Technical Support
support@nuvationenergy.com

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

Software Reference Manual (Multi-Stack)

Nuvation BMS™ Grid Battery Controller

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

Table of Contents

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 (Multi-Stack): Nuvation BMS™ Grid Battery Controller* manual describes the software configuration settings used within Nuvation BMS™ Grid Battery Controller. The Grid Battery Controller is used in conjunction with multiple Stack Controllers or Battery Controllers to provide pack-level monitoring and aggregation for multi-stack battery packs.

This document is designed as a companion to the example configuration file provided by Nuvation Energy.

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.).

2.4. Stack States

Stacks in the pack are managed by the Grid Battery Controller through distinct states defined for each stack. These states are outlined in the table below.

Table 1. Stack States

State	Description
Uninstalled	A stack in not available for communication or functionality of any kind
Installed	A stack is physically installed in the system and available for communication
Enabled	A stack is included in both aggregation and control operations at the pack level
Ready	A stack is enabled, safe, and ready to be connected to the DC bus
Connected	A stack is ready and electrically connected to the DC bus

The above states are integrated into all aspects of pack level functionality and are referenced throughout the following sections.

3. Configuration Settings

This section highlights the most important settings required to achieve a working Grid Battery Controller configuration for your system.

To operate your Grid Battery Controller in a pack containing one or more Stack Controllers or

Battery Controllers, each Stack Controller or Battery Controller must meet the following requirements:

- It must be running the same firmware version as the Grid Battery Controller you are using.
- It must be configured for the battery stacks in use in the pack.
- It must be assigned a unique network address on the `ETH1` (Internal) Grid Battery Controller network.

Please refer to the *Operator Interface Guide* for instructions on upgrading, configuring, and assigning addresses at the stack level. Refer to the *Operator Interface Manual (Multi-Stack)* for instructions on upgrading firmware or loading configuration for a Grid Battery Controller.

The most important aspects of a Grid Battery Controller configuration file are discussed in the following sections.

3.1. Network Settings

The Ethernet IP addressing used by each stack in the pack must match the network configuration of the Grid Battery Controller.

The recommended network topology has each Stack Controller or Battery Controller connected to an Ethernet switch. This switch is then connected to the Grid Battery Controller on the internal Ethernet port (labeled `ETH1`).

The Grid Battery Controller `ETH1` interface has the following settings by default:

- IP: `192.168.4.10`
- Subnet: `255.255.255.0`
- Gateway: `192.168.4.1`

Each stack is assigned a static IP address on the Grid Battery Controller internal `192.168.4.xxx` subnet. The Grid Battery Controller Ethernet bridges must be configured to match this IP address assignment. There is one instance of the Ethernet bridge component for each stack in the pack.

`gbc_eth_bridge[n].ip_address`

- The IP address of stack `n`. This is set to the Stack Controller or Battery Controller IP address.

`gbc_eth_bridge[n].port`

- The port of the stack `n` request bridge. This should be set to `8080`.

`gbc_eth_bridge[n].publish_port`

- The port of the stack `n` publish bridge. This should be set to `8081`.

3.2. Pack Topology

Battery pack topology configuration is required to specify which stacks are connected in each bank as well as the number of banks within the pack. The following registers must be configured for each stack in the pack (the `pack_stack` component instances correspond in order to the `gbc_eth_bridge` component instances).

`pack_stack[n].installed`

- Flag indicating if stack `n` is installed or uninstalled. Set to `1` if the stack is present.

`pack_stack[n].enabled`

- Flag indicating if stack `n` is enabled. Set to `1` to actually use the stack.

`pack_stack[n].bank_index`

- The index of the bank where stack `n` is located. Set to `0` if only one bank is in use.

Stacks that are installed are considered physically present in the system. The Grid Battery Controller will attempt to connect to and communicate with the SCs for these stacks using the corresponding `gbc_eth_bridge` configuration.

A stack must be enabled before the Grid Battery Controller will aggregate that stack's data into pack-level view or connect the stack to the DC bus.

3.3. Control Settings

Nuvation BMS™ Grid Battery Controller includes a pack connection routine that intelligently manages connection and disconnection of all enabled stacks in a pack. The connection routine will guarantee no stack is connected if its connection will cause excessive in-rush current due to voltage imbalances. During the charging and discharging of the stacks, the routine will automatically connect stacks when appropriate. It is guaranteed that if one stack is ready that stack can be connected. Configuration of this routine is achieved through the following registers.

`pack_control.enabled`

- Set to `1` to enable automatic control of stack connection
- Set to `0` to require manual control of stack connection

`pack_control.auto_connect`

- Setting this to `1` will connect on boot. It will also re-connect after a fault condition has cleared.
- Setting this to `0` will require operator triggered connection on boot. It will also require operator triggered connection after a fault condition has been cleared.

`pack_control.stack_delay`

- Set to the settling delay in microseconds. This prevents multiple stacks from connecting in rapid succession.

`pack_control.stack_nominal_resistance`

- The nominal total ohmic resistance of a stack in milliohms. It is assumed all stacks have equivalent resistance.

`pack_control.max_connect_current`

- The maximum allowable stack inrush current during connection in milliamps. Typically this should be set below the smallest absolute fault threshold for stack current.

`pack_control.connect_policy`

- The connection policy which dictates which stack will be the first to connect.
- Set to `0` for the 'maximum power' policy. This will first connect the ready stack with the median stack voltage. Typically this is used when it is unknown whether the pack will be charging or discharging in the near future.

- Set to `1` for the 'charge' policy. This will first connect the ready stack with the lowest stack voltage. Typically this is used when the pack will be charged in the near future.
- Set to `2` for the 'discharge' policy. This will first connect the ready stack with the highest stack voltage. Typically this is used when the pack will be discharged in the near future.

By default, this feature is disabled. When disabled, stacks must be connected and disconnected individually through the Operator Interface or via one of the supported Communication Protocols.

When the Grid Battery Controller pack connection routine is disabled, *NO* validation of stack voltages or SOC occurs before connecting stacks. Unless each stack is using a pre-charge circuit designed with this behavior in mind, use of the pack connection routine is recommended.

3.4. Operational Limits

The operational limits of the pack are primarily defined by the configuration at the stack level. The Grid Battery Controller aggregates the limits and signals from each stack to produce overall pack-level limits and signals automatically.

A number of additional triggers are present within the Grid Battery Controller itself that define fault thresholds at the pack level.

3.4.1. Stack Safety

Each enabled stack fault and warning states propagate up to the Grid Battery Controller through triggers. These triggers provide flexible error handling and allow stack level redundancy so a single stack fault won’t cause a pack fault and halt the entire system. The triggers which are the most important in handling pack safety are described below.

Table 2. Stack Safety Operational Limits

Register	Setting
`pack_fault_ready_stacks.thresh`	The minimum number of stacks in the ready state required for the pack to be considered safe. For no fault tolerance this should be set to (enabled stack count - 1). For full fault tolerance this should be set to 0.
`pack_warn_ready_stacks.thresh`	The minimum number of stacks in the ready state required for the pack to indicate a warning. Typically this is set to (enabled stack count - 1).
`pack_warn_stacks.thresh`	The count of enabled stacks in the warning state, above which the pack will indicate a warning. Typically this is set to 0, which results in any enabled stack warning indicating a pack warning.

With the above triggers any system can be configured to tolerate some stacks faulting. For example: a four stack system which can still operate with at least two stacks ready would set `pack_fault_ready_stacks.thresh = 1`. This indicates that a pack fault should stop operation if only one stack is ready. In another example: a four stack system which cannot operate unless all four stacks are ready would set `pack_fault_ready_stacks.thresh = 3`. This indicates that if only three stacks were ready (i.e. a single stack had faulted) it is considered a pack fault and should stop operating. In general the formula for this is: (minimum ready stacks - 1).

3.4.2. Communication Safety

Pack level communication must be stable and error free to ensure the battery is safely operating. Faults are tripped if any of the installed stack's Ethernet bridges experience connectivity issues. Fault behavior is tuned using the following trigger registers.

`gbc_fault_eth_conn.thresh`
- The threshold that is tripped when a bridge fault occurs.

`gbc_fault_eth_conn.time_hyst`
- The microsecond duration that the fault must be present before it trips.

`gbc_fault_eth_conn.end_time_hyst`
- The microsecond duration that the fault must be cleared before it resets.

3.4.3. Stack Watchdog

Nuvation BMS™ Grid Battery Controller will periodically update the watchdogs timers (`sc_controller_wdt`) of all stacks that are connected to the GBC. If an Stack Controller becomes disconnected from the GBC, this local Stack Controller watchdog can be configured to trip a fault at the stack level.

`gbc_sc_wdt_reset.period`
- The period at which all connected Stack Controller watchdogs will be reset.

`gbc_sc_wdt_reset.enable`
- Set this to 1 to enable the update of stack watchdogs by the GBC.

If this feature is enabled, then it assumes that the external controller heartbeat is not in use and it's fault should be disabled.

3.4.4. External Controller Heartbeat

Nuvation BMS™ Grid Battery Controller can be configured to require a heartbeat signal from an external controller in order to keep all stacks online and out of fault state. A write to the MESA controller heartbeat register is expected at least once during the watchdog period. The Grid Battery Controller will write to all enabled stacks watchdog registers (sc_controller_wdt) to keep them stack out of fault state.

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

`gbc_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 and the stack watchdog should be enabled.

3.5. Communication Protocols

Nuvation BMS™ Grid Battery Controller supports the following interfaces for connection with external systems:

- 10/100/1000 Ethernet for Modbus TCP and Operator Interface connectivity
- RS-485 for Modbus RTU

3.5.1. RS-485 Modbus RTU

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

`gbc_modbus_rtu.device_address`

- Set to the desired Modbus RTU slave device address

4. Troubleshooting

4.1. Faults and Warnings

When a pack level fault occurs, all stacks are disconnected. The following sections describe the different faults and the conditions that trigger them. 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. Ethernet Connection

This fault is related to communication faults between the Grid Battery Controller and any of the Stack Controller that is configured as installed `pack_stack[n].installed`.

`gbc_fault_eth_conn`

- A Grid Battery Controller could not comminicate with one or more installed Stack Controller(s).

This fault is usually triggered due to:

- GBC network configuration issues. Refer to Network Settings for details.
- IP networking configuration is different between the Stack Controller and Grid Battery Controller. Verify that `gbc_int_ethernet` register settings are compatible with the Stack Controller IP register settings.
- The expected Stack Controller is not on the network or receiving power.

To find the Stack Controller(s) that have a connection fault, refer to the register `gbc_eth_bridge_status[*].is_connected`. The pattern of data should match the installed stacks at `pack_stack[*].installed`.

4.1.2. Controller Heartbeat

`gbc_fault_controller_wdt`

- Fault indicating that an external controller did not provide a heart beat update within the configured period. Refer to external controller heartbeat on details of this configuration.

4.1.3. Ready Stacks

This fault represents a configurable minimum limit to how many stacks must be ready and still have the pack remain connected and manage the battery power. Refer to Stack Safety for details of this operation and configuration.

`pack_fault_ready_stacks`

- Fault indicating that the minimum number of ready stacks for the pack has been exceeded.

Refer the `Stack Status` page on the Operator Interface to observe which stacks have been either disabled or have faulted. Details about this screen are covered in the Operator Interface Manual (Multi-Stack).

4.1.4. Firmware Mismatch

`gbc_fault_fw_mismatch`

- Fault indicating that the difference between the Grid Battery Controller and the expected Stack Controller or Power Interface firmware versions. Perform a firmware upgrade to the affected stacks.

To determine which stacks have this particular fault, refer the component `gbc_fw_mismatch[].mismatch`.

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 Grid Battery 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