GOMSPACE

User Manual

NanoPower BP8

User manual for the NanoPower BP8 battery pack

MAN 1034929
November 5, 2025
Revision 4.1.0

User Manual NanoPower BP8
User manual for the NanoPower BP8 battery pack

Document reference: MAN 1034929
Source reference: doc-nanopower-bp8-user-manual
Date: November 5, 2025
Revision: 4.1.0

GomSpace A/S
Langagervej 6, 9220 Aalborg East
Denmark
Phone: +45 71 741 741
www.gomspace.com

Contents

List of Figures

List of Tables

List of Abbreviations

CAN Controller Area Network.

CSP Cubesat Space Protocol.

ESD Electrostatic Discharge.

GOSH GomSpace Shell.

I2C Inter-Integrated Circuit.

MCU Microcontroller Unit.

SOC State of Charge.

UART Universal Asynchronous Receiver/Transmitter.

1 Introduction

The NanoPower BP8 is a high-capacity lithium-ion battery pack with an integrated protection system, cell balancing, cell fault detection, heating system and telemetry sampling. This manual describes how to handle, prepare and utilize the features of the NanoPower BP8.
Please refer to the NanoPower BP8 datasheet [1] regarding electrical and mechanical characteristics.

1.1 Overview

1.1.1 BP8 components

The NanoPower BP8 (see Figure 1.1) arrives with three additional items, the BP8 ground breaker, an Electrostatic Discharge (ESD) cable and a GomSpace Shell (GOSH) cable, see Figure 1.2. These components are necessary for handling and installing the NanoPower BP8, see Section 1.2.

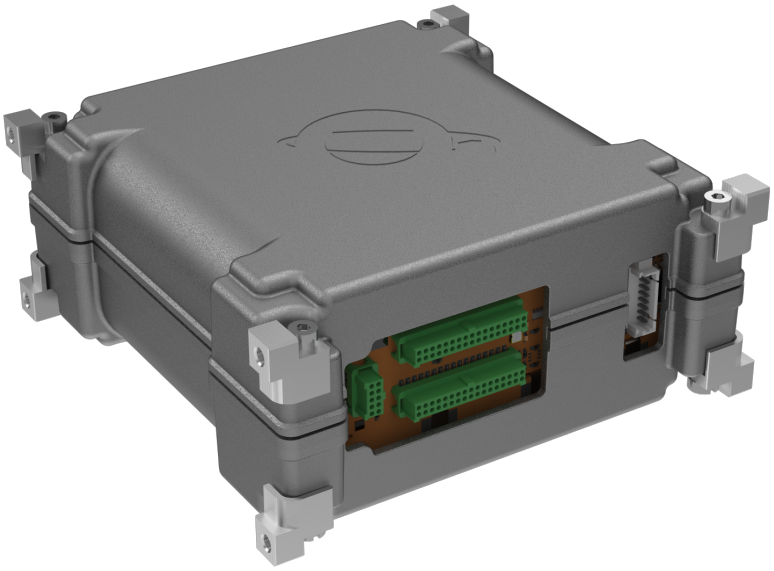
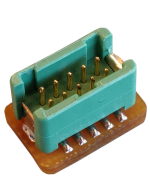
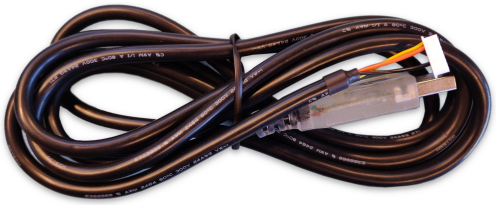


Figure 1.1: The NanoPower BP8



(a) Ground breaker



(b) GOSH cable

(c) ESD cable

Figure 1.2: The NanoPower BP8 accessories

1.1.2 Connector placement

The NanoPower BP8 has a total of four connectors; J1, J2, J3 and J5, see Figure 1.3 for the placement of the connectors and Table 1.1 for a description of each. Please refer to the NanoPower BP8 datasheet [1] for the pinout of each connector.

Designator	Description
J1	Main Power connector
J2	Secondary Power connector
J3	Ground breaker connector
J5	GOSH connector for configuration

Table 1.1: Overview of connectors

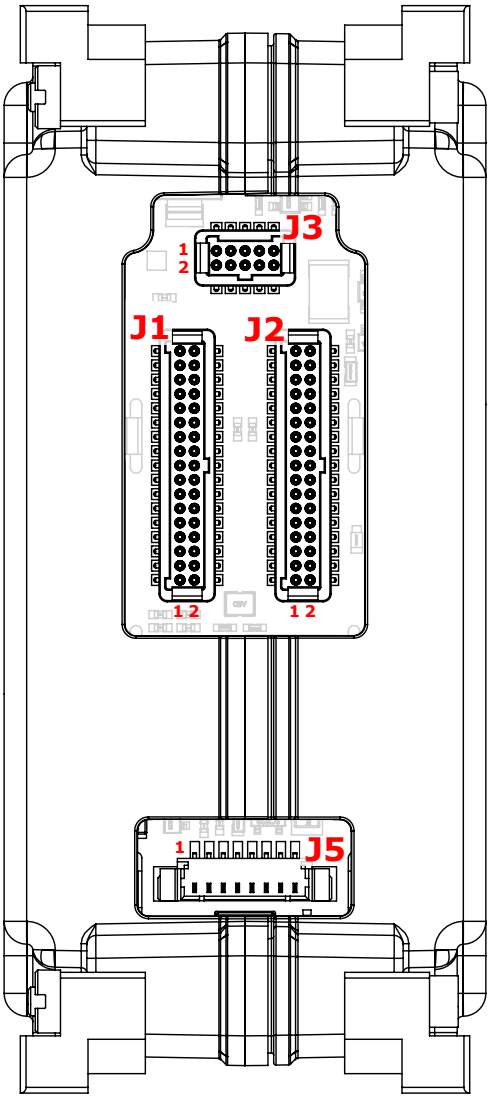


Figure 1.3: Placement of connector J1, J2, J3 and J5

1.2 Unpacking and handling

1.2.1 Handling, Grounding and Cleaning

The NanoPower BP8 system employs components based on Field Effect Transistors (FETs) and requires anti-static handling precautions. It is crucial that the NanoPower BP8 has the same ground potential as the system it connects to, before other connections are made. This can be achieved using the ESD cable supplied with the NanoPower BP8, see Figure 1.2c. The ESD cable can be removed only when the BP8 has connection to the system ground through either J1 or J2.

Please use an ESD mat and an ESD wrist strap as a minimum when handling the NanoPower BP8, and wear gloves to avoid fingerprints on the product. If any cleaning of the parts are required prior to flight, use ESD-safe cleaning methods and a neutral, non-reactive, isopropyl alcohol (IPA) solvent.

> ⚠ **CAUTION:** Do not touch or handle the product without proper ESD grounding!

1.2.2 Electrical integration

The NanoPower BP8 contains battery cells capable of delivering very high currents. Caution must therefore be taken to avoid short circuit when integrating the BP8. To limit the chance of faults when connecting the NanoPower BP8 to an electrical system the following conditions must be met:

- The ground breaker at J3 is disconnected
- The battery pack must be charged according to Section 1.2.3.
- The first pack in a string must be connected through J1 to the system.

> ⚠ **CAUTION:** Do not insert the ground breaker before the NanoPower BP8 is connected through J1.

1.2.3 Charging

Before using the NanoPower BP8, it is important to charge the device. Charging is done through connector J1 or J2, please refer to the datasheet for pinouts [1]. For BP8s with serial number 400 or higher, it is important to note that charging is only possible when the kill switch is released. The BP8 should be charged according to the specifications in Table 1.2.

Step	Method	Specification
Step 1	Constant current (CC)	charge at <= 4 A until 32 V per battery pack
Step 2	Constant voltage (CV)	charge at 32 V until 50 mA per battery pack

Table 1.2: Recommended charging cycle

> **NOTE:** Before connecting multiple BP8 in parallel, a powersupply or the P80 can be used to charge them individually; connecting one BP8 at the time and charging according to Table 1.2.

1.2.4 Storage conditions

The NanoPower BP8 will always have a small amount of self-discharge, and it is therefore important that actions are taken to prolong the battery life before storing it for a longer period. To prolong the BP8's battery life, the storage conditions given in Table 1.3 are recommended.

Parameter	Condition	Note
Temperature	between −20 °C and 20 °C	
Charge level	between 30 % and 60 %	Voltage of 28.8 V–30.8 V

Table 1.3: Recommended storage conditions

When the recommended storage conditions in Table 1.3 are respected, the battery pack can be stored according to Table 1.4.

Type	Time	Condition
Shelf storage	12 months	Ground breaker disconnected
Integrated storage	3 months	Ground breaker connected and kill switch enabled

Table 1.4: Total storage time before reconditioning of the BP8 is needed

NOTE: The performance of the battery cells will degrade over time, even when stored in optimal conditions. Further details on current consumption and cell self-discharge can be found in the NanoPower BP8 datasheet [1].

1.2.5 Mechanical integration

The NanoPower BP8 comes with 8 x M2.5 mounting holes, located on each corner of the battery pack, see Figure 1.4. The BP8 must be mounted in the satellite structure using M2.5 bolts with maximal thread engagement of 2.5 mm. It is advised not to unscrew the aluminum corner brackets and ensure careful handling to maintain the parallel orientation of brackets towards the satellite structure.

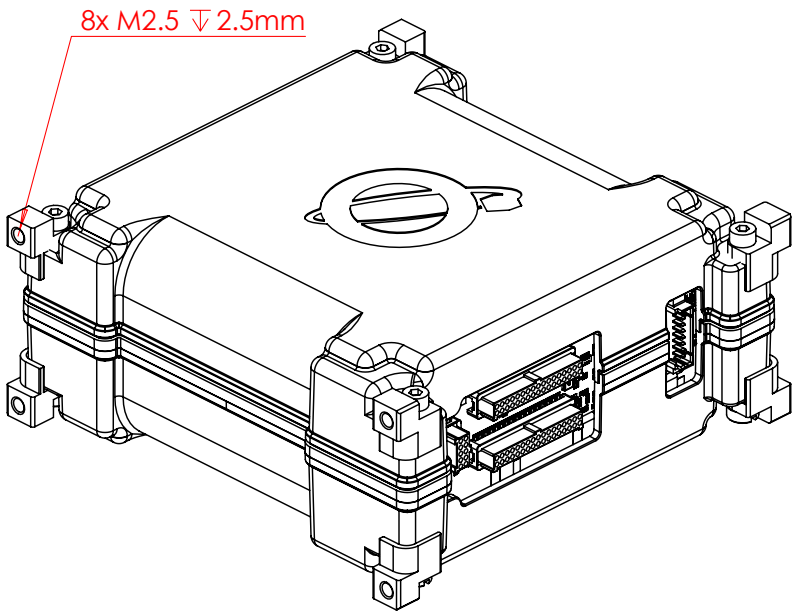


Figure 1.4: Mechanical interfaces on the NanoPower BP8

Remark that the BP8 corner brackets have screw lock Helicoil inserted, and therefore no thread lockers such as Loctite is needed. GomSpace recommends using stainless steel M2.5 screws of property class 70 for mounting the NanoPower BP8. As the locking capability change over the first few uses of a screw lock Helicoil, the following torque values are recommended, see Table 1.5.

Number #	Mouting torque
1	0.97 N m
2-5	0.75 N m
>5	0.72 N m

Table 1.5: Recommended mounting torque per usage for screw lock Helicoil

NOTE: Please note that the recommended torque values are valid for stainless steel class 70 screws.

1.3 Getting started

This section describes how to get started with the NanoPower BP8 using the GOSH interface, and without having a P80 available. The GOSH interface provides a text-based interface to configure and test the NanoPower BP8. Before going forward, please refer to Section 1.2 for how to handle the BP8.

1.3.1 Requirements

To get started with GOSH on the BP8, only a PC along with the items listed in Table 1.6 are required:

Manufactorer	Part number	Description	Included
GomSpace	101651	NanoPower BP8	✓
GomSpace	101647	Ground breaker	✓
GomSpace	109421	GOSH cable	✓
Harwin	G125-3043496L4	Female connector	-
Harwin	G125-MW10150M94	Pre-crimped wire	-

Table 1.6: Items needed to getting started with the NanoPower BP8.

The female connector and pre-crimped wire listed in Table 1.6 are used to create a jumper that enables the MCU, by connecting the ENA-pin in connector J1 to GND, please refer to the datasheet for pinout [1]. This provides power to the NanoPower BP8 MCU and enables communication over GOSH. This connector is not provided by GomSpace and must be realised by the customer.

1.3.2 Hardware setup

The GOSH interface is accessed through connector J5, using the GOSH cable provided by GomSpace.

NOTE: The MCU is powered internally and therefore no power is included in the GOSH cable.

To power up the MCU and enable the GOSH interface, the following steps are needed:

1. Connect the USB end of the GOSH cable to a PC and connect the Picoblade end in J5
2. Insert the jumper connector at J1 that connects the EN-pin to GND (see Section 1.3.1)
3. Insert the ground breaker at J3 (this connects the negative battery terminal to GND)

⚠ **CAUTION:** Ensure that any PC connected to the GOSH cable has a proper grounded AC plug. The external power supply ground and PC ground must be the same. If a laptop is used, it is recommended to disconnect the AC charger.

1.3.3 Software setup

To start communicating with the BP8, open any serial interface from the connected PC running 500000-8N1 (8 data bits, no parity, 1 stop). For Linux users the program `minicom` or `tio`, for Windows users the program PuTTY is recommended. From the terminal run the command `help` to get an overview of the available commands, please refer to Section 4 for a description of the NanoPower BP8 specific commands.

1.4 Integration procedure

The BP8 is designed to interface with a P80 power supply unit. When connecting one or more BP8s to a P80, it is important to ensure proper grounding and voltage potentials before the systems are connected. This is done by following the procedure below:

1. Connect GND in J5 (pin 3) between all BP8s and connect this to the EPS / system GND
2. Connect the main connector J1 from the BP8s to the EPS, followed by any daisy chained BP8s
3. When all BP8s are connected, disconnect the GND harness in J5
4. Use the P80 RBF or the killswitch (KS) to inhibit power from the BP8s
5. Insert the GND breakers in the BP8s one at a time
6. Remove the RBF (or disengage the KS) to power on the system

NOTE: Use the debug connector J5 on the BP8 to connect the grounds of the system, before connecting the main connector.

NOTE: Check that the voltage of all BP8s are within 200 mV of each other before connecting them in parallel.

2 Configuration and operation

This chapter describes how to use and configure the features of the NanoPower BP8 before flight, and how to operate it during flight.

2.1 Overview

The NanoPower BP8 has many features that run autonomously, while others need to be configured or controlled externally. Features that protect the cells/battery are built into the hardware and run autonomously to ensure the battery operates safely, even when the Microcontroller Unit (MCU) is disabled. The higher-level features such as State of Charge (SOC) estimation or heater control runs in software on the onboard MCU. The features are divided as follows:

Hardware specific

- Over- and undervoltage protection
- Battery/cell fault monitoring
- Cell balancing

Software specific

- Autonomous or manual battery heating
- Telemetry sampling
- SOC estimation
- Battery cut off (irreversible), for decommission purpose.

The features of the NanoPower BP8 are, like many other GomSpace products, configured and operated through the GomSpace parameter system. A description of the parameter system is found in Section 3, and an overview of the GOSH commands used to interface with it is found in Section 4.

2.2 Interfaces and protocols

The NanoPower BP8 has three physical communication interfaces: Inter-Integrated Circuit (I2C), Controller Area Network (CAN) and Universal Asynchronous Receiver/Transmitter (UART), see Table 2.1.

Interface	Max baud	Function	Comment
UART	500.000	Debugging and configuration	3.3V TTL, 8N1
CAN	1.000.000	Configuration and operation	ISO 11898-2 compliant External 120 ohm termination is needed
I2C	400.000	Configuration and operation	3.3V TTL, 7-bit address Multi-master mode

Table 2.1: BP8 Communication Interfaces

The main interfaces used to talk with the NanoPower BP8 are I2C and CAN, while UART should only be used for debugging and configuration. The communication setup is done through the board table, see Section 3.2.1.

NOTE: The UART interface should only be used for debugging and configuration. It is recommended to be the first interface used, see Section 1.3.

The Cubesat Space Protocol (CSP) is used to route data between the physical interfaces and the parameter system. The CSP layer supports CAN and I2C, see Figure 2.1 for an overview.

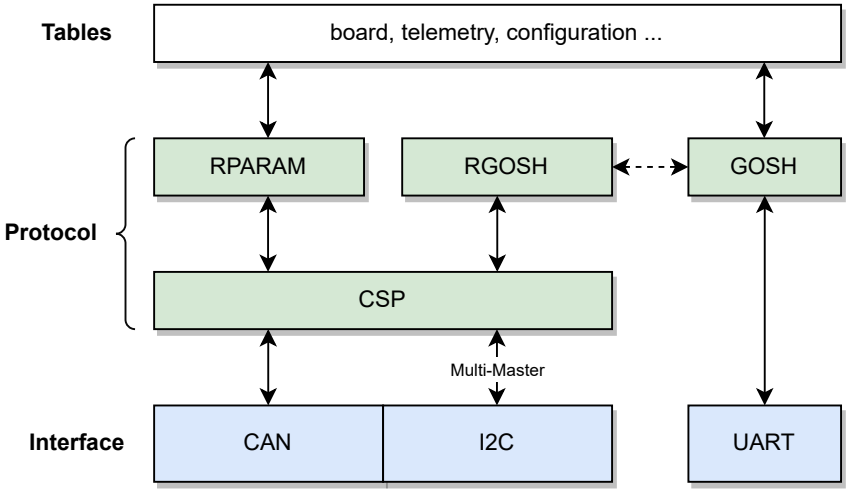


Figure 2.1: NanoPower BP8 communication stack

2.2.1 CSP, RGOSH and RPARAM

The CSP protocol is a small universal network layer delivery protocol similar to TCP/IP but more suitable for smaller networks with less overhead. Most of the GomSpace subsystems utilize this protocol. Please refer to [2] for further information.

In addition to CSP, there are two main protocols for controlling the NanoPower BP8; RGOSH and RPARAM. The RGOSH protocol can be used to execute every GOSH command remotely and is described in detail in [2]. The RPARAM protocol can be used to read, change, and save parameters remotely, and is described in detail in the [2].

For further information on the implementation of RGOSH and RPARAM over CSP, please refer to the NanoPower BP8 client source [3] and [2].

2.3 Multi-pack setup

To increase the total power capacity of a NanoPower system, multiple BP8s can be connected in parallel. The BP8 can be connected in parallel with another BP8, using the secondary power connector J2. In multi-pack setups, J2 of the first NanoPower BP8 must be connected to J1 of the second BP8 in the string, please see Figure 2.2.

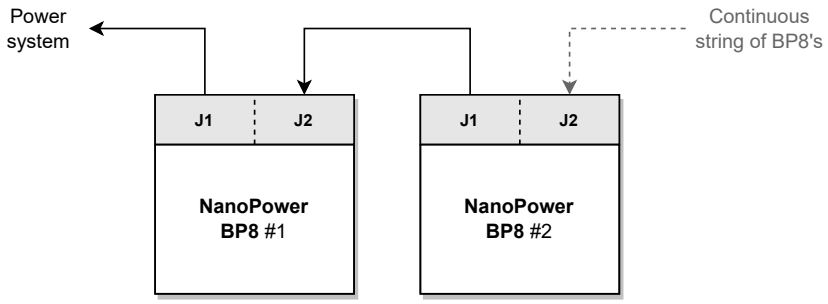


Figure 2.2: A string of NanoPower BP8's connected in parallel

Before connecting two BP8 battery packs in parallel, it is important to ensure that the voltage of both packs is the same. If the voltage difference between the packs is larger than 200 mV, each NanoPower BP8 should be charged to the same level before connecting, please refer to Section 1.2.3.

⚠ **CAUTION:** When connecting multiple NanoPower BP8's in parallel, each must be charged at the same voltage level such that large balancing currents are reduced at startup.

2.3.1 Enable loop

Only two BP8s can be controlled individually in a single string. The enable signals must be crossed to maintain the enable signal of two BP8s in parallel, as illustrated in Figure 2.3. This utilizes the EN_LOOP pin; please refer to the BP8 datasheet [1] for pinout.

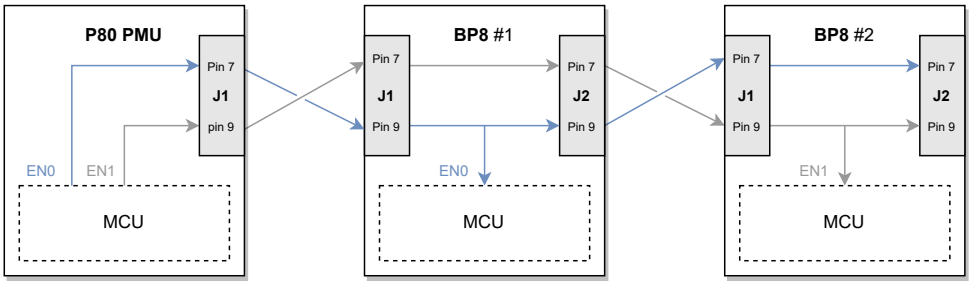


Figure 2.3: Enable signal routing with P80 and two BP8 units

NOTE: If more than two NanoPower BP8's are connected in a single string, these cannot be controlled individually by the enable signal.

2.4 Battery heater

The NanoPower BP8 comes with a battery heater and four temperature sensors to measure the battery's temperature. The heater can be activated in autonomous mode, adjusting the heater output based on the temperature measurements and battery voltage. Alternatively, the heater can be manually controlled to allow for an external controller to be implemented.

2.4.1 Autonomous mode

The autonomous mode is based on the hysteresis control principle. The purpose of the controller is to keep the batteries within a certain temperature range if the battery voltage is over a user-defined critical level (`vbat_critical`). If the battery voltage drops below this limit this is an indication of some system(s) improper operation in the satellite, and the heater will not turn on, until entering a state with definite more energy in the batteries. The heater controller block diagram is illustrated in Figure 2.4.

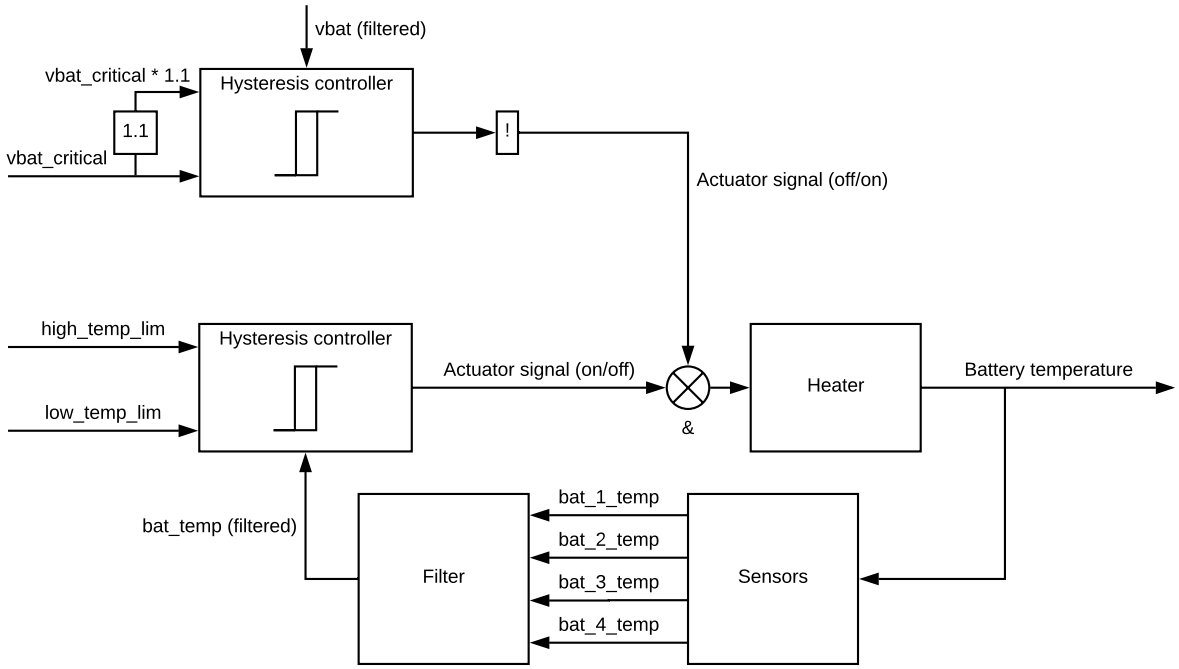


Figure 2.4: Block diagram of heating control system

There are four different temperature sensors which measure the temperature of the cells within the battery pack. These are lowpass filtered and combined into a single value `bat_temp`. The lowpass filters introduce a minor time delay in the measurements, however, this is compensated for in the controller. If one temperature measurement deviates by more than 10 °C compared to the rest, it is discarded before filtering, as it indicates an error in the measurement. The battery voltage is lowpass filtered as well.

⚠ **CAUTION:** The `vbatt_critical` level must be at least 10 % below that of `vbatt_max`, as the heater controller implements 10 % hysteresis.

NOTE: The four temperature measurements `bat_n_temp` ($n = 1, 2 \ldots 4$) can be affected by dissipated heat from the cell balancing circuit while charging.

An example of the controller system behavior is given in Figure 2.5.

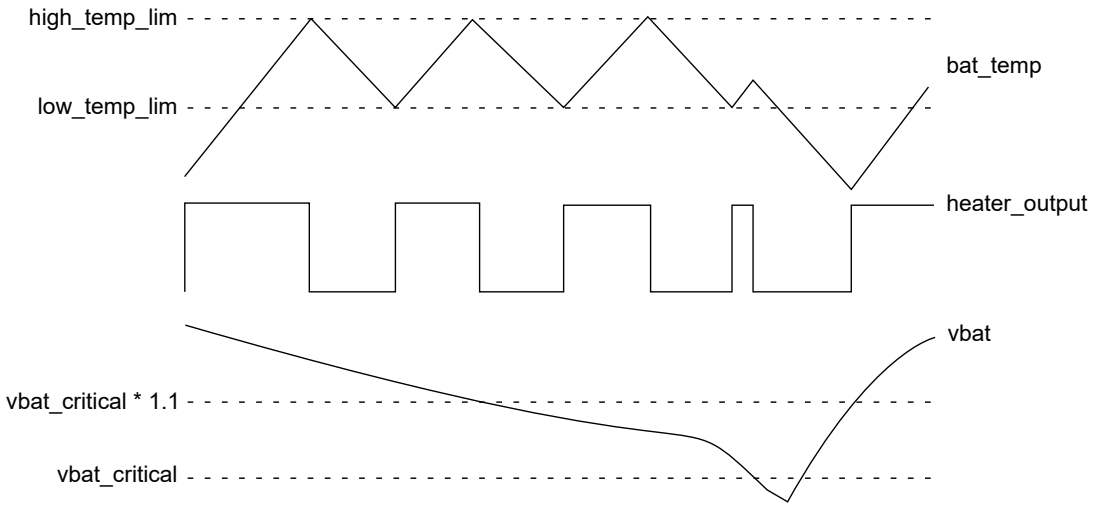


Figure 2.5: Example behavior of heating control system

Configuring the heater variables (`low_temp_lim`, `high_temp_lim` and `vbat_critical`) and enabling the autonomous mode are done through the configuration table described in Section 3.2.3. Note that the value of `vbat_critical` should not necessarily match the critical level defined in the NanoPower P80 but should be configured to meet the demands of the satellite system it integrates in.

NOTE: The battery voltage level is ignored by the controller if `vbat_critical` is set to 0.

The heater is turned off prior to launch on most GomSpace missions, as it is deemed more important to use all energy to bring the satellite into a known state (detumbled etc.). However, this compromise is obtained by factoring in different risks in the specific missions and may only apply to some missions

2.4.2 Manual mode

For testing and/or implementation of an external controller, the heater can be turned on manually. This is done by setting the manual heat timer (`heat_manual`) to a value different from 0. The heater will then be active for as many seconds as set, and the timer will countdown itself, giving an option to continuously read the remaining period. Resetting the timer to zero will stop the heater. The timer is found in the control table described in Section 3. It should be noted that the manual timer overrides the autonomous controller if it is enabled.

2.5 Battery health monitoring

The NanoPower BP8 consists of 8 lithium-ion cells in series. The battery pack balances the cells to the same voltage when charged to 32 V. The balancing is implemented with a cell discharge path on each cell. The discharge path opens/closes based on the cell voltage, thus controlling the current to each cell when charging.

At the end of life for a single cell, the voltage level will often drop. Due to small differences in the cells, it will happen at different times for the different cells in the battery pack. When one or more cells encounter(s) this sudden voltage drop, it degrades the whole battery pack, by lowering the total available voltage. However, if more battery packs are connected in parallel the other packs will compensate, and the difference in the voltage level will be less significant. This will, however, result in wasted energy going to the pack with the malfunctioning cell(s).

Therefore, to make such cell faults visible, a dedicated circuitry monitors the voltage of each cell, see Figure 2.6. If either of the cells drops below a critical level, the fault-checking circuitry raises a fault flag, which is saved in the parameter `bat_fault` in the telemetry table in Section 3.

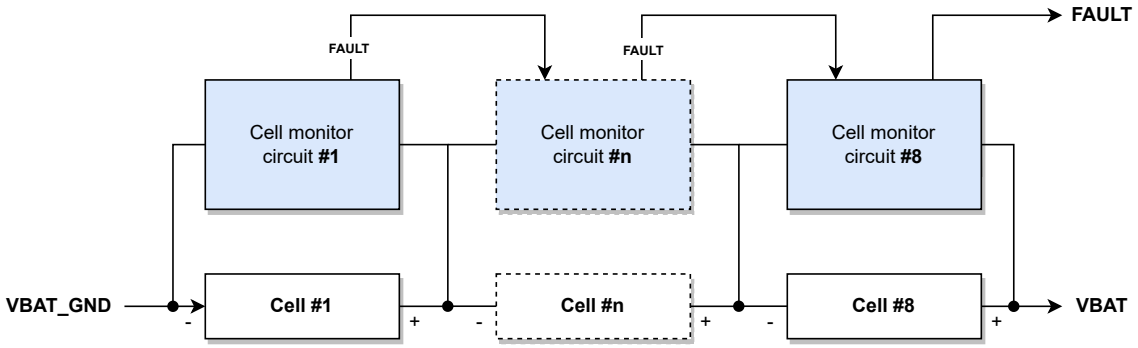


Figure 2.6: Cell setup

NOTE: The fault latches (in hardware), which means the voltage only needs to drop below the limit for one cell, for the fault signal to be present.

The fault signal can be reset by the parameter `fault_reset`. This way it can be checked if the fault results from a single event. Find the mentioned parameters in the table descriptions in Section 3.2.4. In the case of a permanent cell fault in a satellite with more packs in parallel, the operator can disconnect the battery pack, which has the faulty cell to prolong the life of the satellite. This operation results in capacity reduction of the satellite power system. For information about the disconnection procedure, please refer to Section Section 4.3.

3 Parameter System

The parameter system on the NanoPower BP8 controls all of the functionality. It consists of five different tables, each containing a number of variables. The tables are stored as different copies in different stores.

3.1 Stores

Each table is stored in different dynamic versions in different stores with different levels of accessibility. This is shown in Table 3.1. The parameter system will upon boot load each table from the leftmost available store (referring to Table 3.1). This is either the persistent, protected or the flash store. Each table is stored with a CRC value. If the CRC fails while loading, the parameter system will try to load the table from the next store. If all dynamic table versions get corrupted, then the table obtains non changeable default values, which will be the same for every NanoPower BP8.

Table	Persistent (writable FRAM)	Protected (write protected FRAM)	Flash
board	-	✓	✓
configuration	✓	✓	✓
calibration	-	✓	✓
control	-	-	-
telemetry	-	-	-

Table 3.1: The parameter tables availability in different stores

For instance, if the writable version of the configuration table, stored in the persistent table, gets corrupted it will fall back to the protected version. If also this one fails, it falls back to the version stored in the flash. If this is also corrupt, the last fall back is to the default version. Refer to Section 4 for information regarding changing parameters and saving to different stores.

⚠ **CAUTION:** The write protected tables should be changed only on ground.

⚠ **CAUTION:** The values of the parameters can be different in each store. Make sure to save configuration in each available store before flight.

NOTE: Some of the variables in telemetry table are also persistently stored, such as boot count, but as they change dynamically there is no fall back versions.

3.2 Parameter tables

The content of the different tables are described in the following subsections.

3.2.1 Board

This table holds all critical configuration of the NanoPower BP8. It should not be changed at flight.

Table 3.2: Parameter table 0: 'board'.

Name	Address	Type	Description
uid	0x0	string	Hardware UID string, do not change this **Max string length:** 16 **Default:**
type	0x10	uint8	Hardware type/variant, do not change this **Default:** 0
rev	0x11	uint8	Hardware revision, do not change this **Default:** 0
hostname	0x12	string	Hostname for CSP and GOSH prompt **Max string length:** 16 **Default:** BP8
addr	0x22	uint8	Address used for CSP and/or I2C **Default:** 16 **Valid range:** 1 <= addr <= 31
csp_rtable	0x23	string	CSP routing table **Max string length:** 96 **Default:** 0/0 CAN
can_brate	0x84	uint32	CAN interface bitrate **Unit:** bps **Default:** 1000000 **Valid range:** 125000 <= can_brate <= 1000000
i2c_brate	0x88	uint32	I2C interface bitrate **Unit:** bps **Default:** 400000 **Valid range:** 100000 <= i2c_brate <= 400000

Continued on next page

Table 3.2: Parameter table 0: 'board'. (Continued)

Name	Address	Type	Description
i2c_mode	0x8c	uint8	Protocol to use over I2C **Default:** 0 **Valid values:** '0': CSP '1': GSPP (legacy)
app_mode	0x8d	uint8	Mode of application **Default:** 0 **Valid values:** '0': integration '1': flight

The `app_mode` parameter must be changed before flight, to make the battery decommission (fuse burn) feature available, see Table 3.3.

Functionality	Integration mode (0)	Flight mode (1)
Fuse burn	no	yes

Table 3.3: Application modes

3.2.2 Calibration

This table holds the values used for calibration of the different sensors. It should not be changed.

> ⚠ **CAUTION:** The factory calibration will be lost by changing calibration values.

Table 3.4: Parameter table 2: 'calibration'.

Name	Address	Type	Description
in_i_gain	0x0	float	Gain on current in **Default:** 1.0
in_i_offs	0x4	int16	Offset current in **Unit:** mA **Default:** 0
heat_i_gain	0x8	float	Gain on heater current **Default:** 1.0

Continued on next page

Table 3.4: Parameter table 2: 'calibration'. (Continued)

Name	Address	Type	Description
heat_i_offs	0xc	int16	Offset heater current **Unit:** mA **Default:** 0
out_i_gain	0x10	float	Gain on current out **Default:** 1.0
out_i_offs	0x14	int16	Offset current out **Unit:** mA **Default:** 0
vbat_gain	0x18	float	Gain on battery voltage **Default:** 1.0
vbat_offs	0x1c	int16	Offset battery voltage **Unit:** ddegC **Default:** 0
battemp1_offs	0x1e	int16	Offset on battery temperature sensor 1 **Unit:** ddegC **Default:** 0
battemp2_offs	0x20	int16	Offset on battery temperature sensor 2 **Unit:** ddegC **Default:** 0
battemp3_offs	0x22	int16	Offset on battery temperature sensor 3 **Unit:** ddegC **Default:** 0
battemp4_offs	0x24	int16	Offset on battery temperature sensor 4 **Unit:** ddegC **Default:** 0

3.2.3 Configuration

This table holds run time configuration of the NanoPower BP8. Only the version in the persistent store should be changed at flight.

Table 3.5: Parameter table 1: 'configuration'.

Name	Address	Type	Description
auto_heat_en	0x0	bool	Enables autonomous heater **Default:** True
low_temp_lim	0x2	int16	Limit for auto heater low temperature (heater activates if temp is under this limit) **Unit:** ddegC **Default:** 20
high_temp_lim	0x4	int16	Limit for auto heater high temperature (heater stops if temp is over this limit) **Unit:** ddegC **Default:** 50
vbat_critical	0x8	uint32	Voltage limit for auto heater (heater stops if vbat is under this limit and will first start again when vbat is 10% over this limit) **Unit:** mV **Default:** 25000
fuse_burn_en	0xc	bool	Enables that a fuse burn can be done **Default:** False
log_mask	0xd	string	Settings applied to log groups and appenders during boot, overwriting the default levels set in the code. Format: <group|appender>=<level> (CSV list). Level: 0 (off), e (error), w (warning), n (notice), d (debug), t (trace) Example: "i2c=0,csp=t" **Max string length:** 64 **Default:**

3.2.4 Control

This table holds parameters which will control the application. This table cannot be saved.

Table 3.6: Parameter table 3: 'control'.

Name	Address	Type	Description
soc_reset	0x0	bool	Reset state of charge **Default:** False
fault_reset	0x1	bool	Setting this to true, will reset the battery fault signal **Default:** False

Continued on next page

Table 3.6: Parameter table 3: 'control'. (Continued)

Name	Address	Type	Description
fuse_burn	0x2	string	If set to correct burn key (the uid), it burns the battery fuse **Max string length:** 16 **Default:**
heat_manual	0x12	uint16	If set to value > 0, it activates heater in the amount of seconds and downcount itself **Unit:** s **Default:** 0 **Valid range:** 1 <= heat_manual <= 600

3.2.5 Telemetry

This table holds all telemetry collected by the application. Parameters are auto persistent and should not be changed externally.

Table 3.7: Parameter table 4: 'telemetry'.

Name	Address	Type	Description
uptime	0x0	uint32	How long the unit has been running (since last reset/boot) **Unit:** s
bootcount	0x4	uint16	Number of times the unit has booted
bootcause	0x6	uint16	Boot cause **Valid values:** '0': Unknown '1': Brownout (power on) '3': Watchdog, happens if OS task(s) halts '4': SW triggered boot, refer to resetcause
resetcause	0x8	uint16	Reset cause **Valid values:** '0': Unknown cause /hardware triggered boot '3': Stack overflow '4': Exception (error), illegal addressing etc. '5': Reset from GOSH '6': Reset from CSP '8': Insufficient RAM '100': Error in fuse burning procedure

Continued on next page

Table 3.7: Parameter table 4: 'telemetry'. (Continued)

Name	Address	Type	Description
soc	0xc	float	State of charge of the battery pack: 0.0 = empty, 1.0 = full. SOC can display a negative value when discharging below 23.6V. Charging to 32V will provide SOC of approx. 0.76, or 2100mAh. The reference for SOC is 2750mAh.
int_temp	0x10	int16	Internal MCU temperature **Unit:** ddegC **Invalid:** -32768
bat_avr_temp	0x14	float	Average temperature of battery temp sensors **Unit:** degC **Invalid:** -32768
bat_1_temp	0x18	int16	Temperature of battery sensor 1 **Unit:** ddegC **Invalid:** -32768
bat_2_temp	0x1a	int16	Temperature of battery sensor 2 **Unit:** ddegC **Invalid:** -32768
bat_3_temp	0x1c	int16	Temperature of battery sensor 3 **Unit:** ddegC **Invalid:** -32768
bat_4_temp	0x1e	int16	Temperature of battery sensor 4 **Unit:** ddegC **Invalid:** -32768
vbat	0x20	uint16	Battery voltage **Unit:** mV **Invalid:** 65535
i	0x24	float	Current in/out to battery pack **Unit:** A **Invalid:** 65535
in_i	0x28	uint16	Current in to battery pack **Unit:** mA **Invalid:** 65535

Continued on next page

Table 3.7: Parameter table 4: 'telemetry'. (Continued)

Name	Address	Type	Description
heater_i	0x2a	uint16	Current consumed by heater **Unit:** mA **Invalid:** 65535
out_i	0x2c	uint16	Current out of battery pack (including heater current) **Unit:** mA **Invalid:** 65535
o_volt_count	0x2e	uint16	Number of registered over voltage conditions (vbat over 33.2 V) **Default:** 0
bat_fault	0x30	bool	True if a battery cell fault is detected (voltage of one or more cells has once dropped below 2.3 V) **Default:** False

NOTE: The temperature measurements of type 'int16' such as `int_temp` are measured in `ddegC` (0.1 °C) while measurements of type 'float' such as `bat_avr_temp` are measured in °C.

4 Commands

The NanoPower BP8 features the console-like interface GOSH, as described in Section 1.3. This chapter describes the NanoPower BP8 specific GOSH commands and the GOSH commands needed to operate the parameter system.

4.1 Parameter System

The regular operation and configuration of the NanoPower BP8 is carried out through the parameter system. Entering the command `param` returns a list of subcommands, please refer to [2] for further information regarding the parameter system.

Listing 4.1: Commands used to interface the parameter system

```
BP8 # param
Parameter System (local)
  select        Select working table
  list          List all parameters
  tableinfo     Show table information
  export        Export parameters to stdout
  set           Set parameter value
  get           Get parameter value
  load          Load table
  save          Save table
  storeinfo     Show store information
  clear         Clear/invalidate store slot
  lock          Lock a store
  unlock        Unlock a store
```

4.1.1 Saving configuration

To change a configuration and save it in all stores, the following procedure should be followed.

- Unlock the flash storage by entering `param unlock flash`.
- Unlocking the flash requires a power on reboot of the MCU. This can be done by removing the ground breaker shortly (note that it automatically enables protection again after a second power cycle).
- Select the configuration table after boot by entering `param select configuration`.
- Unlock the protected storage by entering `param unlock protected` (note, that it automatically enables lock upon boot).
- Change the parameters in the working table (for instance entering `param set vbat_critical 0`).
- Save the changed working table to the different stores:
  - Enter `param save configuration flash` to save in flash.
  - Enter `param save configuration protected` to save in protected.
  - Enter `param save configuration persistent` to save in persistent.
- Perform a power on reboot.

4.1.2 Starting heater

Enter `param select control` to select the control table to work on. Entering the command `param list` will now show the control table, see Listing 4.2.

Listing 4.2: Control parameter table

```
BP8 # param select control
BP8 # param list
Table control (3)
  0x0000 soc_reset      BL false
  0x0001 fault_reset    BL false
  0x0002 fuse_burn      STR ""
  0x0012 heat_manual    U16 0
```

The content is the volatile working copy of the table (only version for the control table). Enter `param set heat_manual 30` to set the value of `heat_manual`. This starts the battery heater for 30 seconds. Entering `param get heat_manual` shows the value of the parameter.

4.2 Clearing telemetry

Some of the parameters in the telemetry table are persistent as covered in Section 3.1. These should not be cleared through the parameter system, but using specific GOSH commands, which will clear them in the internal states as well. Entering the command `clr_telemetry` returns a list of subcommands, see Listing 4.3.

Listing 4.3: Commands used to clear static telemetry

```
BP8 # clr_telemetry
bp8: Clear static telemetry
  bootcount         Select working table
  o_volt_count      List all parameters
```

Each sub command resets one (or two) of the persistent parameters in the telemetry table. For instance, the boot count is cleared by entering `clr_telemetry bootcount`. After pressing enter the boot count will reset to zero.

> ⚠ **CAUTION:** By resetting any of the persistent telemetry variables, vital historic information about the battery pack is lost

4.3 Battery decommission

For decommissioning purposes or when experiencing a cell fault as described in Section 2.5, it might be necessary to permanently disconnect the battery pack. This means, that the battery cells are disconnected from any circuitry, ensuring no thermal runaway.

> ⚠ **CAUTION:** A thermal runaway in the battery cell can potentially lead to fire or explosion, which will destroy the satellite and can lead to space debris.

The decommission feature is physically realised by burning a fuse, which normally connects the raw positive battery terminal to the output supply exposed on J1 and J2. The decommission procedure is programmed with safety to minimize the risk of both random bit flips in MCU RAM and operator errors, as an unintended disconnection could be catastrophic to a mission.

The steps to do a battery disconnect (fuse burn) follows:

1. Ensure the battery pack is in flight mode (`app_mode`) in the board table. This should be done prior to flight.
2. Enable the fuse burn function to execute (`fuse_burn_en`) in the configuration table. This should only be done prior to a fuse burn.
3. Command the actual fuse burn, by setting the parameter `fuse_burn` (a string) equal to the UID (uid) of the pack in the control table. This is the last step, and 60 seconds after setting it, the fuse will burn, disconnecting the battery pack.

An already activated burn can be stopped by undoing step 2 or 3 within the 60 second period. It is then necessary to reboot the NanoPower BP8, to reset the internal states in the procedure.

These steps are demonstrated in Section 4.3.1, with the exception that the parameter `app_mode` is set to integration mode (i.e., 0) for the simulation, whereas it should be set to flight mode (i.e., 1) for executing the disconnection.

> **NOTE:** GomSpace advises as the last thing in a decommissioning phase, the battery pack(s) be disconnected.

> ⚠ **CAUTION:** Activating a fuse burn will permanently kill the battery pack. Only do this at the end of use.

4.3.1 Simulating battery disconnection

It is possible to make a dry run of the disconnection procedure. This is done by setting the `app_mode` to integration mode followed by step 2 and 3 as described in Section 4.3. The fuse does not actually burn, but the logs print on GOSH interface as if it were burning (but with one log message declaring a dry run).

It should be noted that the configuration table is updated only in the persistent store because this is the only table that should be changed during flight. If the table should be updated prior to the mission, this can be done by following the procedure given in Section 4.1.1.

Listing 4.4: Dry run of battery disconnection procedure

```
BP8 # param select board
BP8 # param list
Table board (0)
  0x0000 uid            STR "103885-44"
  ...
BP8 # param select configuration
BP8 # param set fuse_burn_en 1
BP8 # param list
Table configuration (1)
  ...
  0x000C fuse_burn_en   BL  true
  ...
BP8 # param select control
BP8 # param set fuse_burn "103885-44"
BP8 # 40.056000 N fuse: only dry run (integration mode)
40.059000 W fuse: burning fuse in 60 sec (killing battery pack)
100.056000 W fuse: burning fuse
```

NOTE: The `param list` commands in the example are not necessarily but are used to confirm the parameters values and changes made.

5 References

[1] **GomSpace**
Datasheet 1034928
NanoPower BP8
Cited on pages 1–4, 6, 10

[2] **GomSpace**
Manual 1018560
NanoSoft - Product Interface Application
Cited on pages 9, 22

[3] **GomSpace**
Software library
nanopower-bp8_client
Cited on page 9