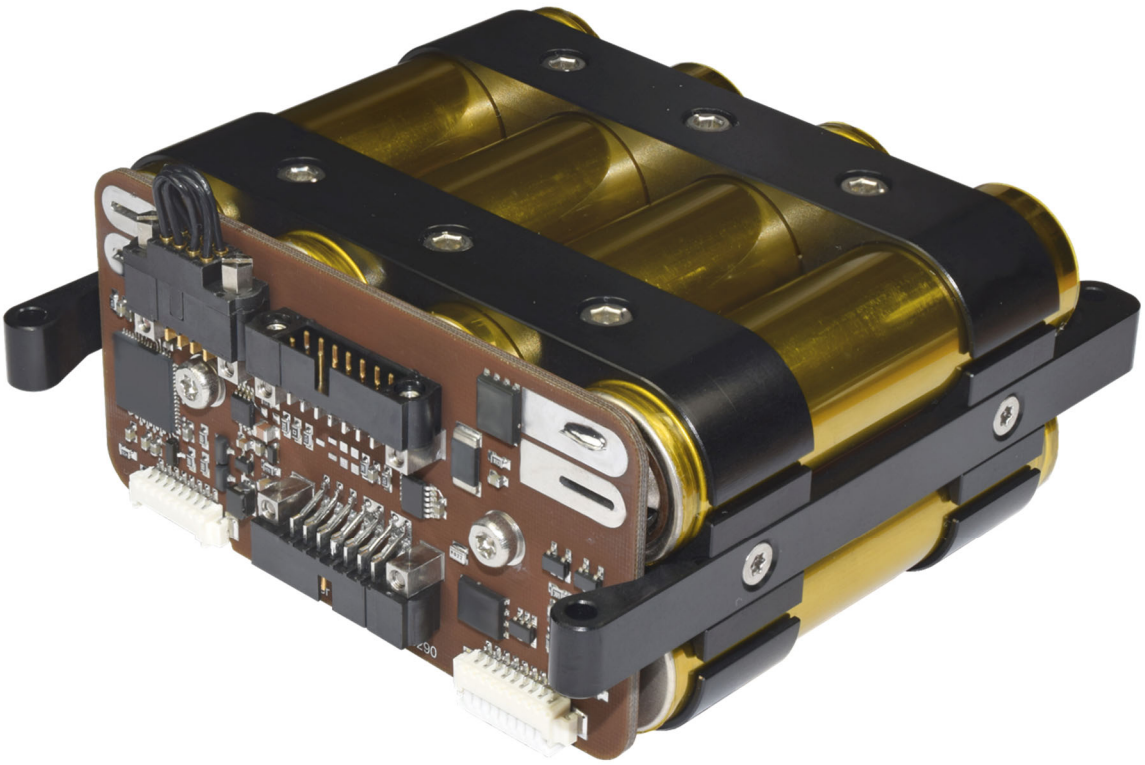


NanoPower
BPX

Manual

High-capacity battery pack for nanosatellites

GomSpace A/S | Langagervej 6 | DK-9220 Aalborg East | Denmark
gomspace.com | sales@gomspace.com | T: +45 71 741 741

1 Table of Contents

2 Changelog

Date	Revision	Author	Description
8-7-2016	1.0	KLK	First release
15-8-2016	1.1	KLK	Chapter 3.1 added
15-3-2017	1.2	ANM	Manual heat command added
17-5-2017	1.3	PNN	Page 9 Battery heater under voltage cut off in auto mode 1
23-7-2017	1.4	ANM	Added note on EEPROM cmds, chapter 4.2.5, and changed reply data for manual heat cmd, chapter 4.2.2.
27-06-2024	1.5	LAV	Added section 3.2 with electrical integration

3 Unpacking and handling precautions

Warnings:

This high-performance battery contains batteries capable of delivering very high currents. Be very careful to avoid shorts.

Balance out charge between NanoPower BPX battery packs when connecting them in parallel.

The NanoPower BPX system employs components based on FETs and therefore requires anti-static handling precautions to be observed.

Please use an ESD mat and a wrist strap as a minimum. Wear gloves to avoid fingerprints on the board. If any cleaning of the parts are required prior to flight, use only ESD safe cleaning methods and a neutral, non-reactive, IPA solvent.

Do not touch or handle the product without proper grounding!

3.1 ESD Grounding

The BPX must be properly ESD grounded before connecting it to a subsystem. It is very important that the BPX has the same ground potential as the system it connects to.

System ground must be connected to the BPX's PCB ground through the ESD cable supplied with the BPX.

The ESD cable can be removed only when the BPX has connection to the system ground through the PBAT1 or PBAT2 connector.

Figure 1 ESD cable

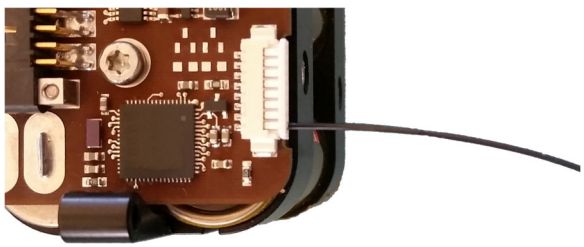


Figure 2 ESD cable inserted into BPX connector P2

3.2 Electrical Integration

The NanoPower BPX contains battery cells capable of delivering very high currents. Caution must therefore be taken to avoid short circuit when integrating the BPX.

To limit the risk of faults when connecting the NanoPower BPX to an electrical system the ground breaker at PGND1 is disconnected.
After connected to the EPS through PBAT1/PBAT2, the ground breaker can be inserted.

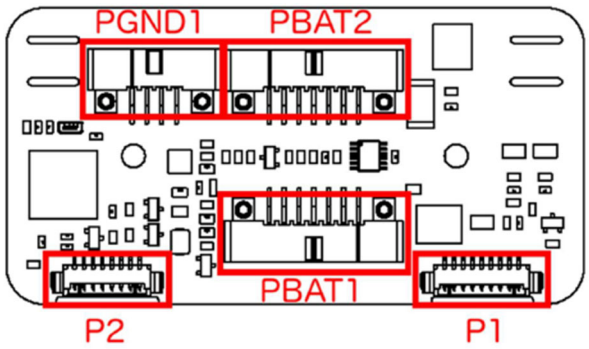


Figure 4 Location of connectors

4 Software

4.1 Terminal (GOSH)

As an extension of the BPX interface, there is a command line utility available. This command system works by interpreting commands given on the serial port of the BPX and calling the BPX C-library functions. This means that they are available directly on the serial port by typing commands. This is great for debugging, configuring and getting started with the BPX.

The GomSpace Shell (GOSH) currently runs on almost all GomSpace products and on the ground-station PC in a small application called CSP-term that is freely available. This means the BPX commands can be executed from several different sources, depending on where you have access to a GOSH shell. When issuing a command from the Ground-station, TNC or OBC, the underlying CSP-protocol will ensure to route the commands correctly to the BPX and back again.

NOTICE: When the BPX is off, it is advised to remove the serial cable. Leak current on the Rx line can in some case keep powering the electronics.

4.1.1 Commands

By typing ‘`help`’, in any GOSH shell, a list of commands will be printed. Most commands available are not subsystem specific. To see the version of the BPX type ‘`cmp ident`’. Example:

```
bpx # cmp ident
 Hostname: BPX
 Model:    BPX
 Revision: v2.0
 Date:     Dec 19 2013
 Time:     09:51:22
```

To see the UID of the unit type ‘`board getuid`’.

The standard BPX commands are all prefixed with the word ‘`bpx`’. In order to list the BPX commands type ‘`bpx <tab>`‘

4.2 Command and Data Interface

The NanoPower BPX is an independent network node designed for the CSP protocol and I²C bus communication. The CSP protocol is implemented in all GomSpace products and provides a highly capable and integrated networking architecture that can be utilized across multiple physical link implementations to cover both the space and ground segment.

4.2.1 How to send a command

If you have a NanoMind OBC in your satellite, drivers for the all NanoPower systems are already included. If you do not, there is also the option of writing a custom driver, based on the CSP interface specification.

1. **Using client library (C-interface)** All the functions and commands in GOSH are using the C-interface in `<bpx/io.h>` header file. These functions are also available to be called from any part of your own C-code. In order to get BPX housekeeping data just issue a C-call to the function `bpx_hk_get()`. The C-interface will take care of generating the correct request and sending the request over the CSP network to the BPX, wait for the reply, decode the data and represent it nicely in a C data structure called ‘`bpx_hk_t`’. This makes it very easy to write a housekeeping collector or send commands to power subsystems on and off from anywhere on the satellite.

2. **Write your own driver** If you do not have a GomSpace OBC, or a GomSpace ground station with the C client library or GOSH, GomSpace is happy to share with you the source-code for the CSP network protocol and the C-library. This way you can port the nanopower drivers to your own CPU and architecture. Please contact GomSpace for more information about this option.

Information like uptime, identification etc. can be accessed through the standard CSP interface, please see http://www.libcsp.org for more information.

4.2.2 CSP Network Interface

Applies to latest version of BPX - Your system may have other commands or may not support some of these commands. If you are in doubt please contact GomSpace for further help

Mnemonic	Port	Request/Reply Data	Bytes	Description
GET_HK	9	empty	0	Send empty packet to request housekeeping struct.
		struct bpx_hk_t;	struct	Reply: Data-structure (see below)
RESET_COUNTERS	15	uint8 magic=0x42;	1	Send this command to reset boot counter magic = 0x42
		none	x	
CONFIG_CMD	17	uint8 cmd;	1	Use this command to control the config system. cmd=1: Restore default config
		none	x	
CONFIG_GET	18	none	x	Use this command to request the BPX config.
		struct eps_config_t	struct	Reply: Data-structure (see below)
CONFIG_SET	19	struct eps_config_t	struct	Use this command to send a config to the BPX and save it.
		none	x	
MAN_HEAT_ON	20	uint16_t heat_time	2	Use this command to start heating manually for a specific period of time (s). This works with any configurations.
		uint8 reply	1	reply = 0x01 means manual heat on OK
MAN_HEAT_OFF	21	none	x	Use this command to interrupt and stop manual heating.
		uint8 reply	1	reply = 0x01 means manual heat off OK

Information like uptime, identification etc. can be accessed through the standard CSP interface, please see libcsp.org for more information.

4.2.3 Housekeeping Format:

The housekeeping data is a structure as specified below:

```
typedef struct {
    uint16_t cur_charge;      //! Charge current in mA
    uint16_t cur_discharge;   //! Discharge current in mA
    uint16_t cur_heater;      //! Heater current in mA
    uint16_t vbatt;           //! Battery voltage in mV
    int16_t bat_temp1;        //! Battery temperature 1 in degC
    int16_t bat_temp2;        //! Battery temperature 2 in degC
    int16_t bat_temp3;        //! Battery temperature 3 in degC
    int16_t bat_temp4;        //! Battery temperature 4 in degC
    uint32_t counter_boot;    //! Number of reboots
    uint8_t bootcause;        //! Cause of last reset
} bpx_hk_t;
```

Reset cause can be 0=Power On Reset, 1=External Reset, 2=Brown Out Reset, 4=WDT reset, 8=JTAG reset.

4.2.4 I²C Bus Specification

The I²C interface is used for commanding the NanoPower and for receiving housekeeping and status messages. NanoPower operates on the I²C bus as multi-master node, which is either as *slave receiver* or as *master transmitter*. NanoPower transmits at 400 kbit and can receive at anything from up to 400 kbit. The CSP network stack takes care of the I²C bus addressing and framing format. The CSP/ I²C frame looks like this:

<start><write><csp-header><data><stop>

Where the CSP header is 4 bytes big endian (For more information on this, see libcsp.org and read the specification for the csp_if_i2c interface), and the data field contains the request/reply as specified in the command and data handling section of this datasheet.

Paramter	Condition	Min	Typical	Max	Unit
I²C speed transmit			400		kbit
I²C speed receive		0		400	kbit
I²C / CSP address	Default (can be changed through GOSH)		7		

I²C Slave Mode

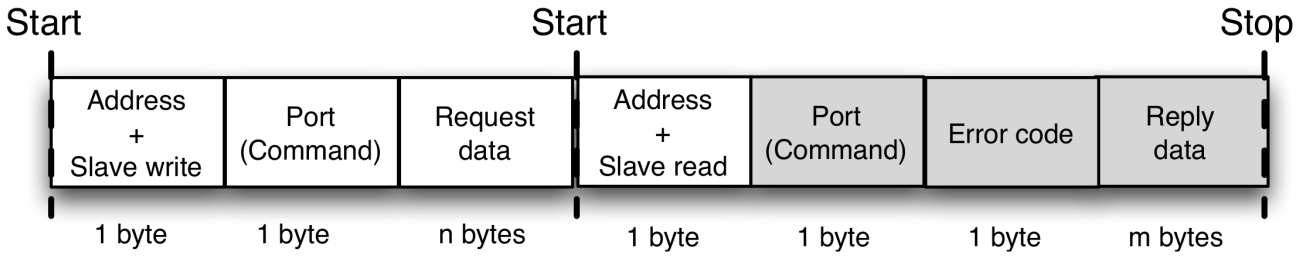
For users, which do not wish to support the CSP protocol, or the multi-master I²C, interface. The BPX comes with a separate slave-mode I²C interface. Setting the '`board i2cslave`' to 1 can enable this interface.

NOTE: When the BPX is set to slave mode, the CSP commands like "`bpx hk`", "`cmp ident`", etc in GOSH does not work. Furthermore, the slave interfaces, ie. "`bpxslave`" commands do not work either as the unit cannot be both slave and master at the same time.

NOTE: Commands "`RESET_COUNTERS`", "`CONFIG_CMD`" and "`CONFIG_SET`" are writing to the eeprom, which can't finish within an I²C transaction. Expect these commands to be finished within 500 ms after execution.

This mode of operation disables the use of the CSP stack, and uses a slave-mode only protocol instead. A limited set of the CSP commands is available in this mode. An I²C master wishing to communicate with the BPX device, should send a single byte specifying the command number, followed by any command arguments. The command number should match the CSP port number from multi-master mode.

The BPX returns the same 1-byte command number, followed by an error code. A successful command will return an error code of 0. Errors are marked by returning a non-zero error code. If a command is marked erroneous, the remaining bytes of the reply will be undefined and should be discarded. The following figure shows the command sequence on the I²C bus. White boxes are data sent from the I²C master, while grey boxes marks data read from the BPX system:



As in CSP mode, all values of more than 1 byte must be transmitted in big endian byte order.

Additional Slave Mode Commands
Since the CSP stack is not used, the command set has been extended with a ping and reboot command. The CSP service handler normally handles these commands.

Mnemonic	Port	Request / Reply Data	Bytes	Description
PING	1	`uint8_t value`	1	One byte ping value. Any value can be used.
		`uint8_t value`	1	The BPX replies with the same value as in the ping request.
REBOOT	4	`uint8_t magic[4]`	4	Magic sequence must be: 0x80, 0x07, 0x80, 0x07
		`none`		The BPX is rebooted, so no reply is generated. A stop condition should be sent after the request, instead of the repeated START.

4.3 Configuration

The BPX allows the user to setup a number of configurations like auto-heater parameters. This is done in two different config structs through the standard command system through the '`bpx conf`' commands:

```
bpx # bpx conf
  get               Conf get
  set               Conf set
  edit              Edit local config
  print             Print local config
  restore           Restore config from default
```

Furthermore, there are dedicated GOSH commands to handle the default configs:

```
bpx # conf
  restore                   Configuration restore
  store_default             Store configuration as default
  batt_critical_level       Set/get the voltage where heater stops
```

To view the current config of the BPX type '`bpx conf get`' in GOSH and to edit the config type '`bpx conf edit`'. To send and save the configuration on BPX type '`bpx conf set`'.

On the BPX, a configuration has two instances: a working config and a default config:
The working config is the one currently used by BPX and it can be edited through the '`conf get`', '`conf edit`' and '`conf set`' commands.

The default config can be seen as a backup config which can only be set before launch through GOSH by by '`conf store_default`'. There are three ways of restoring to the default config:
By typing '`conf restore`' in BPX GOSH

By using the restore command over I^2C ('`bpx conf restore`' in GOSH)
If a checksum error is found in the working config, the default config is restored

The config struct is defined as

```
typedef struct __attribute__((packed)) {
    uint8_t battheater_mode;  //! Mode for battheater  [0=OFF,1=Auto]
    int8_t battheater_low;    //! Turn heater on at [degC]
    int8_t battheater_high;   //! Turn heater off at [degC]
} bpx_config_t;
```

The BPX will turn on the heater if the mode is set to auto (1) and the average temperature of the temperature sensors is below the battheater_low value. It will turn off the heater when the average temperature is above the battheater_high value or if mode is set to off (0). It will also stop if the filtered battery voltage is under the batt_critical_level. It will turn on again when the filtered battery voltage is 10 % over batt_critical_level. The batt_critical_level is configurable only through GOSH.