NUVATION BMS™
NUVATION ENGINEERING

nuvationbms.com
nuvationbms@nuvation.com

SILICON VALLEY HEADQUARTERS
151 GIBRALTAR CT
SUNNYVALE, CA 94089 USA
408.228.5580

WATERLOO DESIGN CENTER
332 MARSLAND DR., SUITE 200
WATERLOO, ON N2J 3Z1 CANADA
519.746.2304

Nuvation BMS™ Software User Guide
2017-02-07, 17.01.1

Table of Contents

Terminology

Topology

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

Cell

A Cell is the smallest unit of energy storage distinguishable by the BMS. 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.

Pack

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

Hardware Modules

Nuvation BMS™ hardware is partitioned into four types of modules:

Cell Interface (CI)

Exposes interfaces to monitor and balance connected battery Cells.

Power Interface (PI)

Exposes interfaces to monitor current and voltage as well as control contactors. This board interfaces directly with high voltage and high current components.

Stack Controller (SC)

Controls a single Stack of series connected battery Cells. It operates as a “master” for all connected Power Interfaces and Cell Interfaces.

Grid Battery Controller (GBC)

Interfaces with multiple Stack Controllers to aggregate measurements and control signals for battery packs containing multiple stacks of cells.

Software Building Blocks

Nuvation BMS™ software is a scalable distributed system built out of a few fundamental building blocks:

Register

Each piece of data in the BMS (whether it is measurement, calculation, or configuration) is stored in a register. Each register has an associated type that defines how the value is interpreted and any associated units. Registers range in size from as small as one byte up to as large as eight bytes. Registers are primarily accessed by name, but also have a unique address that is used internally within the BMS for efficient communication.

Register File

A group of related registers are collected together into a register file. A register file is assigned a base address by the system and individual registers within the file are always contiguous in the address space.

Component

A component combines a register file with any associated processing or logic within the BMS related to those registers. For example, a trigger component would contain the register file for trigger configuration and state as well as the actual trigger processing functions. Numerous triggers are used within the BMS, but they are fundamentally derived from the same base component implementation.

Tools

User Interface

The primary user interface for Nuvation BMS™ is accessed through a web browser. This browser interface can be launched by double-clicking on the HTML file present in your Customer Starter Kit software release.

Register Map

The full register map of the BMS is available through the ‘Registers’ page, which is accessed through the gear icon in the top right corner. Each component within the BMS can be explored with the list of available registers. Component and register names documented here correspond with the names used in the configuration file.

It is possible to inspect BMS registers through the ‘Registers’ page. Registers can be read and written by name. Three text fields are provided to specify the: component name, component instance index, and register name.

Lock/Unlock

By default the user interface restricts operations to connect/disconnect of the pack, clearing faults, and reading registers. To configure and upgrade the BMS the interface must be unlocked. To unlock a user can click the ‘Unlock’ item found in the menu accessed through the gear icon in the top right corner. The default password is “*thebestbms*”.

Configuration

System-wide configuration for the BMS can be uploaded though the ‘Configure’ page found in the gear icon menu in the top right corner. Configuration files are formatted as follows:

- Plain text (ASCII). Any standard text editor should work (e.g. Windows Notepad, Notepad++, etc.).
- Any lines starting with a leading ‘#’ are treated as comments.
- Any value starting with a leading ‘@’ followed by a register name resolves to an integer address.
- Each non-comment line is a register-value pair and is formatted as *component[index].register = value*. If no *index* is specified, it is assumed to be zero.

An example of valid Nuvation BMS™ configuration syntax is as follows:

```
# Current limiting
stack_current_limit.voltage_cell_min = 32000
stack_current_limit.voltage_cell_low = 32000
stack_current_limit.voltage_cell_high = 64000
stack_current_limit.voltage_cell_max = 64000

# SC GPO: GPO 0 and 1 indicate overall safety
sc_gpo[0].inverted = 1
sc_gpo[0].enabled = 1
sc_gpo[0].address = @stack_safety[0].safe
sc_gpo[1].inverted = 0
sc_gpo[1].enabled = 1
sc_gpo[1].address = @stack_safety[0].safe
```

Configuration files are downloaded by clicking the 'Download' button. The downloaded files are in the same format as uploaded files and can be edited and uploaded at a later time.

Compressed Syntax

To perform operations across multiple instances of a component a configuration file entry may contain a compressed syntax. The syntax follows the format below:

```
component[start_index:end_index:<skip>:<skip_count>].name
```

An example is:

```
Name: therm[0:3:8:4].installed
therm[0].installed=0
therm[1].installed=0
therm[2].installed=0
therm[3].installed=0
therm[8].installed=0
therm[9].installed=0
therm[10].installed=0
therm[11].installed=0
therm[16].installed=0
therm[17].installed=0
therm[18].installed=0
therm[19].installed=0
therm[24].installed=0
therm[25].installed=0
therm[26].installed=0
therm[27].installed=0
```

Using this syntax a configuration file can be efficiently created without excessive duplication.

Firmware Upgrade

A firmware upgrade for all modules within the BMS can be initiated from the ‘Upgrade’ page, accessed in the menu via the gear icon in the top right corner. This tab displays the current version present on the BMS as well as the version available for upgrade. The upgrade is initiated by clicking the ‘Upgrade’ button and completes after all connected BMS modules have upgraded.

Stack Configuration Quick Start

This section highlights the most important settings required to achieve a working BMS stack configuration for your system.

Warning: It is assumed that your BMS has been pre-calibrated by a Nuvation Engineer. The level of detail provided here is insufficient to safely configure a "blank system" from scratch.

Note: This section was written for the BMS firmware version 4.57.0. There is no guarantee other versions are compatible with the information outline here.

Topology

Configuring your battery stack topology involves specifying the following:

- The number of Cell Interfaces connected to your Stack Controller
- The type of Cell Interfaces connected
- Which cell voltage taps are actually connected to cells
- Which thermistor inputs are actually connected to thermistors

Cell Interfaces

The Cell Interfaces come in a number of variants; CI-12, and CI-16 which support 12x5V and 16x5V cells respectively. All CIs have 8 thermistor inputs. The following components configure how many CIs are present in the system and what type of CI is used.

sc_linkbus

cicount The integer of how many CIs are present in the system.

ci

softwareid An ID that sets the type of CI. (CI-12 = 0, CI-16 = 1).

Cells

The Cell Interface can connect to less than the maximum supported number of cells. The following components configure how many cells are present on each Cell Interface.

cell

installed A boolean flag that sets if the cell input is connected to a cell.

Thermistors

The Cell Interface can connect to less than the maximum supported number of thermistors. The following components configure how many thermistors are present on each Cell Interface.

therm

installed A boolean flag that sets if the thermistor input is connected to a thermistor.

Thresholds

The operational limits of your battery pack are captured in the form of voltage, temperature, and current thresholds. Properly adjusting these thresholds for your cells and application is essential to achieving a working system.

Cell Voltage

The following is a list of components which set cell voltage thresholds.

stack_fault_cell_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The millivolt threshold that is tripped when a cell voltage rises above this value.

time_hyst The microsecond duration that a cell voltage must remain above the threshold before it trips.

end_time_hyst The microsecond duration that a cell voltage must remain below the threshold before it resets.

stack_warn_cell_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The millivolt threshold that is tripped when a cell voltage rises above this value.

time_hyst The microsecond duration that a cell voltage must remain above the threshold before it trips.

end_time_hyst The microsecond duration that a cell voltage must remain below the threshold before it resets.

stack_warn_cell_lo

trig A boolean flag that indicates if this threshold has been tripped.

thresh The millivolt threshold that is tripped when a cell voltage falls below this value.

time_hyst The microsecond duration that a cell voltage must remain below the threshold before it trips.

end_time_hyst The microsecond duration that a cell voltage must remain above the threshold before it resets.

stack_fault_cell_lo

trig A boolean flag that indicates if this threshold has been tripped.

thresh The millivolt threshold that is tripped when a cell voltage falls below this value.

time_hyst The microsecond duration that a cell voltage must remain below the threshold before it trips.

end_time_hyst The microsecond duration that a cell voltage must remain above the threshold before it resets.

stack_cell_balancer

minenablevoltage The millivolt threshold that is tripped when a cell voltage rises above this value.

stack_current_limit

voltage_cell_max The high taper end point for cell voltage.

voltage_cell_high The high taper start point for cell voltage.

voltage_cell_low The low taper start point for cell voltage.

voltage_cell_min The low taper end point for cell voltage.

stack_soc

vfull The voltage of the maximum cell at which point the pack is considered full.

vempty The voltage of the minimum cell at which point the pack is considered empty.

Cell Temperature

The following is a list of components which set cell temperature thresholds. Separate thresholds are available for charging and discharging as most modern cells have different limitations in these two modes of operation. Charge faults and warnings will only trip if the cells are charging, while discharge faults and warnings will only trip if the cells are discharging.

stack_fault_discharge_therm_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The centigrade threshold that is tripped when a cell temperature rises above this value.

time_hyst The microsecond duration that a cell temperature must remain above the threshold before it trips.

end_time_hyst The microsecond duration that a cell temperature must remain below the threshold before it resets.

stack_fault_charge_therm_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The centigrade threshold that is tripped when a cell temperature rises above this value.

time_hyst The microsecond duration that a cell temperature must remain above the threshold before it trips.

end_time_hyst The microsecond duration that a cell temperature must remain below the threshold before it resets.

stack_warn_discharge_therm_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The centigrade threshold that is tripped when a cell temperature rises above this value.

time_hyst The microsecond duration that a cell temperature must remain above the threshold before it trips.

end_time_hyst The microsecond duration that a cell temperature must remain below the threshold before it resets.

stack_warn_charge_therm_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The centigrade threshold that is tripped when a cell temperature rises above this value.

time_hyst The microsecond duration that a cell temperature must remain above the threshold before it trips.

end_time_hyst The microsecond duration that a cell temperature must remain below the threshold before it resets.

stack_trig_discharge_therm_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The centigrade threshold that is tripped when a cell temperature rises above this value.

time_hyst The microsecond duration that a cell temperature must remain above the threshold before it trips.

end_time_hyst The microsecond duration that a cell temperature must remain below the threshold before it resets.

stack_trig_charge_therm_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The centigrade threshold that is tripped when a cell temperature rises above this value.

time_hyst The microsecond duration that a cell temperature must remain above the threshold before it trips.

end_time_hyst The microsecond duration that a cell temperature must remain below the threshold before it resets.

stack_warn_discharge_therm_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The centigrade threshold that is tripped when a cell temperature falls below this value.

time_hyst The microsecond duration that a cell temperature must remain below the threshold before it trips.

end_time_hyst The microsecond duration that a cell temperature must remain above the threshold before it resets.

stack_warn_charge_therm_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The centigrade threshold that is tripped when a cell temperature falls below this value.

time_hyst The microsecond duration that a cell temperature must remain below the threshold before it trips.

end_time_hyst The microsecond duration that a cell temperature must remain above the threshold before it resets.

stack_fault_discharge_therm_lo

trig A boolean flag that indicates if this threshold has been tripped.

thresh The centigrade threshold that is tripped when a cell temperature falls below this value.

time_hyst The microsecond duration that a cell temperature must remain below the threshold before it trips.

end_time_hyst The microsecond duration that a cell temperature must remain above the threshold before it resets.

stack_fault_charge_therm_lo

trig A boolean flag that indicates if this threshold has been tripped.

thresh The centigrade threshold that is tripped when a cell temperature falls below this value.

time_hyst The microsecond duration that a cell temperature must remain below the threshold before it trips.

end_time_hyst The microsecond duration that a cell temperature must remain above the threshold before it resets.

stack_cell_balancer

maxenabletemperature The temperature threshold above which balancing is disabled.

stack_current_limit

temperature_charge_max The high taper end point for thermistor temperature in charge mode.

temperature_discharge_max The high taper end point for thermistor temperature in discharge mode.

temperature_charge_high The high taper start point for thermistor temperature in charge mode.

temperature_discharge_high The high taper start point for thermistor temperature in discharge mode.

temperature_charge_low The low taper start point for thermistor temperature in charge mode.

temperature_discharge_low The low taper start point for thermistor temperature in discharge mode.

temperature_charge_min The low taper end point for thermistor temperature in charge mode.

temperature_discharge_min The low taper end point for thermistor temperature in discharge mode.

Stack Voltage

The following is a list of components which set stack voltage thresholds.

stack_fault_voltage_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The millivolt threshold that is tripped when the stack voltage rises above this value.

time_hyst The microsecond duration that the stack voltage must remain above the threshold before it trips.

end_time_hyst The microsecond duration that the stack voltage must remain below the threshold before it resets.

stack_warn_voltage_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The millivolt threshold that is tripped when the stack voltage rises above this value.

time_hyst The microsecond duration that the stack voltage must remain above the threshold before it trips.

end_time_hyst The microsecond duration that the stack voltage must remain below the threshold before it resets.

stack_warn_voltage_lo

trig A boolean flag that indicates if this threshold has been tripped.

thresh The millivolt threshold that is tripped when the stack voltage falls below this value.

time_hyst The microsecond duration that the stack voltage must remain below the threshold before it trips.

end_time_hyst The microsecond duration that the stack voltage must remain above the threshold before it resets.

stack_fault_voltage_lo

trig A boolean flag that indicates if this threshold has been tripped.

thresh The millivolt threshold that is tripped when the stack voltage falls below this value.

time_hyst The microsecond duration that the stack voltage must remain below the threshold before it trips.

end_time_hyst The microsecond duration that the stack voltage must remain above the threshold before it resets.

stack_current_limit

voltage_stack_max The high taper end point for stack voltage.

voltage_stack_high The high taper start point for stack voltage.

voltage_stack_low The low taper start point for stack voltage.

voltage_stack_min The low taper end point for stack voltage.

Stack Current

The following is a list of components which set stack current thresholds.

stack_fault_current_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The milliampere threshold that is tripped when the stack current rises above this value.

time_hyst The microsecond duration that the stack current must remain above the threshold before it trips.

end_time_hyst The microsecond duration that the stack current must remain below the threshold before it resets.

stack_warn_current_hi

trig A boolean flag that indicates if this threshold has been tripped.

thresh The milliampere threshold that is tripped when the stack current rises above this value.

time_hyst The microsecond duration that the stack current must remain above the threshold before it trips.

end_time_hyst The microsecond duration that the stack current must remain below the threshold before it resets.

stack_warn_current_lo

trig A boolean flag that indicates if this threshold has been tripped.

thresh The milliampere threshold that is tripped when the stack current falls below this value.

time_hyst The microsecond duration that the stack current must remain below the threshold before it trips.

end_time_hyst The microsecond duration that the stack current must remain above the threshold before it resets.

stack_fault_current_lo

trig A boolean flag that indicates if this threshold has been tripped.

thresh The milliampere threshold that is tripped when the stack current falls below this value.

time_hyst The microsecond duration that the stack current must remain below the threshold before it trips.

end_time_hyst The microsecond duration that the stack current must remain above the threshold before it resets.

stack_cell_balancer

maxenablecurrent The maximum current threshold above which balancing is disabled.

minenablecurrent The minimum current threshold below which balancing is disabled.

stack_current_limit

maxchargecurrent The milliampere threshold that is tripped when the stack current rises above this value.

maxdischargecurrent The milliampere threshold that is tripped when the stack current rises above this value.

lpfgain The microsecond window to average current limit over.

Passive Balancing

When multiple cells are stacked 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 SOC or a single high SOC cell will dominate the performance of the large battery stack. The act of equalizing SOC 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 SOC are pulled low 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.

Cell Balancing

The following is a list of components which set the balancing parameters.

stack_cell_balancer_stat

measurewhilebalancing A boolean flag that sets if cell voltage measurements should be taken and used while balancing.

stack_cell_balancer

enabled A boolean flag that enables the passive balancing algorithm.

voltagedelta A millivolt goal that the passive balancing algorithm will bring the cell voltages together.

Digital I/O

The Stack Controller and Power Interface provide multiple digital interfaces to interact with external equipment/systems (contactors, LEDs, panel switches, charger/inverters, battery controllers, etc). All interfaces provide some level of customization to allow Nuvation BMS™ to integrate well with any system.

Stack Control

The stack control enables an external system to take the stack online and offline, as well as provides pre-charge contactor capabilities and time sequencing.

stack_control

requested_state Setting this register requests the connection or disconnection of the stack.

connect_delay The delay in microseconds before the stack engages under safe conditions.

disconnect_delay The delay in microseconds before the stack disengages.

precharge_delay The delay in microseconds before pre-charging terminates.

precharge_max_current The maximum current below which pre-charge can be terminated.

Contactor Control

There are 4 contactor outputs available on the Power Interface. The following component configures the behavior of each contactor output.

stack_contactor

address The hexadecimal register address of a boolean flag which should be used to control the contactor.

enabled A boolean flag which allows the value at the address to control the contactor.

inverted A boolean flag which inverts the contactor control.

General Purpose Outputs

There are 4 GPOs available on the Stack Controller. The following component configures the behavior of each GPO.

sc_gpo

address The hexadecimal register address of a boolean flag which should be used to control the GPO.

enabled A boolean flag which allows the value at the address to control the GPO.

inverted A boolean flag which inverts the GPO control.

General Purpose Inputs

There are 4 GPIs available on the Stack Controller. The following component configures the behavior of each GPI.

sc_gpi

enabled A boolean flag which enables the GPI value to be populated into the value field.

inverted A boolean flag which inverts the GPI value.

Isolated CAN

The Stack Controller provides an isolated CAN-BUS interface for interacting with external systems. This interface is customizable to integrate well with a majority of common external systems. 64 registers can be selectively reported and more can be added at the end via the bulk report feature. The following components configure the behavior of the CAN interface.

sc_canbus

enabled A boolean flag which enables the CAN-BUS interface.

errratewindow The microsecond window to average communication errors over when calculating the error rate.

basecanaddress The hexadecimal address to start all CAN message addresses from.

reportintervalus The microsecond period to repeat CAN message broadcasts.

sc_canbus_rpt

address The hexadecimal register address of a value to report over CAN-BUS.

sc_canbus_bulkrpt

baseaddress The hexadecimal register address to start bulk reading from.

baseenableaddress The hexadecimal register address to check if the reading should be reported.

offset The hexadecimal offset to increase the reading and enable addresses between each read.

numtoread The hexadecimal number of registers to bulk read.

RS-485 Modbus RTU

sc_modbus_rtu

device_address The address of the modbus RTU slave device corresponding to this SC.

Safety

The thresholds for cell voltage, cell temperature, stack voltage and stack current for fault and warning conditions define an overall safe state of the battery system. This configuration approach allows a high degree of customization to easily integrate Nuvation BMS™ into any battery system. It is very important to properly define the overall safe state if the system has been configured to look at the safe status. Moreover, Nuvation BMS™ allows customization of acceptable error rates so that the system can be as lenient to communication errors as desired.

Communication Safety

Each communication bus has configurable acceptable error rates. The following components configure the acceptable rates.

sc_linkbus

errwindow The microsecond duration to average linkbus communication errors over.

wdtperiod The microsecond duration of acceptable zero successful linkbus communication.

sc_fault_linkbus_rx

thresh The percentage threshold that is tripped when linkbus rx error rate rises above this value.

time_hyst The microsecond duration that linkbus rx error rate must remain above the threshold before it trips.

end_time_hyst The microsecond duration that linkbus rx error rate must remain below the threshold before it resets.

sc_fault_linkbus_wdt

thresh The boolean flag that is tripped when linkbus wdt equals this value.

time_hyst The microsecond duration that linkbus wdt must remain at the threshold before it trips.

end_time_hyst The microsecond duration that linkbus wdt must remain below the threshold before it resets.

sc_canbus

errwindow The microsecond duration to average CAN-BUS communication errors over.

sc_stackbus

errratewindow The microsecond duration to average stackbus communication errors over.

rxwdtperiod The microsecond duration of acceptable zero successful received stackbus communication.

txwdtperiod The microsecond duration of acceptable zero successful transmitted stackbus communication.

sc_fault_stackbus_rxwdt

thresh The boolean flag that is tripped when stackbus rxwdt equals this value.

time_hyst The microsecond duration that stackbus rxwdt must remain at the threshold before it trips.

end_time_hyst The microsecond duration that stackbus rxwdt must remain below the threshold before it resets.

sc_fault_stackbus_txwdt

thresh The boolean flag that is tripped when stackbus txwdt equals this value.

time_hyst The microsecond duration that stackbus txwdt must remain at the threshold before it trips.

end_time_hyst The microsecond duration that stackbus txwdt must remain below the threshold before it resets.

pi_afe

wdt_period The microsecond duration of acceptable zero successful AFE communication.

sc_fault_pi_afe_rx

thresh The percentage threshold that is tripped when pi_afe rx error rate rises above this value.

time_hyst The microsecond duration that pi_afe rx error rate must remain above the threshold before it trips.

end_time_hyst The microsecond duration that pi_afe rx error rate must remain below the threshold before it resets.

sc_fault_pi_afe_wdt

thresh The boolean flag that is tripped when pi_afe wdt equals this value.

time_hyst The microsecond duration that pi_afe wdt must remain at the threshold before it trips.

end_time_hyst The microsecond duration that pi_afe wdt must remain below the threshold before it resets.

Calibration

The Cell Interface, Stack Controller and Power Interface offer a level of calibration which allows the system to be tuned to achieve the desired measurement accuracy. The preset values that ship with the BMS from Nuvation can be adjusted also.

Cell Interface

The CIs can be configured to use any thermistor. The following component sets up the transfer function between the measured voltage and the reported temperature.

stack_therm_poly

This component contains the votlage to temperature conversion function:

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

where the coefficients are defined in the following registers:

coeff0 The float value of the 0th order coefficient.

coeff1 The float value of the 1st order coefficient.

coeff2 The float value of the 2nd order coefficient.

coeff3 The float value of the 3rd order coefficient.

coeff4 The float value of the 4th order coefficient.

coeff5 The float value of the 5th order coefficient.

coeff6 The float value of the 6th order coefficient.

Stack Controller

The SC can be configured to use any battery stack capacity by setting the following component.

stack_soc

nominal_capacity The milliampere-hour capacity of the battery stack.

Power Interface

The PI can work with a wide range of current shunt resistances and stack voltages. The following components adjust the transfer functions.

pi_afe_iadc

This component contains the ADC to current conversion function:

$$I(currentadc) = MULTIPLIER(currentadc/DIVIDER)$$

where the multiplier and divider are defined in the following registers:

multiplier The integer to multiply currentadc by to convert it into a current value.

divider The integer to divide currentadc by to convert it into a current value.

fsample The Hertz value of currentadc sampling frequency.

pi_afe_vadc

This component contains the ADC to voltage conversion function:

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

where the multiplier and divider are defined in the following registers:

multiplier The integer to multiply voltageadc by to convert it into a stack voltage value.

divider The integer to divide voltageadc by to convert it into a stack voltage value.

Stored Configuration

The Stack Controller and Power Interface can store configuration values between power cycles. Nuvation BMS™ comes with a stored preset configuration to use as a starting point.

Loading Configuration

The configuration is automatically loaded after a power cycle, but sometimes a power cycle is not desired to restore the configuration. The following components have settings to load.

sc_persist

load A boolean flag to load persistent parameters.

error A read-only boolean flag which indicates if the load operation was unsuccessful.

pi_persist

load A boolean flag to load persistent parameters.

error A read-only boolean flag which indicates if the load operation was unsuccessful.

Saving Configuration

The configuration file needs to indicate to Nuvation BMS™ that the settings are to be persisted. Also, a save can be triggered manually to save any settings tweaked during integration. The following components have settings to save.

sc_persist

save A boolean flag to save persistent parameters.

error A read-only boolean flag which indicates if the save operation was unsuccessful.

pi_persist

save A boolean flag to save persistent parameters.

error A read-only boolean flag which indicates if the save operation was unsuccessful.

Example Configuration

The following is an example of how to configure Nuvation BMS™ for a system composed of:

- 28 ICR18650-26A cells, connected in series to form the example battery stack
- 14 NTCALUG03A103H thermistors placed between cells
- 2xCI-16 with 14 cells each
- RSA-10-50 current shunt
- 2 contactors connected to coil1 and coil2
  - Coil1 connects to contactor between charger and battery stack
  - Coil2 connects to contactor between load and battery stack

Warning: This example is to be used as a reference only for understanding how to go about configuring Nuvation BMS™. It is by no means a feature-complete example. Nuvation is not responsible for how the BMS will behave if this example configuration is uploaded to the BMS.

Configuration Prerequisite Calculations

Before we jump into setting configuration values, there are a few things to calculate so that the values used will have the best chance of creating a well-behaving BMS without requiring multiple iterative tuning passes.

Thermistor Coefficients

The thermistor voltage is read by a 10k pull-up to 3.00V. The first step in calculating the coefficients is to create a table in excel or any other 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. Vadc is calculated by:

$$V_{adc} = 3.0 * (Resistance/(Resistance + 10000))$$

Using the line plot feature, create a graph of Vadc vs Temperature and turn on the trend line. Modify the trend line to be a 6th order polynomial type. Display the equation on the chart. The equation will look like:

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

We now have the coefficient numbers for the stack_therm_poly component.

Current Shunt

The current shunt voltage is read by the AFE chip on the Power Interface. The current shunt is a 5mOhm resistor, meaning it 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:

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

where current is 10000mA and Vadc is 50mV.

For our current shunt, the conversion factor is 0.000027939.

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. The multiplier and divider registers only accept whole positive numbers. 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. Experimental calibration is always recommended if possible.

We now have the multiplier and divider for the pi_afe_iadc component.

Stack Voltage

The stack voltage is read by the AFE chip on the Power Interface. The conversion factor is PI-type specific as each model of the PI contains a different divider circuit. The multiplier and divider work very similar to current shunt multiplier and divider. The starting values, depending on PI model type, is shown in the following table:

PI-Model	multiplier	divider
A	361	25000
B	334	10000
C	923	10000
D	181	1000

For our PI model type, the multiplier is 361 and divider is 25000.

We now have the multiplier and divider for the pi_afe_vadc component.

Capacity

The battery stack capacity is required for the SOC algorithm. To calculate the capacity of this battery stack, we first realize that the milliampere-hour capacity of a single cell is the same milliampere-hour capacity of the overall battery stack. The nominal capacity is provided in the datasheet and is 2600mAh. We now have the battery stack capacity for the stack_soc component.

Creating the Configuration File

Now that we have gone through the prerequisite math, we can now write an initial configuration file. To ease digestion, the configuration will be presented in tabular form to allow comments to be made on each item.

Table 3.1: Sample Register Configurations

Register	Value	Description
Cell Interfaces		
sc_linkbus.cicount	2	There are 2 CIs in the system
ci[0:1].softwareid	1	We are using CI-16s
Cells		
cell[0:13:16:2].installed	1	14 cells are installed on each CI
Thermistors		
therm[0:6:8:2].installed	1	7 thermistors are installed on each CI
Thresholds		
stack_fault_cell_hi.thresh	4200	From datasheet
stack_fault_cell_hi.time_hyst	1000000	1 second delay
stack_fault_cell_hi.end_time_hyst	1000000	1 second delay
stack_warn_cell_hi.thresh	4175	25mV lower than high fault thresh
stack_warn_cell_hi.time_hyst	0	Instant trip
stack_warn_cell_hi.end_time_hyst	10000000	1 second delay
stack_warn_cell_lo.thresh	2800	50mV higher than low fault thresh
stack_warn_cell_lo.time_hyst	0	Instant trip
stack_warn_cell_lo.end_time_hyst	10000000	1 second delay
stack_fault_cell_lo.thresh	2750	From datasheet
stack_fault_cell_lo.time_hyst	1000000	1 second delay
stack_fault_cell_lo.end_time_hyst	1000000	1 second delay
stack_cell_balancer.minenablevoltage	3950	250mV lower than high fault thresh
stack_current_limit.voltage_cell_max	4150	25mV lower than high warn thresh
stack_current_limit.voltage_cell_high	3950	Same as balancing enable thresh

Continued on next page

Table 3.1 – continued from previous page

Register	Value	Description
stack_current_limit.voltage_cell_low	3000	200mV higher than low warn thresh
stack_current_limit.voltage_cell_min	2850	50mV higher than low warn thresh
stack_soc.vfull	4140	10mV lower than current limit cell max
stack_soc.vempty	2860	10mv higher than current limit cell min
stack_fault_charge_therm_hi.thresh	45	From datasheet
stack_fault_charge_therm_hi.time_hyst	5000000	5 second delay
stack_fault_charge_therm_hi.end_time_hyst	5000000	5 second delay
stack_fault_discharge_therm_hi.thresh	60	From datasheet
stack_fault_discharge_therm_hi.time_hyst	5000000	5 second delay
stack_fault_discharge_therm_hi.end_time_hyst	5000000	5 second delay
stack_trig_charge_therm_hi.thresh	43	2 degrees below high fault thresh
stack_trig_charge_therm_hi.time_hyst	0	Instant trip
stack_trig_charge_therm_hi.end_time_hyst	5000000	5 second delay
stack_trig_discharge_therm_hi.thresh	58	2 degrees below high fault thresh
stack_trig_discharge_therm_hi.time_hyst	0	Instant trip
stack_trig_discharge_therm_hi.end_time_hyst	5000000	5 second delay
stack_warn_charge_therm_hi.thresh	41	2 degrees below high trig thresh
stack_warn_charge_therm_hi.time_hyst	0	Instant trip
stack_warn_charge_therm_hi.end_time_hyst	5000000	5 second delay
stack_warn_discharge_therm_hi.thresh	56	2 degrees below high trig thresh
stack_warn_discharge_therm_hi.time_hyst	0	Instant trip
stack_warn_discharge_therm_hi.end_time_hyst	5000000	5 second delay
stack_warn_charge_therm_lo.thresh	5	2 degrees above low fault thresh
stack_warn_charge_therm_lo.time_hyst	0	Instant trip
stack_warn_charge_therm_lo.end_time_hyst	5000000	5 second delay
stack_warn_discharge_therm_lo.thresh	-8	2 degrees above low fault thresh

Continued on next page

Table 3.1 – continued from previous page

Register	Value	Description
stack_warn_discharge_therm_lo.time_hyst	0	Instant trip
stack_warn_discharge_therm_lo.end_time_hyst	5000000	5 second delay
stack_fault_charge_therm_lo.thresh	0	From datasheet
stack_fault_charge_therm_lo.time_hyst	5000000	5 second delay
stack_fault_charge_therm_lo.end_time_hyst	5000000	5 second delay
stack_fault_discharge_therm_lo.thresh	-20	From datasheet
stack_fault_discharge_therm_lo.time_hyst	5000000	5 second delay
stack_fault_discharge_therm_lo.end_time_hyst	5000000	5 second delay
stack_cell_balancer.maxenabletemperature	38	2 degrees below max charge temperature
stack_current_limit.tmax	40	From datasheet
stack_current_limit.thigh	36	2 degrees below max balance temperature
stack_current_limit.tlow	2	2 degrees above min charge temperature
stack_current_limit.tmin	0	From datasheet
stack_fault_voltage_hi.thresh	117600	28x high fault thresh
stack_fault_voltage_hi.time_hyst	1000000	1 second delay
stack_fault_voltage_hi.end_time_hyst	1000000	1 second delay
stack_warn_voltage_hi.thresh	116900	28x high warn thresh
stack_warn_voltage_hi.time_hyst	0	Instant trip
stack_warn_voltage_hi.end_time_hyst	1000000	1 second delay
stack_warn_voltage_lo.thresh	78400	28x low warn thresh
stack_warn_voltage_lo.time_hyst	0	Instant trip
stack_warn_voltage_lo.end_time_hyst	1000000	1 second delay
stack_fault_voltage_lo.thresh	77000	28x low fault thresh
stack_fault_voltage_lo.time_hyst	1000000	1 second delay
stack_fault_voltage_lo.end_time_hyst	1000000	1 second delay
stack_current_limit.vstackmax	116200	28x max cell voltage current limit
stack_current_limit.vstackhigh	110600	28x high cell voltage current limit

Continued on next page

Table 3.1 – continued from previous page

Register	Value	Description
stack_current_limit.vstacklow	84000	28x low cell voltage current limit
stack_current_limit.vstackmin	79800	28x min cell voltage current limit
stack_fault_current_hi.thresh	2600	From datasheet
stack_fault_current_hi.time_hyst	1000000	1 second delay
stack_fault_current_hi.end_time_hyst	1000000	1 second delay
stack_warn_current_hi.thresh	1300	From datasheet
stack_warn_current_hi.time_hyst	0	Instant trip
stack_warn_current_hi.end_time_hyst	1000000	1 second delay
stack_warn_current_lo.thresh	-2600	From datasheet
stack_warn_current_lo.time_hyst	0	Instant trip
stack_warn_current_lo.end_time_hyst	1000000	1 second delay
stack_fault_current_lo.thresh	-5200	From datasheet
stack_fault_current_lo.time_hyst	1000000	1 second delay
stack_fault_current_lo.end_time_hyst	1000000	1 second delay
stack_cell_balancer.maxenablecurrent	-100	From datasheet
stack_cell_balancer.minenablecurrent	-2600	From datasheet
stack_current_limit.maxchargecurrent	2600	From datasheet
stack_current_limit.maxdischargecurrent	-5200	From datasheet
stack_current_limit.lpfgain	10000000	10 second low-pass filter
Passive Balancing		
stack_cell_balancer_stat.measurewhilebalancing	0	Only want valid voltage readings
stack_cell_balancer.enabled	1	Enable cell balancing
stack_cell_balancer.voltagedelta	5	Allow 5mV delta between max and min cell
Digital I/O		
stack_contactor[0].address	@stack_safety[0].safetocharge	Use safetocharge boolean flag
stack_contactor[1].address	@stack_safety[0].safetodischarge	Use safetodischarge boolean flag
stack_contactor[0:1].enabled	1	Enable coil1 and coil2

Continued on next page

Table 3.1 – continued from previous page

Register	Value	Description
stack_contactor[0:1].inverted	0	Apply boolean flags as-is
BMS Safety		
sc_linkbus.actualscanrate	500000	2Hz update rate
sc_linkbus.errwindow	2500000	5x actual scan rate
sc_linkbus.wdtperiod	5000000	10x actual scan rate
sc_fault_linkbus_rx.thresh	15	Allow 15% linkbus errors
sc_fault_linkbus_rx.time_hyst	1000000	1 second delay
sc_fault_linkbus_rx.end_time_hyst	0	Instant reset
sc_fault_linkbus_wdt.thresh	1	Trip when Linkbus WDT expires
sc_fault_linkbus_wdt.time_hyst	0	Instant trip
sc_fault_linkbus.wdt.end_time_hyst	500000	Delay for 1 scan
sc_stackbus.errratewindow	1000000	1 second averaging window
sc_stackbus.rxwdtperiod	500000	1 scan rate window
sc_stackbus.txwdtperiod	500000	1 scan rate window
sc_fault_stackbus_rxwdt.thresh	1	Trip when Stackbus rxwdt expires
sc_fault_stackbus_rxwdt.time_hyst	0	Instant trip
sc_fault_stackbus.rxwdt.end_time_hyst	500000	Delay for 1 scan
sc_fault_stackbus_txwdt.thresh	1	Trip when Stackbus txwdt expires
sc_fault_stackbus_txwdt.time_hyst	0	Instant trip
sc_fault_stackbus.txwdt.end_time_hyst	500000	Delay for 1 scan
pi_afe.rx_err_window_us	1000000	1 second averaging window
pi_afe.wdt_period_us	500000	1 scan rate window
sc_fault_pi_afe_rx.thresh	15	Allow 15% pi-afe errors
sc_fault_pi_afe_rx.time_hyst	1000000	1 second delay
sc_fault_pi_afe_rx.end_time_hyst	0	Instant reset
sc_fault_pi_afe_wdt.thresh	1	Trip when pi-afe wdt expires

Continued on next page

Table 3.1 – continued from previous page

Register	Value	Description
sc_fault_pi_afe_wdt.time_hyst	0	Instant trip
sc_fault_pi_afe_wdt.end_time_hyst	500000	Delay for 1 scan
pi_gfd.enabled	0	Disable ground-fault detection
Calibration		
stack_therm_poly.coeff0	151.68	Calculated value
stack_therm_poly.coeff1	-352.94	Calculated value
stack_therm_poly.coeff2	549.33	Calculated value
stack_therm_poly.coeff3	-482.08	Calculated value
stack_therm_poly.coeff4	223.69	Calculated value
stack_therm_poly.coeff5	-51.518	Calculated value
stack_therm_poly.coeff6	4.5693	Calculated value
stack_soc.nominal_capacity	2600	Calculated value
pi_afe_iadc.multiplier	31	Calculated value
pi_afe_iadc.divider	1110000	Calculated value
pi_afe_iadc.fsample	800	800Hz sample period
pi_afe_vadc.multiplier	361	PI-A Multiplier value
pi_afe_vadc.divider	25000	PI-A Divider value
Stored Configuration		
sc_persist.save	1	Save these settings
pi_persist.save	1	Save these settings

Tuning the system

After uploading the configuration file to Nuvation BMS™, tuning is likely required to accurately calibrate the system and adjust the timing/behavior of faults. Leaving sc_persist.save and pi_persist.save as 0 in the configuration file allows temporary configurations to be uploaded. That way if something goes awry, a simple power cycle of the BMS will restore the previously saved configuration.

Isolated CAN Support

Nuvation BMS™ implements a flexible and customizable CAN protocol which maps BMS registers to CAN messages. The external CAN protocol supports both individual and bulk reporting capabilities. The parameters for CAN are:

- Baud: 500 kbit/s
- CAN ID: Extended
- CAN payload length: 8-byte
- Maximum individual messages: 64
- Maximum bulk messages: 4

Standard Reports

Nuvation BMS™ is design to work with a standard set of reported messages, as well as report intervals suitable for most systems. Below is a table of the standard registers:

Table 4.1: Standard Register Reports

Message	Register	Unit
Clock	@sc_clock.seconds	Seconds
Stack Voltage	@stack_power.voltage	mV
Stack Current	@stack_power.current	mA
State of Charge	@stack_soc.soc	%
Depth of Discharge	@stack_soc.dod	mAhr
Maximum Cell Voltage	@stack_cell_stat.max	mV
Minimum Cell Voltage	@stack_cell_stat.min	mV
Average Cell Voltage	@stack_cell_stat.avg	mV
Maximum Temperature	@stack_therm_stat.max	C
Minimum Temperature	@stack_therm_stat.min	C
Average Temperature	@stack_therm_stat.avg	C
Overall Safe	@stack_safety.safe	Boolean
Safe to Charge	@stack_safety.safetocharge	Boolean
Safe to Discharge	@stack_safety.safetodischarge	Boolean
Charge Current Limit	@stack_current_limit.charge_current_limit	mA
Charge Percent Limit	@stack_current_limit.charge_current_percent	%
Discharge Current Limit	@stack_current_limit.discharge_current_limit	mA
Discharge Percent Limit	@stack_current_limit.discharge_current_percent	%
Stack Control Connection State	@stack_control.connection_state	Enumeration

These reports are ordered and will have CAN IDs offset from the configured ID found in "*sc_canbus.basecanaddress*" register. For example, using "*sc_canbus.basecanaddress = 0x1000*":

- 0x1000: Clock
- 0x1001: Stack Voltage
- etc...
- 0x1012: Stack Control Connection State

The recommended report interval for data is 500ms. Any further configuration and customization can be done with the components found in the "Isolated CAN" section.

MESA Modbus Support

Nuvation BMS™ implements the SunSpec battery models defined in the Modular Energy Storage Architecture (MESA) as the top-level Modbus interface to the product. These specifications are available for download at http://mesastandards.org/mesa-downloads/. Currently, the draft 3 version is implemented.

A good introduction to system-level MESA concepts can be found here: http://mesastandards.org/downloads/MESA-PCS-Specification_D2.pdf.

Modbus Protocol Support

Nuvation BMS™ supports the following Modbus protocols.

Modbus RTU

This protocol is used in serial communications. The Stack Controller or Grid Battery Controller support connecting over the RS485 port. The default configuration is as follows:

- Baudrate: 38400
- Parity: even
- Data bits: 8
- Stop bits: 1

The device address must be configured via the modbus RTU component. For the Stack Controller the component is "*sc_modbus_rtu*". For the Grid Battery Controller the component is "*gbc_modbus_rtu*".

Modbus TCP

This protocol is used for communications over TCP/IP networks. The Stack Controller and Grid Battery Controller support connecting over port 502.

Supported Models

The MESA standards contain a number of 'models' that can be implemented by vendors to describe a storage device at various levels of detail. The models implemented by Nuvation BMS™ are described below.

Detailed register maps are provided by the standard in an Excel spreadsheet format here: http://mesastandards.org/wp-content/uploads/2015/10/Energy-Storage-Information-Models_D3-2015-10-26-Update.xlsx.

Common Model

This model primarily contains information to identify the device (e.g. manufacturer, model, serial number) as well as the version of software running on the device. A full description of the Common Model can be found in the SunSpec specification bundle.

S801

This model describes an energy storage device at the highest possible level. State of charge and overall alarm and warning states are found here. All mandatory points are implemented. The Modbus address of this model is 40070.

S802

This model describes a battery storage device. Critical operational information included at this level are charge and discharge current limits. All mandatory points are implemented. The Modbus address of this model is 40094.

S803

This model describes a lithium-ion battery in detail. Voltage, temperature, and current statistics are available at the pack and stack level within this model. All mandatory and most optional points are implemented. The Modbus address of this model is 40116.

End Model

This model marks the end of the implemented Modbus address space.

Accessing MESA Models

MESA models are located contiguously in the Modbus address space starting at a base address of 40000. The Common Model is always located first in this space. The End Model is always last in order and is used to denote the end of MESA Modbus registers. Each model located between these two has a numeric

identifier as well as a length. A handy tool that can be used to explore the MESA Modbus registers for Nuvation BMS™ is *modpoll.exe*. It is available for free download at http://www.modbusdriver.com/modpoll.html.

Using the tool, the Common Model can be polled from a Stack Controller or Grid Battery Controller using the following command (assuming the device has an IP address of 192.168.1.21)

```
modpoll.exe -m tcp -0 -r 40000 -c 70 192.168.1.21

modpoll 3.4 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2002-2013 proconX Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: MODBUS/TCP
Slave configuration...: address = 1, start reference = 40000 (PDU), count = 70
Communication.........: 192.168.1.21, port 502, t/o 1.00 s, poll rate 1000 ms
Data type.............: 16-bit register, output (holding) register table

-- Polling slave... (Ctrl-C to stop)
[40000]: 21365
[40001]: 28243
[40002]: 1
[40003]: 66
.
.
.
[40068]: 4660
[40069]: -32768
```

As another example, the complete S802 model for a system with one stack could be polled using the following command:

```
modpoll.exe -m tcp -0 -r 40094 -c 22 192.168.1.21

modpoll 3.4 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2002-2013 proconX Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: MODBUS/TCP
Slave configuration...: address = 1, start reference = 40094 (PDU), count = 22
Communication.........: 192.168.1.21, port 502, t/o 1.00 s, poll rate 1000 ms
Data type.............: 16-bit register, output (holding) register table

-- Polling slave... (Ctrl-C to stop)
[40094]: 802
[40095]: 20
.
```

```
.
.
[40114]: -2
[40115]: -32768
```

To access the common model using Modbus RTU (assuming Nuvation BMS ™ is connected to serial port COM1 and its address is 0x1):

```
modpoll.exe -m rtu -0 -r 40000 -c 70 -b 38400 COM1

modpoll 3.4 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2002-2013 proconX Pty Ltd
Visit http://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: Modbus RTU
Slave configuration...: address = 1, start reference = 40000 (PDU), count = 70
Communication.........: COM1, 38400, 8, 1, even, t/o 1.00 s, poll rate 1000 ms
Data type.............: 16-bit register, output (holding) register table

-- Polling slave... (Ctrl-C to stop)
[40000]: 21365
[40001]: 28243
[40002]: 1
[40003]: 66
.
.
.
[40068]: 4660
[40069]: -32768
```

Notes:

NUVATION BMS™
NUVATION ENGINEERING

SILICON VALLEY HEADQUARTERS
151 GIBRALTAR CT
SUNNYVALE, CA 94089 USA
408.228.5580

WATERLOO DESIGN CENTER
332 MARSLAND DR., SUITE 200
WATERLOO, ON N2J 3Z1 CANADA
519.746.2304

nuvationbms.com
nuvationbms@nuvation.com