balancing voltage **bvmin** can be configured: a cell with a voltage lower than this value will not be balanced.

Cell Balancing – LTC12 and LTC18

For LTC devices, the MCU determines one cell from each group of six cells to balance. Once a cell has been chosen for discharge, its associated shunting resistor is enabled. BMSS24 and BMSS18 hardware supports 24 ohm, 1W balancing resistors, which will shunt about 170ma at 4V. The best candidate for balancing will have a cell voltage that is:

- the highest voltage of any other cell in its six cell group,
- higher than the configured **bvmin** value, and
- higher than the minimum cell voltage + 2.5mv.

Approximately once a minute, discharge candidates are reevaluated, new discharge candidates are determined, and then charge shunting is resumed.

Cell Balancing – A123

A123 MBBs contain integrated balancing hardware. A123 MBBs will balance any cell with a voltage higher than a specified balance target. A balance target is calculated by the MCU that is:

- higher than the configured **bvmin** value, and
- higher than the minimum cell voltage + 3.0mv.

Thermistors

The BMSS24 has two LTC devices. Each LTC has on an on-die thermistor and support for five external thermistors. There are two 10-pin thermistor connections on the BMSS24, one for each LTC. The thermistor numbering from 1 to 5 runs right to left for both connectors.

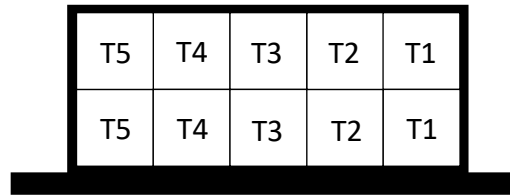


Figure 23 – BMSS24 Thermistor Connectors

The BMSS18 has a single LTC device. It supports an on-die thermistor and nine external thermistors. Two ten-pin thermistor connectors are present: the pinout of the thermistor connector for thermistors 1 through 5 is identical to the figure above. For thermistors 6 through 9, the same connector is used with one pair left disconnected.

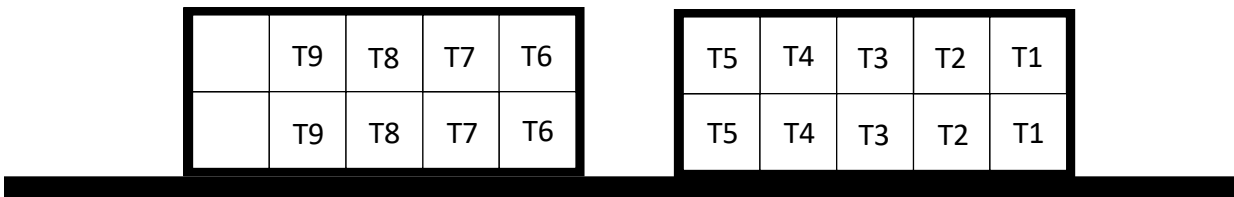


Figure 24 – BMSS18 Thermistor Connectors

The thermistor connector is a 10p Molex 87833-1031. The matching connector is a Molex 51110-1060 with crimp terminal 50394-8054. A harness with pigtail wire connections is available. This can be soldered to a wire lead thermistor or connected directly to the thermistor connection on an OEM battery pack.

The MCU has been designed to work with a 1% 10K NTC thermistor, with a “B value” of 3435K. One possible choice is the Vishay NTCLE413E2103F520.

By default, thermistor operation is disabled and no thermistor alerts will be reported.

The **show thermistor** command shows thermistor status. The following example shows four LTC devices and their thermistor readings. In this example, die thermistor readings are shown for all LTCs, but only LTC A2 has external thermistors connected. The readings all show in parenthesis because the thermistors have been detected but not enabled.

```
sys> sh th
ltc| die | T1 | T2 | T3 | T4 | T5
-----|-----
A1 | (30C)
A2 | (30C) (24C) (24C) (23C) (25C)
A3 | (39C)
A4 | (30C)
```

The **enable thermistor all** command will enable all discovered thermistors. There are also commands to enable all die thermistors “**en th die**”, all external thermistors “**en th ext**”,

and all thermistors on a particular device “**en th A1**”, as well as commands to enable and disable individual thermistors (e.g., “**en th A1 T2**”).

```
bms> enable thermistor all
bms> sh th
ltc|  T0  |  T1  |  T2  |  T3  |  T4  |  T5
-----|-----
A1 |  25C
A2 |  22C          16C    16C    15C    16C
A3 |  22C
A4 |  22C
```

A thermistor that is “enabled” will report alerts. There are three alerts associated with thermistors: **TCENSUS**, **HITEMP**, and **LOWTEMP**.

If an enabled thermistor is disconnected and not detected, this will result in serial output and in the **TCENSUS** alert.

```
mcu> ltc A2:T2 NOT DETECTED!
ltc A2:T3 NOT DETECTED!
ltc A2:T4 NOT DETECTED!
ltc A2:T5 NOT DETECTED!
```

HITEMP and LOWTEMP Conditions

The thermistor maximum and minimum threshold temperatures are set by the configuration parameters **thmax** and **thmin**.

If a thermistor is enabled and if the measured thermistor temperature is greater than **thmax**, the **HITEMP** condition is TRUE and a message is printed. The **HITEMP** condition can also be used to control an output which could enable a fan.

If, for example, **thmax** is set to 40, and the thermistor temperature reaches 41C, the following message will be printed:

```
ltc A1:T2 41C, HITEMP!
```

If a thermistor is enabled and if the measured thermistor temperature is lower than **thmin**, the **LOWTEMP** condition is TRUE and a message is printed.

ONBOARD CHARGING

With onboard charging, one or more chargers are installed in the EV and AC Line Power is connected to the chargers using an EV charge plug.

The MCU supports the J1772 Type 1 and J1772 Type 2 charge protocols. These protocols define the functions of both the Electric Vehicle Supply Equipment (EVSE) and the Electric Vehicle (EV). J1772 Type 1 is typically used in North America and supports single phase 110V AC and 220V AC line power. J1772 Type 2, typically used in Europe, supports one or three-phase 220V

AC line power. J1772 Type1 and J1772 Type2 require different charge plugs but there is otherwise a large overlap between the two protocols.

The MCU also supports a “direct plug-in” option. This option allows the user to directly connect line power to the charger without using a J1772 EVSE. The three options (**J1772**, **J1772T2**, and **direct**) are mutually exclusive and are set by using the **set plug** command.

The MCU also supports onboard charging using the Zero EV CCS; this case is handled separately: see *ZEV CCS Charging*, below.

The MCU monitors for charge plug insertion to begin a charge, and once charging has started, the MCU controls the chargers, typically by using CAN messages.

The MCU has a sleep and autostart feature that uses the **KSI** and **Proximity** inputs. If the MCU is not charging, the MCU will enter a “sleep” mode if **KSI** is not present. The **Proximity** input has a threshold detector that can wake the MCU. In the case of J1772 and J1772T2 charging, plugging in the charge cable will trigger the proximity detector and wake up the MCU.

The sleep and autostart functions can also be used for non J1772/J1772T2 charging applications. In this case, **KSI** is wired to the Key Switch as before. The autostart function can be initiated by applying a ground to the **Proximity** input in order to wake the MCU and start the charge. Similarly, releasing the ground on **Proximity** will stop the charge.

The sleep and autostart functions can be disabled by connecting the MCU **KSI** and **12V** inputs together. In this case, the MCU can be powered up and down by providing or removing 12V power.

The **KSI** input has an associated condition, also called **KSI**.

“Time of use” charging is supported, in which case, **Proximity** charge plug detection will wake the MCU, but the MCU can be configured to wait several hours for the EVSE to become active.

Charge voltage, charge current, and maximum charge time are configured in the MCU and used during charging to control the chargers.

The MCU supports several CAN-enabled chargers, including the Thunderstruck TSM2500 and Elcon chargers. The MCU supports up to four parallel chargers for faster charging. When multiple chargers are configured, each charger is individually CAN addressed. Work is divided evenly between the chargers and statistics are gathered and recorded on each charger separately.

The MCU supports three step charging: a Bulk charge, a Finishing charge, and a Float charge. The Bulk charge phase is used by all battery cell chemistries. The Finishing charge is optional and typically used by Lead Acid batteries. The Float (or “trickle”) charge, also optional, is used to maintain pack charge for an extended period of storage.

It is the responsibility of the user to configure charge parameter values appropriate for the pack.

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

Up to four “charge profiles” can be created for onboard charging: each profile contains a separate copy of all necessary parameters for charging. In operation, a “profile selection” switch can determine which profile is used.

Charging normally completes when the charging current drops below a minimum configured value. The MCU will also stop charging if an error condition occurs, such as: the charge plug becomes unplugged, a cell High Voltage Cutoff error occurs, CAN communication is lost between the MCU and Charger, or the maximum configured charge time is reached.

There are two Output Functions relevant to charging: the **charge** output is TRUE when charging, and the **plugin** output is TRUE when the charge plug is plugged-in.

The serial interface has commands to monitor charging progress and retrieve charge history.

Charge Plugs and Line Powering

J1772 Type 1

SAE J1772 defines the physical connector and protocols used between the charging station or “Electric Vehicle Supply Equipment” (EVSE), and the Electric Vehicle (EV). This standard is widely used and provides single phase 220V 30-50A or 110V at 15A to an EV onboard charger.

The system diagram, below, shows the main components of a J1772 Type 1 charging system, including the charge plug, the charger, the EV pack, and the MCU. Also shown are the CAN bus used to communicate with the charger and connections to BMS measurement modules.

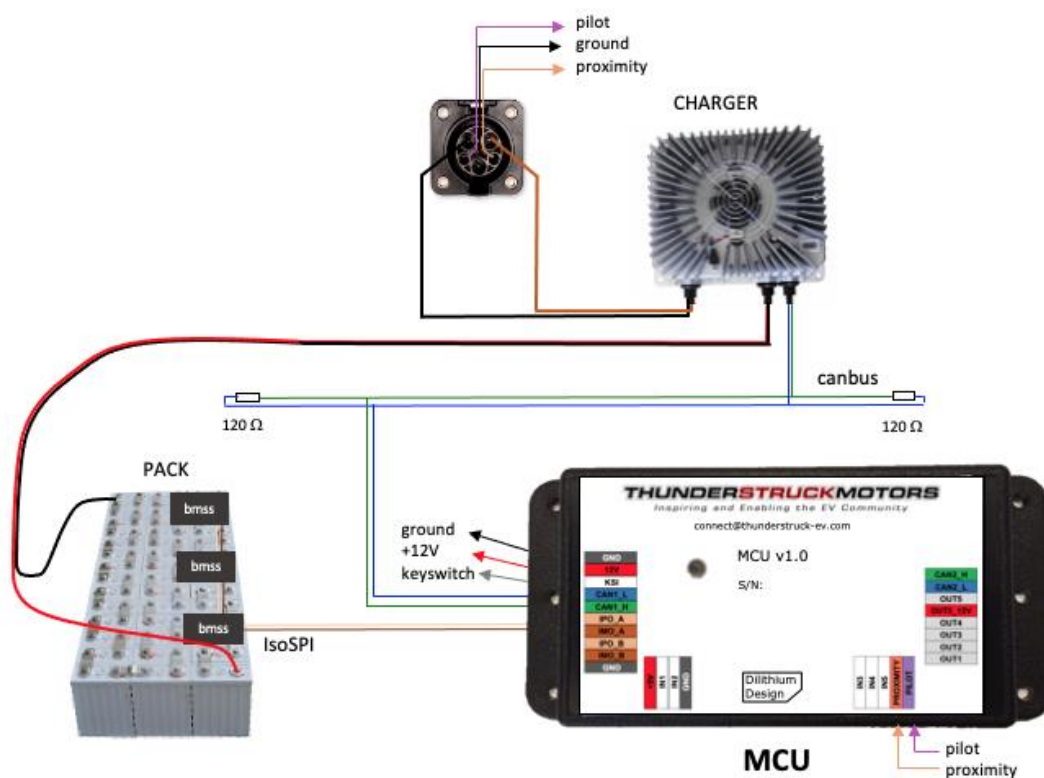


Figure 25 – J1772 Type 1 System Diagram

The J1772 Type 1 connector has five conductors: three conductors for line power, a **Proximity** signal, and a **Pilot** signal. **Proximity** is used to determine if the charger plug is connected and is wired directly to the proximity input on the MCU. **Pilot** has a dual function: it is used by the EVSE to advertise how much charge current is available, and it is used by the EV to enable or disable charging. It is wired directly to the Pilot input on the MCU.

The following diagram shows the connections to the J1772 Type 1 EV connector:

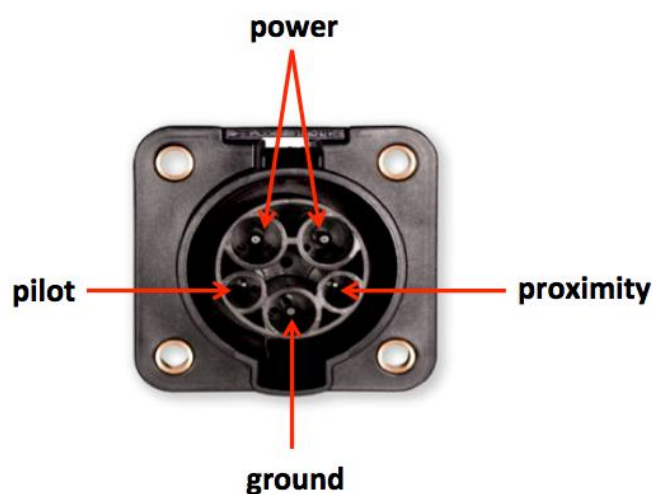


Figure 26 – J1772 Type 1 EV Connector

The J1772 Type 1 charge plug is a latching plug with a charger release button. Resistors in the charge plug are connected to the charger release button, to **Proximity**, and to **Ground**. The resistance between **Proximity** and **Ground** allows the EV to determine if the plug is “disconnected”, “connected”, or “locked”. Expected readings are:

Proximity	Plug State
Inf	disconnected
480 Ω	connected
150 Ω	locked

The **Proximity** signal provides important safety features. Should the plug become unlocked while charging, the MCU will detect this and will immediately stop charging. This helps prevent arcing at the contacts if the plug is suddenly removed. The MCU supports the Output Function (**plugin**) that indicates when the plug is connected; this signal can be used to prevent the EV from being driven if the charge plug is connected.

The **Pilot** signal allows the EVSE to advertise how much line current is available for charging. The EVSE indicates this by placing a 1KHz square wave on the Pilot signal; the duty cycle of the square wave indicates the available charging current. If the duty cycle is $\leq 50\%$, the line current in amps is the duty cycle times 6/10. As an example: a duty cycle of 50% advertises 30A of line current.

The **Pilot** signal also allows the EVSE to detect the presence of the EV and allows the EV to enable and disable charging. The EV provides a series diode and 2.7K resistor to ground and bridges an additional 1.3K resistor to ground to enable charging.

IMPORTANT: When wiring the J1772 signals, ensure that there is a good ground connection between the EV Chassis, J1772 Ground, and MCU GND. A poor ground connection can result in erratic operation.

For more information on J1772 see http://en.wikipedia.org/wiki/SAE_J1772

J1772 Type 2

The International Electrotechnical Commission (IEC) has adopted and extended the SAE J1772 standard for use internationally. In its IEC 62196 standard, “J1772 Type 1” and J1772 Type 2” protocols are defined. J1772 Type 1 uses the same plug and signaling protocols as SAE J1772.

J1772 Type 2 uses a different plug which allows the use of either one- or three-phase power, common in Europe. If only one-phase power is available, a simple adapter cable can be used that mechanically converts from one connector to the other. The J1772 Type 2 connector is commonly known as the “Mennekes” connector. The figure below shows the J1772 Type 2 connector.



Figure 27 – J1772 Type 2 Connector

The following table lists the J1772 Type 2 signals and the correspondence to J1772 Type 1 signals.

J1772 T2 Abbreviation	J1772 T2 Signal	J1772 Signal
PP	Plug Present	Proximity
CP	Control Pilot	Pilot
PE	Protective Earth	Ground
N	Neutral	Power
L1	Phase 1	Power
L2	Phase 2	n/a
L3	Phase 3	n/a

The following diagram shows a J1772 Type 2 connector wired with a charger on each phase.

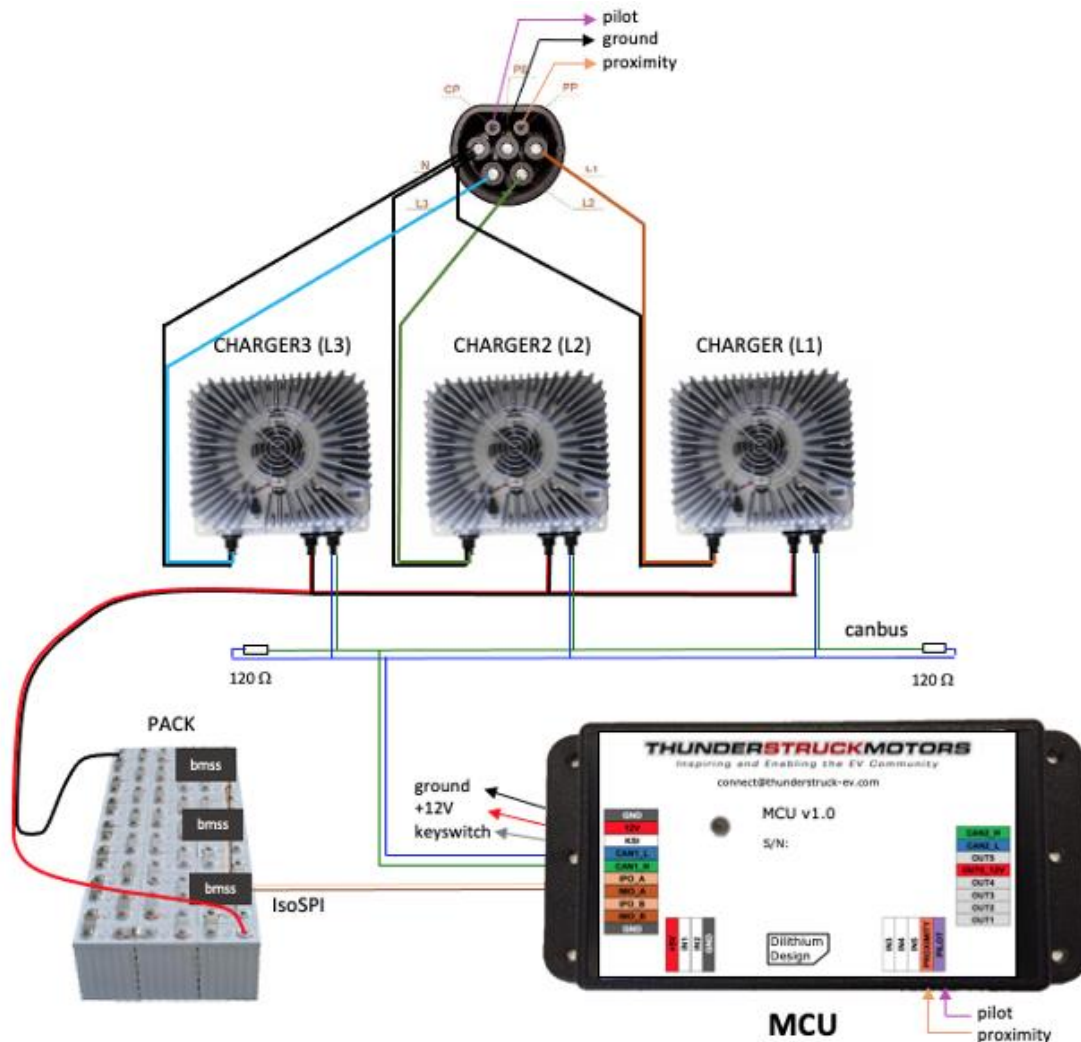


Figure 28 – J1772 Type 2 - One Charger Per Phase

J1772 Type 2 has some minor differences to the J1772 Type 1 protocol. J1772 Type 2 supports a removable charge cable without a pushbutton lock: the **proximity** signal only supports “connected” and “disconnected” (no “locked”). And, since charge cables can be rated at different line currents; the allowable values of the proximity resistance have been extended in order to communicate the charge cable ampacity to the EV.

The J1772 Type 2 **Proximity** resistor values are:

Proximity	Cable Rating
1.5 K	13A
680 Ω	20A
220 Ω	32A
100 Ω	63A

The J1772 Type 2 **Pilot** protocol works similarly to J1772 Type 1. As with Type 1, if the duty cycle is $\leq 50\%$, the line current in amps (per phase) is calculated by the same formula (amps =

duty cycle times 6/10). For duty cycles larger than 50%, the J1772 Type 2 protocol has extensions in in order to handle higher currents.

Direct Plug

In a solar power or marine application, J1772 charging may not be available or appropriate. The charge plug parameter may be set to “direct” (**set plug direct**). With this setting the MCU will not perform the J1772 pilot protocol.

In the most barebones implementation, the **Proximity** input is left disconnected. In this case, autostart is not available, and the wiring must connect the KSI input to the 12V input power. Charging current must be manually entered (**maxc**). Since there is no notion of charge plug presence, a charge cycle begins when CAN messages start arriving from the onboard charger. This would be the case, for example, if the onboard charger were plugged into line power. When a charge cycle completes, the MCU will stop sending CAN messages to the charger and will require that the Charger stop sending CAN messages for a “warmdown period” before it considers the charge cycle complete and before it will allow a new charge cycle to begin. This would occur, for example, if the onboard charger were unplugged from line power.

Alternately, the sleep/autostart and charge plug presence functions can still be used with the **direct** option. In this case, **KSI** is wired to the Key Switch as before. The autostart function can be initiated by applying a ground the **Proximity** input which will wake the MCU and start the charge. Similarly, releasing the ground on **Proximity** will stop the charge.

ZEVCSS Onboard Charging

When the ZEVCSS option is enabled, different mechanisms are used to implement autostart, determine line ampacity, and determine charge plug presence. See the Section *ZEVCSS Charging*.

Charge Plug State

In all, the charge plug is in one of five “states”: **disconnected**, **connected**, **locked**, **active**, and **waiting for disconnect**.

In the **disconnected** state, the MCU is prepared to start a charge. The **connected** state means that the charge plug has been inserted. In the case of J1772, there is an additional **locked** state. The **active** state is used in the direct plug option and means that CAN messages are being received from the charger.

Once a charge has completed, the plug enters the **waiting for disconnect** state. This state is used to ensure that the plug be completely disconnected (or that charger CAN messages stop) before the MCU will start a new charge session.

The valid states and state transitions depend on the configuration of the charge plug.

- J1772 uses: **disconnected**, **connected**, **locked**, and **waiting for disconnect**.
- J1772T2 uses: **disconnected**, **connected**, and **waiting for disconnect**.
- Direct Plug with proximity uses: **disconnected**, **connected**, and **waiting for disconnect**.

- Direct Plug without proximity uses: **disconnected**, **active**, and **waiting for disconnect**.

Charging

Output Functions

The following Conditions are relevant to onboard charging:

- **power** on when power is on.
- **charge** on when charging
- **plugin** on when the charge plug is plugged in.

The MCU has a supervisory circuit which monitors for a change in the **Proximity** resistor value. It will wake the MCU if it detects a high to low transition of proximity resistance indicating the presence of the charge cable.

When the MCU powers on or wakes up, it enables the **power** Logical Output. The MCU checks if the charge plug is connected, and if so, the MCU will attempt a new charge cycle. When a charge completes, the MCU will remain powered if the **KSI** input is on, otherwise it will disable the **power** Logical Output and go to sleep.

Creating a **power** Output Function and mapping it to OUT5 allows the MCU to switch up to 1.5A of 12V power to external equipment when it is powered on.

Target Current

Before CAN messages can be sent to the chargers, the MCU must determine Target Charging Voltage and Target Charging Current. For the bulk charge phase, Target Charging Voltage is determined by the configuration parameter **maxv**.

Target Charging Current might be directly set by using the parameter **maxc**. Alternately, Target Charging Current might be calculated from line current. In turn, Line current may be directly configured (by setting the parameter **linec** to a value specified in amps) or if **linec** is configured to the value **J1772**, the J1772 duty cycle measurement determines line current. (If plug is set to **J1772T2**, a modified calculation is used and calculated line current may be reduced based on the ampacity of the charge cable, as determined by the **Proximity** reading). Once available line current is determined, Target Charging Current is determined based on a power calculation which converts Line Watts to Charge Watts. To do this calculation, it is necessary to know line voltage, line current, and target voltage. If **linev** is configured, then it is used for Line Voltage. If **linev** NOT provided, a heuristic is used³.

If both **linec** and **maxc** are provided, and there will be two ways to determine Target Current. In this case, the MCU will choose the minimum of the two.

³ If the J1772 duty cycle is less than 20% then the MCU assumes the line voltage is 110V, otherwise the MCU assumes a line voltage of 220. This is driven by commonly available EVSE equipment used in the North American market.

The Pseudo code, below, captures the algorithm used

```
// best guess for line current
if linec is defined
    if linec is J1772
        LineCur = DutyCycle * (6/10)
    else if linec is specified in amps
        LineCur = linec

// determine line voltage
if linev is defined
    LineVol = linev
else if (DutyCycle > 25)
    LineVol = 220
else
    LineVol = 110

// calculate target current using a power calculation
TC = (LineVol * LineCur * .9) / maxv

if maxc is defined
    TargetCurrent = min(maxc, TC)
else
    TargetCurrent = TC
else
    TargetCurrent = maxc
```

If multiple chargers are defined, then available Target Current is divided among the chargers on each phase to ensure that the calculated line current limit is not exceeded.

J1772 Startup

When the charge plug is first inserted to start a J1772 Charge, several events occur:

The MCU:

1. If the MCU is in sleep mode, the the plug in of the charge cable will cause the MCU to wake.
2. The MCU determines the presence of the EVSE by measuring the the **Proximity** input. It then connects the **Pilot** “charge enable” resistor and starts a **Pilot** duty cycle measurement.
3. Once the MCU completes the duty cycle measurement, it then determines Charge Voltage and Charge Current and begins sending CAN messages to the chargers to begin charging.

The EVSE

1. The EVSE detects MCU presence using the **Proximity** signal. The EVSE then starts sending a square wave on the **Pilot** signal to advertise its ampacity.
2. When the EVSE detects the **Pilot** “charge enable” resistor, it will close its contactors allowing line current to flow to the EV Chargers.

When the EV Chargers receive line power, they will power up and begin sending CAN messages.

The EVSE has a delay before the **Pilot** square wave starts and the chargers have a delay before powering up. Still, this entire sequence of events typically happens in 3 to 5 seconds. If the startup sequence does not complete quickly enough, the MCU will terminate the charge session.

Slowstart Option

Some public EVSEs have been found to have several tens of seconds of startup delay and so the default startup time can be extended (to up to 45 seconds) with the **SLOWSTART** option.

J1772 Time Of Use Charging

Time of Use charging is supported. In this case, the charge plug is inserted into the EV several hours before charging is to begin. Upon charge plug insertion, the MCU will start as above. The EVSE then may perform the initial startup sequence but “decide” to delay charging until some later time. The EVSE will suspend the charge and then wait for the low-rate period to begin. When the time comes, the EVSE attempts to restart the charging.

The MCU supports this operation with the configuration parameter **evsewait**, specified in hours. If this parameter is set, the MCU will autostart, will “forgive” a short, suspended, charge such as described above, but will remain powered up and will wait for the charge to start later.

Direct Plug Startup

With this option, grounding **Proximity** starts the MCU. Target Current is specified by **maxc**. Releasing the ground on **Proximity** is the equivalent to charge plug removal.

Bulk Charge

Onboard charging begins in the Bulk Charge phase. Typically, two types of CAN messages are used for charging. The first, sent to the charger, provides the charger with Target Voltage and Target Current, and the second message, sent from the charger, reports actual Pack Voltage and Charge Current. These messages are usually sent twice a second, both from MCU to Charger and from Charger to MCU. If there are multiple chargers, each charger must have a unique CAN address and charge statistics are collected on each charger. If the MCU stops receiving periodic messages from the Chargers, charging will terminate with a termination reason of **CHARGER_RX_TIMEOUT**.⁴

Onboard charging can terminate for several other reasons, given in the following table:

⁴ The timeout is only enforced for the first charger. If multiple chargers (e.g., **charger2**, **charger3** or **charger4**) are defined, their participation is optional and a CAN message timeout does not stop the charge.

Termination Reason	Description
NORMAL	Normal Charge completion
HWFAULT	Hardware Fault Alert
CCENSUS	Cell Census Alert
TCENSUS	Thermistor Census Alert
HVC	HVC Alert
LVC	LVC Alert
HITEMP	Thermistor Overtemperature Alert
LOWTEMP	Thermistor Undertemperature Alert
NOTLOCKED	Configuration Not Locked Alert
PLUG_DISCONNECTED	Charge Plug became disconnected
CELL_LOOP	Cell Loop not OK
EVSE_TIMEOUT	Timeout waiting for Pilot
CHARGER_RX_TIMEOUT	Charger CAN message timeout
NO_PACK	No pack connected
CHARGE_TIMEOUT	Charge timeout

Figure 29 – Onboard Charge Termination Reasons

In all, the bulk charge phase is controlled by the configuration parameters: **linev**, **linec**, **maxv**, **maxc**, **termc** and **termt**. The charge cycle normally completes when the charge current drops below the configured minimum charge current, set by the configuration parameter **termc**.

Finishing Charge

Lead Acid Batteries require a multi-stage charging algorithm. After the bulk charge phase a “finishing charge” phase may be defined. The Finishing Charge phase is controlled by the configuration parameters: **fin_maxv**, **fin_maxc** and **fin_termt**. This phase is enabled if **fin_maxv** is set to something other than 0.

The Finishing Charge phase will normally complete when the charge voltage rises above **fin_maxv**. This contrasts with the Bulk Charge phase, which terminates when the charging current drops below **termc**.

For additional information about lead acid battery charging, see https://www.trojanbattery.com/pdf/TrojanBattery_UsersGuide.pdf.

Float Charge

Once 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 can keep them topped up if the EV sits for an extended period of nonuse. However, this phase may apply to other battery chemistries as well.

A Float Charge phase is controlled by the configuration parameters: **flt_maxv**, **flt_maxc**, and **flt_termt**. This phase is enabled **flt_maxv** is set to something other than 0. Note that the value of **flt_termt** may be set to 0, which in this case means “forever”.

Top Balancing

The “topbalance” feature can be used to balance the pack. If this option is enabled, charging proceeds and terminates normally, with one exception: the charge does not stop with a cell HVC

alert. When that occurs, charging is paused. When the **HVC** alert clears, charging resumes and the procedure repeats. The idea is that each such cycle will add more charge to lower cells and the pack gradually comes into balance. Note that if this technique is used, it is recommended to also use the parameter **hvcc** to prevent the charger from cycling on and off rapidly.

To enable this feature, the **TOPBALANCE** option needs to be enabled.

Charge History

Once charging starts, the MCU begins collecting per-charger statistics for the charge session, including Watt Hours delivered.

When the charge stops, a charge history record is written which records charge time, reason for completion, and charge session statistics such as highest voltage delivered and watt hours.

Charge Configuration

This section summarizes onboard charge configuration.

Plug Type

The plug type must be configured:

- **J1772** for J1772 Type 1 charging.
- **J1772T2** for J1772 Type 2 charging.
- **direct** for “direct plug” charging.

Input Functions

The following Input Functions are relevant to onboard charging:

- **pselect** is used for profile selection. If **pselect** is not mapped, then charge profile 1 is always used for onboard charging
- **plugin** is used to indicate that a charge plug is inserted. This input is used for ZEVCCS onboard charging.
- **loop** is used to as a “go/no-go” to indicate that charging is allowed.

The **loop** Input Function allows external circuitry to control the charge process. This input function may be mapped to any input. If **loop** is mapped to an input that measures voltage (IN1 to IN4), then the input must be +5V to be considered “TRUE”. If the **loop** is mapped to IN5 (which measures resistance), then it must be 0R to be considered “TRUE”.

Options

The following options are relevant to onboard charging:

- **SLOWSTART** – This option allows additional time for the EVSE to begin charging
- **TOPBALANCE** – This option allows the “topbalance” feature. See above.

Chargers

The MCU supports up to four chargers (named: **charger**, **charger2**, **charger3**, and **charger4**). Chargers are defined using the **set charger** command. The charger type indicates both the charger type 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
  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
  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 charger to “none”.

```
evcc> set charger2 none
```

With **J1772 Type 2**, three-phase power can be used and chargers may be associated with a phase: one of L1, L2 or L3. Here is an example:

```
evcc> set charger tsm2500 L1
evcc> show config
  charger : tsm2500    L1
  maxv    : 158.0V
  maxc    : 12.0A
  termc   : 0.5A
  termt   : 720.0hr
evcc>
```

Charge Profiles

Up to four charging profiles may be defined, numbered from 1 to 4. Each profile contains a copy of all charging parameters. By default, Profile 1 is created and cannot be deleted. Only one profile may be edited at a time: a profile is chosen for editing by using the command **edit profile <n>**, where n is from 1 to 4. This command will create the profile if it does not exist; and subsequent edits to profile parameters will affect that profile. A profile may be deleted by using the command **delete profile <n>**.

In the following example, there is one profile to start. Profile 2 is created using the command **edit profile 2**. Note that the **show config** output changes if there is more than one

profile, and that the profile that is currently being edited is marked with a (*). Also, note that **linev** and **linec**, normally optional parameters, are displayed once they are defined.

```
evcc> sh config
--charging-----
  charger   : tsm2500
  maxv      : 20.0V
  maxc      : 0.0A
  termc     : 2.0A
  termt     : 72.0hr
evcc>
evcc> edit profile 2
evcc> sh config
--charging-----
  profile   : 1          2 (*)          3          4
  charger   : tsm2500    tsm2500
  maxv      : 20.0V      20.0V
  maxc      : 0.0A       2.0A
  termc     : 2.0A       2.0A
  termt     : 72.0hr     72.0hr
evcc>
evcc> set linev 220
evcc> set linec 25.5
evcc> sh config
--charging-----
  profile   : 1          2 (*)          3          4
  charger   : tsm2500    tsm2500
  linev     :           220.0V
  linec     :           25.5A
  maxv      : 20.0V      20.0V
  maxc      : 0.0A       2.0A
  termc     : 2.0A       2.0A
  termt     : 72.0hr     72.0hr
evcc>
```

In order to use a charge profile when charging, the **pselect** Input Functions must be mapped to an Input. When charging, **pselect** is used to select the profile.

The MCU measures the resistance to GND at this input and determines four possible selections: “inf”, 20K, 5K, and “0”, using the following formula:

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”

If **pselect** input is left unconnected, it will read “open” (or “infinite” resistance), and maps to “inf”. If **pselect** is shorted to ground it will measure “0”. If a resistor is connected between the **pselect** input and ground, the remaining two choices “20K” and “5K” can be selected. The user might connect the input to a switch to GND to enable two profiles, or connect it to a multi-position switch and a resistor network and enable up to four profiles.

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

In the following example, **IN5** is mapped to **pselect**. Note that the output of **show config** changes to show the **pselect** mappings; all **pselect** mappings are set to choose profile 1.

```
evcc> set in5 pselect
in5      : pselect
evcc> sh config
--charging-----
profile  :      1          2 (*)          3          4
charger  : tsm2500      tsm2500
linev    :          220.0V
linec    :          25.5A
maxv     :      20.0V      20.0V
maxc     :      0.0A       2.0A
termc    :      2.0A       2.0A
termt    :      72.0hr     72.0hr
pmap     :
  inf     :      x
  20K     :      x
  5K      :      x
  0       :      x
evcc>
```

Then the profile map is edited to make “inf” at **pselect** choose profile 2.

```
evcc> set pmap inf 2
evcc> sh config
--charging-----
profile  :      1          2 (*)          3          4
charger  : tsm2500      tsm2500
linev    :          220.0V
linec    :          25.5A
maxv     :      20.0V      20.0V
maxc     :      0.0A       2.0A
termc    :      2.0A       2.0A
termt    :      72.0hr     72.0hr
pmap     :
  inf     :          x
  20K     :      x
  5K      :      x
  0       :      x
evcc>
```

Charger Support

TSM2500

See *TSM2500 Series High Efficiency Intelligent Charger, ThunderStruck User Manual Ver 1.0.8*.
<http://www.thunderstruck-ev.com/images/companies/1/ThunderStruck-TSM2500-ManualV1.08.pdf>

The CAN connections for chargers purchased in 2020 or later are found on the two-pin connector shown in the *Input Cables* section of the charger manual. CANL is connected with a blue wire and CANH is connected with a green wire.

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

The MCU 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 MCU uses the CAN address 0x18e54024 for messages TO the charger and 0x18eb2440 FROM the charger to the MCU).

The TSM2500 can report the following errors shown in the “trace charger” output:

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

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, ensure that only one charger is powered.

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

To do this, power up the MCU and provide line power to the charger. Then type the following command:

```
mcu> set charger2 tsm2500_42 program
```

The MCU will then print

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

Proceed [Y/N] ?

Enter "y", and the MCU prints

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

and then 5-10 seconds later it prints

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

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

ELCON PFC

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

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

The MCU 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 MCU uses the CAN address 1806e5f4 for messages TO the charger and 18ff50e5 FROM the charger to the MCU).

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

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

LEAR

The MCU supports some Lear chargers. This support is limited to the “control message” to the charger with CAN ID “0x050” and the status message from the charger with CAN ID “0x617”. It has been found that not all Lear chargers use these messages due to different firmware and may not support this message set.

The MCU only defines a single Lear charger type, named “lear”.

Having more than one Lear charger is not supported. Having one Lear charger and another non-Lear charger is not supported. The Lear charger runs at 500kbps.

Debugging

Show

The **show** command gives high level charging status, including charge time, target voltage and target current, and the status of each charger.

```
evcc> sh
state      : CHARGE, CC, CV
plug state: CONNECTED
chg time   : 9 mins
voltage(t): 120.0V
current(t): 14.0A
charger    : tsm2500
status     : 1135 msgs sent; 820 msgs received
voltage    : 118.7V
current    : 12.8A
charge     : 244Wh
evcc>
```

Tracing

The command **trace state** shows real-time information about charge progress, including information about plug state as well as charge state.

```
evcc> trace state
state tracing is now ON
evcc> 00:06:41.5 Plug State: DISCONNECT_WAIT => DISCONNECTED
00:06:46.0 Plug State: DISCONNECTED => CONNECTED
00:06:46.0 Charge State: STANDBY => STARTUP
00:06:46.3 Charge State: STARTUP => CHARGE, Charge Type: J1772, CurCalc: MAXC
00:07:21.1 Plug State: CONNECTED => DISCONNECT_WAIT
00:07:21.2 Charge State: CHARGE => WARMDOWN, term rsn=PLUG DISCONNECTED
00:07:26.2 Charge State: WARMDOWN => STANDBY
00:07:26.2 Plug State: DISCONNECT_WAIT => DISCONNECTED
```

The command **trace charger** decodes charger CAN messages.

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

Charge History

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

```
evcc> show history
```

num	term reason	charge time	charger	watt hours	maximum voltage	maximum current	ending current
last	normal	217 mins	tsm2500	2431Wh	120.0V	12.8A	2.0A
- 1	plug disc	1 mins	tsm2500	14Wh	115.0V	12.8A	12.8A

DC FAST CHARGING

The MCU supports two types of DC Fast Charging (DCFC): *ChargeMo* and *ZEVCCS*. Only one type of DCFC may be enabled at a time. In this document, the terms “offboard charging” and “DC Fast Charging” are used interchangeably.

ChargeMo

ChargeMo operates with charge stations using the CHAdeMO specification. The CHAdeMO vehicle side receptacle is required. This connector provides all connections to the EVSE charger including several 12V power and signaling connections, a CAN interface, and high current DC in the range of 200-500 Volts DC.

The picture below is the CHAdeMO connector for a Nissan Leaf (as viewed from the outside of the EV). The table lists recommended connections necessary from the connector to the MCU and EV. (The color assumes that an OEM Leaf Charge port connector assembly is used).

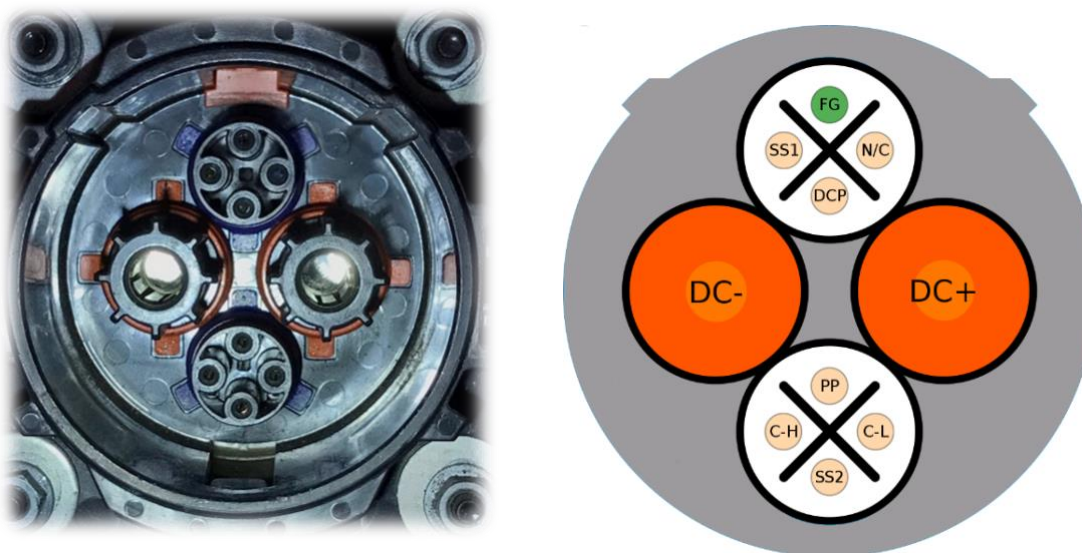


Figure 30 - CHAdeMO Charge Port Connector

Signal	Color (Leaf)	Signal Name	MCU	EV
FG	Black	Ground		EV Ground
DCP	Blue	Charge Permission	OUT2	
SS1	Yellow	Charge Sequence 1	IN3	Trigger Coil +
SS2	Green	Charge Sequence 2	IN4	Trigger Coil -
PP	Brown	Connection Check	PROXIMITY	
C-L	Pink	CAN LO	CAN2 LO	
C-H	White	CAN HI	CAN2 HI	
DC +	Orange	Power +		Contactor 1 to Pack B+
DC -	Orange	Power -		Contactor 2 to Pack B -

Figure 31 – ChargeMo Charge Port Connections

The following is a systems connection diagram.

Notes

- 1) Power wiring gauge must meet current capacity requirements.
- 2) Power wiring is fused at the source based on wire ampacity.
- 3) 12V wiring is fused at the source.
- 4) See Inlet Port Pinout Diagram for connection locations.

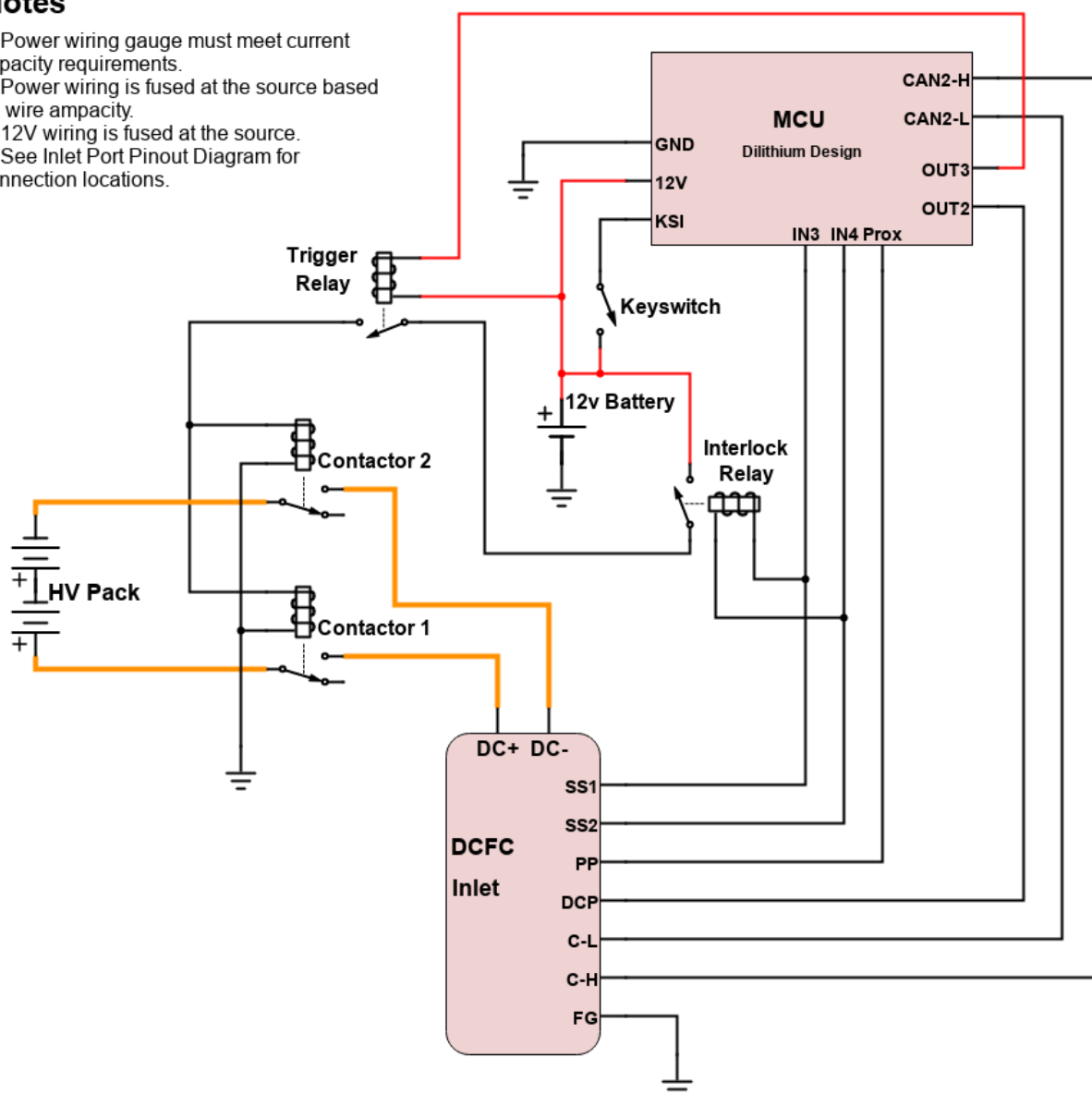


Figure 32 – ChargeMo Onboard Equipment Connections

In addition to the ChargeMo inlet port, two power contactors are needed: Contactor 1 and Contactor 2, which need to be rated to battery voltage and charge current, and use a 12V coil. Two additional 12V relays are needed: a “trigger relay” and an “interlock relay”; these should be rated to 20A.

ChargeMo assumes a pack current sensor and requires that the pack be protected with BMS thermistors.

Installation

Rigid mounting is recommended for the inlet port. High voltage connections should be insulated, and wiring protected from damage and short circuits.

Follow the onboard equipment connection and inlet port pinout diagrams above for system connections. Use a voltmeter to verify connections before testing the system.

NOTE: Properly placed cell thermistors and conservative BMS cell temperature limits are recommended for fast charging so that charging will stop if thermistor readings exceed configured limits.

Cell temperatures generally increase during charging, and there will be a delay between increasing internal cell temperatures and external thermistor readings. It is recommended that conservative values be used, and that actual temperatures be verified during several charging sessions.

Configuring the MCU for ChargeMo

Determining maximum continuous charge Amps, Volts, and temperature for the pack.

Power up the MCU and enter the user interface to configure ChargeMo settings as shown below. This example sets the MCU CAN2 network to 500 kbps and enables ChargeMo on CAN2.

```
mcu> set can2br 500
mcu> enable chargemo can2
```

The following internal input and output maps will be automatically set (if they are not previously mapped for other purposes):

```
mapping IN3 to CSEQ1
mapping IN4 to CSEQ2
mapping OUT2 to CPERM
mapping OUT3 to CCONT
```

The user interface will show an “offboard charging” profile is also created. This is a separate profile from the “onboard charging” profile(s). Edit the offboard profile similarly to the example below.

```
mcu> edit profile offboard
mcu> set maxv 400
mcu> set maxc 50
mcu> set termc 5
mcu> set termt 60
mcu> set thmax 45
mcu> set thmin 5
mcu> show config
```

```
. . .

--bms-----
arch       : LTC12
hvc        : 3.400v
```

```

lvc      : 2.400v
bvmin    : 3.000v
thmax    : 45C
thmin    : 5C
--offboard charging (edit)-----
protocol : chargemo
  cseq1   : in3
  cseq2   : in4
  cperm   : out2
  ccont   : out3
maxv     : 400.0V
maxc     : 50.0A
termt    : 60min
termc    : 5.0A

```

ChargeMo Troubleshooting

The MCU **trace** commands offer live reporting which can help in troubleshooting. The **show** command returns the present system status including error reports, basic BMS stats and a charge system snapshot.

The commands **trace state**, **trace chargemo**, **show**, and **show history** can be used to verify proper operation of ChargeMo. Charge termination options cited in the **show history** report are listed in the figure below.

Termination Reason	Description
NORMAL	Normal Charge completion
HWFAULT	Hardware Fault Alert
CCENSUS	Cell Census Alert
TCENSUS	Thermistor Census Alert
HVC	HVC Alert
LVC	LVC Alert
HITEMP	Thermistor Overtemperature Alert
LOWTEMP	Thermistor Undertemperature Alert
NOTLOCKED	Configuration Not Locked Alert
HIPACK	Pack voltage high
PLUG_DISCONNECTED	Charge Plug became disconnected
CELL_LOOP	Cell Loop not OK
CHARGER_RX_TIMEOUT	Charge timeout
NO_SEQ1	SEQ1 signal timeout
NO_SEQ2	SEQ2 signal timeout
SYS_ERROR	System error
OUTPUT_CURRENT_LOW	Output current low
OUTPUT_CURRENT_NEGATIVE	Output current negative
OUTPUT_VOLTAGE_MISMATCH	Voltage mismatch
EV_INCOMPATIBLE	Incompatible
CHARGER_NOT_RUNNING	Charger not running
EV_STOP	EV stop

Figure 33 – ChargeMo Charge Termination Reasons

See the *Onboard Charging / Debugging* section above for examples of debug output reports.

ZEV CCS Charging

The ZeroEV CCS system terminates the CCS fast charging protocol and has all connections to the CCS charge plug, and handles EV power up when a charge is started. When fast charging, the MCU performs BMS functions and it acts as a charge controller for the ZEVCCS system. The MCU provides pack target charge voltage and current as well as BMS status such as **HVC** or **HITEMP** alerts.

ZEVCCS does not perform onboard charging. However, the ZEVCCS system manages the charge plug and a special interface is used between the ZEVCCS system and the MCU so the MCU can be notified of charge plug presence and its ampacity. Charge plug presence, normally provided by the **Proximity** input, is instead provided by a discrete input from the ZEVCCS equipment (and is called **plugin**). Charge plug ampacity, normally provided by the pilot signal, is determined by the ZEVCCS system, and communicated to the MCU by a CAN message. Other than charge plug management, the MCU performs onboard charging as normal: it maintains the configuration for the charger, terminates the CAN messages to and from the charger, determines when charging is to stop, and maintains a charger history.

Enabling ZEVCCS charging is performed by the **enable zevccs** command.

```
sys> enable zevccs can1
mapping IN3 to PLUGIN
sys>
```

This command will attempt to assign the Input Function **plugin** to a default input. (If **IN3** is used for other purposes, **plugin** can be assigned to any of **IN1** to **IN4**).

Now create and edit the offboard charge profile:

```
evcc> edit profile offboard
evcc> sh config
--offboard charge profile (edit)-----
  protocol : zevccs
   plugin  : in3
   maxv    : 20.0V
   maxc    : 2.0A
--onboard charge profile(s)-----
  charger  : tsm2500
   maxv    : 20.0V
   maxc    : 2.0A
   termc   : 2.0A
   termt   : 72.0hr
evcc>
```

Set the Target Voltage and Target Current for DCFC.

```
evcc> set maxv 320
evcc> set maxc 25
evcc> sh config
--offboard charge profile (edit)-----
  protocol : zevccs
   plugin  : in3
```

```

maxv      : 320.0V
maxc      : 25.0A
--onboard charge profile(s) -----
charger   : tsm2500
maxv      : 20.0V
maxc      : 2.0A
termc     : 2.0A
termt     : 72.0hr
evcc>

```

Note that when using the ZEVCCS system for both offboard and onboard charging, it is necessary to define a profile for onboard charging. This is done in the normal way, see *Onboard Charging, above*.

When ZEVCCS is enabled, the MCU will broadcast BMS status by sending the following five messages:

```

ZEV_BMS_LIMITS      0x0351
ZEV_BMS_SOC          0x0355
ZEV_BMS_STATUS       0x0356
ZEV_BMS_ERRORS       0x035a
ZEV_BMS_STATUS2      0x035b

```

Two messages are expected from the ZEVCCS Fast Charge Controller:

```

ZEV_CCS_COMMAND      0x0357
ZEV_CCS_DIAGNOSTIC   0x036a

```

The ZEV_CCS_COMMAND message is used to disable any Ground Fault monitoring circuitry, if present, and it provides the ampacity of the line connection, needed for Onboard Charging.

The ZEV_CCS_DIAGNOSTIC command contains debug information.

It can be verified that these messages are received in the MCU by typing **show**:

```

evcc> show
state      : STANDBY
plug state: DISCONNECTED
ZEV_CMD rx: 56
  curlim   : 22.4A
ZEV_DIAG rx: 7
  diag     : 10 03
    D0     : BMS_FLT
    D1     : INC_CHARGER CHARGE_MODE
evcc

```

The commands **trace state**, **trace charger**, and **show history**, can be used to verify proper operation.

Tracing ZEV CCS messages:

```

evcc> tr charge
charge tracing is now ON

```



```

evcc> 01:30:16.1 zevccs      : CCS_CMD; curlim= 22.4A
01:30:16.6 zevccs      : CCS_CMD; curlim= 22.4A
01:30:17.1 zevccs      : CCS_CMD; curlim= 22.4A
01:30:17.6 zevccs      : CCS_CMD; curlim= 22.4A
01:30:17.6 zevccs      : CCS_DIAG; bvs=105.7V, cvs=115.2V, d= 10 03
01:30:18.1 zevccs      : CCS_CMD; curlim= 22.4A
01:30:18.6 zevccs      : CCS_CMD; curlim= 22.4A

```

Here is an example of **trace state** output while onboard charging (simulating charge plug removal):

```

mcu> trace state
state tracing is now ON
mcu> 00:17:23.3 Plug State: DISCONNECTED => CONNECTED
00:17:23.3 Charge State: STANDBY => STARTUP
00:17:23.9 Charge State: STARTUP => CHARGE, Charge Type: ZEV_OB, CurCalc: ZEVCCS
00:17:46.0 Plug State: CONNECTED => DISCONNECTED
00:17:46.1 Charge State: CHARGE => WARMDOWN, term rsn=PLUG DISCONNECTED
00:17:51.1 Charge State: WARMDOWN => STANDBY

```

An example of **show** while using ZEVCCS onboard charging:

```

evcc> sh
state      : CHARGE
plug state: CONNECTED
ZEVCMDB rx: 2114
curlim    : 22.4A
ZEVDIAG rx: 262
diag      : 10 03
D0        : BMS_FLT
D1        : INC_CHARGER CHARGE_MODE
chg time   : 0 mins
voltage(m) : 310.0V
current(m)  : 10.0A
charger    : tsm2500
status     : 26 msgs sent; 19 msgs received
voltage    : 120.0V
current    : 12.8A
charge     : 5.06Wh
evcc>

```

INSTRUMENTATION

State of Charge Measurement

State of Charge (SOC) is the EV equivalent of a “Fuel Gauge”. SOC is a measurement of EV pack charge as a percentage (0 is “empty”, and 100 is “full”). The MCU computes SOC in two ways: by Charge (or Coulomb) Counting and by Voltage Estimation.

Voltage Estimation uses BMS measured cell voltages and configured HVC and LVC limits to estimate SOC. Voltage Estimation is not very accurate but may be used when Charge Counting is not available and is also used in addition to Charge Counting as a consistency check.

To configure SOC by Charge Counting, a Current Sensor needs to be installed and pack capacity (in kilowatt-hours) must be configured. Charge counting adjusts the SOC up or down as the pack is charged and discharged and has ways to synchronize the SOC value periodically. At the end of a complete, and successful, charge cycle, for example, the SOC value is typically set to 100. The SOC can also be manually reset by using the **socres** Input Function.

LEM DHAB Current Sensors

The LEM DHAB sensors are dual range, bidirectional sensors. The high range determines the working limits of the sensor, and the low range is used to achieve high accuracy at low current. The MCU firmware automatically selects which range to use when measurements are taken.

Sensor	Model	Version	Low Range	High Range
dhabs44	DHAB S/44	V1	+/-20A	-420A to 230A
dhabs125	DHAB S/125	V2	+/-25A	+/-200A
dhabs124	DHAB S/124	V2	+/-75A	+/-500A
dhabs133	DHAB S/133	V2	+/-75A	+/-750A
dhabs137	DHAB S/137	V2	+/-75A	+/-1000A

Figure 34 - DHAB Sensor Support

See the diagram below for correct orientation at installation.

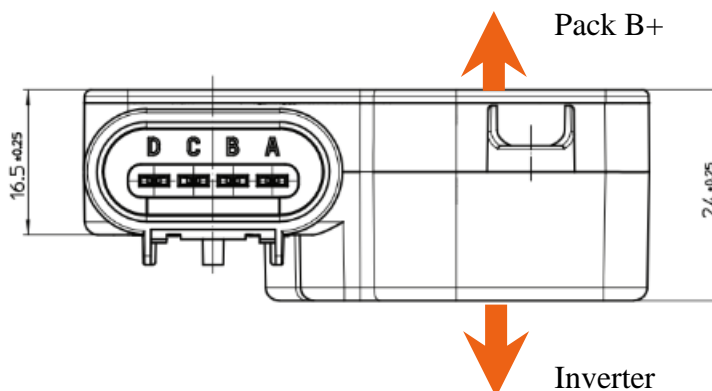


Figure 35 – DHAB V1 Hall Sensor Installation

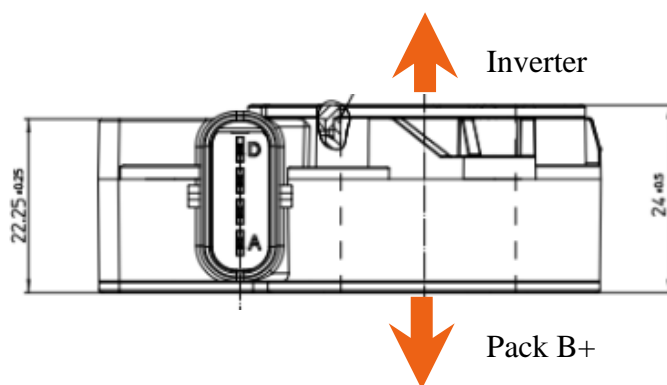


Figure 36 – DHAB V2 Hall Sensor Installation

The sensors have an integrated waterproof connector. The mating connector is a Tyco 1-1456426-5. Connector B on the MCU can connect directly to a DHAB sensor as it has all necessary connections. Connections to the Hall Sensor are made using the following table. Note that the connections to the DHAB sensors vary by version.

MCU Connector	MCU Pin Name	DHAB Signal	V1: S/44	V2: S/124, S/125, S/133, S/137
B1	+5V	Vcc	B	A
B2	IN1	High Range	A	B
B3	IN2	Low Range	D	D
B4	GND	Gnd	C	C

Figure 37 – DHAB Hall Sensor Connections

LEM HTFS Current Sensors

The LEM HTFS Hall sensors are single range, bidirectional sensors. There are several models: HTFS-200, HTFS-400, HTFS-600, and HTFS-800. These sensors have a measurement output (V_o) and a reference input (V_r). The reference input sets the “0” value of the sensor and can be overdriven to shift the working range of the sensor.

Sensor	Model	with Offset	Range
htfs200	HTFS-200	-	+/- 300A
		yes	-420A to 180A
htfs400	HTFS-400	-	+/- 600A
		yes	-820A to 380A
htfs600	HTFS-600	-	+/- 900A
htfs800	HTFS-700	-	+/- 1200A

Figure 38 – HTFS Sensor Support

The LEM HTFS datasheet can be found at:

https://www.lem.com/sites/default/files/products_datasheets/htfs_200_800-p.pdf.

The HTFS sensors are PCB mount. In the past, Dilithium Design has provided HTFS-200 and HTF-400 sensors mounted on a PCB board. In addition, the PCB has an onboard resistor which sets the V_o Offset voltage input in order to shift the working range.

See diagrams below for pinouts and orientation for the HTFS sensors

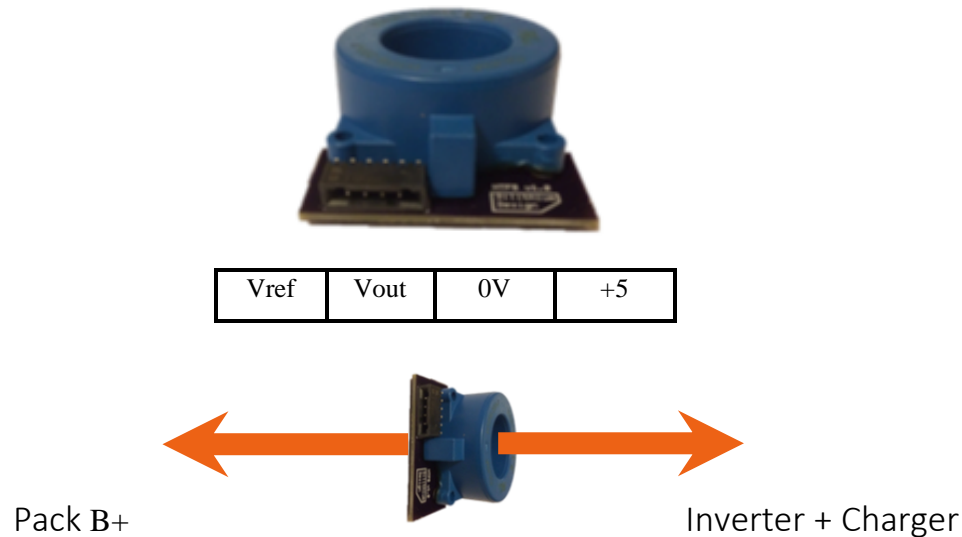


Figure 39 – HTFS Sensor Installation Direction

The following table indicates the MCU connections:

MCU Connector	MCU Pin Name	HTFS Signal	HTFS PCB Connector
B1	+5V	+5V	4
B2	hall1	Vout	2
B3	hall2	Vref	1
B4	GND	Gnd	3

Figure 40 – HTFS Hall Sensor Connections

Isabellenhütte IVT-S Current Sensors

The MCU supports the Isabellenhütte IVT-S current sensor. This sensor is a CAN enabled “shunt” sensor and can make both voltage and current measurements. There are several models of IVT-S sensors in different current ranges.



Figure 41 - Isabellenhütte IVT-S Current Sensor

When the IVT-S sensor is powered up it will start sending CAN messages. Once enabled, the MCU will automatically configure and use it. All models use the same CAN interface and so it is not necessary to configure the current range explicitly.

More information can be found at <https://www.isabellenhuetteusa.com/product/ivt-s-series/>.

Sensor Configuration

The HTFS and DHAB sensors have two analog inputs: hall1 and hall2, which must be configured in the MCU:

```
inst> set hall1 in1
inst> set hall2 in2
```

The pack capacity needs to be configured:

```
inst> set packkwh 60
```

And the sensor type needs to be configured, for example:

```
inst> set isensor dhabs137
inst> sh config
--soc-----
  isensor   : DHABS137
    hall1   : in1
    hall2   : in2
  packkwh   : 60.0

...
inst>
```

To verify proper operation, the show inputs command may be used:

```
sys> show inputs
in1  2.54v  hall1    DHABS137 H = 23.5A
in2  3.23v  hall2    DHABS137 L = 27.0A
in3  1.62v
in4  4.77v
in5   inf
prox  inf    J1772 disconnected
pilot not detected
ksi   1      detected
```

Once **isensor** is configured and the inputs connected, SOC by Charge Counting is enabled.

SOC synchronization will automatically be reset to 100 at the conclusion of a successful charge. Optionally, the **socres** input function may be assigned to an input. This allows the SOC to be manually reset to “full” when the input is grounded.

MCU EV Display

The MCU Electric Vehicle Display displays pack State of Charge, Pack Current, Voltage, and Watts, Thermistor Temperatures, as well as BMS alarms. The MCU Display uses the same hardware as the existing Dilithium Display used with the BMSC product.

The Display hardware uses the 4D systems gen4 32-DCT-CLB. This display is a 3.2” 240x320 pixel LCD capacitive touch display with an attached bezel. The datasheet can be found at http://www.4dsystems.com.au/product/gen4_uLCD_32D/.

With bezel, the display measures 98.8mm x 72.6mm. The display is mounted on a flat surface with a rectangular opening (approximately 80mm x 56mm) and affixed to the surface using double-sided tape.

Mechanical drawings can be found at http://www.4dsystems.com.au/productpages/gen4-uLCD-32D-CLB/downloads/gen4-uLCD-32D-CLB_drawing_R_1_0.PDF

A PCB is mounted directly on the back of the display and connects to the display using a 30pin flat cable. The PCB has two connectors: a 5-pin systems connector and a 4-pin connector, but only the 5-pin system connector is used. When mounting the display, the 5-pin Systems connector is towards the bottom.

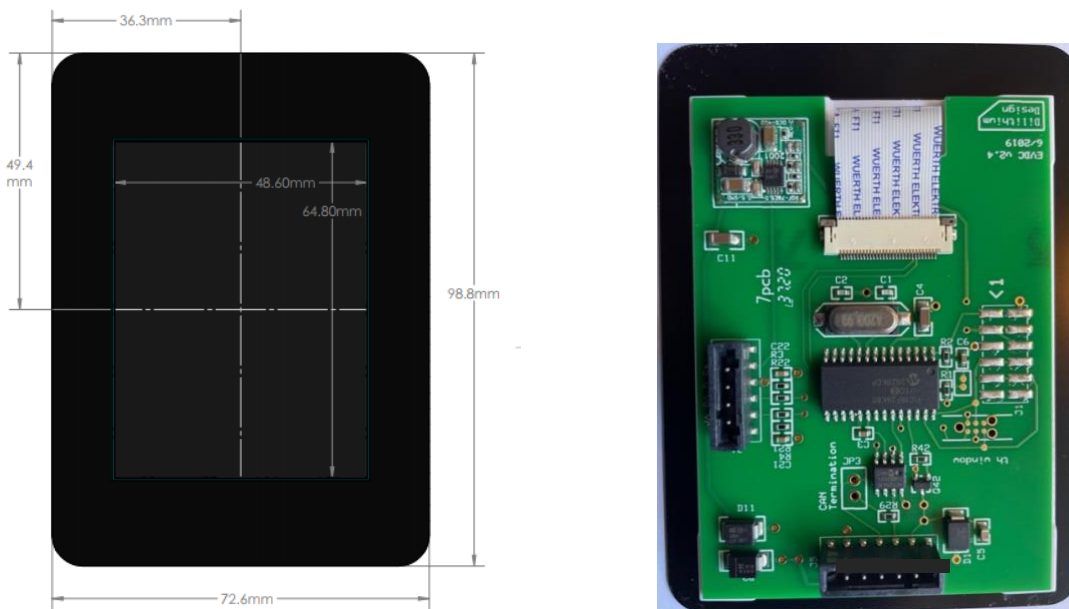


Figure 42 – MCU EV Display Hardware

The System Connector provides 12V power, and CAN; one pin is not used.

GND	12V	CANL	CANH	
-----	-----	------	------	--

The **GND** and **12V** inputs on the System Connector provide Display power. Power should be enabled when the EV is being driven or being charged. The maximum 12V power consumption is approximately 100ma. Display power may be provided directly from the MCU OUT5 output.

The **CANL** and **CANH** inputs can connect to either the CAN1 or CAN2 MCU CAN networks. Configuring the CAN rate is not necessary: when the Display is power cycled, it will autobaud to either 250Kb or 500Kb.

A CAN termination resistor in the Display may be enabled by installing the bridge between the right two pins of jumper JP3, located adjacent to the 5p Systems Connector.

Startup Screen

When the Display is first powered up, it shows a Startup screen for a few seconds. During this time, the Display attempts to communicate with the MCU. The Display will then show both its build date as well as the firmware version number of the MCU, assuming the MCU is connected. In the example below, the display build date is “Aug 2021” and the MCU firmware version is “1.0.3”.

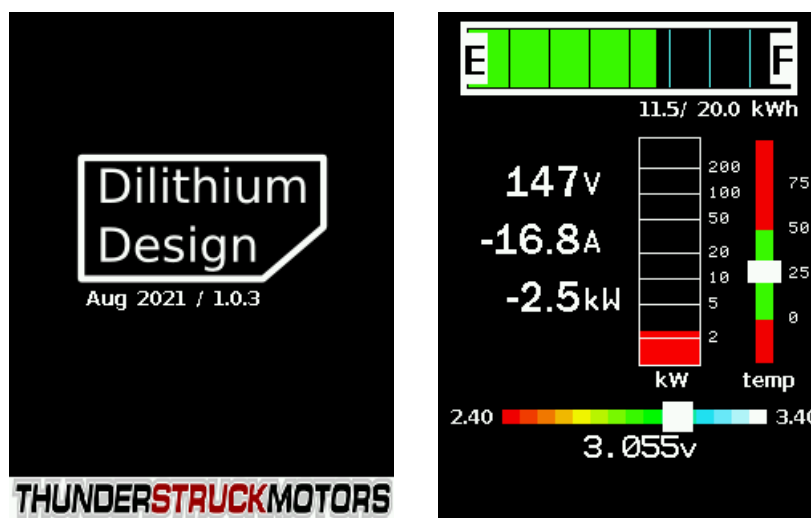


Figure 43 – MCU Display Startup and Status Screens

While the Startup screen is being displayed, it is possible to “freeze” it by touching the screen before it transitions to the next screen. This gives time to make note of the firmware version numbers if needed. Touching the screen again will unfreeze the display.

Status Screen

The **Status** Screen is the main screen. The Status screen is divided into several regions: State of Charge, Numeric Results, Wattmeter, Thermistors, Cell Voltage Summary, and Message.

The **State of Charge** (top) is a digital fuel gauge. This gauge shows the current SOC as calculated by the MCU. It also shows the estimated Kilowatt Hours of pack capacity remaining.

The **Numeric Results** (middle left) shows the Pack Voltage, Pack Amps, and Pack Watts.

The **Kilowatt** meter (middle center) shows Pack Watts. This scale is logarithmic. The bar will be red for power taken from the pack (as when driving) and is green for power being added to the pack (as during regeneration or charging).

The **Thermistor** (middle right) is an optional meter that is shown if any thermistors are enabled. The markings are in C. The green range are determined by the configured parameters **thmax** and **thmin**. The white bar indicates the range of thermistor temperatures from low to high; in the example, all thermistor temperatures are within 25C by a few degrees.

The **Cell Voltage Summary** lies just below the middle of the screen. It shows the range of cell voltages present within the pack as well as the average cell voltage. The scale is calculated from the configured **hvc** and **lvc** values in the BMS. In the example above:

- 2.40 is the BMS programmed Low Voltage Cutoff Voltage
- 3.40 is the BMS programmed High Voltage Cutoff Voltage
- 3.055v is the average cell voltage of all cells in all packs

The “white” indicator bar of the Cell Voltage Summary indicates the range of cell voltages present in the pack. The left border of the white indicator is the lowest cell voltage in the pack, the right border is the highest cell voltage in the pack and the width of the indicator gives an overall indication of pack balance.

In the example above, the pack is somewhat unbalanced which is indicated by the width of the white bar.

The **Message** region is at the bottom of the display.

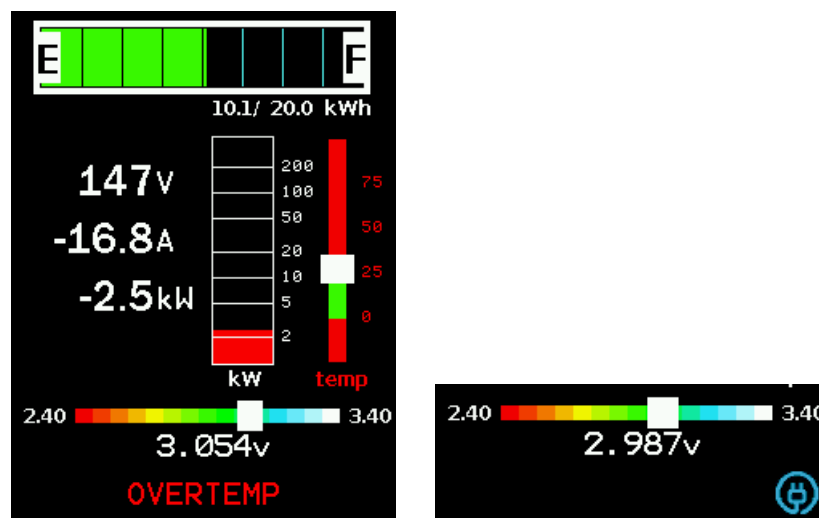


Figure 44 - MCU Display Message Region

If there are there are no alerts present, the Message region will be blank. If, for example, a thermistor reading exceeds the **thmax** setting, an **HITEMP** alert will be generated and a blinking message will appear in the **Message** area.

In all, the following messages are supported:

- **HARDWARE** – indicates a diagnostic failure by one of the measurement devices
- **HVC** – high voltage cutoff alert
- **LVC** – low voltage cutoff alert
- **CCENSUS** – cell census alert
- **HITEMP** – thermistor overtemperature alert
- **LOTEMP** – thermistor undertemperature alert
- **TCENSUS** – thermistor census alert
- **NOTLOCKED** – configuration not locked alert.

The **Message** area also has a charge plug indicator at the right. If the charge plug is shown in blue, it indicates that the charge plug is connected but the MCU is not charging. When charging, the plug is shown in red.

From the **Status** screen, the user can navigate to the **Cell Map** by touching the display anywhere above the **Cell Voltage Summary**.

Cell Map

The **Cell Map** shows a graphical representation of all cells. Cells are arranged in groups and each row of cells represents a cell group. The BMS architecture determines the nominal cell group size: cell groups may contain 12, 13 or 18 cells. Cell groups are connected in series to create a “pack”. Multiple packs may then be paralleled together. The pack is shown by the colored bar to the left of each group of cells: each pack is represented by a different color. Pack 1 is turquoise, Pack 2 is brown, and Pack 3 is cornflower blue, pack 4 is gold, up to 8 packs can be shown.

In the cell map, a color is assigned to every cell based on its voltage. The order is roughly: red, yellow, green, blue, white, where a “red” cell is low and a “white” cell is high.

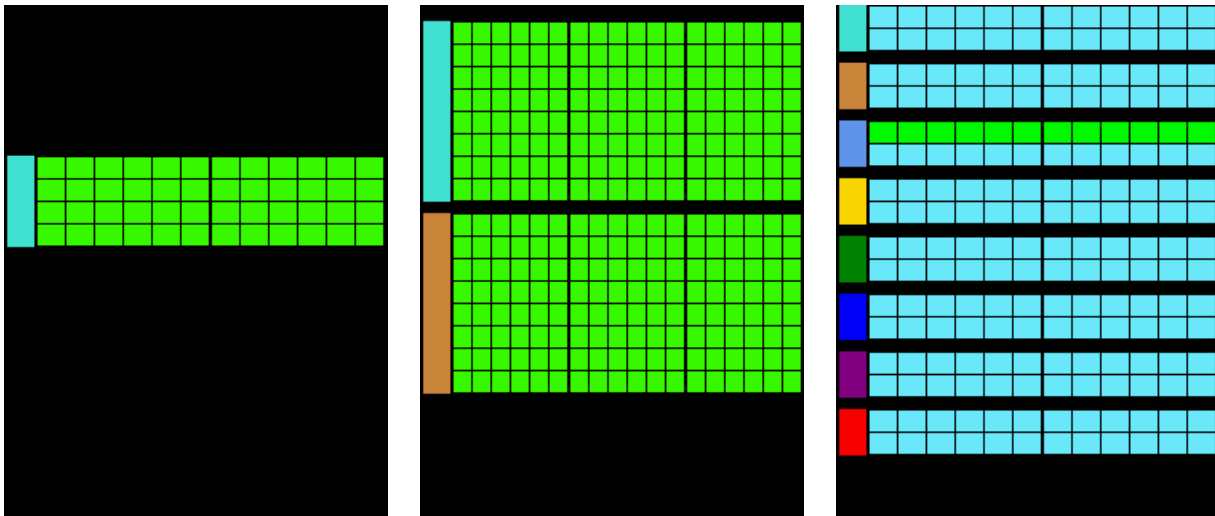


Figure 45 - MCU Display Cell Map Examples

- The example on the left shows a single pack, with four 12-cell groups.
- The example in the middle shows two packs, each with eight 18-cell groups.
- The example to the right shows eight packs, each with two 12-cell groups.

Note that in the first two examples the cells are in good balance with each other and show the same color. In the last example, the first group of the third pack (with cells in “green”) is lower than the other cells.

This screen updates in real time and can be used to track the performance of the cells under load.

To Navigate from this screen, return to **Status** by touching the Cell Map display

Charge Progress

When charging, the Display reverts to a charge progress screen, that shows voltage, amps, and Watt Hours. This screen is purposely dim as it may be on for several hours while charging.

Touching the charge progress screen will return to the Status Screen where all operations are available.

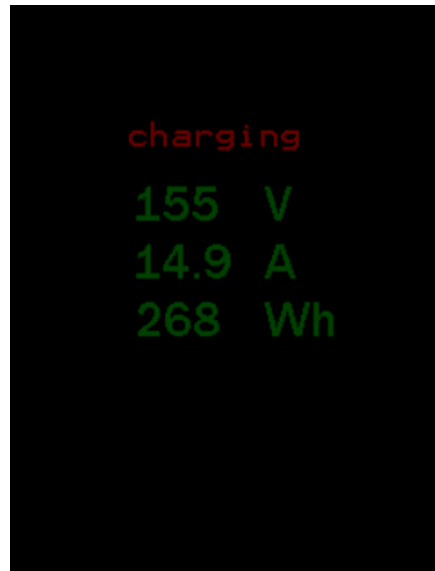


Figure 46 – MCU Display Charging Progress Screen

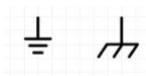
Analog Fuel Gauge

Analog Fuel gauges typically have three connections: +12V, Ground, and “sender”.



Figure 47 – Analog Fuel Gauge Typical Connections

Note that the “Ground” connection on the meter might be indicated by “-“, or by a symbol:



Conventionally, the Fuel Gauge is connected to a gas tank float “sender” which is a variable resistor depending on the level of the float. Usually, this connection only uses one wire as the return connection is made directly to chassis ground. The fuel gauge and the sender are a matched set, and, although there are common values, there are no standard ones.

The MCU uses Pulse Width Modulation (PWM) to drive the meter. To configure an analog fuel gauge, the **fuelg** PWM output must be mapped to a PWM-capable physical output (one of **OUT1**, **OUT2**, or **OUT3**).

```
inst> set out1 fuelg
```

PWM settings must also be configured in order to calibrate the gauge. There are five settings needed; one for each of 0%, 25%, 50%, 75%, and 100%. Each PWM setting is a number from 0 to 255. When the MCU updates the gauge, it interpolates between the calibration values to determine appropriate PWM values from 0 to 255.

An example:

```
inst> set fuelg 0 30
inst> set fuelg 25 80
inst> set fuelg 50 110
inst> set fuelg 75 150
inst> set fuelg 100 200
inst> show config
--soc-----
  isensor   : DHABS137
   hall1    : in1
   hall2    : in2
  packkwh   : 20.0
--gauges-----
  fuelg     : out1, pwm (0:30 25:80 50:110 75:150 100:200)
```

When a **set fuelg <s> <p>** command is entered, the gauge will operate with the newly entered duty cycle. (And will time out after ten or twenty seconds). Repeat the process for each setting so that the gauge reads correctly for all five settings: 0%, 25%, 50%, 75%, and 100%.

The Display has been tested with a variety of fuel gauges, including VDO and Bosch.

Analog Temperature Gauge

Analog Temperature gauges work similarly to analog fuel gauges. The gauge sender input must be connected to a PWM capable output on the MCU and the MCU output **tempg** must be mapped. Finally, the PWM setpoints must be configured. For example:

```
inst> set out2 tempg
inst> set tempg 0 30
inst> set tempg 25 80
inst> set tempg 50 110
inst> set tempg 75 150
inst> set tempg 100 200
```

Speedhut Gauges

The MCU supports CAN capable Speedhut gauges for fuel State of Charge, pack current, pack voltage, and temperature. The MCU refers to these gauges as **sh_fuel**, **sh_vat**, (or **sh_vatr**), and **sh_temp**. Only one of **sh_vat** and **sh_vatr** may be enabled; the result is identical except that the **sh_vatr** option negates the sign of the ammeter reading.

The gauges are enabled with the **enable** command. This command has two parameters: the gauge type and the CAN interface.

To enable all three gauges on CAN2, the following commands are used:

```
inst> enable sh_fuel can2
```

```
inst> enable sh_vat can2
inst> enable sh_temp can2
```

Curtis 3100r

The Curtis 3100r display is a CAN controlled 52mm round display. This gauge is limited to use with a single pack of 96 cells or fewer.



Figure 48 – 3100R Display

This display can display Pack State of Charge, has a 7-segment alphanumeric LCD display, graphical icons, an LED, backlight and two buttons.

Once the 3100r is connected to power and connected to the MCU via the CAN network, the display may be enabled by the command:

```
mcu> enable 3100r
```

Pack **SOC** is shown in the top 10 segment SOC arc. (see *State of Charge Measurement*, above)

The 3100r **LED** blinks when the pack is in HVC or LVC. The **LED** is on for Thermistor over-temperature. The **Wrench** Icon blinks if there is a configuration error: configuration not locked, cell census error or thermistor census error. The **Battery** Icon is on when the pack is balancing.



Figure 49 – 3100R Display Examples

The left-hand button sequences the alphanumeric display between **pack voltage**, **minimum cell voltage**, **maximum cell voltage**, **average cell voltage**, and **standard deviation**.

- The pack voltage is shown in tenths of a volt.
- The minimum cell voltage shows the cell number, an "underbar" and then three digits which is the cell voltage in hundredths, but with no period. (So "354" is 3.54v).
- The maximum cell voltage shows the cell number, an "upperbar" and the value.
- The average cell voltage shows three "middle bars" and a value.
- The standard deviation shows "Sd", "space" and then three digits of standard deviation. (So "014" is 0.014v).

The right-hand button toggles the **backlight**.

The 3100r can be programmed to operate at a 250Kb or 500Kb CAN data rate. If the CAN data rate must be changed, use the following MCU commands (example: change from 250Kb to 500Kb with the 3100R on MCU CAN1):

```
mcu> set can1br 250
mcu> set 3100rbr 500
mcu> set can1br 500
```

This process sets the MCU to the 3100r data rate, changes the 3100r data rate, then changes the MCU data rate to match the new 3100r rate.

Torque

Torque is an application that runs on a smartphone or tablet. It implements the OBDII protocol, supports many standard PIDs and allows the user to customize the display to define meters and to define what data is to be displayed. For a reference on Torque, see:

https://torque-bhp.com/wiki/Main_Page

In use, the smartphone typically uses Bluetooth to communicate with a Bluetooth to OBDII adapter, which acts as a gateway to the CAN network in the vehicle. OBDII Bluetooth adapters are readily available.

In the MCU, OBDII may be enabled with the command:

```
mcu> enable obdii
```

This will enable OBDII on both CAN1 and CAN2. An OBDII reply will be sent to the same interface that the request arrived on.

The following table lists the OBDII PID support in the MCU. This table also has the necessary information useful to define custom meters within Torque.

Name	PID	Torque ModeAndPID	Equation	Min	Max	Units	Scale
RunTime	0x1f	011f	int16(B:A)	0	65535	seconds	
Version	0xdd80	22dd80	A				1
Revision		22dd80	B				1
ALERTS	0xdd81	22dd81					
HARDWARE			BIT(A:6)	0	1		
CCENSUS			BIT(A:5)	0	1		
TCENSUS			BIT(A:4)	0	1		
HVC			BIT(A:3)	0	1		
LVC			BIT(A:2)	0	1		
HITEMP			BIT(A:1)	0	1		
LOTEMP			BIT(A:0)	0	1		
Pack Voltage	0xdd83	22dd83	int16(B:A)/10			volts	0.1
Pack Current	0xdd84	22dd84	((signed(B)*256)+A)/10			amps	0.1
Pack SOC	0xdd85	22dd85	A	0	100	percent	1
Cell Min	0xdd86	22dd86	int16(B:A)/10000			volts	0.0001
Cell Max	0xdd87	22dd87	int16(B:A)/10000			volts	0.0001
Cell Mean	0xdd88	22dd88	int16(B:A)/10000			volts	0.0001
Cell STD	0xdd89	22dd89	int16(B:A)/10000			volts	0.0001

Figure 50 – OBDII PID Support

DC/DC Converter Support

The MCU supports the Volt and Delphi DC/DC converters. These DC/DC use a default CAN baud rate of 500Kb/sec. The ddtype parameter chooses the DCDC converter type, the ddvoltage parameter configures the DC setpoint.

An example of configuration that enables the DCDC converter:

```
inst> set ddtype volt
inst> set ddvoltage 13.6
inst> enable dc dc can2
```

To disable the DC/DC converter:

```
inst> set ddtype none
```

CAN SUPPORT SUMMARY

The table below gives a list of CAN services supported in the MCU. CAN Services of type “Client” respond to requests from either CAN1 or CAN2 and transmit responses on the interface that received the request. These include CANopen, OBDII, and J1939. CANopen is used to interface with the Dilithium Display.

CAN Services of type “Client/server” only operate on a single CAN interface. Messages received are only processed if they arrive on the configured interface. Messages transmitted are only sent on the configured interface.

CAN Services of type “TX only” only transmit messages on a single CAN interface; they never receive messages.

CAN Service	Type	Reference
canopen	Client	CANopen
obdii	Client	OBDII
j1939	Client	J1939
a123	Client/server	A123 MBB
zevccs	Client/server	ZEV CCS Charging
chargemo	Client/server	ChargeMo Charging
obcharge	Client/server	Onboard Charging
3100r	Client/server	Curtis 3100r
dcde	TX Only	DC/DC converter
sh_fuel	TX Only	Speedhut
sh_temp	TX Only	Speedhut
sh_vat	TX Only	Speedhut
sh_vatr	TX Only	Speedhut

Figure 51 – CAN Services

Examples of the **enable** and **disable** commands to enable CAN services are given below:

```
sys> enable obdii
sys> enable 3100r can2
sys> disable canopen
```


SID Support

The following table shows SID values supported by the MCU.

SID	RANGE	SENT BY	RECEIVED BY	PROTOCOL	MESSAGE NAME
0x000		MCU		CO/3100R	NMT
0x050		MCU	A123	A123	BCM Command
0x050		MCU	LEAR	LEAR	Charge Control
0x100		MCU	FC EVSE	ChargeMo	EV Battery
0x101		MCU	FC EVSE	ChargeMo	EV Charge Time
0x102		MCU	FC EVSE	ChargeMo	EV Status
0x108		FC EVSE	MCU	ChargeMo	CH CAPABILITY
0x109		FC EVSE	MCU	ChargeMo	CH STATUS
0x193		MCU	DISPLAY	CO/DISPLAY	PDO1 MISO
0x1d4		MCU	VOLT	VOLT	DC/DC Command
0x1d7		MCU	DELPHI	DELPHI	DC/DC Command
0x1fb		C3100R	MCU	CO/3100R	PDO1 MISO
0x20X	X=0..0x0f	A123	MCU	A123	STD
0x212		VOLT	MCU	VOLT	Charge Status
0x213		DISPLAY	MCU	CO/DISPLAY	PDO1 MOSI
0x27b		MCU	C3100R	CO/3100R	PDO1 MOSI
0x28X	X=0..0x0f	A123	MCU	A123	STD2
0x293		MCU	DISPLAY	CO/DISPLAY	PDO2 MISO
0x30X	X=0..0x0f	A123	MCU	A123	XD1
0x304		MCU	VOLT	VOLT	Charge Control
0x30E		MCU	VOLT	VOLT	Charge Enable
0x313		DISPLAY	MCU	CO/DISPLAY	PDO2 MOSI
0x351		MCU	ZEVCCS	ZEVCCS	BMS Limits
0x355		MCU	ZEVCCS	ZEVCCS	BMS SOC
0x355		MCU	SPEEDHUT	SPEEDHUT	Speedhut SOC
0x356		MCU	ZEVCCS	ZEVCCS	BMS Status
0x356		MCU	SPEEDHUT	SPEEDHUT	BMS Status
0x357		ZEVCCS	MCU	ZEVCCS	ZEVCCS Command
0x357		MCU	SPEEDHUT	SPEEDHUT	Speedhut Temp
0x35a		MCU	ZEVCCS	ZEVCCS	BMS Errors
0x35b		MCU	ZEVCCS	ZEVCCS	BMS Status2
0x36a		ZEVCCS	MCU	ZEVCCS	CCS Diagnostic
0x38X	X=0..0x0f	A123	MCU	A123	XD2
0x393		MCU	DISPLAY	CO/DISPLAY	PDO3 MISO
0x40X	X=0..0x0f	A123	MCU	A123	XD3
0x411		MCU	IVT-S	IVT-S	IVT Command
0x413		DISPLAY	MCU	CO/DISPLAY	PDO3 MOSI
0x48X	X=0..0x0f	A123	MCU	A123	XD4

0x511		IVT-S	MCU	IVT-S	IVT Reply
0x521		IVT-S	MCU	IVT-S	IVT AMPS
0x522		IVT-S	MCU	IVT-S	IVT V1
0x523		IVT-S	MCU	IVT-S	IVT V2
0x524		IVT-S	MCU	IVT-S	IVT V3
0x5f0		NLG5	MCU	BRUSA	Charge Status
0x5fb		C3100R	MCU	CO/3100R	SDO MISO
0x617		LEAR	MCU	LEAR	Charge Status
0x618		MCU	NLG5	BRUSA	Charge Control
0x67b		MCU	C3100R	CO/3100R	SDO MOSI
0x700		MCU	A123	A123	Diag_Req
0x77b		C3100R	MCU	CO/3100R	Heartbeat
0x7df		MCU	A123	A123	Diag All Req
0x7df		OBDII	MCU	OBDII	OBDII Broadcast
0x7e8		MCU	OBDII	OBDII	OBDII Response

ChargeMO

When ChargeMo is enabled, the MCU will broadcast the following three messages:

SID	msg	period	B0	B1	B2	B3	B4	B5	B6	B7
0x100	EV_BATTERY_MSG	100ms	Cterm				VLo	Vhi	Rate	
0x101	EV_CHARGE_TIME_MSG	100ms		0xff	TermT	0		CapLo	CapHi	
0x102	EV_STATUS_MSG	100ms	ProtId	TVLo	TVHi	TC	FF	SF	SOC	

Cterm - charge termination current
 Vlo, Vhi - maximum charge voltage
 Rate - set to 100
 TermT - max charge time, in minutes
 CapLo, CapHi - pack capacity in KWH
 ProtId - protocol ID
 TVLo, TVHi - target voltage
 TC - target current
 FF - fault flag
 SF - status flag
 SOC - state of charge

The MCU expect the following two messages from the ChargeMo Controller:

SID	msg	timeout	B0	B1	B2	B3	B4	B5	B6	B7
0x108	CM_CAPABILITY_MSG	2 sec	ProtId	MaxVLo	MaxVHi	MaxC	ThVLo	ThVHi		
0x109	CM_STATUS_MSG	2 sec	ProtId	VLo	VHi	C		SF	T1	T2

ProtId - protocol Id
 MaxVLo, Hi - maximum voltage available
 MaxC - maximum current available
 ThLo, Hi - threshold voltage
 Vlo, Vhi - Voltage

C	- Current
SF	- Status Fault flag
T1	- Remaining Charge Time 10s
T2	- Remaining Charge Time

ZEVCCS

When ZEVCCS is enabled, the MCU will broadcast the following messages:

SID	msg	period	B0	B1	B2	B3	B4	B5	B6	B7
0x351	BMS_LIMITS	100ms	CVLo	CVHi	CCLo	CCHi	DCLLo	DCLHi	DVLLo	DVLHi
0x355	BMS_SOC	100ms	SOC				SOCHLo	SOCHHi		
0x356	BMS_STATUS	100ms	VLo	VHi	CLo	CHi	TempLo	TempHi		
0x35a	BMS_ERRORS	100ms	ProtId	TVLo	TVHi	TC	FF	SF	SOC	
0x35b	BMS_STATUS2	100ms	ProtId	TVLo	TVHi	TC	FF	SF	SOC	

CVHi, CVLo	- target charge voltage
CCHi, CCLo	- target charge current
DCLHi, DCLLo	- discharge current limit
DVHi, DVLo	- discharge voltage limit
SOC	- battery SOC (in percent)
SOCHi, SOCLo	- battery SOC (in tenths of a percent)
VHi, VLo	- battery voltage
CHi, CLo	- battery current (signed)
TempHi, TempLo	- battery temperature (signed)

The MCU expect the following two messages from the ZECCS Controller:

SID	msg	timeout	B0	B1	B2	B3	B4	B5	B6	B7
0x357	CCS_COMMAND	n/a	ISODis	CLLo	CLHi					
0x36a	CCS_DIAGNOSTIC	n/a	D0	D1	BVSLo	BVSHi	CVSLo	CVSHi		

ISODis	- disables ground fault monitoring circuitry
CLHi, CLLo	- charge plug current limit
D0	- diag info
D1	- diag info
BVSHi, Lo	- battery voltage sense
CVSHi, Lo	- CCS voltage sense

VOLT, LEAR, and BRUSA NLG5 Chargers

SIDs are used for VOLT, LEAR and BRUSA NLG5 chargers.

SID	type	sent by	msg
0x212	VOLT	charger	CHARGE_STATUS
0x617	LEAR	charger	CHARGE_STATUS
0x5f0	NLG5	charger	CHARGE_STATUS

SID	type	sent by	msg
0x304	VOLT	MCU	CHARGE_CONTROL
0x050	LEAR	MCU	CHARGE_CONTROL
0x618	NLG5	MCU	CHARGE_CONTROL

VOLT and DELPHI DC/DC Converters

SIDs are used for VOLT and DELPHI DC/DC converters.

SID	type	sent by	msg
0x1d4	VOLT	MCU	DCDC_ENABLE
0x1d7	DELPHI	MCU	DCDC_ENABLE

A123

A123 battery modules use SID CAN ID ranges for different functions and within a range a different CAN Id is used for each module ... the MCU supports up to 16 modules. When using A123 modules, and so it is recommended that other SID based devices (such as the Dilithium Display) be used on a different CAN network.

SID	sent by	msg
0x050	MCU	BCM Command
0x200-0x20f	A123	STD
0x280-0x28f	A123	STD2
0x300-0x30f	A123	XD1
0x380-0x38f	A123	XD2
0x400-0x40f	A123	XD3
0x480-0x48f	A123	XD4
0x700	MCU	Diag_Req
0x7df	MCU	Diag All Req

For more information on A123 CAN messages see A123 documentation.

SpeedHut Gauges

When enabled, the MCU will transmit the following messages with the given period. Note that these gauges are “receive only” and do not transmit any CAN messages.

SID	gauge	period	B0	B1	B2	B3	B4	B5
0x355	fuel	1s	SOC					
0x356	vat/vatr	125ms	VLo	Vhi	Alo	AHi	Tlo	Thi
0x357	temp	250ms	VLo	Vhi	Temp			

SOC - State of charge, a number from 0 to 100.

VHi,Lo - pack voltage in Volts

AHi,Lo - pack AMPS

THi,Lo - highest thermistor reading in C

Temp - highest thermistor reading in C.

IVT-S

The following messages are used by the IVT current sensor:

SID	sent by	msg
0x411	MCU	IVT_Command

0x511	IVT-S	IVT Reply
0x521	IVT-S	IVT AMPS
0x522	IVT-S	IVT V1
0x523	IVT-S	IVT V2
0x524	IVT-S	IVT V3

The IVT Command and IVT Reply are used for configuration. The remaining messages report current and the three voltage measurements. For more information, see IVT-S documentation.

OBDDII Client

OBDDII is a CAN based management protocol widely used in automotive applications. OBDDII is a request / response protocol: a diagnostic tool typically sends a request in one CAN message and a vehicle ECU (Engine Control Unit) sends a reply.

SID	sent by	Msg
0x7df	OBDDII Manager	OBDDII request
0x7e8	MCU	OBDDII reply

The OBDDII protocol supports up to eight individually addressable ECUs as well as a “broadcast” function. The MCU currently does not support ECU numbering, and only the broadcast function is used. The manager sends the CAN message with SID 0xdf for its request message, and the MCU responds with the CAN SID 0x7e8.

OBDDII defines a “Parameter ID” or “PID” which identifies the type of data to be retrieved or set. A set of standard PIDs are defined, and custom, vehicle specific PIDs (“extended PIDS”) may also be defined and used.

An OBDDII “service” indicates how to operate upon the PIDs. The MCU supports Service 1 (“show current data”) and Service 0x22 (“vehicle specific service”).

For a reference on OBDDII, the CAN structure, and the definition of standard PIDs, see the following:

https://en.wikipedia.org/wiki/On-board_diagnostics

https://en.wikipedia.org/wiki/OBD-II_PIDs

Standard PIDs are identified by a single byte, and the MCU supports two standard PIDs: the “PID value supported” (PID=0x00) and “run time” (0x1f).

To request the current “run time” data the following message is sent:

SID	B0	B1	B2
0x7df	0x02	0x01	0x1f

A SID of 0x7df is used for the request. The first byte of the message, B0, is the number of bytes that follow. The second byte of the message (B1) is the “service type”, 1 means “get current

data”. The third byte of the message (B2) is the PID value. In this case the PID value 0x1f is for “run time”.

In reply, the MCU will send:

SID	B0	B1	B2	B3	B4
0x7e8	0x04	0x41	0x1f	A	B

The SID is 0x7e8 for the reply. B0, length, is 4. B1, “current data reply” is 0x41. B2 (PID) is 0x1f, “run time”, and then A and B are low and high bytes of the run time in seconds.

Vehicle specific PIDs are identified by two bytes; an example is Pack Voltage, which has the value 0xdd83. The full list of PIDs supported by the MCU is specified below.

To request the current pack voltage data the following message is sent:

SID	B0	B1	B2	B2
0x7df	0x03	0x22	0xdd	0x83

The first byte of the message, B0, is the number of bytes that follow, three in this case. The second byte of the message (B1), “service type” is 0x22 or “custom service”. The third and fourth bytes of the message are the PID value (0xdd83), in this case it is Pack Voltage.

In reply, the MCU might send:

SID	B0	B1	B2	B3	B4	B5
0x7e8	0x05	0x62	0xdd	0x83	0x58	0x06

B0, length, is 5. B1, “custom service reply” is 0x62. B2 and B3 (PID) is 0xdd83 (Pack Voltage), and B4 and B5 are low and high bytes of pack voltage (in this case 0x0658 = 162.4V).

C3100R

The C3100R uses CANopen messages with NodeId 0x7b. The following messages are supported:

SID	sent by	Msg
0x000	MCU	NMT
0x1fb	C3100R	PDO1 MISO
0x27b	MCU	PDO1 MOSI
0x5fb	C3100R	SDO MISO
0x67b	MCU	SDO MOSI
0x77b	C3100R	Heartbeat

For message contents, see Curtis C3100 documentation.

MCU DISPLAY

MCU uses CANOpen messages with NodeId 0x13 to communicate with the MCU Display. The MCU acts as a CANOpen client. The CANOpen service is enabled by default and is active on both CAN1 and CAN2. The MCU display can request configuration and state data but does not perform configuration. The GFM, if present, also sends an unsolicited CANOpen message to the display to report ground fault status.

The following messages are supported:

SID	sent by	Msg
0x193	MCU	PDO1 MISO
0x193	GFM	PDO1 MISO
0x213	DISPLAY	PDO1 MOSI
0x293	MCU	PDO2 MISO
0x313	DISPLAY	PDO2 MOSI
0x393	MCU	PDO3 MISO
0x413	DISPLAY	PDO3 MOSI

PDO1 MOSI (SID 0x213) – MCU Configuration Data Request

This message is used by the Display to request configuration data from the MCU. The first byte is a control field which determines the data to be requested.

message	SID	B0
Options	0x213	0x00
Obcharge1	0x213	0x01
Obcharge2	0x213	0x02
Obcharge3	0x213	0x03
Obcharge4	0x213	0x04
Obcharge5	0x213	0x05
Obcharge6	0x213	0x06
MCUCFG	0x213	0x10
BMSCCFG1	0x213	0x11
BMSCCFG2	0x213	0x12
GRPMAP0	0x213	0x18
GRPMAP1	0x213	0x19
GRPMAP2	0x213	0x1a

PDO1 MISO (SID 0x193) – MCU Configuration Data Reply

This message is sent by the MCU in reply to the PDO1 MOSI request for configuration data. The first byte is the control field which indicates the type of data.

message	SID	B0	B1	B2	B3	B4	B5	B6	B7
Options	0x193	0x00		evccOptions		bmsOptions		instOptions	
Obcharge1	0x193	0x01	profile	ch1type	ch2type	ch3type	ch4type	termc	
Obcharge2	0x193	0x02	profile	maxv		maxc		termt	
Obcharge3	0x193	0x03	profile	lineV		lineC		evseWaitT	
Obcharge4	0x193	0x04	profile	ch1phase	ch2phase	ch3phase	ch4phase		
Obcharge5	0x193	0x05	profile	finmaxv		finmaxc		fintermt	

Obcharge6	0x193	0x06	profile	fltmaxv		fltmaxc		fltterm	
mcucfg	0x193	0x10		version	revision	arch			
Bmscfg1	0x193	0x11		hvc		lvc		bvmin	
Bmscfg2	0x193	0x12		hvcc		lvcc		hvcldel	lvclldel
GRPMAP0	0x193	0x18		G0	G1	G2	G3	G4	G5
GRPMAP1	0x193	0x19		G6	G7	G8	G9	G10	G11
GRPMAP2	0x193	0x1a		G12	G13	G14	G15		

PDO1 MISO (SID 0x193) – GFM Status

This message is sent by the GFM. This unsolicited message is sent every 1 second.

message	SID	B0	B1	B2	B3	B4	B5	B6	B7
GFM Summary	0x193	0x0f	Status	Isolation		Leakage		Location	

PDO2 MOSI (SID 0x313) – MCU Status Data Request

This message is used by the Display to request status data from the MCU.

SID	B0	B1	B2	B3	B4	B5	B6	B7
0x313	0	MSRate	PSRate	CVRate	THRate	SOCRate	CMRate	0

MSRate, PSRate, CVRate, THRate, SOCRate and CMRate give the period in 50ms ticks for each of the messages: MCU Summary, Pack Summary, Cell Voltage Summary, Threshold Summary, SOC Summary and CellMap Summary. Multiple summary reply messages may be requested by a single request message.

PDO2 MISO (SID 0x293) – MCU Status Data Reply

These messages are used to report MCU Summary, Pack Summary, Cell Voltage Summary, Thermistor Summary and SOC Summary data by the MCU.

message	SID	B0	B1	B2	B3	B4	B5	B6	B7
MCU Summary	0x293	0x01		ChargeKw		Charge state	Plug state	Alerts	
Pack Summary	0x293	0x02		Pack Voltage		Pack Current			
Cell Voltage Summary	0x293	0x03	cell count	CV Low		CVMean		CVHi	
Thermistor Summary	0x293	0x04	Th count	THMin	THMax			tTHLo	iTHHi
SOC Summary	0x293	0x05	SOC	PackKWHr		PackMaxKWHr			

Each of these messages repeat at the requested rate for up to 2 seconds and then stop. If a continuous display is required, it is necessary for the Display to periodically re-send the PDO2 MOSI message.

PDO3 MISO (SID 0x393) – MCU Cell Map Summary Reply

This message is used to report Cell Map Summary. The cell map represents each cell as a 4 bit value, and the cell map data for two cells can fit into a single byte. The number of messages required to report a cell group depends on the cell group size. The first byte of the message indicates the architecture and the message type. The data for every cell is displayed on the MCU display as a colored rectangle representing its voltage.

For the LTC12 architecture, each cell group contains 12 cells; one message is needed per group. The encoding is:

SID	B0	B1	B2	B3	B4	B5	B6	B7
0x393	0	0	CM01	CM23	CM45	CM67	CM89	CM10_11

For the A123 architecture, each cell group contains 13 cells; one message is needed per group. The encoding is:

SID	B0	B1	B2	B3	B4	B5	B6	B7
0x393	1	CM_12	CM01	CM23	CM45	CM67	CM89	CM10_11

For the LTC18 architecture, each cell group contains 18 cells; two messages are needed per group. The encoding is:

SID	B0	B1	B2	B3	B4	B5	B6	B7
0x393	2	0	CM01	CM23	CM45	CM67	CM89	CM10_11
0x393	3	0	CM12_13	CM14_15	CM16_17			

These messages repeat at the requested rate for up to 2 seconds and then stop. If a continuous display is required, it is necessary for the Display to periodically re-send the PDO2 MOSI message.

EID Support

The following table shows EID values supported by the MCU.

EID	RANGE	SENT BY	RECEIVED BY	PROTOCOL	MESSAGE NAME
0x14ebd0d8		Manager	MCU	J1939	J1939 Read
0x14efd0d8		Manager	MCU	J1939	J1939 Write
0x14ff01d0		MCU	Manager	J1939	OBCHG1
0x14ff02d0		MCU	Manager	J1939	OBCHG2
0x14ff03d0		MCU	Manager	J1939	OBCHG3
0x14ff04d0		MCU	Manager	J1939	OBCHG4
0x14ff05d0		MCU	Manager	J1939	OBCHG5
0x14ff06d0		MCU	Manager	J1939	OBCHG6
0x14ff10d0		MCU	Manager	J1939	MCUCFG0
0x14ff11d0		MCU	Manager	J1939	BMSCFG1
0x14ff12d0		MCU	Manager	J1939	BMSCFG2
0x14ff18d0		MCU	Manager	J1939	GRPMAP0
0x14ff19d0		MCU	Manager	J1939	GRPMAP1
0x14ff1ad0		MCU	Manager	J1939	GRPMAP2
0x14ff20d0		MCU	Manager	J1939	MCUSUM
0x14ff21d0		MCU	Manager	J1939	PACKSUM
0x14ff22d0		MCU	Manager	J1939	CVSUM
0x14ff23d0		MCU	Manager	J1939	THSUM

0x14ff24d0		MCU	Manager	J1939	SOCSUM
0x14ff30d0		MCU	Manager	J1939	CELLMAP0
0x14ff31d0		MCU	Manager	J1939	CELLMAP1
0x1806e0f4		MCU	Charger_e0	STEALTH	Charge Control
0x1806eXf4	X=5, 6, 7, 8, 9	MCU	Charger_eX	ELCON	Charge Control
0x18e54X24	X=0, 1, 2, 3	MCU	Charger_4X	CH4100	Charge Control
0x18eb244X	X=0, 1, 2, 3	Charger_40	MCU	CH4100	Charge Status
0x18ff50e0		Charger_e0	MCU	STEALTH	Charge Status
0x18ff50eX	X=5, 6, 7, 8, 9	Charger_eX	MCU	ELCON	Charge Status
0x1a5a5a5a		MCU	Charger	CH4100	Set CAN Address
0x1de08XY	X=0,1; Y=0..15	Manager	MCU	Dilithium	Read Cell Data
0x1de09XY	X=0,1; Y=0..15	MCU	Manager	Dilithium	Group 1 Cell Data
0x1de0aXY	X=0,1; Y=0..15	MCU	Manager	Dilithium	Group 2 Cell Data
0x1de0bXY	X=0,1; Y=0..15	MCU	Manager	Dilithium	Group 3 Cell Data
0x1de0cXY	X=0,1; Y=0..15	Manager	MCU	Dilithium	Read Thermistor Data
0x1de0dXY	X=0,1; Y=0..15	MCU	Manager	Dilithium	Group 1 Therm Data
0x1de0eXY	X=0,1; Y=0..15	MCU	Manager	Dilithium	Group 2 Therm Data
0x1de11XY	X=0,1; Y=0..15	MCU	Manager	Dilithium	Group 4 Cell Data
0x1de12XY	X=0,1; Y=0..15	MCU	Manager	Dilithium	Group 5 Cell Data

J1939 READ Request – Configuration Data

The J1939 Read Request is used by the manager to request both state data and configuration data.

J1939 READ	B0	B1	B2	B2	B4	B5	B6	B7
0x14ebd0d8	PGN	0	Rate	Repeat	0	0	0	Profile

The PGN indicates the data type to retrieve. When configuration data is retrieved, the PGN determines the type of the data and in the cases of charge parameters, the “profile” field is used as an index into the data. The Rate and Repeat fields are ignored. The following configuration data is supported:

PGN	data	
0x00	Options	evcc, bms, and inst options
0x01	Obcharge1	Assigned chargers, termc
0x02	Obcharge2	Maxv, maxc, termt
0x03	Obcharge3	Linev, linvc, EVSEWaitT
0x04	Obcharge4	Charger phase
0x05	Obcharge5	Finmaxv, finmaxc, fintemt
0x06	Obcharge6	Fltmaxv, fltmaxc, flttermt
0x11	BMSCCFG1	Hvc, lvc, bvmin
0x12	BMSCCFG2	Hvcc, lvcc, hvclpdelay, lvclpdelay
0x18	GRPMAP0	Group map configuration
0x19	GRPMAP1	Group map configuration
0x1a	GRPMAP2	Group map configuration

J1939 READ Reply - Configuration Data

The J1939 implementation for configuration data uses identical message structures as for the CANopen implementation, described above, only the CAN ID changes. In particular, the numbering of the J1939 PGN is the same as for the CANopen “control” field, and the structure of the CAN message body for the reply is identical.

Message Type	EID	Message Body
Options	0x14ff00d0	See PDO1 MISO – MCU Configuration Data Reply
Obcharge1	0x14ff01d0	“
Obcharge2	0x14ff02d0	“
Obcharge3	0x14ff03d0	“
Obcharge4	0x14ff04d0	“
Obcharge5	0x14ff05d0	“
Obcharge6	0x14ff06d0	“
MCUCFG	0x14ff10d0	“
BMSCCFG1	0x14ff11d0	“
BMSCCFG2	0x14ff12d0	“
GRPMAP0	0x14ff18d0	“
GRPMAP1	0x14ff19d0	“
GRPMAP2	0x14ff1ad0	“

J1939 WRITE Request – Configuration Data

The J1939 implementation to write configuration data uses similar structures as for the J1939 READ reply, described above. The EID used is different, however the message bodies for the write request is identical to the J1939 READ Reply – Configuration Data, just described.

J1939 WRITE	B0	B1	B2	B2	B4	B5	B6	B7
0x14efd0d8	PGN	<identical to J1939 READ Reply, above>						

As an acknowledge of a successful write operation, the MCU will return a J1939 READ Reply message.

J1939 READ Request – State Data

The J1939 Read Request is used by the manager to request data. In order to optimize the message traffic for display, one request may result in multiple replies sent with a given period. The Rate and Repeat parameters determine the message rate and number of messages.

J1939 READ	B0	B1	B2	B2	B4	B5	B6	B7
0x14ebd0d8	PGN	0	Rate	Repeat	0	0	0	Profile

J1939 READ Reply - State Data

J1939 Broadcast messages are used by the MCU to report data. The J1939 implementation for state data uses identical message structures as for the CANopen implementation, described above, only the CAN ID changes. In particular, the numbering of the J1939 PGN is the same as for the CANopen “control” field, and the structure of the CAN message body for the reply is identical.

Message Type	EID	Message Body
MCUSUM	0x14ff20d0	See PDO2 MISO - MCU Status Data Reply
PACKSUM	0x14ff21d0	“
CVSUM	0x14ff22d0	“
THSUM	0x14ff23d0	“
SOCSUM	0x14ff24d0	“
CELLMAP0	0x14ff30d0	See PDO3 MISO - Cell Map Summary Reply
CELLMAP1	0x14ff31d0	“

CH4100, ELCON and Stealth Chargers

EIDs are used for these chargers.

EID	type	sent by	msg	notes
0x1806e0f4	STEALTH	MCU	CHARGE_CONTROL	
0x18ff50e0	STEALTH	CHARGER	CHARGE_STATUS	
0x1806e5f4, 0x1806e6f4, 0x1806e7f4, 0x1806e8f4, 0x1806e9f4	ELCON	MCU	CHARGE_CONTROL	Multiple chargers are supported, each with unique ids
0x18ff50e5, 0x18ff50e6, 0x18ff50e7, 0x18ff50e8, 0x18ff50e9	ELCON	CHARGER	CHARGE_STATUS	
0x18e54024, 0x18e54124, 0x18e54224, 0x18e54324	CH4100	MCU	CHARGE_CONTROL	Multiple chargers are supported, each with unique ids
0x18eb2440, 0x18eb2441, 0x18eb2442, 0x18eb2443	CH4100	CHARGER	CHARGE_STATUS	

For message contents, see respective charger documentation.

Cell Voltage Retrieval

The MCU supports proprietary messages to retrieve individual cell voltages. To request cell voltages from an LTC, a manager may send the command MGR_READ_LTCn_CV.

The MGR_READ_LTCn_CV message uses an EID with value:

0x1de080n, where “n” is a hex number from 0 to 15.

Request messages carry no data.

With the LTC12 and LTC18 architectures, values of “n” from 0 to 7 request voltages from LTCs 1 through 8 on IsoSPI A, and values of “n” from 8 through 15 request voltages from LTCs 1 through 8 on IsoSPI B.

Each voltage is in tenths of a millivolt. Examples are given below.

With the A123 architecture, values of “n” from 0 to 15 request voltages from MBB 1 through MBB 16.

Each request will generate from three to five replies based on architecture. Each reply contains up to four cell voltages and is of the form:

B0	B1	B2	B2	B4	B5	B6	B7
CellV_n		CellV_n+1		CellV_n_2		CellV_n+3	

The LTC12 architecture has 12 cells per LTC and will send three replies; the A123 architecture has 13 cells per MBB and will send four replies, and the LTC18 architecture has 18 cells per LTC and will send five replies.

The encoding of the MCU_LTCn_GRPm_CV reply messages is:

0x1dfgg0n, where “gg” is the group number and “n” is the LTC number in the request.

The group numbers are as follows:

- 0x09 – cells 1 through 4
- 0x0a – cells 5 through 8
- 0x0b – cells 9 through 12
- 0x11 – cells 13 through 16
- 0x12 – cells 17 and 18

Example trace:

The manager sends the 0x1de0801 message which requests the data for LTC2.

```

25.8 Tx      01DE0801  0                                ; request
26.3 Rx      01DF0901  8  3A 78 41 78 2F 78 24 78 ; group 1
26.6 Rx      01DF0B01  8  21 78 29 78 2E 78 26 78 ; group 3
26.9 Rx      01DF0A01  8  15 78 22 78 3E 78 2F 78 ; group 2

```

There are three replies for the 12 cells; note that they have arrived “out of order”. These have the following values:

```

C1:  3A 78 = 0x783a = 30778 = 3.0778v
C2:  41 78 = 0x7841 = 30775 = 3.0775v
C3:  2F 78 = 0x782F = 30767 = 3.0767v
C4:  24 78 = 0x7824 = 30756 = 3.0756v

```

```

C5:  15 78 = 0x7815 = 30741 = 3.0741v
C6:  22 78 = 0x7822 = 30545 = 3.0754v
C7:  3E 78 = 0x783E = 30782 = 3.0782v
C8:  2F 78 = 0x782F = 30767 = 3.0767v
C9:  21 78 = 0x7821 = 30753 = 3.0753v
C10: 29 78 = 0x7829 = 30761 = 3.0761v
C11: 2E 78 = 0x782E = 30766 = 3.0766v
C12: 26 78 = 0x7826 = 30758 = 3.0758v

```

Thermistor Retrieval

The MCU supports proprietary messages to retrieve thermistor readings. To request thermistor temperatures from an LTC, a manager may send the command `MGR_READ_LTCn_CV`.

The `MGR_READ_LTCn_TH` message uses an EID with value:

`0x1de0c0n`, where “n” is a hex number from 0 to 15.

Request messages carry no data.

With the LTC12 and LTC18 architectures, values of “n” from 0 to 7 request thermistor readings from LTCs 1 through 8 on IsoSPI A, and values of “n” from 8 through 15 request thermistor readings from LTCs 1 through 8 on IsoSPI B. With the A123 architecture, values of “n” from 0 to 15 request thermistor readings from MBB 1 through MBB 16.

A request will generate one or two replies based on device. A reply can contain up to six thermistor readings. LTC12 devices have one on-die thermistor and support for five external thermistors and will send one reply; LTC18 devices have one on-die thermistor and support for nine external thermistors and will send two replies; the A123 architecture has one thermistor per MBB and will send one reply.

Thermistor values are reported in Centigrade. They are encoded into a single 8-bit signed number; with the values of 0x80 unused and the values 0x7f and 0x81 reserved:

0x00 - 0x7e	: temperature readings from 0C to 126C
0x7f	: connection is shorted
0x80	: unused
0x81	: no thermistor detected
0x82 – 0xff	: temperature readings from -126 to -1C

In practice, the accuracy below 0C will be low.

Example trace:

In this example, there is an LTC18 device on ISOSPI B. The manager sends the `0x1de0c08` message which requests the data for LTC B1.

```

25.8 Tx      01DE0C08 0                                ; request B1
26.3 Rx      01DF0C08 8 15 14 14 81 81 81            ; group 1
26.6 Rx      01DF1008 8 13 13 13 81 00 00            ; group 2

```

Note that thermistor reply messages only have a six byte payload. The die temperature is given in the first byte of the 0x1df0c08 message (0x15 = 21C). In the example, external thermistors 1 and two are connected, thermistors 3 to 5 are not. As this device is an LTC18, a second message is present, and it gives the thermistor readings for thermistors 6 through 9. In the example, thermistors 6, 7, and 8 (with a value of 0x13 = 19C). The 9th thermistor is not connected, and the message has two unused bytes.

SERIAL PORT MESSAGES

There are several types of messages that are autonomously sent to the serial port output:

Startup Banner

When the MCU is powered up, it prints a banner message with the version number.

```
*****
*                MCU - Dilithium Design                *
*                v1.0.91; Mar 20 2021 11:10:29          *
*****
```

LTC Discovery Messages

On startup, the MCU automatically discovers how many measurement devices are connected and prints a summary:

```
ltc A1 detected
ltc A2 detected
ltc A3 detected
ltc A4 detected
ltc A5 detected
ltc A6 detected
```

Changes in topology and diagnostic failures of the LTCs (or MBBs) are printed:

```
ltc A1 detected
ltc A2 detected
ltc A3 detected, diagnostic failure=0x1000
ltc A4 detected
ltc A5 NOT DETECTED!
ltc A6 NOT DETECTED!
```


Cell HVC and LVC Messages

When a cell reports an HVC or LVC alert, it is printed:

```
1/c25: in HVC !
1/c26: in HVC !
1/c25: OK
1/c26: OK
1/c12: in LVC !
1/c11: in LVC !
```

Thermistor Temperature Messages

When an enabled thermistor temperature exceeds the **thmax** threshold (or drops below the **thmin** threshold) a message is printed. The message includes the LTC number and thermistor number.

```
ltc: A2:T3 HITEMP!
```

Then, when it recovers a message is printed as follows.

```
ltc A2:T3 18C
```

IsoSPI Error Messages

If an IsoSPI error occurs, a debug dump is noted:

```
bms> 0103 000f ffff ffff ffff ffff 4c66
      2703 000f ffff ffff ffff ffff 4c66
      0003 000f ffff ffff ffff ffff 4c66
      0203 000f ffff ffff ffff ffff 4c66
      2703 000f ffff ffff ffff ffff 4c66
      0203 000f ffff ffff ffff ffff 4c66
      0303 000f ffff ffff ffff ffff 4c66
      2703 000f ffff ffff ffff ffff 4c66
      0203 000f ffff ffff ffff ffff 4c66
      0903 000f ffff ffff ffff ffff 4c66
      2703 000f ffff ffff ffff ffff 4c66
      0703 000f ffff ffff ffff ffff 4c66
```

SERIAL PORT COMMANDS

Serial port commands are organized into five groups or “contexts”: “SYStem,” “BMS,” “EVCC,” “INSTrumentation,” and “MCU.”

The prompt shows the current context. To start, the user is in the MCU context and the prompt is “**mcu>**.”

- The **system** command navigates to the SYSTEM context.
- The **bms** command navigates to the BMS context.
- The **evcc** command navigates to the EVCC context.
- The **inst** command navigates to the INSTrumentation context.
- The **mcu** or **x** command (shortcut for eXit) navigate to the MCU context.

The contexts follow the functional partitioning of the MCU. The commands **help**, **show**, **show config**, and **reset config**, depend on context. The command **help** may be typed in any context and will give a list of commands available at that context. So, for example, the command **show config** in the EVCC context will print out charging related configuration and the command **reset config** in the EVCC context will only reset EVCC related configuration.

In many cases, command shortcuts are available. This is shown with the use of uppercase and lowercase letters in the command help. For example, the help lists the command **SHoW CONfig**. The case of the letters indicates that this command may be entered as: **show config**, **sh config**, or simply **sh con**.

Although certain commands like **help** or **show config** are context dependent, most commands can be typed in any context. It is possible, for example, to configure EVCC parameters while in the SYSTEM context.

help

The **help** command prints context dependent help. This command does not take any options.

Generally, **help** prints out the “command verbs” (such as **set** and **show**) and “command objects” (such as **config** and **version**) that are available in the given context.

The help system is hierarchical. The “top level” help generally does not define all parameters or the precise syntax of the commands. More information by typing a partial command. For example, by typing the command **set** (in the SYSTEM context) will give a list of commands available:

```
sys> set
  SET INn <ifunc>      - map IN1 .. IN5 to input function
  SET OUTn <ofunc>     - map OUT1 .. OUT5 to output function
  SET CAN1BR <br>      - set CAN baud rate for CAN1
  SET CAN2BR <br>      - set CAN baud rate for CAN2
```

In this example, the list of available of “input functions” for the command **set in1** can be printed by typing:

```
sys> set in1
syntax: 'set in1 [HALL1|HALL2|LOOP|-LOOP|PSELECT|PLUGIN|SOCRES|-SOCRES|NONE]'
```

At the MCU context, the **help** command shows hints about navigation and prints a few globally applicable commands.

mcu> help

```
mcu> help
--navigation-----
  SYS                - System
  BMS                 - Battery Management
  EVCC                - Charging
  INST                - Instrumentation
  eXit                - exit to MCU
```

```
--status and configuration-----
  SHoW                - MCU status
  SHoW CONfig          - MCU configuration
  REset CONfig         - reset configuration to defaults
--firmware and database-----
  SHoW VERsion         - firmware version
  UPGRADe              - upgrade firmware
```

Below, help output for each context (SYS, BMS, EVCC and INST) is given.

sys> help

```
sys> help
--system-----
  SHoW                - system status
  SHoW CONfig          - system configuration
  REset CONfig         - reset system configuration
  SHoW INputs          - MCU input status
  SHoW OUTputs         - MCU output status
  SHoW CAN             - CAN interface status
--hardware configuration-----
  SEt INn <ifunc>       - map IN1 .. IN5 to input function
  SEt OUTn <ofunc>      - map OUT1 .. OUT5 to output function
  SEt CAN1BR <br>       - set CAN baud rate for CAN1
  SEt CAN2BR <br>       - set CAN baud rate for CAN2
--services-----
  ENable OBDII         - OBDII client
  ENable J1939         - J1939 client
  ENable CANOPEN       - CANOPEN client
  ENable CAN1DISFIL    - enable SID filter on CAN1
  DISable <service>    - disable a service
--diagnostics-----
  MEasure <in>          - measure an input, <in> := [IN1 .. IN5, PROX, PILOT, KSI]
  ENable <out>          - enable an output, <out> := [OUT1 .. OUT5]
  DISable <out>         - disable an output
  TRace <target>        - enable debug message trace
  TRace- <target>       - disable debug message trace
  TRace OFF            - turn off tracing
```

bms> help

Note that **help** at the bms prompt is architecture dependent; the first line of the help indicates the architecture.

```
bms> help
--bms (LTC12)-----
  SHoW                - BMS status
  SHoW CONfig          - BMS configuration
  REset CONfig         - reset BMS configuration to defaults
  SHoW LTC             - Linear Technologies measurement chips
  SHoW CMAP            - cell group map
  SHoW CELLS           - cell data
  SHoW STATS           - cell statistics
  SHoW THERMISTOR       - thermistors
--configuration-----
  SEt ARCH <arch>       - set BMS architecture
  SEt CMAP <l> <p> <g> - map ltc to cell group
  LOCK                - lock cell group map and cells
  -- discharge current limit --
  SEt DCL <a>          - discharge current limit in amps
  SEt DCLMINTMP <t>    - DCL min temp
```

```

    SET DCLMAXTMP <t>      - DCL max temp
    SET DCLDRLOTMP <t>     - DCL derate lo temp
    SET DCLDRHITMP <t>     - DCL derate hi temp
-- per-cell --
    SET HVC <v>            - high voltage cutoff threshold
    SET LVC <v>            - low voltage cutoff threshold
    SET HVCDELAY <s>       - delay asserting HVC after detection
    SET LVCDELAY <s>       - delay asserting LVC after detection
    SET HVCC <v>           - HVC clear threshold
    SET LVCC <v>           - LVC clear threshold
    SET BVMIN <v>          - discharge balancing voltage minimum
    SET LOWPACKV <v>       - low pack voltage threshold
    SET HIPACKV <v>        - high pack voltage threshold
-- thermistor --
    SET THMAX <t>          - thermistor max temperature
    SET THMIN <t>          - thermistor max temperature
    ENable THERMISTOR      - enable thermistors
    DISable THERMISTOR     - disable thermistors
--diagnostics-----
    REset STATS            - reset cell statistics

```

evcc> help

```

evcc> h
--evcc-----
    SHow                  - charge status
    SHow CONfig            - charge configuration
    REset CONfig           - reset EVCC configuration
--configuration-----
    SET PLUG <p>           - charge plug type
    SET EVSEWAIT <m>       - EVSE wait time
    SET CHARGER ...        - defines charger 1
    SET CHARGER2 ...       - defines charger 2
    SET CHARGER3 ...       - defines charger 3
    SET CHARGER4 ...       - defines charger 4
-- bulk charge parameters--
    SET LINEV <v>          - line voltage
    SET LINEC <c>          - line current
    SET MAXV <v>           - maximum charging voltage
    SET MAXC <a>           - maximum charging current
    SET TERMC <a>          - termination charging current
    SET TERMT <s>          - maximum charge time
--lead acid charge parameters--
    SET FIN_MAXV <v>       - finishing charge max voltage
    SET FIN_MAXC <a>       - finishing charge max current
    SET FIN_TERMT <s>      - finishing charge max charge time
    SET FLT_MAXV <v>       - float charge max voltage
    SET FLT_MAXC <a>       - float charge max current
    SET FLT_TERMT <s>      - float charge max charge time
--profiles--
    EDIT PROFILE <n>       - choose profile to edit; <id> := [1|2|3|4|OFFBOARD]
    DELETE PROFILE <n>     - choose profile to delete; <id> := [2|3|4]
    SET PMAP <ps> <id>    - map pselect to profile
--options--
    ENable ZEVCCS <cif>    - ZEVCCS charging
    ENable CHARGEMO <cif>- CHARGEMO charging
    ENable OBCHARGE <cif>- onboard charging
    ENable TOPBALANCE      - see documentation
    ENable SLOWSTART       - see documentation
    DISable <option>       - disable an option
--diagnostics-----
    SHow HISTory           - charge history

```

```

REset HIStory      - reset charge history
TRace CHarger      - enable onboard charger tracing
TRace STATE        - enable onboard charger state tracing
TRace- <t>          - disables trace
TRace OFF          - disables all tracing

```

inst> help

inst> **help**

```

--instrumentation-----
  SHow              - instrumentation status
  SHow CONFIG        - instrumentation configuration
  REset CONFIG       - reset INST configuration
--options-----
  ENable ISENSORREV  - reverse polarity on current sensor
  ENable 3100R <cif>  - Curtis 3100R Gauge
  ENable IVTSAFACTORY - IVT-S factory settings
  ENable SH_FUEL <cif> - Speedhut Fuel Gauge (0x355)
  ENable SH_VAT <cif>  - Speedhut Volts/Amps/Temp Gauge (0x356)
  ENable SH_VATR <cif> - Speedhut VAT; Amps reverse polarity
  ENable SH_TEMP <cif> - Speedhut Temperature Gauge (0x357)
  ENable DCDC <cif>   - DC/DC converter
  DISable <option>    - disable an option
--configuration-----
  SET ISENSOR <m>      - current sensor model
  SET ISENSORADJ <n>   - sensor adjustment in percent
  SET PACKKWH <kwh>    - pack capacity in KWh
  SET FUEL <s> <p>      - fuel gauge PWM setpoint
  SET TEMPG <s> <p>     - temp gauge PWM setpoint
  SET PWMPERIOD <p>    - PWM period, <p>:= [150|275|400]
  SET FANONTEMP <t>    - temp to set the FAN output ON, <t>:= [0..100]
  SET FANOFFTEMP <t>   - temp to set the FAN output OFF
  SET 3100RBR <br>     - 3100R CAN Baud Rate, <br> := [125|250|500]
  SET DDTYPE <t>       - DCDC Converter

```

show

The **show** command is used to print out configuration and status. The **show** and **show config** commands, without additional parameters, are context dependent. The **show** command can also take a parameter. The following table lists the possible options.

show / MCU	
show	MCU Status
show config	MCU Configuration
show version	Firmware version number
show / SYSTEM	
show	System status
show config	System configuration
show can	CAN configuration and status
show input	Input configuration and status
show output	Output configuration and status
show / BMS	
show	BMS status

show config	BMS configuration
show cmap	Cell Map configuration
show ltc	LTC configuration (LTC12 and LTC18 architectures only)
show mbb	MBB configuration (A123 architecture only)
show cells	Cell status
show stats	Cell statistics
show thermistor	Thermistor status
show / EVCC	
show	Charging status
show config	Charging configuration
show history	Charge history
show / INST	
show	Instrumentation status
show config	Instrumentation configuration

mcu> show

```
mcu> show
pack
  voltage : 229.50v
  cells   : 72
  mean    : 3.186v
  std dev : 0.008v
  alerts  : not locked
  current : 27.2A
  SOC     : 100%
  uptime: 7 hour(s), 36 minute(s), 57 second(s)
```

sys> show

```
sys> show
--inputs-----
in1  2.54v  hall1  DHABS137 H = 23.5A
in2  3.23v  hall2  DHABS137 L = 27.3A
in3  1.63v
in4  4.77v
in5   inf
prox inf    J1772 connected
pilot      not detected
ksi        1 detected
--outputs-----
out1  240    fuelg    soc = 100%
out2  ---
out3  ---
out4  GND    hvc lvc NOTLOCKED lowpack hipack
out5  ---    charge disabled
pilot      EVSE disabled
--can1-----
canlbr : 250 Kbps
services : obcharge, canopen
tx       : 16223 [16219, 1, 1, 2], overflow= 5
rx       : 3443 [3443, 0] overflow= 5, RX Passive
--can2-----
```

```

can2br      : 250 Kbps
services    :
tx          : 64500 [64500, 0, 0], TX Passive, abt= 241, err1= 64499
rx          :      0 [0, 0], RX Early Warn

```

bms> show

```

bms> show
pack
  voltage : 229.50v
  cells   : 72 (not locked)
  mean    : 3.186v
  std dev : 0.008v
  alerts  : not locked

```

evcc> show

When not charging, **show** would print:

```

evcc> show
state      : STANDBY
plug state: DISCONNECTED

```

When charging, **show** prints additional information:

```

evcc> show
state      : CHARGE
plug state: CONNECTED
chg time   : 1 mins
duty cycle = 49%, 29.4A available
voltage(t) : 120.0V
current(t)  : 12.0A
charger    : tsm2500
  status    : 19 msgs sent; 20 msgs received
  voltage   : 117.0V
  current   : 12.8A
  charge    : 3.60Wh

```

inst> show

```

inst> show
soc        : 61%
SOC (Vest) : 63%
pack wh/   : 19741/32.0kwh [19200+271]

```

mcu> show config

At the MCU context, **show config** concatenates all configuration from all contexts.

sys> show config

At the SYS context, **show config** shows the SYS configuration:

```

sys> show config
--inputs-----
in1      : hall1
in2      : hall2
--outputs-----
out1     : fuelg
out5     : charge
--can-----

```

```

can1br    : 250 Kbps
can2br    : 250 Kbps
--services-----
canopen   : enabled
obcharge  : can1

```

bms> show config

At the BMS context, **show config** shows the BMS configuration:

```

bms> show config
arch      : LTC12
hvc       : 3.400v
lvc       : 2.400v
bvmin     : 3.000v
thmax     : 50C

```

evcc> show config

In the EVCC context, **show config** command shows EVCC configuration. The output of **show config** becomes progressively more complex as more features are enabled.

A simple example with only onboard charging defined:

```

evcc> show config
--charging-----
plug      : J1772
charger   : tsm2500
maxv      : 20.0V
maxc      : 2.0A
termc     : 2.0A
termt     : 72.0hr

```

An example with additional parameters defined:

```

evcc> show config
--charging-----
plug      : J1772
charger   : tsm2500
charger2  : tsm2500_42
charger3  : tsm2500_43
charger4  : elcon
linev     : 220.0V
linec     : 30.0A
evsewait  : 4.0hr
maxv      : 155.0V
maxc      : 15.0A
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   : topbalance
           : slowstart

```


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

Example output with multiple charge profiles and a profile switch enabled is shown below:

```
evcc> sh config
--charging-----
plug      : J1772
profile   :      1          2 (*)          3          4
charger   : tsm2500      tsm2500
linev     :                220.0V
linec     :                25.5A
maxv      :      20.0V      20.0V
maxc      :      0.0A      2.0A
termc     :      2.0A      2.0A
termt     :      72.0hr      72.0hr
pmap      :
  inf     :      x
  20K     :      x
  5K      :      x
  0       :      x
```

inst> show config

```
inst> show config
--soc-----
isensor    : DHABS137
  hall1    : in1
  hall2    : in2
packkwh    : 20.0
--gauges-----
fuelg      : out1, pwm (0:95 25:125 50:150 75: 175 100:240)
```

show version

```
mcu> show version
v1.0.4; Sep  2 2021 11:10:29
```

show can

```
sys> show can
--can1-----
can1br     : 250 Kbps
services   : obcharge, canopen
tx         : 16219 [16219, 0, 0, 0]
rx         : 3443 [3443, 0]
--can2-----
can2br     : 250 Kbps
services   :
tx         :      0 [0, 0, 0]
rx         :      0 [0, 0]
```

show inputs

```
sys> show inputs
in1  2.54v hall1 DHABS137 H = 23.5A
in2  3.23v hall2 DHABS137 L = 27.0A
in3  1.62v
in4  4.77v
in5  inf
prox inf J1772 disconnected
```

```

pilot      not detected
ksi        1      detected

```

show outputs

```

sys> show outputs
out1  240    fuelg    soc = 100%
out2  ---
out3  ---
out4  ---
out5  ---    charge disabled
pilot      EVSE disabled

```

show cmap

```

bms> show cmap
ltc|pack|group| cells
-----|-----|-----|-----
A1 | 1 | 1* | (c1 -c12 ) . . . . .
A2 |  | 2* | (c13 -c24 ) . . . . .
A3 |  | 3* | (c25 -c36 ) . . . . .
A4 |  | 4* | (c37 -c48 ) . . . . .
A5 |  | 5* | (c49 -c60 ) . . . . .
A6 |  | 6* | (c61 -c72 ) . . . . .

```

The LTC number, pack, and group are shown. If the group has an asterisk (*) following, then the assignment was automatic.

Each cell entry is represented as a character, according to the following rule:

<space>	the cell was neither detected nor configured.
.	the cell was detected but not configured
?	the cell was configured but not detected
X	the cell was configured and detected

show ltc

The **show ltc** command is used for the LTC12 and LTC18 architectures.

```

bms> sh ltc
ltc| status | rev | Van | Vdig | Vref | SOC | SUM | conf | disc
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----
A1 | ok | 6811-6 | 5.094v | 3.206v | 3.006v | 38.19v | 38.18v | 0x00000 | 0x00fff
A2 | ok | 6811-6 | 5.130v | 3.209v | 3.003v | 38.19v | 38.18v | 0x00000 | 0x00fff
A3 | ok | 6811-6 | 5.042v | 3.214v | 3.004v | 38.25v | 38.24v | 0x00000 | 0x00fff
A4 | ok | 6811-6 | 5.087v | 3.210v | 3.003v | 38.24v | 38.23v | 0x00000 | 0x00fff
A5 | ok | 6811-6 | 5.070v | 3.219v | 3.004v | 38.32v | 38.32v | 0x00000 | 0x00fff
A6 | ok | 6811-6 | 5.098v | 3.191v | 3.003v | 38.36v | 38.36v | 0x00000 | 0x00fff

```

The **status** value should be **ok**. If not, then this field will show a hex failure value.

The **rev** shows the LTC chip hardware revision. Expect to see either ltc6804, ltc6811, and ltc6813

The values **Van**, **Vdig**, **Vref** are internal voltage measurements on the chip. The nominal values are 5v, 3v, and 3v respectively.

The values of **SOC** and **SUM** is the total voltage managed by the chip. These numbers are calculated in two different ways. The LTC can measure the cell group voltage (SOC) and the MCU adds up individual cell measurements (SUM). These numbers should be close to each other.

The **conf** is a hexadecimal representation of which cells have been configured for this LTC using the **lock** command. The **disc** field is a hexadecimal representation of which cells have been discovered. Note that an equivalent, more user friendly, representation of this data is given in **show cmap**.

show mbb

The **show mbb** command is used for the A123 architecture.

```
mcu> show mbb
mbb| status | min | max | mean | flt_xx | conf | disc
----|-----|-----|-----|-----|-----|-----|-----
1 | OK | 3.119v | 3.142v | 3.133v | in | 0x01fff | 0x01fff
2 | OK | 3.110v | 3.144v | 3.132v | in | 0x01fff | 0x01fff
3 | OK | 3.126v | 3.189v | 3.177v | in | 0x01fff | 0x01fff
5 | OK | 3.126v | 3.189v | 3.177v | in | 0x01fff | 0x01fff
```

show cells

The **show cells** command lists current cell voltage measurements and status.

```
bms> show cells
c1 - 3.042v +      c13- 3.027v +      c25- 2.718v -      c37- 2.713v -
c2 - 3.472v >HVC  c14- 2.995v      c26- 2.918v      c38- 2.706v -
c3 - 3.421v >HVCC c15- 3.054v +      c27- 2.959v      c39- 2.908v
c4 - 3.012v +      c16- 2.993v      c28- 3.080v ++    c40- 2.913v
c5 - 3.052v +      c17- 3.045v +      c29- 2.706v -      c41- ??????
c6 - 3.001v +      c18- 2.991v      c30- 2.910v      c42- ??????
c7 - 2.799v      c19- 2.766v      c31- 2.708v -      c43- ??????
c8 - 2.788v      c20- 2.773v      c32- 2.920v      c44- ??????
c9 - 2.774v      c21- 2.777v      c33- 2.714v -      c45- -----
c10- 2.771v      c22- 2.783v      c34- 2.980v      c46- -----
c11- 2.783v      c23- 2.783v      c35- 2.722v -      c47- -----
c12- 2.788v      c24- 2.770v      c36- 2.917v      c48- -----
```

If the cell value shown is "-----" then the cell voltage is less than 20mv.

If the cell value shown is "??????" then the is configured, but not present.

An individual cell voltage may be an Indicator Flag. There is room for only one Indicator Flag,

Indicator Flag	Description
>HVC	Cell in HVC
>HVCC	Cell > HVCC
<LVCC	Cell < LVCC
<LVC	Cell in LVC
++	SDM is > 1.5
+	SDM is > 1.0
-	SDM is < 1.0
--	SDM is < 1.5

show stats

The **show stats** command lists current cell statistics.

Mean cell voltage and **std deviation** (standard deviation) are calculated over all cells. The **min** and **max** values are the lowest and highest value of the cell since power on or since last time **reset stats** was entered.

The **variance** is the difference, in volts, of the current cell voltage with the average cell voltage. For an explanation of **sdm**, **min**, **max**, and **delta**, see the text.

```

bms> sh stats
total|-mean cell voltage-----|-standard deviation-----
      | 3.187v                      | 0.006v
pack1|-voltage---min---max--|----deviation-----min---max--delta-
c1   | 3.183v   3.182v 3.183v | -0.004v -0.7s -0.7s -0.6s 0.1s
c2   | 3.186v   3.185v 3.186v | -0.002v -0.3s -0.3s -0.2s 0.1s
c3   | 3.187v   3.186v 3.187v | -0.000v +0.0s +0.0s +0.0s 0.0s
c4   | 3.182v   3.181v 3.182v | -0.005v -0.8s -0.8s -0.7s 0.1s
c5   | 3.196v   3.195v 3.196v | 0.008v +1.3s +1.2s +1.3s 0.1s
c6   | 3.181v   3.180v 3.181v | -0.006v -0.9s -1.0s -0.9s 0.1s
c7   | 3.184v   3.183v 3.184v | -0.003v -0.5s -0.5s -0.4s 0.1s
c8   | 3.182v   3.181v 3.182v | -0.006v -0.9s -0.9s -0.8s 0.1s
c9   | 3.183v   3.182v 3.183v | -0.005v -0.7s -0.7s -0.6s 0.1s
c10  | 3.186v   3.185v 3.186v | -0.002v -0.3s -0.3s -0.2s 0.1s
c11  | 3.185v   3.184v 3.185v | -0.003v -0.4s -0.5s -0.3s 0.2s
c12  | 3.182v   3.181v 3.182v | -0.005v -0.8s -0.9s -0.7s 0.2s
c13  | 3.183v   3.182v 3.183v | -0.004v -0.7s -0.7s -0.6s 0.1s
c14  | 3.186v   3.185v 3.186v | -0.002v -0.2s -0.3s -0.2s 0.1s
c15  | 3.188v   3.187v 3.188v | 0.001v +0.0s +0.0s +0.0s 0.0s
c16  | 3.186v   3.185v 3.186v | -0.001v -0.2s -0.2s -0.1s 0.1s
c17  | 3.186v   3.185v 3.186v | -0.001v -0.2s -0.3s -0.2s 0.1s
c18  | 3.183v   3.181v 3.183v | -0.005v -0.7s -0.8s -0.7s 0.1s
c19  | 3.187v   3.187v 3.188v | -0.000v +0.0s +0.0s +0.0s 0.0s
c20  | 3.182v   3.182v 3.183v | -0.005v -0.8s -0.8s -0.7s 0.1s
c21  | 3.185v   3.184v 3.185v | -0.003v -0.4s -0.4s -0.3s 0.1s
c22  | 3.186v   3.185v 3.186v | -0.001v -0.2s -0.2s -0.1s 0.1s
c23  | 3.187v   3.186v 3.187v | -0.000v +0.0s +0.0s +0.0s 0.0s
c24  | 3.185v   3.184v 3.185v | -0.003v -0.4s -0.4s -0.3s 0.1s

```

show thermistor

```
bms>sh th
ltc|  T0 |  T1 |  T2 |  T3 |  T4 |  T5
-----|-----
A1 | 26C+      NC      NC
A2 | 23C      18C      18C      18C      18C
A3 | 23C
A4 | 23C
```

Value	Discovered	Configured	TCENSUS	HITEMP	LOTEMP
<blank>					
NC	no	yes	yes		
SHORT	yes	yes	yes		
(xx)	yes	no			
xx+	yes	yes		yes	
xx-	yes	yes			yes

show history

The **show history** command displays data about the last sixteen charge cycles.

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

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

The following example shows charge history, with different “termination reasons”. The termination reason contains the reason that the charge cycle stopped. In this example, in the most recent charge attempt, the user disconnected the J1772 plug one minute after charging started. (EVSE disc, 1 mins). The previous attempt (“-1”) shows a normal charge completion with a charge time of 214 minutes and includes the number of watt hours delivered.

```
evcc> show history
| term | charge |      | watt | maximum| maximum| ending|
num | reason | time | charger | hours | voltage| current| current|
-----|-----
last | hitemp | 15 mins|tsm2500 | 383Wh| 120.0V | 12.8A | 12.8A |
- 1 | plug disc| 1 mins|tsm2500 | 19Wh| 120.0V | 12.8A | 12.8A |
- 2 | normal | 214 mins|tsm2500 | 3249Wh| 152.9V | 7.9A | 1.9A |
```

The full set of “term reason” codes is:

Term reason	Description
normal	Normal completion
charge t/o	Charge timeout
plug disc	Charge plug disconnected
hwfault	MCU Hardware alert
ccensus	Cell census alert
tcensus	Thermistor census alert
HVC	High Voltage Cutoff alert
LVC	Low Voltage Cutoff alert
hitemp	Thermistor overtemperature alert

lowtemp	Thermistor undertemperature alert
hipack	High Pack Voltage
notlocked	BMS configuration not locked
cell loop	Cell loop
EVSE t/o	EVSE timeout
chgr t/o	Charger CAN RX timeout
no pack	No pack detected
chgr stop	Charger unexpectedly stopped delivering current
no SEQ1	ChargeMo: no SEQ1 signal
no SEQ2	ChargeMo: no SEQ2 signal
chgr t/o	ChargeMo: charger timeout
sys error	ChargeMo system error
chgr flt	ChargeMo: charger fault
low cur	ChargeMo: output current low
high cur	ChargeMo: output current high
neg cur	ChargeMo: negative current
V mismatch	ChargeMo: voltage mismatch
evincompat	ChargeMo: EV incompatible
ch not on	ChargeMo: Charger not on

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

```
evcc> show history
```

num	term reason	charge time	charger	watt hours	maximum voltage	maximum current	ending 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

The **set** command assigns a value to a parameter. In most cases, this command takes two arguments: a parameter name and a parameter value.

```
set <param> <value>
```

The following table lists the possible options.

set / SYSTEM	Value	Description
set in1 set in2 set in3 set in4 set in5	see <i>Input Functions</i>	Input Function mapping
set out1 set out2 set out3 set out4	see <i>Output Functions</i>	Output Function mapping

set out5		
set can1br	125, 250, 500	CAN1 Baud Rate
set can2br	125, 250, 500	CAN2 Baud Rate
set / BMS		
set arch	1tc12, 1tc18, a123	BMS Architecture
set mbbaddr	1..16	Set mbb address
set cmap	See example below	Cell Map
set hvc	3.0 to 4.6 volts	High Voltage Cutoff Threshold
set lvc	1.8 to 4.0 volts	Low Voltage Cutoff Threshold
set hvcdelay	0 to 10 seconds	Delay before declaring HVC Alert
set lvcdelay	0 to 10 seconds	Delay before declaring LVC Alert
set hvcc	0 to 4.6 volts (0=disable)	High Voltage Cutoff Clear Threshold
set lvcc	0 to 4.0 volts (0=disable)	Low Voltage Cutoff Clear Threshold
set hipackv	0 to 1000 volts (0=disable)	Hi Pack Voltage Threshold
set lowpackv	0 to 800 volts (0=disable)	Low Pack Voltage Threshold
set bvmin	0 to 5.0 volts (0=disable)	Balance Voltage Minimum Voltage
set thmax	0 to 100 degrees C	Thermistor Overtemperature Threshold
set thmin	-20 to 30 degrees C	Thermistor Undertemperature Threshold
set dcl	In amps	Discharge Current limit
set dclmintmp	-20 to 30 degrees C	Hard cutoff
set dclmaxtmp	0 to 100 degrees C	Hard cutoff
set dcldr1otmp	-20 to 30 degrees C	Derate cutoff
set dcldrhitmp	0 to 100 degrees C	Derate cutoff
set / EVCC		
set charger set charger2 set charger3 set charger4	See text below	Set charger type (and optional phase)
set plug	J1772, J1772T2, direct	Plug protocol
set linev	100 to 300 Volts	Line Voltage
set linec	0 to 200 Amps (or J1772)	Line Current
set maxv	0 to 800 Volts	Maximum Charge Voltage
set maxc	0 to 400 Amps	Maximum Charge Current
set termc	1 to 20 Amps	Termination Charge Current
set termt	0 to 672 hours	Termination Charging Time
set fin_maxv	0 to 800 Volts (0=disable)	Finishing Charge Voltage
set fin_maxc	0 to 30 Amps	Finishing Charge Current
set fin_termt	0 to 672 Hours	Finishing Charger Termination Time

set flt_maxv	0 to 800 Volts (0=disable)	Float Charge Voltage
set flt_maxc	0 to 10 Amps	Float Charge Current
set flt_termt	0 to 672 Hours (0="forever")	Float Charger Termination Time
set evsewait	0 to 672 Hours	Time of use "wait time"
set pmap	See text below	Profile Selection Map
set / INST		
set packkwh	0 to 120 KWHr	Pack Capacity
set isensor	Current Sensor Support See Figure	Current Sensor Type
set isensoradj		ISENSOR adjustment
set soc	0 to 100 percent	Sets the SOC value manually
set fuelg <s>	0 to 255	Fuel gauge duty cycle setting for setpoint: 0, 25, 50, 75 and 100.
set tempg <s>	0 to 255	Temp gauge duty cycle setting for setpoint: 0, 25, 50, 75 and 100.
set ddtype	none, volt, delphi	DC/DC Converter type
set ddvoltage	0 to 15.0	DC/DC voltage setpoint
set 3100rbr	125, 250, 500	3100R CAN baud rate

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 chargers are supported:

- **tsm2500**
- **tsm2500_41**
- **tsm2500_42**
- **tsm2500_43**
- **elcon**
- **elcon_e7**
- **elcon_e8**
- **elcon_e9**
- **lear**

The following command sets a single charger

```
evcc> set charger tsm2500
```

The following command sets a second charger

```
evcc> set charger2 tsm2500_42
```

A charger may be deleted using the command

```
evcc> set charger2 none
```


This command is also used to associate a phase with a charger, which is used with J1772 Type 2 charging. The following example defines three chargers and associates each of them with a different phase.

```
evcc> set charger tsm2500 L1
evcc> set charger2 tsm2500_41 L2
evcc> set charger3 tsm2500_42 L3
```

The phase association may be reset to “none” using the command:

```
evcc> set charger tsm2500 none
```

set pmap

This command sets the profile map. The syntax is

```
set pmap <ps> <id>;      <ps> := [inf|20K|5K|0]
                        <id> := [1|2|3|4]
```

So, the following command will map “inf” at pselect to choose profile 2:

```
evcc> set pmap inf 2
```

set cmap (LTC12 or LTC18)

If the BMS architecture is set to LTC12 or LTC18, the **set cmap** is used to set the ltc to cell group mapping. The syntax of this command is

```
set cmap [A|B]<ltc> <pack> <group>
```

where A and B choose the IsoSPI interface, <ltc> is from 1 to 8, <pack> is from 1 to 8 and <group> is from 1 to 8.

set cmap (A123)

If the BMS architecture is set to A123, the **set cmap** is used to set the mbb to cell group mapping. The syntax of this command is

```
set cmap <mbbId> <pack> <group>
```

where <mbbId> is from 1 to 16, <pack> is from 1 to 8 and <group> is from 1 to 8.

enable | disable

The **enable** commands can be used to enable an option (which requires a single parameter, the option name). Or the commands can also be used to enable or disable a CAN service (which requires two parameters: the service name, and the CAN interface). And, finally, these commands can be used to enable or disable thermistors, which supports a variety of possible keywords.

enable disable / SYSTEM	
en dis out1 en dis out2 en dis out3 en dis out4 en dis out5	Manually enable an output
en dis obdii	OBDII service
en dis j1939	J1939 service
en dis canopen	CANopen service
en dis canlsidfil	Enables a filter for SID only traffic on CAN1
enable disable / BMS	
en dis balance	BMS cell discharge balancing
en dis thermistor	see text
en dis a123 <can1 can2>	A123 service
enable disable / EVCC	
en dis zevccs <can1 can2>	ZEVCCS service
en dis chargemo <can1 can2>	Chargemo service
en dis obcharge <can1 can2>	Onboard charger
en dis topbalance	EVCC topbalancing option
en dis slowstart	EVCC slowstart option
enable disable / INST	
en dis isensorrev	Flips polarity of the hall sensor
en dis ivtsfactory	IVT-S factory settings
en dis 3100r <can1 can2>	31004 display service
en dis sh_fuel <can1 can2>	Speedhut fuel gauge
en dis sh_vat <can1 can2>	Speedhut volts / amps / temp gauge
en dis sh_vatr <can1 can2>	Speedhut VAT gauge; amps reverse polarity
en dis sh_temp <can1 can2>	Speedhut temp gauge
en dis dcdc <can1 can2>	DC/DC converter

Example of enabling and disabling options:

```
evcc> enable slowstart
evcc> disable topbalance
```

Some CAN services like CANopen are client services; enabling them will activate them on both CAN1 and CAN2. Requests that arrive on CAN1 will result in a reply on CAN1.

Some CAN services such as the Speedhut gauges are server services; enabling them select which CAN interface the messages are sent and received on.

```
sys> enable obdii
```

enable | disable thermistor

The **enable thermistor** is used to enable thermistor operation. The command **disable thermistor** disables thermistor operation. The list of which thermistors are enabled can be determined by the command **show thermistor**.

If a thermistor is enabled, then if the measured value of the thermistor is greater than **thmax** then the BMS will generate a **HITEMP** alert.

To enable all thermistors that have been detected, use the command:

```
bms> enable thermistor all
```

To enable or disable individual thermistors, use the commands:

```
enable thermistor <ltc> <therm_number>
disable thermistor <ltc> <therm_number>
```

where **<ltc>** is from 1 to 8, and **<therm_number>** is from 1 to 5.

All die thermistors may be enabled using the command:

```
enable thermistor <ltc> die
```

All external thermistors may be enabled using the command:

```
enable thermistor <ltc> ext
```

The thermistor threshold temperature is determined by the configuration parameter **thmax**.

When a thermistor temperature exceeds **thmax**, a **HITEMP** alert is present in the **show** command.

The measured value can be determined by the **show thermistor** command.

Examples:

```
bms> en th 1 1
bms> en th 1 2
bms> en th 2 1
```

edit | delete

The “edit/delete” command verb pair is only used to edit charge profiles.

edit profile <n>

The **edit profile** command can be used to create, if necessary, and set the edit profile. This command takes one parameter, a number from 1 to 4 for the four onboard profiles, or the keyword **offboard** to edit the offboard profile.

```
evcc> edit profile 3
```

Each profile contains the following information:

charger	Charger Type and Phase
charger2	
charger3	
charger4	
evsewait	Time of use charging
linev	Line Parameters
linec	
maxv	Bulk Charge Parameters
maxc	
termc	
termt	
fin_maxv	Finishing Charge Parameters
fin_maxc	
fin_termt	
flt_maxv	Float Charge Parameters
flt_maxc	
flt_termt	

delete profile <n>

The **delete profile** command can be used to delete Profiles 2-4. This parameter takes one parameter, a number from 2 to 4. It is not possible to delete Profile 1.

```
evcc> delete profile 3
```

reset

reset config

The **reset config** command resets configuration to defaults. This command is context dependent. If at the MCU context, all configuration will be reset. Otherwise, only the configuration for the active context is reset.

```
mcu> reset config
SYS config..reset
BMS config..reset
EVCC config..reset
INST config..reset
mcu>
```

reset stats

The **reset stats** command clears out the cell statistics.

Example:

```
bms> reset stats
```

reset history

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

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

measure

The **measure** command is used to verify the A/D inputs. When this command is issued, the MCU will repeatedly measure and print the value of an analog input. The command will run until the user stops the command by typing any character.

The **measure** command takes one argument which can either a Physical Input (IN1, IN2, IN3, IN4, IN5), a fixed function input Proximity, KSI, or can be an Input Function.

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

Examples:

```
sys> measure prox
sys>   in6   395R   prox   J1772 connected
      in6   397R   prox   J1772 connected
      in6   406R   prox   J1772 connected
      in6   410R   prox   J1772 connected
      in6   409R   prox   J1772 connected
      in6   395R   prox   J1772 connected
```

```
sys> measure in4
sys>   in4   1.01v
      in4   1.01v
      in4   1.01v
      in4   1.01v
      in4   1.01v
```

```
sys>
```

```
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
```

trace

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

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

trace charger

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

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

```
00:08:23.1 V=148.6, A= 7.9, W=1173, Wh= 1.12, TMP = 26C
00:08:23.6 V=148.6, A= 7.9, W=1173, Wh= 1.28, TMP = 26C
```

trace can1 | can2

The **trace can1** and **trace can2** 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
```

trace state

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

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

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

trace off

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

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

lock

The **lock** command is used to configure the discovered configuration. See text.

```
bms> lock
configuration locked
```

upgrade

Firmware upgrade is performed by using the serial port and uses a special bootloader application program. To begin the upgrade process, use the upgrade command. For more details, see *Firmware Upgrade*, above.

mcu> **upgrade**

```
***
***                               Starting MCU Upgrade                               ***
*** 1) Exit from the terminal application                                     ***5
*** 2) Start the bootloader and download a new .hex file                   ***
*** 3) Restart                                                             ***
```

Warranty and Support

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

This product is intended for non-commercial use by hobbyists. The warranty does not cover defects arising from miswiring, abuse, negligence, accidents, or reverse engineering. Dilithium Design shall not be responsible for any incidental or consequential damages.

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

Document Revision

Rev 1.0	Mar 2021	Initial Version
Rev 1.1	Sep 2021	Production Release
Rev 1.2	Nov 2021	Minor Edits
Rev 1.3	Aug 2022	Updated Output Mapping, added tempg
Rev 1.4	Aug 2023	Added ChargeMo DCFC details, typos, corrections
Rev 1.5	Dec 2023	Additional CAN interface documentation