NUVATION ENERGY

Technical Support
support@nuvationenergy.com

Sales Inquiries
sales@nuvationenergy.com
855-261-0507

NUVATION BMS™

Software Reference Manual

Single Stack

2018-10-08, Rev. 2.0, Babbage 18.08

Table of Contents

Important Safety Information . . . 1
1. Introduction . . . 2
  1.1. About this Guide . . . 2
2. Background and Terminology . . . 3
  2.1. Battery Topology . . . 3
  2.2. Register Data Model . . . 3
    2.2.1. Registers and Components . . . 3
    2.2.2. Index versus Location . . . 4
    2.2.3. Register Expressions . . . 4
  2.3. Configuration File . . . 7
3. Configuration Settings . . . 7
  3.1. Units . . . 7
  3.2. Battery Parameters . . . 8
    3.2.1. Stack Capacity . . . 8
    3.2.2. Stack Cycle Count . . . 8
    3.2.3. Voltage Full and Empty Thresholds . . . 8
    3.2.4. Current Full Threshold . . . 9
  3.3. Stack Topology . . . 9
    3.3.1. Cell Inputs . . . 9
    3.3.2. Thermistor Inputs . . . 10
  3.4. Operational Limits . . . 10
    3.4.1. Hysteresis Triggers . . . 11
    3.4.2. Cell Voltage Thresholds . . . 12
    3.4.3. Thermistor Temperature Thresholds . . . 13
    3.4.4. Module Temperature Thresholds . . . 14
    3.4.5. Stack Current Thresholds . . . 15
    3.4.6. Stack Voltage Thresholds . . . 15
    3.4.7. External Controller Heartbeat . . . 17
  3.5. Control Settings . . . 17
    3.5.1. Stack Switch Functions . . . 18
    3.5.2. Stack Current Limits . . . 20
    3.5.3. Passive Cell Balancing . . . 21
  3.6. Input/Output Assignment . . . 22
    3.6.1. Nuvation High-Voltage BMS™: Input/Output Assignment . . . 22
    3.6.2. Nuvation Low-Voltage BMS™: Input/Output Assignment . . . 23
    3.6.3. Contactor Outputs . . . 24
    3.6.4. Digital Outputs . . . 25
    3.6.5. Digital Inputs . . . 26
  3.7. Protocol Settings . . . 27
    3.7.1. CAN Bus . . . 27
    3.7.2. RS-485 Modbus RTU . . . 34
  3.8. Measurement Calibration . . . 34
    3.8.1. Thermistor Calibration . . . 34
    3.8.2. Stack Current Calibration . . . 36
    3.8.3. Stack Voltage Calibration . . . 38

3.9. Hardware Settings . . . 38
3.9.1. Module Specific Selection . . . 39
3.9.2. LinkBus Scan Period . . . 39
3.9.3. Fault Pilot Signal . . . 40
3.9.4. Under Voltage Shutdown . . . 41
4. Troubleshooting . . . 42
4.1. Faults and Initialization . . . 42
4.1.1. Cell Voltage Faults . . . 42
4.1.2. Stack Voltage Faults . . . 43
4.1.3. Combined Voltage Faults . . . 43
4.1.4. Thermal Faults . . . 43
4.1.5. Current Faults . . . 44
4.1.6. Precharge Faults . . . 44
4.1.7. Contactor Faults . . . 45
4.1.8. Breaker Faults . . . 45
4.1.9. Watchdog Faults . . . 46
4.1.10. Miscellaneous Faults . . . 46
4.2. Lost/Forgotten IP Address . . . 47
4.2.1. Wireshark (Windows/Linux) . . . 47
4.2.2. Netdiscover (Linux only) . . . 47
5. Registers . . . 48

Important Safety Information

The content in this document must be followed in order to ensure safe operation of Nuvation BMS™.

For Nuvation High-Voltage BMS™, do **NOT** energize the system until all connections to the Cell Interface and Power Interface modules have been made.

Insulated handling is required of any connector carrying potentials over 600Vdc relative to chassis.

For Nuvation Low-Voltage BMS™, do **NOT** connect the `J7: Current Shunt / +V Power` connector to the Battery Controller until all other connections have been made.

Properly insulate or remove any unused wires. Unused wires can couple excessive system noise into Nuvation BMS which can disrupt communication and lead to undesirable behaviors.

Please be aware of high voltages present in your system and follow all necessary safety precautions.

The provided module enclosures are not fire enclosures.

Depending on battery chemistry, there might be a nominal voltage per cell which adds up in series and is always present. There are many different battery chemistries with different current capacities, and so high voltage with high current capacity may be present while connecting the Nuvation BMS. You must use proper electrical safety precautions when handling any part of the Nuvation BMS. Neither Nuvation Energy or any of its employees shall be liable for any direct, indirect, incidental, special, exemplary, personal or consequential harm or damages (including, but not limited to, procurement or substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this product.

The Nuvation BMS relies on your system charger to charge the battery cells; do not leave your charger off while the Nuvation BMS is powered from the stack for prolonged periods of time. The Nuvation BMS should be shut down when the system is in storage to minimize the drain on the cells.

1. Introduction

Thank you for choosing Nuvation BMS™

Nuvation BMS™ manages rechargeable battery cells by limiting operation to within the cell's safe operating range, monitoring the state of the cells, estimating State-of-Charge and State-of-Health, reporting measured data, and interacting with the energy storage system controller.

1.1. About this Guide

This *Software Reference Manual: Single Stack* manual describes the software configuration settings used within both Nuvation High-Voltage BMS™ and Nuvation Low-Voltage BMS™.

This document is designed as a companion to the quick start configuration file available from https://ncloud.nuvationenergy.com.

Important terminology and concepts for Nuvation BMS are reviewed. A detailed breakdown of configuration settings by major areas of interest is presented.

The sections are designed to correspond in order and content to the layout of the example configuration file. This enables efficient cross referencing between the descriptions in this document and actual configuration examples.

This document applies to Nuvation BMS Babbage 18.08 software release (Firmware version 4.88.0, Operator Interface version 0.38.0). Content may be inaccurate or incomplete for other versions.

We thrive on your feedback and what we build is driven by your input. Please submit support tickets to support@nuvationenergy.com.

2. Background and Terminology

Terminology and technical concepts critical to the operation and configuration of Nuvation BMS are presented in this section.

2.1. Battery Topology

Energy Storage Systems are hierarchical in nature. Nuvation Energy has adopted the following definitions for battery pack topology:

Cell

A Cell is the smallest unit of energy storage distinguishable by the battery management system. One Cell, as defined from the perspective of the BMS, may actually consist of one or more electrochemical cells connected in parallel. This subtlety is reflected in the nomenclature for completeness. For example, a "1p" Cell refers to a single electrochemical cell, while a "2p" Cell refers to two electrochemical cells connected together in parallel. From the perspective of the BMS, these topologies appear identical except for the capacity of the Cells.

Group

A Group is a set of Cells connected in series and managed together. For example, 12 "1p" Cells in series are referred to as a "12s1p" Group, while 16 "2p" Cells in series are referred to as a "16s2p" Group. Grouping of Cells is highly application-specific and is defined in how BMS hardware interfaces are physically wired up to Cells.

Stack

A Stack is one or more Groups connected in series. For example, five "14s2p" Groups connected in series are referred as a "5g14s2p" Stack. This Stack may also be described as a "70s2p" Stack.

Bank

A Bank is one or more stacks connected in parallel. For example, three "5g14s2p" Stacks are referred to as a "3x5g14s2p" Bank or simply a "3x70s2p" Bank.

Pack

A Pack is one or more Banks connected in series.

2.2. Register Data Model

2.2.1. Registers and Components

Understanding the Register Data Model is key to understanding how to configure Nuvation BMS.

Nuvation BMS implements all data storage and processing using two important software building blocks

Register

A register is the fundamental unit of data storage within the system. Each register has a unique name and associated type that defines how the value is interpreted. Registers range in size from as small as one byte up to as large as eight bytes.

Component

A component combines a set of related registers with processing rules that operate on those

registers to implement a particular BMS function for the system. A given component may have many instances throughout the system. In this case, its associated registers will have the same number of instances.

Complex behavior within the system is achieved by connecting multiple components together, either through configuration or through hard-wired connections in the firmware itself.

Configuration for a system is completely determined by the state of all configuration registers present within the system. Configuration registers are persisted in non-volatile memory and are loaded automatically upon reset.

External protocols are implemented by mapping (and in some cases aggregating) the appropriate BMS registers to CAN bus messages or Modbus registers.

2.2.2. Index versus Location

Internally to Nuvation BMS firmware, all register indexing is zero-based. That is, if multiple instances of the same register are present, the first instance is always indexed at zero. This convention is reflected in all register expressions and configuration files.

Operator-facing tools such as the Nuvation BMS Operator Interface or the MESA Modbus models use one-based location identifiers to refer to physical, countable entities.

For example, the location of the first cell within a stack is defined as CI 1, Cell 1 - i.e. it is the first cell in the first Cell Interface. The index of this first cell is defined as zero within the firmware.

2.2.3. Register Expressions

Registers are accessed by name in Nuvation BMS tools and configuration. Each register also has a unique address that is used internally within the BMS.

Single Register Instance

This expression is used when assigning to or reading from a single register in the system and is of the form:

```
component_name.register_name
```

where:

- **component_name** is the name of the component within the system
- **register_name** is the name of the register within the component

Range of Register Instances

These expressions build on the single register references by adding an additional range expression in square brackets:

```
component_name[range_expression].register_name
```

The **range_expression** may take any of the following forms:

- **index** - A single instance of `component_name.register_name` at `index`. Note that `cell[0].voltage` is equivalent to `cell.voltage`.
- **start_index:end_index** - All instances from `start_index` through `end_index`. The expression `cell[0:3].voltage` expands into:

```
cell[0].voltage
cell[1].voltage
cell[2].voltage
cell[3].voltage
```

- **start_index:end_index:block_length** - All instances from `start_index` through `end_index` within a repeating block of `block_length` across all instances of the register. The expression `cell[0:3:16].voltage` expands into:

```
cell[0].voltage
cell[1].voltage
cell[2].voltage
cell[3].voltage
cell[16].voltage
cell[17].voltage
cell[18].voltage
cell[19].voltage
cell[32].voltage
cell[33].voltage
cell[34].voltage
cell[35].voltage
cell[N-16].voltage
cell[N-15].voltage
cell[N-14].voltage
cell[N-13].voltage
```

  where `N` is the total number of instances of the register `cell.voltage` within the system.

- **start_index:end_index:block_length:block_count** - All instances from `start_index` through `end_index` within a repeating block of `block_length` repeated `block_count` times. The expression `cell[0:3:16:2].voltage` expands into:

```
cell[0].voltage
cell[1].voltage
cell[2].voltage
cell[3].voltage
cell[16].voltage
cell[17].voltage
cell[18].voltage
cell[19].voltage
```

  In this case, only the first 2 blocks of 16 instances are included, rather than all blocks of 16 instances.

All Register Instances

A compact syntax can be used to expand to all instances of a given register within the system. The expression:

```
component_name[*].register_name
```

expands to all instances of **component_name.register_name** within the system. For example, the expression `cell[*].voltage` expands into:

```
cell[0].voltage
cell[1].voltage
cell[2].voltage
cell[3].voltage
cell[4].voltage
cell[5].voltage
cell[6].voltage
cell[7].voltage
cell[8].voltage
cell[9].voltage
cell[10].voltage
cell[11].voltage
cell[12].voltage
cell[13].voltage
cell[14].voltage
cell[15].voltage
cell[16].voltage
.
.
.
cell[N-3].voltage
cell[N-2].voltage
cell[N-1].voltage
```

where `N` is the total number of instances of the register `cell.voltage` within the system.

Register Address

In some cases, it is necessary to use the address of a register as a configuration value for another register in the system. This is required when assigning input and output pins to functions within the BMS, for example.

The expression:

```
@component_name.register_name
```

expands to the address of the register in the system. The single-instance range expression may be used for register addresses. For example:

```
@component_name[index].register_name
```

expands to the address of **component_name.register_name** at `index` in the system.

2.3. Configuration File

Configuration is stored externally to Nuvation BMS in a plain-text file. This file defines the state of configuration registers as required for a particular system.

The Operator Interface provides tools for importing and exporting configuration files to and from Nuvation BMS as a way to set or retrieve the state of all configuration registers.

The configuration file format is plain ASCII text with the following syntax:

- Any lines starting with a leading `#` are treated as comments.
- Each non-comment line is treated as a register assignment statement.

A register assignment takes on the form `register_expression = value` where:

- `register_expression` is one of the valid Register Expressions previously defined.
- `value` is either a numerical constant, quoted string, IP address, or a valid Register Address.

Any standard text editor can be used to edit configuration files (e.g. Notepad++, etc.).

3. Configuration Settings

The following sections break down a complete system configuration into the major areas of responsibility.

3.1. Units

A standard set of units has been adopted for use within Nuvation BMS for the measurements and configuration settings. Unless otherwise noted, the units used within the firmware should be assumed as defined below.

Table 1. Standard Measurements and Units

Measurement	Units	Application
Voltage	millivolts	Cell and stack voltages
Temperature	degrees Celsius	Thermistor temperatures
Charge Current	negative milliamperes	Stack and pack currents
Discharge Current	positive milliamperes	Stack and pack currents
Charge	milliampere hours	Depth of discharge and throughput
Time	microseconds	Hysteresis time and sampling periods

Every register within the firmware has an associated type that defines the expected units for that register.

3.2. Battery Parameters

These following settings are used to configure State of Charge (SOC) and State of Health (SOH) algorithms for operation with a particular battery chemistry.

3.2.1. Stack Capacity

Battery stack capacity is defined as the total amount of charge that can be extracted from a battery stack when discharging from full to empty, assuming current limits are properly followed. Nominal (or design) capacity is configured as follows:

`stack_soc.nominal_capacity`

- Nominal capacity of the battery stack
- Set to the capacity that would correspond to a full discharge

NOTE

The nominal capacity of the stack is identical to that of the cells used within the stack, where a cell may be one or more electro-chemical cells directly connected in parallel. The actual capacity of the battery stack maybe less than this nominal capacity due to imbalances in SOC and capacity fade of the cells with usage. Correct configuration of the nominal capacity is essential for accurate SOC and SOH estimation.

3.2.2. Stack Cycle Count

In addition to the nominal capacity, the nominal cycle count for the battery stack is required for SOH estimation based upon cycle count.

`stack_soc.nominal_cycle_count`

- Set to the expected cycle life of the battery stack assuming full discharge cycles at the configured maximum operating current
- Set to zero to ignore cycle count for SOH

3.2.3. Voltage Full and Empty Thresholds

The full and empty thresholds correspond to the open-circuit voltages used to define a fully charged and fully discharged battery cell in operation. These thresholds should be defined with respect to the operational zone as configured for a particular application.

This means these settings must be carefully aligned with current limits to ensure the SOC reported by the BMS correctly reflects the configured usable capacity of the battery.

For battery chemistries that do not tolerate overcharge and make use of electrical balancing circuits to keep cells balanced (e.g. Lithium-Ion), the full and empty conditions are typically applied against the maximum and minimum cell voltages. But for chemistries that make use of overcharge during the charging process as a way to balance the stack (e.g. lead acid), the full and empty conditions are typically applied against the average cell voltage.

Nuvation BMS supports definition of the empty and full conditions using either or both min/max and average cell voltages, as configured using the following registers.

`stack_soc.vfull`

- Defines fully charged maximum cell voltage for operation

- Set as per application requirements, or set to zero to disable

`stack_soc.vfullavg`

- Defines fully charged average cell voltage for operation
- Set as per application requirements, or set to zero to disable

`stack_soc.vempty`

- Defines fully discharged minimum cell voltage for operation
- Set as per application requirements, or set to zero to disable

`stack_soc.vemptyavg`

- Defines fully discharged average cell voltage for operation
- Set as per application requirements, or set to zero to disable

The empty and full thresholds must be carefully aligned with the maximum and minimum Cell Voltage Thresholds and Stack Voltage Thresholds for proper SOC operation. Typically, empty and full are set to the voltage where the corresponding current limit has fallen to C/20. In this case, empty will be set slightly above the minimum voltage and full will be set slightly below the maximum voltage.

3.2.4. Current Full Threshold

The current full threshold is another condition that can be applied to determine the full state of the battery. A battery is not fully charged unless both the full voltage condition is met and the battery current is above this threshold.

`stack_soc.ifull`

- Defines a maximum current before the battery is considered full
- Value must be negative to represent a current in the charge direction.
- When the current is above this threshold and the full voltage condition is met, then the battery is considered fully charged

3.3. Stack Topology

Configuring the battery stack topology requires that the following are specified:

- Which cell voltage taps are actually connected to cells
- Which thermistor inputs are actually connected to thermistors

3.3.1. Cell Inputs

Both the Cell Interface and Battery Controller modules may be connected to fewer cells than the maximum supported number of cell inputs. The following registers are used to indicate which cells are actually present in the system. The index `n` is the zero-based index of the cell input.

`cell[n].installed`

- Indicates a cell is physically connected
- Set to 1 if connected
- Set to 0 if not connected

Cell indexing is statically assigned in such a way that `cell[0]` always refers to the first cell in the Battery Controller or first Cell Interface.

For Nuvation High-Voltage BMS™ with multiple Cell Interfaces, `cell[16]` refers to the first cell in the second Cell Interface, `cell[32]` refers to the first cell in the third Cell Interface, and so on.

3.3.2. Thermistor Inputs

Both the Cell Interface and Battery Controller modules can connect to fewer than the maximum supported number of thermistors. The following registers are used to indicate which thermistors are actually present in the stack. The index `n` is the zero-based index of the thermistor input.

`therm[n].installed`

- Indicates a thermistor is physically connected
- Set to 1 if connected
- Set to 0 if not connected

Thermistor indexing is statically assigned in such a way that `therm[0]` always refers to the first thermistor in the Battery Controller or first Cell Interface.

For Nuvation High-Voltage BMS™ with multiple Cell Interfaces, `therm[8]` refers to the first thermistor in the second Cell Interface, `therm[16]` refers to the first thermistor in the third Cell Interface, and so on.

3.4. Operational Limits

The operational limits of a battery stack are captured in the form of voltage, temperature, and current thresholds. These thresholds must be set correctly for your battery cells and DC bus system so that the BMS can:

1. Gracefully control current during charging and discharging to keep the battery within normal operating limits.
2. Warn operators and external systems if the battery is not within normal operating limits
3. Disconnect the battery from the DC bus under a fault condition if the battery is approaching unsafe limits

The operating zone and safe zone are illustrated below graphically. The outermost box around the *Fault* state represents all possible states for the entire system (this is a boundary around a space that has many dimensions, not just two). The boundary around the *All Clear* state represents the subset of all system states that can be reached under normal, fully controlled operation. This is the operating zone. With proper configuration and functional control, a system should never leave the *All Clear* state. The boundary between the *Warning* state and the *Fault* state represents the threshold beyond which the system cannot be safely operated - i.e, this is the edge of the safe zone.

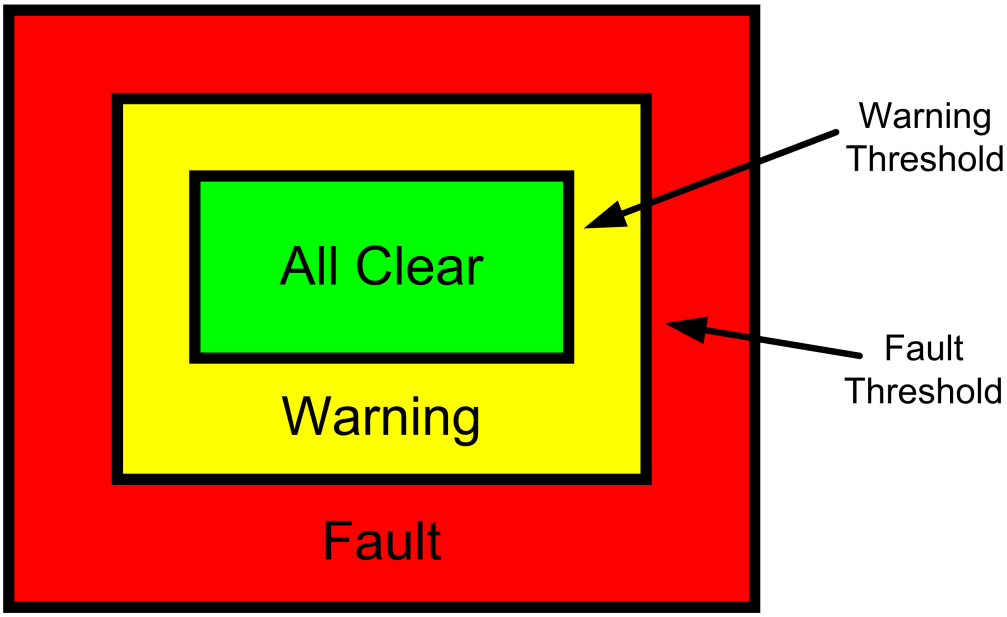

Figure 1. Battery Management System Zones

3.4.1. Hysteresis Triggers

The fundamental building block used to define thresholds throughout Nuvation BMS is the hysteresis trigger component. The following registers are used to configure each trigger discussed in the following sections.

`trigger_name.thresh`

- The input must meet or exceed this threshold to trip the trigger.

`trigger_name.time_hyst`

- The elapsed time that the input must meet or exceed the threshold in order to trip the trigger.
- Set to 0 to configure a trigger that trips instantly.

`trigger_name.end_time_hyst`

- The elapsed time that the input must recover (remain below the threshold) before the trigger will clear.
- Set to 0 to configure a trigger that clears instantly.
- Set to > 0 to configure a trigger that clears after the defined period of time.

`trigger_name.latched`

- Set to 1 for latching trigger behavior
- Set to 0 for non-latching trigger behavior

`trigger_name.disabled`

- Set to 0 to enable the trigger
- Set to 1 to disable the trigger

Latching triggers remain tripped until explicitly cleared through an external request via the Operator Interface or a supported protocol. Non-latching triggers clear automatically after the appropriate end time hysteresis.

3.4.2. Cell Voltage Thresholds

The following diagram illustrates how current limits, State-of-Charge thresholds, and trigger thresholds are configured by cell voltage in typical applications. The horizontal axis represents cell voltage, increasing from left to right. The vertical axis represents the calculated current limit percentage, increasing from bottom to top.

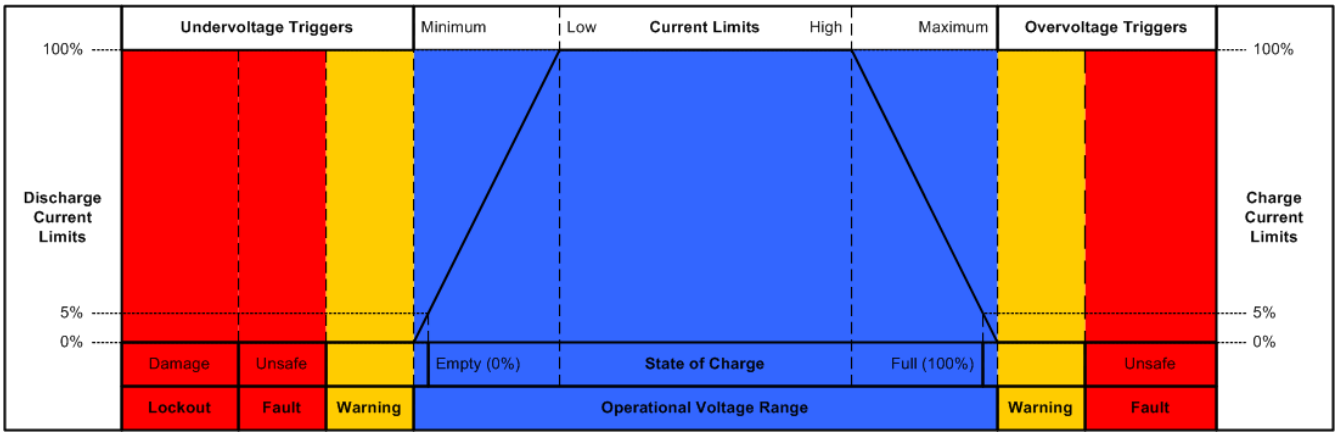

Figure 2. Typical Arrangement of Cell Voltage Thresholds

Most systems will make use of thresholds configured in the following order of decreasing cell voltage.

Table 2. Cell Voltage Operational Limits

Register	Setting
`stack_fault_cell_hi.thresh`	The upper limit of the safe zone as per cell specifications.
`stack_warn_cell_hi.thresh`	The upper limit of the operating zone. Set just above `stack_current_limit.voltage_cell_max`.
`stack_current_limit.voltage_cell_max`	The voltage at which charge limits approach 0%. Set just above `stack_soc.vfull`.
`stack_current_limit.voltage_cell_high`	The voltage at which charge current limits are reduced from 100%. Set as per application requirements.
`stack_current_limit.voltage_cell_low`	The voltage at which discharge current limits are reduced from 100%. Set as per application requirements.
`stack_current_limit.voltage_cell_min`	The voltage at which discharge limits approach 0%. Set just below `stack_soc.vempty`.
`stack_warn_cell_lo.thresh`	The lower limit of the operating zone. Set just below `stack_current_limit.voltage_cell_min`.
`stack_fault_cell_lo.thresh`	The lower limit of the safe zone as per cell specifications.
`stack_uvlo_cell_voltage.thresh`	The low-voltage lockout threshold for system shutdown.

The low-voltage lockout trigger itself does not guarantee the BMS will shutdown when a cell voltage drops below this level. The BMS must have appropriate power switching hardware driven off of this trigger for this shutdown to be functional. Nuvation BMS™ Low-Voltage Battery Controller has this functionality built in, while the Nuvation BMS™ High-Voltage Stack Controller does not.

Typically, cell voltage warning triggers are configured as non-latching, while cell voltage fault triggers are configured as latching.

User-defined triggers are also available for high and low cell voltages.

Table 3. User Defined Cell Voltage Triggers

Register	Setting
`stack_trig_cell_hi.thresh`	User-defined high cell voltage trigger.
`stack_trig_cell_lo.thresh`	User-defined low cell voltage trigger.

3.4.3. Thermistor Temperature Thresholds

Separate configuration thresholds are provided for charging and discharging as many cells have different temperature limits in these two modes of operation. Charge triggers will only trip while the stack is charging, while discharge triggers will only trip while the stack is discharging or resting.

Table 4. Charge Temperature Operational Limits

Register	Setting
`stack_fault_charge_therm_hi.thresh`	The upper limit of the safe charging zone as per cell specification.
`stack_warn_charge_therm_hi.thresh`	The upper limit of the operating charging zone. Set just above `stack_current_limit.temperature_charge_max`.
`stack_current_limit.temperature_charge_max`	The temperature at which current limits approach 0% during charging.
`stack_current_limit.temperature_charge_high`	The temperature at which current limits are reduced from 100% during charging.
`stack_current_limit.temperature_charge_low`	The temperature at which current limits are reduced from 100% during charging.
`stack_current_limit.temperature_charge_min`	The temperature at which current limits approach 0% during charging.
`stack_warn_charge_therm_lo.thresh`	The lower limit of the operating charging zone. Set just below `stack_current_limit.temperature_charge_min`.
`stack_fault_charge_therm_lo.thresh`	The lower limit of the safe charging zone as per cell specification.

Table 5. Discharge Temperature Operational Limits

Register	Setting
`stack_fault_discharge_therm_hi.thresh`	The upper limit of the safe discharging zone as per cell specification.
`stack_warn_discharge_therm_hi.thresh`	The upper limit of the operating discharging zone. Set just above `stack_current_limit.temperature_discharge_max`.
`stack_current_limit.temperature_discharge_max`	The temperature at which current limits approach 0% during discharging.
`stack_current_limit.temperature_discharge_high`	The temperature at which current limits are reduced from 100% during discharging.
`stack_current_limit.temperature_discharge_low`	The temperature at which current limits are reduced from 100% during discharging.
`stack_current_limit.temperature_discharge_min`	The temperature at which current limits approach 0% during discharging.

Register	Setting
stack_warn_discharge_therm_lo.thresh	The lower limit of the operating discharging zone. Set just below stack_current_limit.temperature_discharge_min.
stack_fault_discharge_therm_lo.thresh	The lower limit of the safe discharging zone as per cell specification.

Typically, thermistor temperature warning triggers are configured as non-latching, while thermistor temperature fault triggers are configured as latching.

User defined triggers are also available for high and low thermistor temperature during both charge and discharge.

Table 6. User Defined Charge Temperature Triggers

Register	Setting
stack_trig_charge_therm_hi.thresh	User defined threshold for charge high temperature trigger.
stack_trig_charge_therm_lo.thresh	User defined threshold for charge low temperature trigger.

Table 7. User Defined Discharge Temperature Triggers

Register	Setting
stack_trig_discharge_therm_hi.thresh	User defined threshold for discharge high temperature trigger.
stack_trig_discharge_therm_lo.thresh	User defined threshold for discharge low temperature trigger.

3.4.4. Module Temperature Thresholds

Nuvation BMS actively monitors the temperature of modules containing passive balancing circuits to avoid potential overheating. These trigger settings may be customized for applications that may require operational zones narrower than the defaults.

Table 8. Module Temperature Limits

Register	Setting
sc_fault_ci_therm_hi.thresh	The upper temperature limit for module operation as per specifications.
sc_warn_ci_therm_hi.thresh	The high module temperature warning threshold. Set as per specifications.
sc_warn_ci_therm_lo.thresh	The low module temperature warning threshold. Set as per specifications.
sc_fault_ci_therm_lo.thresh	The lower temperature limit for module operation as per specifications.

User defined triggers are also available for high and low module temperatures.

Table 9. User Defined Module Temperature Triggers

Register	Setting
sc_trig_ci_therm_hi.thresh	The user defined high module temperature threshold.
sc_trig_ci_therm_lo.thresh	The user defined low module temperature threshold.

3.4.5. Stack Current Thresholds

The stack current thresholds are used to define the limits of the operational zone and safe zone for charge and discharge currents. These limits must factor in the specifications of the battery cells as well as the limits of any DC current-carrying paths in the stack.

Table 10. Stack Current Operational Limits

Register	Setting
`stack_fault_current_lo.thresh`	The limit of the safe charging zone as per cell specification and stack design limits.
`stack_warn_current_lo.thresh`	The limit of the operational charging zone as per application requirements.
`stack_warn_current_hi.thresh`	The limit of the operational discharging zone as per application requirements.
`stack_fault_current_hi.thresh`	The limit of the safe discharging zone as per cell specification and stack design limits.

Typically, stack current warning triggers are configured as non-latching, while stack current fault triggers are configured as latching with a small amount of trip time hysteresis (e.g. approximately 100ms - 200ms).

User-defined triggers are also available for charge and discharge currents.

Table 11. User Defined Stack Current Triggers

Register	Setting
`stack_trig_current_lo.thresh`	User defined charge current trigger.
`stack_trig_current_hi.thresh`	User defined discharge current trigger.

Charge current thresholds are specified as negative values while discharge current thresholds are specified as positive values.

3.4.6. Stack Voltage Thresholds

The following diagram illustrates how current limits, SOC thresholds, and trigger thresholds are configured by stack voltage in typical applications. The horizontal axis represents total stack voltage, increasing from left to right.

The vertical axis represents the calculated current limit percentage, increasing from bottom to top.

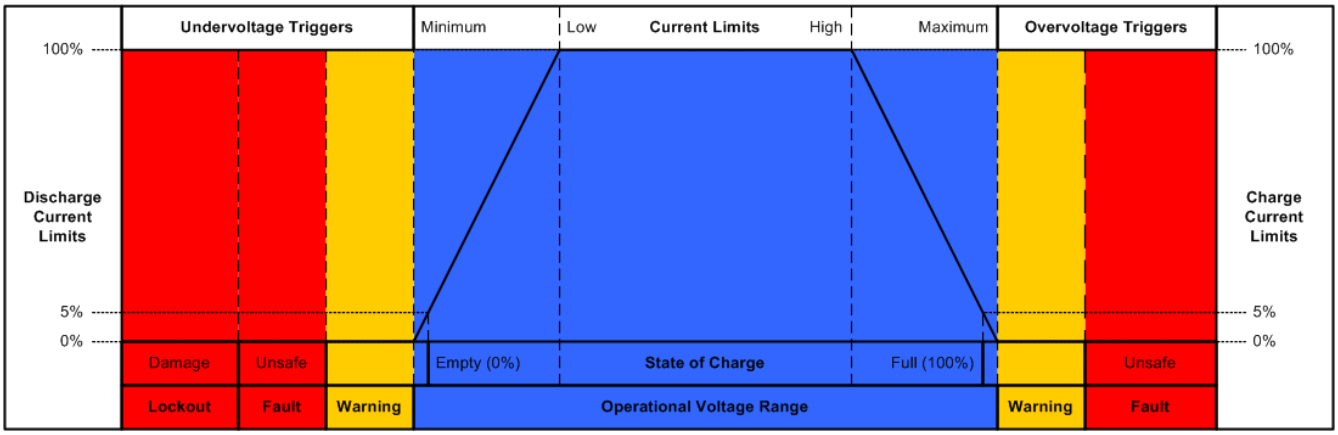

Figure 3. Typical Arrangement of Stack Voltage Thresholds

Stack voltage thresholds are used to define the operating voltage limits of the overall battery stack design. These thresholds are used to enforce the design limits of the battery stack or DC bus as a whole. For example, it may be necessary to limit the overall stack voltage within a certain limited range to maintain compatibility with a specific charger or inverter. Stack voltage limits also are used to ensure that average cell voltage is maintained within specified limits.

Most systems will make use of thresholds configured in the following order of decreasing stack voltage.

Table 12. Stack Voltage Operational Limits

Register	Setting
`stack_fault_voltage_hi.thresh`	The upper limit of the safe zone as per application requirements.
`stack_warn_voltage_hi.thresh`	The upper limit of the operating zone. Set just above `stack_current_limit.voltage_stack_max`.
`stack_current_limit.voltage_stack_max`	The voltage at which charge limits approach 0%. Set as per application requirements.
`stack_current_limit.voltage_stack_high`	The voltage at which charge current limits are reduced from 100%. Set as per application requirements.
`stack_current_limit.voltage_stack_low`	The voltage at which discharge current limits are reduced from 100%. Set as per application requirements.
`stack_current_limit.voltage_stack_min`	The voltage at which discharge limits approach 0%. Set as per application requirements.
`stack_warn_voltage_lo.thresh`	The lower limit of the operating zone. Set just below `stack_current_limit.voltage_stack_min`.
`stack_fault_voltage_lo.thresh`	The lower limit of the safe zone as per application requirements.
`stack_uvlo_stack_voltage.thresh`	The low-voltage lockout threshold for system shutdown.

The low-voltage lockout trigger itself does not guarantee the BMS will shutdown when the stack voltage drops below this level. The BMS must have appropriate power switching hardware driven off of this trigger for this shutdown to be functional. Nuvation BMS™ Low-Voltage Battery Controller has this functionality built in, while Nuvation BMS™ High-Voltage Stack Controller does not.

Typically, stack voltage warning triggers are configured as non-latching, while stack voltage fault triggers are configured as latching.

User-defined triggers are also available for high and low stack voltages.

Table 13. User Defined Stack Voltage Triggers

Register	Setting
`stack_trig_voltage_hi.thresh`	User defined high stack voltage trigger.
`stack_trig_voltage_lo.thresh`	User defined low stack voltage trigger.

Since stack voltage is measured independently from individual cell voltages in Nuvation BMS, another important configuration threshold is the limit for mismatch between the overall stack voltage measurement and the sum of individual cell voltages. Generally this is configured with a trip time hysteresis of one to two seconds.

`stack_fault_voltage_sum.thresh`

- The upper limit of the safe voltage mismatch zone
- Set as per application requirements (typically a few percent of `stack_fault_voltage_hi.thresh`)

3.4.7. External Controller Heartbeat

Nuvation BMS can be configured to require a heartbeat signal from an external controller in order to keep the stack online and out of fault state. A write to the MESA controller heartbeat register is expected at least once during the watchdog period. If Nuvation BMS™ Grid Battery Controller is in use, the Grid Battery Controller will write to this register to keep the stack out of fault state.

`sc_controller_wdt.period`

- Trip time for watchdog if heartbeat disappears
- Set to 5 seconds or as per application requirements

`sc_fault_controller_wdt.disabled`

- Set to 0 to enable controller watchdog
- Set to 1 to disable controller watchdog

If this feature is not used, the watchdog fault should be disabled.

3.5. Control Settings

Nuvation BMS controls the current flowing through a battery stack:

1. During connection or disconnection of the battery to prevent harmful transient current events
2. During operation of a connected battery to keep the battery within its operational limits
3. During operation of a battery to keep the individual cells at a balanced state of charge
4. During a fault condition in order to protect the battery

Control for cases (1) and (4) is achieved through external switching devices that are under the control of Nuvation BMS. This control is limited to hard switching. Control for case (2) is achieved through current limiting signals that are used by chargers and inverters to throttle current dynamically. Control for case (3) is implemented within Nuvation BMS itself through passive balancing loads that are under control of a configurable balancing algorithm.

3.5.1. Stack Switch Functions

Nuvation BMS defines three contactor switch functions for use within typical battery configurations:

1. **Pre-charge Switch** - Connected during pre-charge operation only. Disconnected under fault condition.
2. **Main Switch** - Connected after any pre-charge operation completes. Disconnected under fault condition.
3. **Stack Switch** - Connected whenever either the pre-charge switch or the main switch are connected. Disconnected under fault condition.

While a pre-charge contactor is optional, virtually all systems require a main contactor to protect the battery from unsafe conditions. A stack contactor is often used in conjunction with a main contactor to isolate the battery stack completely from the DC bus and provide a level of contactor redundancy.

Stack connection and disconnection sequencing is illustrated in the state diagram below.

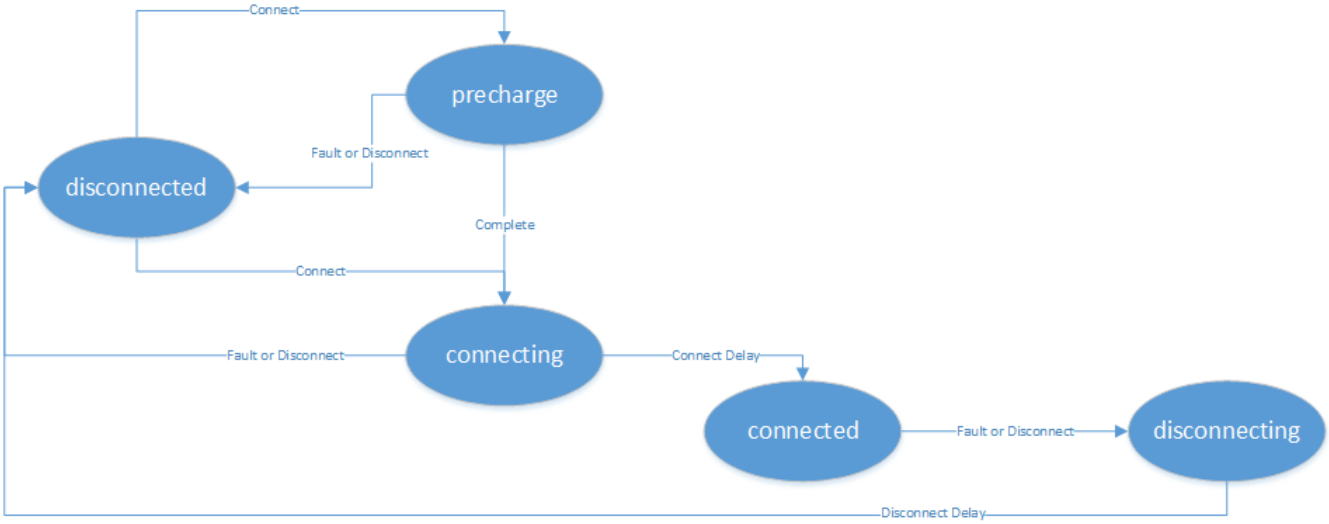

Figure 4. Battery Stack States and Transitions

As a system is connected and disconnected from the DC bus, a configurable sequencing delay is inserted before and after the connected state. During the connected state, the BMS uses current limits to control current flowing into and out of the stack. During all other states, current limits are set to zero. This allows for graceful switching behavior with no current flow under normal connect and disconnect requests.

The switch functions defined here must be mapped to appropriate outputs for use in an actual system. This process is covered in detail in Input/Output Assignment.

Auto-Connection Setting

Sometimes its desired that the Nuvation BMS initiate a connection of a battery stack when:

1. the BMS powers on.
2. after all faults have been cleared.

This operation can be configured with the following register setting.

`stack_control.auto_connect`

- When set to a value of one, the BMS will initiate a connection of the battery stack if there are no faults triggered.
- When set to a value of zero, the BMS waits for a connection request via the OI or through its MESA interface. Note a stack will not connect if there is any faults triggered.

Pre-Charge Switch Settings

If enabled, the pre-charge switch is engaged for a fixed (but configurable) amount of time during the pre-charge state. If the pre-charge over current fault is configured, the pre-charge will terminate if the fault is triggered. At the end of the pre-charge period, the stack current is compared against a maximum threshold value to determine if the pre-charge operation was successful. Upon successful completion, the stack connection sequence continues. Upon failure, a pre-charge fault is tripped.

Pre-charge behavior is configured through the following registers as required for a particular application.

`stack_control.precharge_delay`

- Determines the fixed amount of time the pre-charge path is energized.
- Set based upon pre-charge hardware power and thermal ratings.
- Set to zero to disable pre-charge.

`stack_control.precharge_max_current`

- Determines the maximum current flow at the end of `stack_control.precharge_delay` under which pre-charge can complete successfully.
- Set to ensure any in-rush currents upon main switch connection are within system ratings.
- Set to zero to disable pre-charge.

`stack_fault_precharge_over_current.thresh`

- Sets the over current limit at any time during pre-charge.
- If the fault is triggered, pre-charge will terminate immediately.
- Limits the power dissipated of a pre-charge resistor during a short condition allowing for a smaller resistor.

A pre-charge operation only completes successfully if the stack current magnitude falls below the maximum pre-charge current within the configured delay time.

Sequencing Delays

The following registers are used to configure the sequencing delays used before a stack enters the connected or disconnected state.

`stack_control.connect_delay`

- The delay between current limits engaging and the contactors connecting after a connect request.
- Typically this is between 1 and 5 seconds.

`stack_control.disconnect_delay`

- The delay between current limits disengaging and the contactors disconnecting after a disconnect request.
- Typically this is between 1 and 5 seconds.

The disconnect switching delay only applies to disconnect requests. Under a fault condition, the stack can be disconnected with little to no delay through use of the Fault Pilot signal.

3.5.2. Stack Current Limits

Maximum Operating Currents

The maximum continuous operating charge and discharge currents must be configured for current limiting to function properly. These values correspond to the current limit values that will be used during normal wide-open operation (no throttling).

`stack_current_limit.max_charge_current`

- Magnitude of maximum continuous operating charge current.

`stack_current_limit.max_discharge_current`

- Magnitude of maximum continuous operating discharge current.

The current limits given above are magnitudes only - i.e. both charge and discharge current limits are positive.

Minimum Charge Current

The minimum charge current is the constant charging current that should be applied as the battery reaches the end of its charge cycle. The BMS will ensure that the charge current limit does not fall below this minimum value until the battery has reached its maximum charging voltage.

`stack_current_limit.min_charge_current`

- Minimum charge current to be applied at the end of charge cycle.
- Set as per battery manufacturer recommendations (typically below C/20).

This setting must be configured in conjunction with the full thresholds defined by Stack Capacity. For example, the stack can be considered full once the current limit reaches the minimum charge current.

Current Limiting Response Times

The current limiting control loop can be tuned for stable and responsive behavior in a variety of systems. Two independent settling times are provided to allow independent adjustment of the response to decreases in current limits (attack time) and increases in current limits (decay time).

`stack_current_limit.attack_settling_time`

- Settling time for decreases in current limits (typically this should be no larger than 5-10s).

`stack_current_limit.decay_settling_time`

- Settling time for increases in current limits (typically this is on the order of 10x larger than `stack_current_limit.attack_settling_time`).

Since the attack time determines how quickly the current limits can respond before a potential fault conditions opens a switch, it is critical to have sufficient control bandwidth here to avoid tripping faults.

A non-zero settling time is critical in most applications to avoid oscillations in the presence of noise and other imperfections in high-power control of the DC current in the charger and/or inverter.

3.5.3. Passive Cell Balancing

When multiple cells are connected in series to form a larger battery stack, it is important to ensure each cell is giving equal contributions to the system. The effects of a single low State-of-Charge or a single high State-of-Charge cell will dominate the performance of the large battery stack. The act of equalizing State-of-Charge of multiple series-connected cells is called balancing and there are many flavors of balancing. Nuvation BMS implements a passive balancing solution. Cells with high State-of-Charge are discharged via bleed resistors which are enabled on a per-cell basis. Properly adjusting the algorithm settings for your cells is necessary to achieve a well-performing system.

A number of configurable settings are used to fine tune the passive balancing algorithm for voltage, temperature, current, and duty cycle.

The balancer must be enabled for balancing to take place.

`stack_cell_balancer.enabled`

- Enables or disables balancing operation.
- Set to 1 to enable.
- Set to 0 to disable.

Cell Voltage Settings

Both absolute and relative cell voltage thresholds are used to safely balance a stack of batteries.

Minimum voltage: this absolute threshold determines the voltage below which a cell will not be balanced. This prevents over discharging in a system even with large imbalances.

Delta voltage: this relative threshold is used to determine when a system is balanced. Balancing will take place when the difference between the highest and lowest cell is greater than or equal to this threshold.

`stack_cell_balancer.min_enable_voltage`

- Minimum voltage threshold for balancing.
- Typically, this is set higher than `stack_current_limit.voltage_cell_high`.

`stack_cell_balancer.voltage_delta`

- Delta voltage threshold for balancing.
- Typically, this is set within 5-25mV.

If `stack_cell_balancer.voltage_delta` is set to zero, the system will continue balancing down all cells (even if the difference between min and max is zero) until they reach `stack_cell_balancer.min_enable_voltage`. This mode can be used to passively balance all cells in a stack to a specific open-circuit voltage.

Temperature Settings

An upper temperature limit is implemented to prevent BMS module overheating. If either of the following temperature thresholds are exceeded, balancing is disabled for all cells in the stack.

`stack_cell_balancer.max_enable_temperature`

- The upper cell temperature limit for balancing.
- Set as per application requirements.

`stack_cell_balancer.max_ci_enable_temperature`

- The upper BMS module silicon temperature limit for balancing.
- Set as per BMS module specifications or application requirements.

Current Settings

Balancing enable thresholds based upon stack current allow the balancer to be fine-tuned to run during specific portions of the charge and discharge cycle.

`stack_cell_balancer.min_enable_current`

- The minimum current at which balancing remains enabled.
- This is typically set to a negative value to enable balancing below certain charge currents.

`stack_cell_balancer.max_enable_current`

- The maximum current at which balancing remains enabled.
- This may be a negative or positive value depending upon application requirements.

The two most common use cases are:

1. Balance only while charging. In this case, both the minimum and maximum current thresholds are set to negative values that correspond to the range of charge currents under which balancing should take place.
2. Balance while charging and holding. In this case, the maximum current threshold is set to a slightly positive value so that the stack will balance when it is idle or disconnected. The level of discharge current flow tolerated during idle balancing is application specific and is thus configurable.

3.6. Input/Output Assignment

There are some differences in the Input/Output Assignment for Nuvation High-Voltage BMS™ and for Nuvation Low-Voltage BMS™.

3.6.1. Nuvation High-Voltage BMS™: Input/Output Assignment

The Stack Controller implements the following digital outputs and inputs:

- 4 general purpose digital outputs
- 1 Fault Pilot output signal (driven over the Stack Bus)
- 4 general purpose digital inputs

The Nuvation BMS Fault Pilot signaling mechanism is a dedicated hardware signaling path between the Stack Controller and Power Interface that is used to rapidly open the contactors in the case of

a fault condition or processor failure. It is configurable for advanced applications that may require customized behavior.

The Power Interface implements 4 contactor output drivers and is the module that is typically used to control high-current switching devices.

Contactor outputs and general purpose digital inputs and outputs present on Nuvation BMS are implemented to allow for assignment of pin functions through configuration rather than through hard-wired implementation. This means that the pins connected to any external contactors, switches, or other digital inputs or outputs must be mapped in configuration to the appropriate BMS function for that system.

3.6.2. Nuvation Low-Voltage BMS™: Input/Output Assignment

The Battery Controller supports the following outputs and inputs:

- 4 contactor output drivers
- 4 general purpose digital outputs
- 1 internal Fault Pilot output signal
- 4 general purpose digital input
- 3 hard-wired digital inputs

The Nuvation BMS Fault Pilot signaling mechanism is a dedicated hardware path used to rapidly open the contactors in the case of a fault condition or processor failure. It is also used to drive the fault LED on the module. It is configurable for advanced applications that may not use default behavior.

The contactor outputs and general purpose digital inputs and outputs present on Nuvation BMS are implemented to allow for assignment of pin functions through configuration rather than through hard-wired implementation. This means that the pins connected to any external contactors, switches, or other digital inputs or outputs must be mapped in configuration to the appropriate BMS function for that system.

The Nuvation BMS™ Low-Voltage Battery Controller has 3 hard-wired digital inputs that enable specific BMS functions to be triggered from an external signal.

Fault Clear

When the *Fault Clear* input to the BMS is asserted for a minimum duration of at least 200ms, the BMS will issue a *Clear Fault* operation. If all fault conditions within the BMS have been cleared by this action, the BMS can be commanded to close it's contactors (or the contactors may automatically close if there was a prior request). If faults still exist after this action, the BMS will prevent the contactors from being closed. An operator will have to use the Modbus or HTTP interfaces to the BMS to determine the remaining faults in the system.

Shutdown

When the Shutdown input to the BMS is asserted for a minimum duration of 200ms but less than two seconds, the BMS will issue a graceful shutdown command and disconnect its power. As part of the graceful shutdown sequence, the BMS will:

1. Request disconnect via `stack_control`
2. Wait for `stack_control` to indicate disconnection is complete

3. Wait for all contactors to reach the disconnected state
4. Disconnect power

The BMS will power off at the end of the shutdown sequence. This process could take multiple seconds depending on system state and configuration. If the BMS fails to initiate the shutdown sequence, the Shutdown input can be asserted continuously for more than two seconds which will cause the BMS hardware to disconnect power. This is only recommended in cases where graceful shutdown fails.

Factory Reset

Factory reset loads the factory firmware image and erases the persistent configuration registers stored within the BMS.

To perform a Factory Reset, first fully power off the Battery Controller.Short the Factory Reset (`FAC_RST#`) input to one of the `COMIO` wires and boot the Battery Controller. The Factory Reset should continue to be shorted to `COMIO` for two (2) seconds, and then de-asserted (released from `COMIO`). The process of asserting Factory Reset and de-asserting it should all happen within the first ten (10) seconds of boot up. If successful, the Link Out (J1) LED will blink briefly. After the Link Out LED stops blinking, reboot the Battery Controller to complete the Factory Reset process.

The Factory Reset input to Nuvation BMS is read by the processor only during boot up. No action is taken on this input after the BMS has booted.

3.6.3. Contactor Outputs

Contactor output drivers are assigned to stack switching functions through configuration. Contactors are also configured as directional or non-directional. A directional contactor has a preferred direction for breaking current. Nuvation BMS will open any non-directional contactors or directional contactors aligned with stack current flow first. Directional contactors that are opposed to stack current flow will be opened after a small delay. An optional feedback mechanism can be configured for contactors with a feedback line. Once a contactor's feedback output is wired into a GPI and configured, if the contactor fails to open/close the feedback line will indicate the problem and the BMS can flag a fault.

The following registers are used to configure contactor outputs. The index **n** is the zero-based index of the hardware contactor coil output.

`stack_contactor[n].address_enabled`

- Set to 1 to enable the contactor function specified in `stack_contactor[n].address`

`stack_contactor[n].inverted`

- When set to 0, contactor is **energized** when assigned function value is 1
- When set to 1, contactor is **energized** when assigned function value is 0

`stack_contactor[n].address`

- Determines the function mapped to the contactor
- Set to `@stack_control.precharge_switch_state` to function as pre-charge switch
- Set to `@stack_control.main_switch_state` to function as main switch
- Set to `@stack_control.stack_switch_state` to function as stack switch

`stack_contactor[n].direction`

- Set to 0 for a non-directional contactor that breaks any current
- Set to 1 for a directional contactor that breaks charge current
- Set to 2 for a directional contactor that breaks discharge current

`stack_contactor[n].delay`

- Time to wait before opening the contactor in the non-preferred direction
- This is typically in the range of 50-100ms

`stack_contactor[n].installed`

- Set to 1 to indicate a contactor is physically wired into the BMS
- Only installed contactors will flag hardware failures via the `stack_contactor[n].coil_error` or `stack_contactor[n].feedback_error`

`stack_contactor[n].feedback_enable`

- Set to 1 to enable contactor feedback from a configured input.
- State mismatches between `stack_contactor[n].feedback_value` and `stack_contactor[n].value` will result in `stack_contactor[n].feedback_error` being set to `1`

For advanced applications, contactor outputs may be configured to be driven from any Boolean register within Nuvation BMS.

3.6.4. Digital Outputs

The most commonly used digital output functions are:

- **Charge current disable** - a control signal that is asserted when charging should be disabled
- **Discharge current disable** - a control signal that is asserted when discharging should be disabled
- **Overall safe state** - a safety signal that is asserted when no faults are present within the system
- **Trigger state** - a trigger signal for external devices that is asserted when a specific trigger within the system is tripped

Digital output pins are assigned through the following configuration registers. The index **n** is the zero-based index of the digital input hardware pin.

`sc_gpo[n].address_enabled`

- Set this to 1 to enable the function specified in `sc_gpo[n].address`

`sc_gpo[n].inverted`

- When set to 0, GPO output switch is **closed** when assigned function value is 0
- When set to 1, GPO output switch is **closed** when assigned function value is 1

`sc_gpo[n].address`

- Determines the function mapped to output

The configuration settings for the most common functions are illustrated in the table below.

Table 14. Common GPO Pin Assignments

Function	sc_gpo[n].address	sc_gpo[n].inverted
Charge Current Enable	`@stack_current_limit.charge_current_disable`	0
Discharge Current Enable	`@stack_current_limit.discharge_current_disable`	0
Overall Safe State	`@stack_safety.safe`	1
Trigger State	`@trigger_name.trig`	1

For advanced applications, digital outputs may be configured to be driven from any Boolean register within the BMS.

3.6.5. Digital Inputs

The most commonly used digital input functions are:

- **Clear faults** - hardware input to clear any latched fault conditions
- **Connect request** - request to connect the battery stack to the DC bus

Digital input pins are assigned through the following configuration registers. The index **n** is the zero-based index of the digital input hardware pin.

`sc_gpi[n].address_enabled`
- When set to 1, the state of the input pin is propagated to the destination register address

`sc_gpi[n].inverted`
- When set to 0, GPI input value is 1 if hardware GPI is asserted
- When set to 1, GPI input value is 0 if hardware GPI is asserted

`sc_gpi[n].address`
- The destination register address to populate with the state of the input pin

`sc_gpi[n].rising_edge_triggered`
- When set to 1, the input value will be populated to the destination upon detection of a rising edge

`sc_gpi[n].falling_edge_triggered`
- When set to 1, the input value will be populated to the destination upon detection of a falling edge

If the input is configured as neither rising nor falling edge triggered, the input value is continuously populated into the destination address. The configuration settings for the most common functions are illustrated in the table below.

Table 15. Common GPI Pin Assignments

Function	sc_gpi[n].address	sc_gpi[n].inverted	sc_gpi[n].rising_edge_triggered
Clear Faults	`@stack_safety.clear_faults`	0	1
Connect Request	`@stack_control.requested_state`	0	1
Contactor Feedback	`@stack_contactor[m].feedback_value`	0	0

For advanced applications, digital inputs may be configured to drive any Boolean register within the BMS.

3.7. Protocol Settings

Both the Battery Controller and Cell Interface supports the following interfaces for connection with external systems:

- 10/100 Ethernet for Modbus TCP and Operator Interface connectivity
- CAN BUS
- RS-485 for Modbus RTU

3.7.1. CAN Bus

Nuvation BMS uses a flexible CAN reporting implementation which maps BMS registers to CAN message identifiers. The external CAN protocol supports both individual and bulk reporting capabilities. Remote Transmission Requests (RTR) are not supported.

The parameters for CAN are:

- Baud: 500 kbit/s
- CAN ID: 11-bit Identifier (Base frame format)
- CAN payload length: variable from 1 byte to 8 bytes based on register size

Up to 64 individual registers may be configured for periodic reporting by the BMS. Additionally, 4 configurable bulk report blocks are available for reporting repeating blocks of registers such as cell voltage and temperature.

The basic CAN configuration can be done with the components and registers described below.

`sc_canbus.enabled`
- A flag which enables the CAN bus interface. This must be set to 1 to enable CAN reporting.

`sc_canbus.base_can_address`
- The base CAN bus message ID. Messages are assigned sequential IDs starting at this value.Note that all reports starting at this base ID must fit into the 11-bit CAN bus message ID.

`sc_canbus.report_interval`
- The periodic reporting period for CAN message broadcasts.

`sc_canbus.report_msg_interval`
- The inter-message delay interval for outgoing CAN messages.

`sc_canbus_packets.err_rate_window`
- The time window to average communication errors over when calculating the error rate.

The standard configuration uses the following CAN reporting base settings:

Table 16. Standard Configuration for CAN Reporting

Register	Setting	Note
sc_canbus.enable	1	Set to 1 to enable CAN reports.
sc_canbus.base_can_address	0x100	
sc_canbus.report_interval	500000	
sc_canbus.report_msg_interval	0	
sc_canbus_packets.err_rate_window	30000000	

Individual Register Mapping

Nuvation BMS has a standard set of mapped registers that covers common use cases suitable for most systems. The addresses of the 64 registers associated with this reporting are configured in the following registers.

`sc_canbus_map[0:63].address`

- The register address of a value to map over CAN bus. A value of 0 disables the associated message.

`sc_canbus_map[0:63].command`

- Set to 1 to enable command messages for the associated register. Set to 0 to enable reported messages for the associated register.

These mapped messages are ordered and will have sequential CAN IDs starting at `sc_canbus.base_can_address`.

A standard configuration for CAN messages of individual registers uses the following settings:

Table 17. Standard Configuration for Individual Register CAN Mapping

Register	Setting
sc_canbus_map[0].address	@sc_clock.seconds
sc_canbus_map[1].address	@stack_power.voltage
sc_canbus_map[2].address	@stack_power.current
sc_canbus_map[3].address	@stack_soc.soc
sc_canbus_map[4].address	@stack_soc.dod
sc_canbus_map[5].address	@stack_cell_stat.max
sc_canbus_map[6].address	@stack_cell_stat.min
sc_canbus_map[7].address	@stack_cell_stat.avg
sc_canbus_map[8].address	@stack_therm_stat.max
sc_canbus_map[9].address	@stack_therm_stat.min
sc_canbus_map[10].address	@stack_therm_stat.avg
sc_canbus_map[11].address	@stack_safety.safe
sc_canbus_map[12].address	@stack_safety.safetocharge
sc_canbus_map[13].address	@stack_safety.safetodischarge
sc_canbus_map[14].address	@stack_current_limit.charge_current_limit
sc_canbus_map[15].address	@stack_current_limit.charge_current_percent
sc_canbus_map[16].address	@stack_current_limit.discharge_current_limit
sc_canbus_map[17].address	@stack_current_limit.discharge_current_percent

Register	Setting
sc_canbus_map[18].address	@stack_control.connection_state
sc_canbus_map[19].address	@stack_safety.clear_faults
sc_canbus_map[20].address	@stack_control.requested_state
sc_canbus_map[21].address	@sc_controller_heartbeat.value
sc_canbus_map[22:63].address	0
sc_canbus_map[0:18].command	0
sc_canbus_map[19:21].command	1
sc_canbus_map[22:63].command	0

With `sc_canbus.base_can_address = 0x100` , the above configuration would result in the following CAN message IDs:

Table 18. CAN IDs for Individual Registers Using the Standard Configuration

CAN ID	Message	Unit
0x100	Clock	Seconds
0x101	Stack Voltage	mV
0x102	Stack Current	mA
0x103	State of Charge	%
0x104	Depth of Discharge	mAhr
0x105	Maximum Cell Voltage	mV
0x106	Minimum Cell Voltage	mV
0x107	Average Cell Voltage	mV
0x108	Maximum Temperature	C
0x109	Minimum Temperature	C
0x10A	Average Temperature	C
0x10B	Overall Safe	Boolean
0x10C	Safe to Charge	Boolean
0x10D	Safe to Discharge	Boolean
0x10E	Charge Current Limit	mA
0x10F	Charge Percent Limit	%
0x110	Discharge Current Limit	mA
0x111	Discharge Percent Limit	%
0x112	Stack Control Connection State	Enumeration

Bulk Register Reporting

In addition to individual register reporting, four (4) configurable bulk report blocks are available for reporting repeating blocks of registers such as cell voltage and temperature.

From the receiver's point of view, there is no difference between a message for individual registers and a message for bulk registers. The main difference is in the configuration.

Repeating blocks of CAN bus messages are configured using the following registers.

`sc_canbus_bulkrpt[0:3].baseaddress`

- The register address to start bulk reading from. A value of 0 disables the associated messages from being broadcast.

`sc_canbus_bulkrpt[0:3].baseenableaddress`

- The register address used to enable transmission of a CAN message.

`sc_canbus_bulkrpt[0:3].offset`

- The offset to add to the base addresses between each read.

`sc_canbus_bulkrpt[0:3].numtoread`

- The number of registers to read and report in total for this bulk report.

The messages for bulk reports are ordered and will have sequential CAN IDs. The first bulk report (associated with `sc_canbus_bulkrpt[0]` uses CAN IDs starting at `sc_canbus.base_can_address + 64`.

The next bulk report (associated with `sc_canbus_bulkrpt[1]`) uses CAN IDs starting at (`sc_canbus.base_can_address + 64 + sc_canbus_bulkrpt[0].numtoread`).

Changing `sc_canbus_bulkrpt[n].numtoread` shifts the CAN IDs of the subsequent bulk reports (`sc_canbus_bulkrpt[n+1], ⋯, sc_canbus_bulkrpt[3]`).

A standard configuration for CAN reporting of bulk registers uses the following settings (for a system with only one set of cells and thermistors)

Table 19. Standard Configuration for Bulk Register CAN Reporting

Register	Setting
`sc_canbus_bulkrpt[0].baseaddress`	`@cell.voltage`
`sc_canbus_bulkrpt[0].baseenabledaddress`	`@cell.installed`
`sc_canbus_bulkrpt[0].offset`	3
`sc_canbus_bulkrpt[0].numtoread`	16
`sc_canbus_bulkrpt[1].baseaddress`	`@therm.temperature`
`sc_canbus_bulkrpt[1].baseenabledaddress`	`@therm.installed`
`sc_canbus_bulkrpt[1].offset`	3
`sc_canbus_bulkrpt[1].numtoread`	8
`sc_canbus_bulkrpt[2:3].baseaddress`	0
`sc_canbus_bulkrpt[2:3].baseenabledaddress`	0
`sc_canbus_bulkrpt[2:3].offset`	0
`sc_canbus_bulkrpt[2:3].numtoread`	0

With `sc_canbus.base_can_address = 0x100`, the above configuration would result in the following CAN message IDs:

Table 20. CAN IDs for Bulk Register Using the Standard Configuration

CAN ID	Message	Unit
0x140	Cell 0 Voltage	mV
0x141	Cell 1 Voltage	mV
0x142	Cell 2 Voltage	mV

CAN ID	Message	Unit
0x143	Cell 3 Voltage	mV
0x144	Cell 4 Voltage	mV
0x145	Cell 5 Voltage	mV
0x146	Cell 6 Voltage	mV
0x147	Cell 7 Voltage	mV
0x148	Cell 8 Voltage	mV
0x149	Cell 9 Voltage	mV
0x14A	Cell 10 Voltage	mV
0x14B	Cell 11 Voltage	mV
0x14C	Cell 12 Voltage	mV
0x14D	Cell 13 Voltage	mV
0x14E	Cell 14 Voltage	mV
0x14F	Cell 15 Voltage	mV
0x150	Thermistor 0 Temperature	C
0x151	Thermistor 1 Temperature	C
0x152	Thermistor 2 Temperature	C
0x153	Thermistor 3 Temperature	C
0x154	Thermistor 4 Temperature	C
0x155	Thermistor 5 Temperature	C
0x156	Thermistor 6 Temperature	C
0x157	Thermistor 7 Temperature	C

Some repeating blocks of registers may have instances whose addresses are not immediately adjacent, due to provisioning of registers for systems with a higher channel count.

For example, in a system with two sets of 12 cells, each set of registers may span 16 register blocks. In this scenario, `sc_canbus_bulkrpt[0:3].numtoread` should be set to 32, not 24. `sc_canbus_bulkrpt[0:3].numtoread` should be a multiple of 16 for bulk registers associated with cells regardless of `cell[0:799].installed`.

`sc_canbus_bulkrpt[0:3].numtoread` should be a multiple of 8 for bulk registers associated with thermistor regardless of `therm[0:399].installed`.

A system with two sets of 12 cells and two sets of 8 thermistors would have `sc_canbus_bulkrpt[0].numtoread = 32` and `sc_canbus_bulkrpt[1].numtoread = 16`.

With this modification, the above configuration would result in the following CAN message IDs:

Table 21. CAN IDs for Bulk Register Using the Standard Configuration

CAN ID	Message	Unit	Note
0x140	Cell 0 Voltage	mV	1st cell of 1st set
0x141	Cell 1 Voltage	mV	2nd cell of 1st set
0x142	Cell 2 Voltage	mV	3rd cell of 1st set
0x143	Cell 3 Voltage	mV	4th cell of 1st set
0x144	Cell 4 Voltage	mV	5th cell of 1st set

CAN ID	Message	Unit	Note
0x145	Cell 5 Voltage	mV	6th cell of 1st set
0x146	Cell 6 Voltage	mV	7th cell of 1st set
0x147	Cell 7 Voltage	mV	8th cell of 1st set
0x148	Cell 8 Voltage	mV	9th cell of 1st set
0x149	Cell 9 Voltage	mV	10th cell of 1st set
0x14A	Cell 10 Voltage	mV	11th cell of 1st set
0x14B	Cell 11 Voltage	mV	12th cell of 1st set
0x150	Cell 16 Voltage	mV	1st cell of 2nd set
0x151	Cell 17 Voltage	mV	2nd cell of 2nd set
0x152	Cell 18 Voltage	mV	3rd cell of 2nd set
0x153	Cell 19 Voltage	mV	4th cell of 2nd set
0x154	Cell 20 Voltage	mV	5th cell of 2nd set
0x155	Cell 21 Voltage	mV	6th cell of 2nd set
0x156	Cell 22 Voltage	mV	7th cell of 2nd set
0x157	Cell 23 Voltage	mV	8th cell of 2nd set
0x158	Cell 24 Voltage	mV	9th cell of 2nd set
0x159	Cell 25 Voltage	mV	10th cell of 2nd set
0x15A	Cell 26 Voltage	mV	11th cell of 2nd set
0x15B	Cell 27 Voltage	mV	12th cell of 2nd set
0x160	Thermistor 0 Temperature	C	1st thermistor of 1st set
0x161	Thermistor 1 Temperature	C	2nd thermistor of 1st set
0x162	Thermistor 2 Temperature	C	3rd thermistor of 1st set
0x163	Thermistor 3 Temperature	C	4th thermistor of 1st set
0x164	Thermistor 4 Temperature	C	5th thermistor of 1st set
0x165	Thermistor 5 Temperature	C	6th thermistor of 1st set
0x166	Thermistor 6 Temperature	C	7th thermistor of 1st set
0x167	Thermistor 7 Temperature	C	8th thermistor of 1st set
0x168	Thermistor 8 Temperature	C	1st thermistor of 2nd set
0x169	Thermistor 9 Temperature	C	2nd thermistor of 2nd set
0x16A	Thermistor 10 Temperature	C	3rd thermistor of 2nd set
0x16B	Thermistor 11 Temperature	C	4th thermistor of 2nd set
0x16C	Thermistor 12 Temperature	C	5th thermistor of 2nd set
0x16D	Thermistor 13 Temperature	C	6th thermistor of 2nd set
0x16E	Thermistor 14 Temperature	C	7th thermistor of 2nd set
0x16F	Thermistor 15 Temperature	C	8th thermistor of 2nd set

Inter Report Message Interval

By default the CAN report messages are sent successively one after another. Some CAN receivers will experience issues receiving CAN messages at this maximum rate. The battery management system can be configured to inject a pause between report messages. This delay is configured using the register `sc_canbus[0].report_msg_interval` . The value of this register represents the number

of microseconds inserted between CAN report messages. This delay applies to all CAN report messages. A delay inserted between report messages will also allow better CAN bus arbitration with master devices on the CAN bus that are attempting to send the battery management system CAN command messages.

The time to transfer the total number of CAN report messages plus the inter-report interval per message must be less than the CAN report interval or there will be performance related issues with the CAN bus operation. Any additional CAN traffic on the bus from other devices must also be considered.

Command Message Mapping

The battery management system can be configured to accept specific `CAN-IDs` as command message inputs to perform actions such as:

- connect/disconnect a battery to the DC bus
- clear faults and warnings
- update the controller heartbeat

A `CAN-ID` is configured as a command by the configuration register settings:

```
sc_canbus_map[N].command = <CAN-ID>
sc_canbus_map[N].address = @<component[].register>
```

This command configuration means that if a CAN message with the correct CAN-ID is transmitted on the CAN bus, the BMS will receive the message and write the data received into the configured register address. The data sent with the message must match the expected size and sign of the register. Refer to the table CAN Command Register Configuration for a list of common registers used to control the battery management system.

Table 22. CAN Command Register Configuration

Register	Operation
stack_safety[0].clear_faults	Set to 1 to clear faults.
stack_safety[0].clear_warnings	Set to 1 to clear warnings.
stack_control[0].requested_state	Set to 1 or 0 to connect or disconnect (respectively) a battery from the DC bus.
sc_controller_heartbeat[0].value	Write any value to update the controller heartbeat and prevent the watchdog timer from expiring (if configured)

Special Application Note: Conditional Message Transmission

In some cases, it may be desirable to only broadcast a message when a certain condition is true. For example, an application may require that a message be broadcast when a GPI is set. This can be achieved by utilizing bulk reporting and setting `sc_canbus_bulkrpt[0].baseenabledaddress` to the same thing as `sc_canbus_bulkrpt[0].baseaddress`. For example, the following configuration would result in Nuvation BMS only transmitting a message when `GPI[0]` is 1:

Table 23. Special Application of Bulk CAN Reporting

Register	Setting
`sc_gpi[0].address_enabled`	1
`sc_gpi[0].inverted`	0

Register	Setting
sc_canbus_bulkrpt[0].baseaddress	@sc_gpi[0].value
sc_canbus_bulkrpt[0].baseenabled	@sc_gpi[0].value
sc_canbus_bulkrpt[0].numtoread	1

3.7.2. RS-485 Modbus RTU

The slave device address used by the BMS for Modbus RTU may be customized as required.

`sc_modbus_rtu.device_address`

- Set to the desired Modbus RTU slave device address

3.8. Measurement Calibration

Both the Nuvation BMS™ Low-Voltage Battery Controller and Nuvation High-Voltage BMS™ provide calibration settings which allow the system to be fine-tuned for integration with a variety of measurement sensors. The pre-set values that ship with the BMS can be adjusted as required.

3.8.1. Thermistor Calibration

Nuvation BMS can be configured to use any thermistor. A function that converts measured voltage into temperature must be determined and configured for the particular thermistor in use.

A sixth-order polynomial is used within the BMS to model this transfer function:

$$T(v) = COEFF_0 + COEFF_1 * v + COEFF_2 * v^2 + COEFF_3 * v^3 + COEFF_4 * v^4 + COEFF_5 * v^5 + COEFF_6 * v^6$$

`stack_therm_poly.coeff0`

- Set to $COEFF_0$ (Floating-point value)

`stack_therm_poly.coeff1`

- Set to $COEFF_1$ (Floating-point value)

`stack_therm_poly.coeff2`

- Set to $COEFF_2$ (Floating-point value)

`stack_therm_poly.coeff3`

- Set to $COEFF_3$ (Floating-point value)

`stack_therm_poly.coeff4`

- Set to $COEFF_4$ (Floating-point value)

`stack_therm_poly.coeff5`

- Set to $COEFF_5$ (Floating-point value)

`stack_therm_poly.coeff6`

- Set to $COEFF_6$ (Floating-point value)

The thermistor voltage is read by a 10k pull-up to 3.00V. The first step in calculating coefficients for a thermistor is to create a table in Microsoft Excel™ or equivalent spreadsheet application with

the following columns:

Temperature (C)	Resistance (Ohms)	Vadc (V)
-40	334274	2.91286
-35	241323	2.88063
. . . .	. . . .	. . . .
125	336.75	0.09773

Temperature and resistance values are taken from the datasheet of the thermistor. V_{adc} is calculated using the following formula:

```
Vadc = 3.0 * (Resistance / (Resistance + 10000))
```

Using the line plot feature, create a graph of V_{adc} vs Temperature and turn on the trend line. Then modify the trend line to be a 6th order polynomial type, and display the equation on the chart. The equation will look like:

$$T(V) = 151.68 + (-352.94)V + 549.33V^2 + (-482.08)V^3 + 223.69V^4 + (-51.518)V^5 + 4.5693V^6$$

These polynomial coefficients can then be used to configure Nuvation BMS for this thermistor. An example plot of these measurements is shown in the next figure.

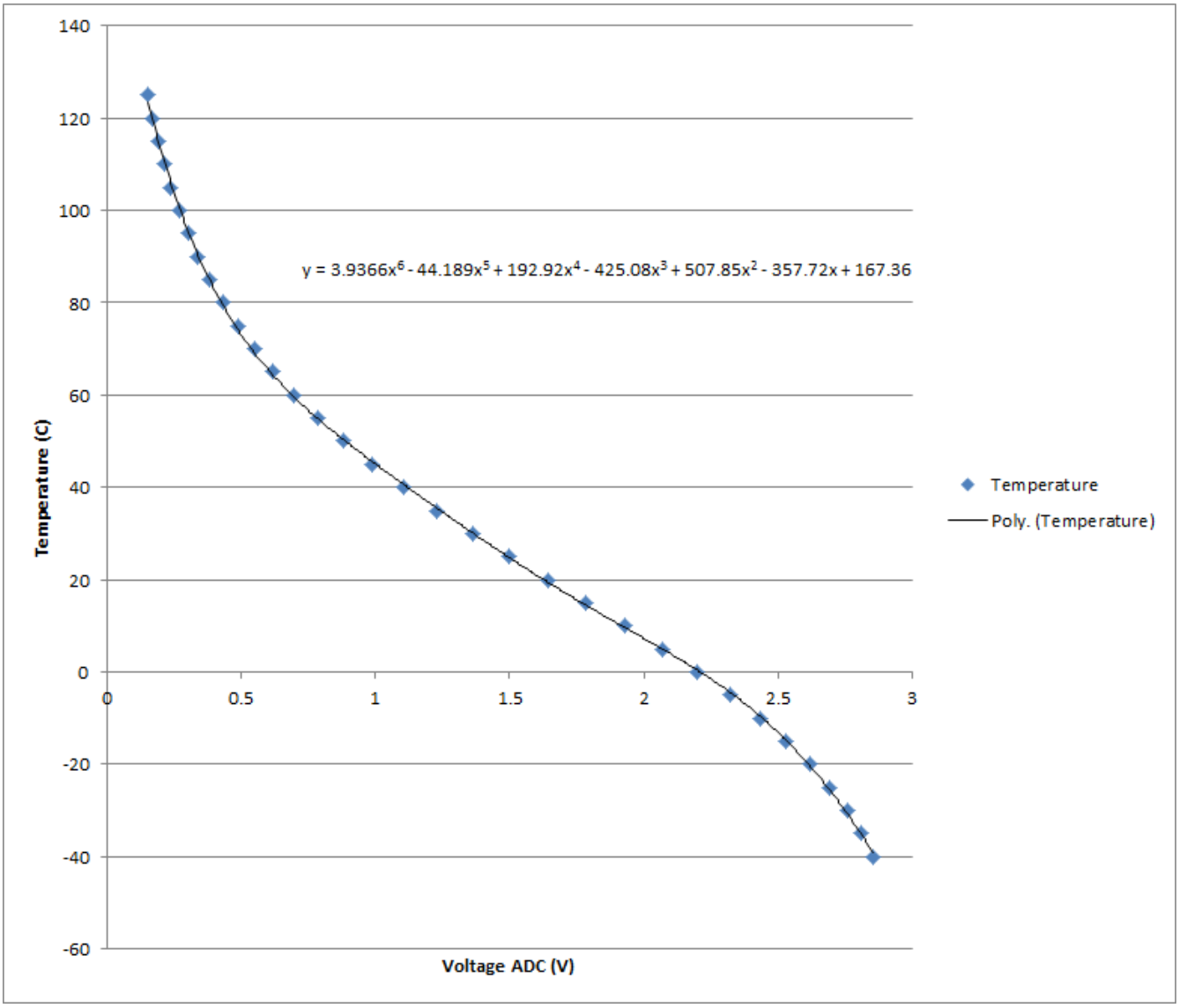

Figure 5. Typical Voltage ADC vs Thermistor Plot

3.8.2. Stack Current Calibration

Current measurements are made using a current sensing analog front end that may be configured for a wide range of current shunt resistances.

Current Shunt Calibration

Calibration of current readings is implemented according to the following formula that converts ADC readings into a current:

```
I (currentadc) = multiplier (currentadc / divider)
```

The multiplier and divider are set in configuration as follows.

`pi_afe_iadc.multiplier`

- Calculated calibration setting

- Set as per current shunt selection

`pi_afe_iadc.divider`

- Calculated calibration setting
- Set as per current shunt selection

Calculating an appropriate multiplier and divider is best illustrated through an example. Assume a 5mOhm current shunt is chosen that will have 50mV across its terminals at 10A. This voltage across the shunt at 10A will be used to determine the multiplier and divider.

The first step is to calculate the conversion factor between the measured voltage and the calculated current value using the following equation:

$$a = current / ((V_{adc}/300) * 2^{31})$$

In this example, the conversion factor is 0.000027939 (10000mA and V_{adc} is 50mV).

The next step is to express this factor according to the following equation:

$$a = MULTIPLIER/DIVIDER$$

The best approach is to use the divider to achieve the desired precision and then use the multiplier to achieve the desired accuracy, keeping in mind that both values must be expressed as integers. In this case, the divider we will start with is 10000000 and the multiplier will then be 279.

Additional work can still be done to refine these values. Searching for more accurate multipliers and dividers through a spreadsheet or trial and error can reduce rounding error. For example a divider of 111000 and a multiplier of 31 are more accurate by a decimal place than the previous values. Running experimental calibration using an external tool to measure current (such as a multimeter) provides the best estimate.

For older versions of firmware, specifically versions before 4.58.0, these settings were different. To migrate multiplier/divider settings from before 4.58.0 to later versions, ensure the multiplier is reduced by a factor of 256. For example: if the old multiplier was 1024, the new multiplier should be 1024/256 = 4.

Charge Deadband Setting

In addition to calibrating current readings, it is also necessary to define the deadband that will be used to determine whether the battery stack is charging, discharging, or at rest (holding).

`stack_charge_status.hold_current`

- The battery stack is considered at rest unless absolute value of current exceeds this threshold
- Set as required by application (typically 5-15 mA)

While this threshold is used to qualify certain aspects of SOC estimation, Coulomb counting takes place continuously regardless of the configured hold current value. Other functions that require knowledge of the charge or discharge state of the stack (e.g. the thermistor temperature

thresholds) also make use of this threshold.

3.8.3. Stack Voltage Calibration

Stack voltage measurements are made using a voltage sensing analog front end that must be configured for Nuvation BMS. The settings will differ depending on whether you have a Nuvation High-Voltage BMS™ or a Nuvation Low-Voltage BMS™.

Calibration operates according to the following formula that converts an ADC reading to a voltage:

```
V (voltageadc) = MULTIPLIER (voltageadc/DIVIDER)
```

The multiplier and divider are set in configuration as follows.

`pi_afe_vadc.multiplier`

- For Battery Controller Default Calibration: set to -1
- For Power Interface (PI-HE) Default Calibration: set to 1

`pi_afe_vadc.divider`

- For Battery Controller Default Calibration: Set to 28436
- For Power Interface (PI-HE) Default Calibration: Set to 1414

While these defaults are likely acceptable for most applications, the calibration can be fine-tuned as needed for particular situations.

For older versions of firmware, specifically versions before 4.58.0, these settings were different. To migrate multiplier/divider settings from before 4.58.0 to later versions, ensure the multiplier is reduced by a factor of 256. For example: if the old multiplier was 1024, the new multiplier should be 1024/256 = 4.

Older Power Interface Models

For High-Voltage BMS systems: All new systems use the PI-HE model since it is the only variant currently in production. However, some existing systems may need to be calibrated for older Power Interface models.

Calibration values for the full list of Power Interface models are shown in the table below.

PI-Model	multiplier	divider
PI-A	1	17729
PI-B	1	7664
PI-C	1	2774
PI-D	1	1414
PI-HE	1	1414

3.9. Hardware Settings

A number of hardware-specific firmware settings are required to configure your Nuvation BMS for operation. In most cases, these settings should be left at the recommended defaults. However, they may be modified for specialized applications.

3.9.1. Module Specific Selection

Battery Controller Selection

For Nuvation Low-Voltage BMS™ systems, the following configuration registers must be set as per the Battery Controller variant in use for your application.

`sc_linkbus.softwareid`

- Selection of Battery Controller variant
- Set to 0 for NUV300-BC-12
- Set to 1 for NUV300-BC-16

`sc_linkbus.cicount`

- Set to 1 for NUV300-BC-12 and NUV300-BC-16

`sc_linkbus.power_mode`

- Set to 2 for NUV300-BC-12 and NUV300-BC-16

Refer to the under voltage shutdown section for details on how to configure this feature for the Battery Controller.

Cell Interface Selection

For Nuvation High-Voltage BMS™ systems, The following configuration registers must be set as per the Cell Interface variant in use for your application.

`sc_linkbus.softwareid`

- Set to 0 for NUV100-CI-12
- Set to 1 for NUV100-CI-16
- Set to 2 for NUV100-CI-4M12

`sc_linkbus.cicount`

- Set to the total number of Cell Interfaces connected to the Stack Controller

Cell Interface Power Source

For Nuvation High-Voltage BMS™ systems, the power required by the Cell Interface may be sourced either from the battery cells themselves or from the 24VDC supply of the BMS through the *LinkBus*.

`sc_linkbus.power_mode`

- Set to 0 to power Cell Interfaces from battery cell power
- Set to 1 to power Cell Interfaces from *LinkBus* power

3.9.2. LinkBus Scan Period

The rate at which cell voltage and temperature measurements are updated is determined by a configurable *LinkBus* scan period. All cell voltages are measured every scan cycle. Thermistor temperature measurement is multiplexed across eight consecutive scan cycles. One thermistor in each group of cells (monitored by a single BMS module) is measured during each cycle. So the effective scan period of any given thermistor in the system is 8 times the *LinkBus* scan period.

`sc_linkbus.scan_period`

- Measurement period for cell voltages.
- This is typically set to 1 second.

Performance of passive balancing is also closely related to the scan period. In Nuvation BMS, a single wiring harness is used to measure cell voltage as well as bleed off passive cell balancing current. It is not possible to make accurate voltage measurements while simultaneously balancing cells. To ensure accurate measurement, balancing current flow must stop before measurement can start (and any transient effects associated with that current flow must be allowed to settle). This means that passive balancing has some duty cycle that is less than 100% in practical systems.

The *LinkBus* has a configurable settling period for balancing that corresponds to the amount of time balancing is left off before voltage measurements are taken.

`sc_linkbus.balance_settle_period`

- Settling period during which balancing is disabled before cell voltage measurement.
- This is typically set to 50-100 milliseconds.

For most systems, values between 50-100ms will yield accurate, stable measurements. If this is coupled with a scan period of 1000ms, balancing duty cycles above 90% are achievable.

For some applications, it may be desirable to de-rate the effective passive balancing current by increasing the balancing settling period as a way to decrease the balancing duty cycle. If the balancing settling period is increased beyond the scan period, the actual scan rate of the system will start to decrease below the configured value. If this is increased too far, the reduced scan period will eventually trip the scan rate watchdog, putting the system into the fault state.

3.9.3. Fault Pilot Signal

The Fault Pilot signal is used to open the contactors through a secondary control path in the case of a fault condition. When the system is unsafe, the Fault Pilot signal should be suppressed to guarantee the contactor coils are de-energized, regardless of the state of the coil control software.

Most applications should drive this signal from a delayed version of the overall fault state of the BMS. This small delay is necessary to allow for the opening of directional contactors according to any delays configured as part of the contactor outputs.

The fault state delay is configured using the following trigger with hysteresis.

`stack_delayed_fault_state.thresh`

- Set to 0

`stack_delayed_fault_state.time_hyst`

- Set according to application (typically, between 100ms and 5s)

`stack_delayed_fault_state.end_time_hyst`

- Set to 0

The Fault Pilot signal is controlled through one of the Digital Outputs.

`sc_gpo_fault_pilot.address_enabled`

- Set to 1 to enable Fault Pilot signal function in `sc_gpo_fault_pilot.address`

`sc_gpo_fault_pilot.inverted`

- Set to 0 so the Fault Pilot is suppressed when the system is unsafe

`sc_gpo_fault_pilot.address`

- Set to `@stack_delayed_fault_state.trig`

For advanced applications, the Fault Pilot signal may be configured to be driven from any Boolean register within the BMS.

3.9.4. Under Voltage Shutdown

The Battery Controller has the hardware feature to shutoff its power when the battery is at a critical low voltage level. This action prevents the battery management system from draining the remaining power from the battery and prevent further damage. The shutdown feature integrates with the same mechanism used by the shutdown input of the Battery Controller. The following registers need to be configured to manage this feature.

It may be possible to implement an under voltage shutdown in the Nuvation High-Voltage BMS™ depending on your application. Please contact support@nuvationenergy.com for more information.

A combined `Under Voltage Lock Out (UVLO)` trigger is used to provide a shutdown input for either a critcally low stack or cell voltage.

`stack_uvlo_cell_voltage.thresh`

- Set to the lowest safe cell voltage for a battery. Must be the lowest voltage possible for battery safety.

`stack_uvlo_stack_voltage.thresh`

- Set to the lowest safe stack voltage for a battery.

The above triggers are logically OR'd together into a combined trigger named `stack_uvlo_combined`. The output of this trigger needs to be directed to the internal shutdown of the Battery Controller using the following register configuration.

`sc_gpo_shutdown.address`

- Set to `@stack_uvlo_combined.trig`

`sc_gpo_shutdown.inverted`

- Set to 1

`sc_gpo_shutdown.address_enabled`

- Set to 1

4. Troubleshooting

4.1. Faults and Initialization

During initial setup and configuration of a Nuvation BMS, there are two main classes of issues that can be encountered:

1. Initialization issues
2. Triggering of faults

The first class of issues are encountered during initial deployment of a Nuvation BMS, particularly while attempting to exit Service Lockout. When this event occurs, an operator of the OI can open the Details|Safety accordian. That screen presents a list of fault and warning triggers that may or may not be initialized. This section will provide details on why the faults/warnings could not be initialized. Once all faults/warnings have been initialized, the Nuvation BMS has been configured and installed as expected (i.e. all inputs are receiving data). When this occurs the software enters its operational mode.

The second class of issues occurs while the Nuvation BMS is operational. One or more faults can be triggered causing the Nuvation BMS to exit it's safety condition and open all contactors. There are many faults that can be triggered. This section will describe the condition the fault monitors and the meaning when that fault is triggered.

The following sections describe different classes of faults/warnings and issues surrounding any initialization. In general, all warnings have a similar trigger condition as their corresponding fault. The following discussion will focus on the term fault and all descriptions can be applied to the compatible warning.

4.1.1. Cell Voltage Faults

These faults are tied directly to the scanning on the Cell Interface modules for the cell voltages. Note that for the Nuvation BMS™ Low-Voltage Battery Controller product, there is one Cell Interface within the enclosure that is not accessible. Additional Cell Interface modules can be added using the `Link Out` port.

`stack_fault_cell_hi`

- An installed cell(s) voltage exceeded the fault threshold

`stack_fault_cell_lo`

- An installed cell(s) voltage was below the fault threshold

Initialization Issues

These faults could fail to initialize through a number of possible conditions such as :

1. Missing or misconfigured Cell Interface modules, preventing the software from scanning for the measurements.
2. Misconfiguration on the number of Cell Interface modules. Refer to the register (sc_linkbus.cicount).
3. Incorrect Cell Interface configuration. For example if the type of Cell Interface (sc_linkbus.softwareid) is incorrect it may prevent the cell voltage measurement.
4. Failure in the ISO SPI of the LinkBus. Any interruptions of the bus can prevent the cell voltages

from being initialized. Such a failure could happen if :

- Not all CI modules are connected.
- Link-bus cables connected to the wrong port (Link-Out instead of Link-In).
- Damage to the link-bus cables.
- Not enough cells are connected to the Nuvation BMS™ Low-Voltage Battery Controller.

4.1.2. Stack Voltage Faults

`stack_fault_voltage_hi`

- The stack voltage measured exceeded the fault threshold.

`stack_fault_voltage_lo`

- The stack voltage measured exceeded the fault threshold.

`stack_fault_voltage_sum`

- The absolute difference between the measured stack voltage and sum of all cell voltages in the stack exceeded the fault threshold.

Initialization Issues

These faults have the following initialization issues:

Fault	Initialization Issues
`stack_fault_voltage_hi`	• PI AFE is configured to be disabled
`stack_fault_voltage_lo`	
`stack_fault_voltage_sum`	Same issues regarding cell voltage initialization

4.1.3. Combined Voltage Faults

`stack_fault_combined_voltage_hi`

- Fault which combines the high cell and stack voltage faults via an OR operation.

`stack_fault_combined_voltage_lo`

- Fault which combines the low cell and stack voltage faults via an OR operation.

These faults have the same intitialization issues as the cell and stack voltage faults and will be uninitialized if either measurement fails to initialize.

4.1.4. Thermal Faults

`stack_fault_discharge_therm_hi`

- Fault that is triggered when any thermistor measurement exceeds the fault threshold during discharge

`stack_fault_discharge_therm_lo`

- Fault that is triggered when any thermistor measurement is below the fault threshold during discharge

`stack_fault_charge_therm_hi`

- Fault that is triggered when any thermistor measurement exceeds the fault threshold during charge

`stack_fault_charge_therm_lo`

- Fault that is triggered when any thermistor measurement is below the fault threshold during charge

`stack_fault_therm_hi`

- Fault that is OR combination of the charge/discharge high faults

`stack_fault_therm_lo`

- Fault that is OR combination of the charge/discharge low faults

The prior two faults are used for aggregating thermal faults for both the charge/discharge states.

`sc_fault_ci_therm_hi`

- Fault that is triggered when any CI silicon temperature measurement is above the fault threshold

`sc_fault_ci_therm_lo`

- Fault that is triggered when any CI silicon measurement is below the fault threshold

These two prior faults use measurements directly from the CI hardware silicon. These measurements are uncalibrated and can vary substantially in value. However the trend in temperature measurement follows the ambient environment. Use of these faults are highly coupled with the thermal environment of the hardware and must be carefully configured and assessed before use.

Initialization Issues

All of the temperature measurements are accomplished through the same ISO SPI bus used to measure the cell voltages. Thus the failures to initialize the temperature measurements are exactly the same. Refer to cell voltage initialization for further details.

4.1.5. Current Faults

`stack_fault_current_hi`

- Fault that is triggered when any stack current exceeds the fault threshold.

`stack_fault_current_lo`

- Fault that is triggered when any stack current exceeds the fault threshold.

Note that both of these prior faults could trigger if the current shunt is not properly connected or has not been cofigured for it’s proper measurement range.

Initialization Issues

The stack current is measured from the Power Interface analog front end. The following issues can contribute to these faults not initializing:

1. PI AFE is not enabled
2. Stack bus failure between the Stack Controller and Power Interface

4.1.6. Precharge Faults

`stack_fault_precharge_timeout`

- Fault that is triggered at the end of the pre-charge period if the measured current exceeds the maximum pre-charge current.

`stack_fault_precharge_over_current`

- Fault that is triggered at any time during the pre-charge connection period if the stack current exceeds the fault thereshold.

Initialization Issues

These faults are directly related to the current faults detailed previously. Refer to current initialization for details.

4.1.7. Contactor Faults

`stack_fault_coil_fail`

- Fault that is generated when there is a difference between the commanded state of the contactor coil and the observed state read from the Power Interface hardware. This fault is a consistency check on the drive state of the coil. This fault will trigger when contactors are unconnected or have a short in their circuit. If the fault pilot signal is asserted, this fault will trigger and can not be used to determine if there is a inconsistent contactor drive state.

`stack_fault_contactor_feedback_fail`

- Fault that is generated when there is a mismatch between the contactor state and the contactor feedback signal provided. Note that the feedback is provided through a GPI.

`sc_fault_pi_interlock`

- Fault that is generated when the interlock function of the High-Voltage BMS has been activated. The interlock when enabled is activated through the interlock circuit having an open condition (i.e. via an e-stop or door switch).

Initialization Issues

All of these faults are dependant on the stack bus communication between the Stack Controller and Power Interface in the High-Voltage BMS product. The following table summarizes these initialization issues:

Fault	Initialization Issues
`sc_fault_pi_interlock`	• Stack bus failure
`stack_fault_coil_fail`	• Power Interface AFE disabled • Stack bus failure
`stack_fault_contactor_feedback_fail`	• GPI not configured • Power Interface AFE disabled • Stack bus failure

4.1.8. Breaker Faults

`stack_fault_breaker_tripped`

- Fault that is triggered to indicate that the breaker has tripped (i.e. opened)

`stack_fault_breaker_conflict`

- Fault that is triggered to indicate that the breaker state differs from the expected state of the breaker.

Initialization Issues

Failure to initialize these faults is caused by an invalid configuration of the GPI(s) used to read the breaker state.

4.1.9. Watchdog Faults

`sc_fault_linkbus_wdt`

- Fault indicating that there was a communication failure over the linkbus (connecting Stack Controller to Cell Interface).

`sc_fault_stackbus_rxwdt`

- Fault indicating that there was a receive communication failure over the stackbus (connecting Stack Controller to Power Interface).

`sc_fault_stackbus_txwdt`

- Fault indicating that there was a transmit communication failure over the stackbus (connecting Stack Controller to Power Interface).

`sc_fault_pi_afe_wdt`

- Fault indicating that there the AFE was not communicating with the Power Interface.

`sc_fault_controller_wdt`

- Fault indicating that an external controller to the battery management system was not updating its watchdog timer (via the MESA heatbeat).

Initialization Issues

The `sc_fault_linkbus_wdt` can be uninitialized if it is misconfigured (i.e. incorrect number of CIs) Please refer to the configuration of the CI for these types of failures.

The `sc_fault_stackbus_rxwdt` and `sc_fault_stackbus_txwdt` should always initialize as it is core communicaiton to the battery management system and does not have configuration associated with it. Please contact Nuvation Energy at support@nuvationenergy.com if you are experiencing this fault.

The `sc_fault_pi_afe_wdt` can be uninitialized due to disabling the PI AFE or a high amount of noise on the DC bus.

The `sc_fault_controller_wdt` can be uninitialized if there is no external controller updating the heartbeat. In this case this fault should be disabled.

4.1.10. Miscellaneous Faults

`sc_fault_config`

- Fault indicating that there was an error reading the non-volatile storage of the battery management system configuration. A default configuration is used when this fault occurs

and the battery management system will fail to exit Service Lockout. Once a configuration has been successfully imported into the battery management system and saved, this fault should not occur. Note that during an upgrade of the Stack Controller and Power Interface, the configuration is deleted to avoid incompatibility with the upgraded version.

`sc_fault_fw_mismatch`

- Fault indicating that the Stack Controller and Power Interface firmware versions are different. Perform an upgrade to the desired battery management system firmware version.

4.2. Lost/Forgotten IP Address

If a Nuvation BMS has been configured with a static IP address and it has been forgotten, follow the steps below to recover it.

The term *module* used below refers to the Battery Controller or Stack Controller as it applies to your system.

Depending on the network interface used on the PC, this process may not work due differing security configurations. If the IP discovered is the IP of the PC, the network interface is not suitable and another one will need to be used. This seems to be particularly problematic with USB to Ethernet dongles.

4.2.1. Wireshark (Windows/Linux)

1. Download/install Wireshark on a PC (https://www.wireshark.org/)
2. Connect the PC directly to the Ethernet port
3. Start a Wireshark capture on the network interface connected to the module
4. In the 'filter' field, enter in `arp.isgratuitous` and press enter
5. Either reboot the module, or unplug/plug the Ethernet cable
6. The module should send a 'Gratuitous ARP' on the Ethernet network. In Wireshark the 'Info' field looks like: `Gratuitous ARP for <IP> (Request)` where the `<IP>` is the address for the module
7. Once that is complete, update the PC network settings to match the SC and connect the UI to re-configure

4.2.2. Netdiscover (Linux only)

1. Install `netdiscover` on a PC (on Debian based systems use: `sudo apt install netdiscover`)
2. Plug the PC directly into the module Ethernet port
3. Run `sudo netdiscover -i <interface> -p` where `<interface>` is the network interface connected to the module
4. Either reboot the module, or unplug/plug the Ethernet cable
5. The module address and MAC will show up in `netdiscover` once an ARP packet is sent
6. Once that is complete, update the PC network settings to match the SC and connect the UI to re-configure

5. Registers

C

cell, 10
  installed, 9, 30, 31
  voltage, 30

P

pi_afe_iadc
  divider, 37
  multiplier, 37
pi_afe_vadc
  divider, 38
  multiplier, 38

S

sc_canbus
  base_can_address, 27, 28, 28, 29, 30, 30, 30
  enable, 28
  enabled, 27
  numtoread, 30
  report_interval, 27, 28
  report_msg_interval, 27, 28
sc_canbus.report_msg_interval, 32
sc_canbus_bulkrpt, 30, 30, 30
  baseaddress, 30, 30, 33, 34
  baseenableaddress, 30
  baseenabled, 34
  baseenabledaddress, 30, 33
  numtoread, 30, 30, 30, 31, 34
  offset, 30, 30
sc_canbus_map
  address, 28, 28
sc_canbus_packets
  err_rate_window, 27, 28
sc_clock
  seconds, 28
sc_controller_heartbeat
  value, 29
sc_controller_wdt
  period, 17
sc_fault_ci_therm_hi
  thresh, 14
sc_fault_ci_therm_lo
  thresh, 14
sc_fault_config, 47
sc_fault_controller_wdt, 46
  disabled, 17
sc_fault_fw_mismatch, 47
sc_fault_linkbus_wdt, 46
sc_fault_pi_afe_wdt, 46
sc_fault_pi_interlock, 45
sc_fault_stackbus_rxwdt, 46
sc_fault_stackbus_txwdt, 46
sc_gpi
  address, 26, 26
  address_enabled, 26, 33
  falling_edge_triggered, 26
  inverted, 26, 26, 33
  rising_edge_triggered, 26, 26
  value, 34, 34
sc_gpo
  address, 25, 26
  address_enabled, 25
  inverted, 25, 26
sc_gpo_fault_pilot
  address, 40, 41
  address_enabled, 40
  inverted, 41
sc_gpo_shutdown
  address, 41
  address_enabled, 41
  inverted, 41
sc_linkbus
  balance_settle_period, 40
  cicount, 39, 39
  power_mode, 39, 39
  scan_period, 40
  softwareid, 39, 39
sc_modbus_rtu
  device_address, 34
sc_trig_ci_therm_hi
  thresh, 14
sc_trig_ci_therm_lo
  thresh, 14
sc_warn_ci_therm_hi
  thresh, 14
sc_warn_ci_therm_lo
  thresh, 14
stack_cell_balancer
  enabled, 21
  max_ci_enable_temperature, 22
  max_enable_current, 22
  max_enable_temperature, 22
  min_enable_current, 22
  min_enable_voltage, 21, 21
  voltage_delta, 21, 21
stack_cell_stat
  avg, 28
  max, 28
  min, 28
stack_charge_status
  hold_current, 37
stack_contactor
  address, 24

address_enabled, 24
coil_error, 25
delay, 25
direction, 25
feedback_enable, 25
feedback_error, 25, 25
feedback_value, 25, 26
installed, 25
inverted, 24
value, 25
stack_control, 23, 23
connect_delay, 19
connection_state, 29
disconnect_delay, 20
main_switch_state, 24
precharge_delay, 19, 19
precharge_max_current, 19
precharge_switch_state, 24
requested_state, 26, 29
stack_switch_state, 24
stack_control.auto_connect, 19
stack_current_limit
attack_settling_time, 20, 20
charge_current_disable, 26
charge_current_limit, 28
charge_current_percent, 28
decay_settling_time, 20
discharge_current_disable, 26
discharge_current_limit, 28
discharge_current_percent, 28
max_charge_current, 20
max_discharge_current, 20
min_charge_current, 20
temperature_charge_high, 13
temperature_charge_low, 13
temperature_charge_max, 13, 13
temperature_charge_min, 13, 13
temperature_discharge_high, 13
temperature_discharge_low, 13
temperature_discharge_max, 13, 13
temperature_discharge_min, 13, 14
voltage_cell_high, 12, 21
voltage_cell_low, 12
voltage_cell_max, 12, 12, 16
voltage_cell_min, 12, 12
voltage_stack_high, 16
voltage_stack_low, 16
voltage_stack_max, 16
voltage_stack_min, 16, 16
stack_delayed_fault_state
end_time_hyst, 40
thresh, 40
time_hyst, 40
trig, 41
stack_fault_breaker_conflict, 46
stack_fault_breaker_tripped, 46
stack_fault_cell_hi, 42
thresh, 12
stack_fault_cell_lo, 42
thresh, 12
stack_fault_charge_therm_hi, 43
thresh, 13
stack_fault_charge_therm_lo, 44
thresh, 13
stack_fault_ci_therm_hi, 44
stack_fault_ci_therm_lo, 44
stack_fault_coil_fail, 45
stack_fault_combined_voltage_hi, 43
stack_fault_combined_voltage_lo, 43
stack_fault_contactor_feedback_fail, 45
stack_fault_current_hi, 44
thresh, 15
stack_fault_current_lo, 44
thresh, 15
stack_fault_discharge_therm_hi, 43
thresh, 13
stack_fault_discharge_therm_lo, 43
thresh, 14
stack_fault_precharge_over_current, 45
stack_fault_precharge_timeout, 45
stack_fault_therm_hi, 44
stack_fault_therm_lo, 44
stack_fault_voltage_hi, 43
thresh, 16, 17
stack_fault_voltage_lo, 43
thresh, 16
stack_fault_voltage_sum, 43
thresh, 17
stack_power
current, 28
voltage, 28
stack_safety
clear_faults, 26, 29
safe, 26, 28
safetocharge, 28
safetodischarge, 28
stack_soc
dod, 28
nominal_capacity, 8
nominal_cycle_count, 8
soc, 28
vempty, 9, 12
vemptyavg, 9
vfull, 9, 12

vfullavg, 9
stack_therm_poly
coeff0, 34
coeff1, 34
coeff2, 34
coeff3, 34
coeff4, 34
coeff5, 34
coeff6, 34
stack_therm_stat
avg, 28
max, 28
min, 28
stack_trig_cell_hi
thresh, 13
stack_trig_cell_lo
thresh, 13
stack_trig_charge_therm_hi
thresh, 14
stack_trig_charge_therm_lo
thresh, 14
stack_trig_current_hi
thresh, 15
stack_trig_current_lo
thresh, 15
stack_trig_discharge_therm_hi
thresh, 14
stack_trig_discharge_therm_lo
thresh, 14
stack_trig_voltage_hi
thresh, 17
stack_trig_voltage_lo
thresh, 17
stack_uvlo_cell_voltage
thresh, 12, 41
stack_uvlo_combined
trig, 41
stack_uvlo_stack_voltage
thresh, 16, 41
stack_warn_cell_hi
thresh, 12
stack_warn_cell_lo
thresh, 12
stack_warn_charge_therm_hi
thresh, 13
stack_warn_charge_therm_lo
thresh, 13
stack_warn_current_hi
thresh, 15
stack_warn_current_lo
thresh, 15
stack_warn_discharge_therm_hi
thresh, 13
stack_warn_discharge_therm_lo
thresh, 14
stack_warn_voltage_hi
thresh, 16
stack_warn_voltage_lo
thresh, 16

T

therm, 10
installed, 10, 30, 31
temperature, 30
trigger_name
disabled, 11
end_time_hyst, 11
latched, 11
thresh, 11
time_hyst, 11
trig, 26