GOMSPACE

NanoPower

P60 PDU-200

Manual

Software Documentation

GomSpace ApS • Alfred Nobels Vej 21C • DK-9220 Aalborg East • Denmark
www.gomspace.com • info@gomspace.com • +45 96356111

Table of Contents

1. Change log

Date	Revision	Author	Description
14-06-2016	1.0	KBA	Initial version 1.0
21-10-2016	1.1	KBA	Fixed error in parameter table 1 documentation

2. Introduction

The GomSpace NanoPower P60 EPS is a flexible and modular system to rapidly implement power supply systems based on customer requirements. The P60 PDU-200 is part of the NanoPower P60 EPS modular power supply system.

This manual describes the P60 PDU-200 firmware version 1.2:

- Command Interface
- CSP Client API
- Parameter Tables

2.1 Unpacking and Handling Precautions

Warning The P60 PDU-200 is an ESD sensitive device. Proper precautions must be observed during the handling of the device.

Please use an ESD mat and a wrist strap as a minimum. Please wear gloves to avoid fingerprints on the anodized aluminum parts, as these are particularly difficult to rinse off. If any cleaning of the parts are required prior to flight, use only ESD safe cleaning methods and a neutral, non-reactive, IPA solvent.

3. Overview

The P60 PDU-200 can be operated through the Cubesat Space Protocol (CSP) over CAN, I^2C or serial port. This integrates well with other GomSpace products, and GomSpace provides a client library for commanding the P60 PDU-200 from other systems running CSP. Refer to the Client API page for documentation of the client library API.

For integration and testing purposes, a command line interface is also available as explained in Command Interface. The command line interface uses the client API internally, so command names generally reflects the API function names.

The block diagram below shows the different P60 PDU-200 modules.

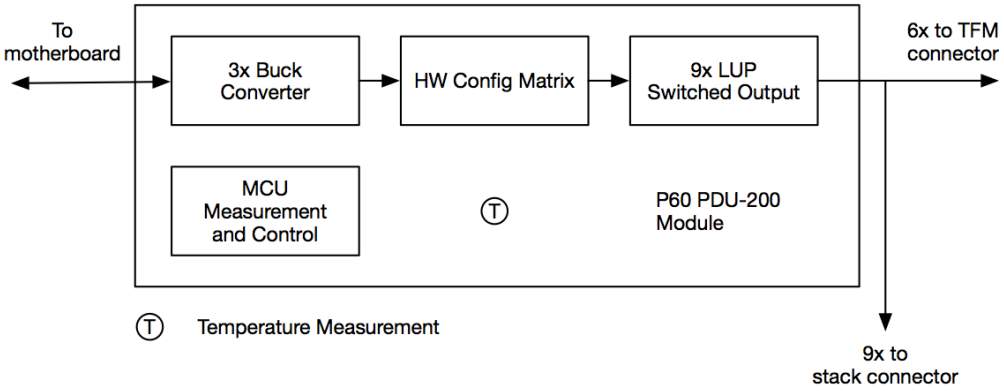


Fig. 3.1: P60 PDU-200 Block Diagram

4. Command Interface

For testing and debugging purposes, the P60 PDU-200 can be operated using the GomSpace Shell (GOSH) interface. The console interface is available in the FPC Debug connector. The serial connection is 8 databits, 1 stopbit, no parity at 500000 baud. On Linux, the serial connection can be opened using the minicom or tio tools (tio was formerly known as gotty).

You can now execute commands in the GOSH command interpreter. An example command is `cmp ident` which lists identification info of the current subsystem:

```
p60-pdu # cmp ident
Hostname: p60-pdu
Model:    NanoPower P60 PDU-200
Revision: v0.1
Date:     Nov  3 2015
Time:     09:53:08
```

A list of available commands can be shown by running the `help` command in GOSH. Pressing the *tab* key automatically completes commands and subcommands, and shows a usage string. Commands specific to the P60 PDU-200 are assembled under the `p60pdu` command group. These commands are also available in `csp-term` for operating the P60 PDU-200 via CSP:

```
p60-pdu # p60pdu
  node                Set node
  timeout             Set timeout in milliseconds
  hk                  Get HK
  gndwdt_clear        Clear GND WDT
```

Note: All of the `p60pdu` commands run over loopback CSP using GOSH command wrappers around the Client API

4.1 Global Settings

The P60 PDU-200 command line interface is written such that there are a few global variables that can be set first. The default values of the variables are:

- The P60 PDU-200 CSP address is set to 3.
- Timeout is set to 5000 ms.

These values can be set using the `p60pdu node <node>` and `p60pdu timeout <timeout>`. To show the current setting, run each command without an argument.

5. Operation

The P60 PDU-200 has three configuration parameter tables:

- Board parameters (table 0)
- Configuration parameters (table 1)
- Calibration parameters (table 2)

P60 PDU-200 telemetry parameters are available in parameter table 4.

See Parameter Tables for details on the individual parameter tables.

5.1 Board Parameters

The Board parameters will normally be configured from factory.

The P60 PDU-200 does not support RS422 USART and rs422 should is set to 0.

CSP address is controlled by parameter is `csp_addr`. The default CSP address is 3 for the P60 PDU-200. If multiple P60 PDU-200 modules are included, they will all use different CSP address, e.g. 3, 5 and 6

5.2 Configuration Parameters

The P60 PDU-200 has a total of 9 output channels that all can be turned on and off. Each channel can be configured with a name. Default names are listed in table below. Seven characters are available for channel names. Channel names are set by parameter `out_name[]`.

Table 5.1: P60 PDU-200 Output channels

Channel	Description	Default name
0	Output channel 0	ch0
1	Output channel 1	ch1
2	Output channel 2	ch2
3	Output channel 3	ch3
4	Output channel 4	ch4
5	Output channel 5	ch5
6	Output channel 6	ch6
7	Output channel 7	ch7
8	Output channel 8	ch8

5.2.1 Channel Output Control

The 9 output channels can all be turned on and off manually and/or automatically. From factory the P60 PDU-200 will be configured not to turn any output channels on automatically.

To turn a channel on or off manually, set the appropriate `out_en[]` parameter.

To turn a channel on manually with a delay, set the appropriate `out_on_cnt[]` parameter to turn the channel on after X seconds. X must be larger than zero.

To turn a channel off manually with a delay, set the appropriate `out_off_cnt[]` parameter to turn the channel off after X seconds. X must be larger than zero.

5.2.2 Battery Mode Dependant Output Control

The P60 PDU-200 output channels can be turned on and off automatically depending on the battery mode (see *Battery Voltage Level*).

Battery mode FULL

In battery mode FULL, all output channels are left unchanged.

Battery mode NORMAL

To turn a channel on automatically without delay when battery mode is NORMAL, set the appropriate `init_out_norm[]` parameter to 1.

To turn a channel on automatically with a delay when battery mode is NORMAL, set the appropriate `init_on_dly[]` parameter to turn the channel on after X seconds. X must be larger than zero. In this case the `init_out_norm[]` parameter should be 0, otherwise the channel will turn on immediately when battery mode is NORMAL.

To turn a channel off automatically with a delay when battery mode is NORMAL, set the appropriate `init_off_dly[]` parameter to turn the channel off after X seconds. X must be larger than zero. In this case the `init_out_norm[]` parameter should be 1 or the `init_on_dly[]` parameter should have a value smaller than the `init_off_dly[]` parameter.

Battery mode SAFE

To keep a channel turned on if battery mode is SAFE mode, set the appropriate `init_out_safe[]` parameter to 1 to keep the channel on when battery mode is SAFE. Otherwise the channel will be turned off when entering SAFE mode.

The parameters `init_on_dly[]` and `init_off_dly[]` are not used in battery mode SAFE.

Each channel has a `safe_off_dly[]` parameter to allow a delay before turning the channel off when entering battery mode SAFE. This allows controlled shutdown of e.g. a payload or flight computer when entering the SAFE mode. Set the `safe_off_dly[]` parameter to turn the channel off after X seconds when entering battery mode SAFE. X must be larger than zero.

Battery mode CRITICAL

In battery mode CRITICAL, all output channels will turned off automatically without any delay.

5.2.3 Current Limitation

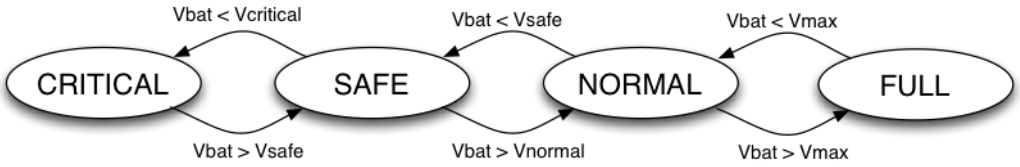
All channels have two current limitation parameters, `cur_lu_lim[]` and `cur_lim[]`.

The parameter `cur_lu_lim[]` is a hard limit, and if the channel consumes more than this limit it will be turned off immediately. It will be turned on again automatically after 5 seconds.

The parameter `cur_lim[]` is a soft limit, meaning that if the channel over a period of time on average consumes more than this limit it will be turned off. The average is calculated over time as an Exponential Moving Average (EMA). The channel will be turned on again automatically after 5 seconds.

5.2.4 Battery Voltage Level

The P60 PDU-200 module has a software low voltage protection and will automatically turn channels on and off depending on battery voltage level and a set of battery level configuration parameters (`batt_max`, `batt_norm`, `batt_safe` and `batt_crit`). The software low voltage protection is a four state system with a CRITICAL, a SAFE, a NORMAL and a FULL mode.



In normal mode everything is nominal but should the battery voltage drop below batt_safe, the P60 PDU-200 module will change its output switch configuration to a safe mode configuration. This allows the P60 PDU-200 module to switch off of all non-essential systems and leave a simple low power mode running.

Should the battery voltage continue to drop below `batt_crit`, the P60 PDU-200 module will switch off all channels.

The four battery voltage level parameters (`batt_max`, `batt_norm`, `batt_safe` and `batt_crit`) are each configurable and must be set to match the battery and the channel control (see section *Channel Output Control*) must be set accordingly to ensure that daughter boards are turned on/off as required by mission.

The following table shows the default settings for battery voltage levels.

Table 5.2: Recommended settings for battery voltage levels

Battery Pack Voltage	**8 V**	**16 V**	**32 V**
`batt_max`	8300 mV	16600 mV	32000 mV
`batt_norm`	7000 mV	14000 mV	28000 mV
`batt_safe`	6700 mV	13400 mV	26800 mV
`batt_crit`	6500 mV	13000 mV	26000 mV

5.3 Calibration Parameters

The P60 PDU-200 has a set of calibration parameters for voltage and current measurements.

The calibrations parameters are configured from factory and normally don't need to be changed.

For voltages, there is a gain parameter. The gain is a floating-point number.

For currents, there is a gain and an offset parameter. The gain is a floating-point number. The offset is a signed 16-bit integer that controls the offset in mA.

The following table lists the default nominal values for each calibration parameter:

Table 5.3: P60 PDU-200 Calibration values

Name	Voltage gain	Current gain	Current offset
Output channel 0	25.0	1.0	0
Output channel 1	25.0	1.0	0
Output channel 2	25.0	1.0	0
Output channel 3	25.0	1.0	0
Output channel 4	25.0	1.0	0
Output channel 5	25.0	1.0	0
Output channel 6	25.0	1.0	0
Output channel 7	25.0	1.0	0
Output channel 8	25.0	1.0	0
VBAT	25.0	n/a	n/a
VCC	3.2	n/a	n/a

5.4 Telemetry Parameters

The P60 PDU-200 will provide the measured voltage and current for each output channel in parameter `v_out[]` and `c_out[]`. In case of over-current (see section *Current Limitation*) on an output channel, the `latchup[]` parameter will increase by 1 each time an over-current is detected and the output channel is turned off by the current monitoring function.

The output enabled mode of each output channel is provided in the `out_en[]` parameter.

The battery voltage measured by the P60 PDU-200 is provided in the parameter `batt_v`. The battery voltage is monitored and the battery mode is provided in the parameter `batt_mode`. The `batt_mode` can have one of four values:

Table 5.4: P60 PDU-200 Battery mode values

Battery Mode value	Battery Mode description
1	CRITICAL
2	SAFE
3	NORMAL
4	FULL

The battery current is also measured and is provided in the parameter `batt_c`.

Temperature of the P60 PDU-200 is measured in one location on the module and is available in parameter `temp`.

5.5 Configuration

The P60 PDU-200 module has two sets of configuration, the running configuration and the default configuration. The default configuration is a fallback configuration that will be restored automatically if the ground watchdog is

triggered (see *Ground Watchdog*)

The P60 PDU-200 has a set of commands to manage the configuration:

```
p60-pdu # config
  status              Show FRAM Lock status
  lock                Lock FRAM upper tables 0x1000 to 0x1800
  unlock              Unlock FRAM upper tables 0x1000 to 0x1800
  gnd_wdt             Get or set gnd wdt timeout value
  update_default      Update default factory settings in FRAM
  verify_default      Verify default factory settings in FRAM
```

5.6 Ground Watchdog

The P60 PDU-200 module has a ground watchdog that needs to be activated at least every 48 hours if enabled. The purpose of the ground watchdog is to enable the P60 PDU-200 to revert to the default configuration if for some reason the running configuration prevents the P60 PDU-200 from operating.

If the ground watchdog is triggered, the P60 PDU-200 will restore to the default configuration and then do a reset of the P60 PDU-200.

To enable the ground watchdog and set the timeout value, run the following command form the GOSH terminal:

```
config gnd_wdt <timeout in seconds>
```

The minimum value is 172800 seconds (48 hours). The maximum value 31536000 seconds (365 days). To disable the ground watchdog, set the timeout to 0 seconds.

Note: The P60 PDU-200 will restore to the default configuration and then do a reset of the P60 PDU-200 if the ground watchdog is triggered.

5.7 Bus Watchdogs

The P60 PDU-200 module has two satellite bus watchdogs, one for I^2C and one for CAN, which allows the P60 PDU-200 to monitor if the satellite bus is operational. Both satellite bus watchdogs can be enabled or disabled and has individual timeout values, `wdt_i2c` and `wdt_can` respectively. The P60 PDU-200 can be configured to reboot if the satellite bus watchdog is triggered, this is controlled by the `wdt_i2c_rst` and `wdt_can_rst`, respectively.

When configured and enabled (`wdt_i2c_rst`/`wdt_can_rst` is 1 and `wdt_i2c`/`wdt_can` > 0), the P60 PDU-200 will reboot if no traffic is observed on the satellite bus for `wdt_i2c`/`wdt_can` seconds.

Note: The satellite bus watchdogs does not run when the battery mode is CRITICAL.

5.8 CSP Watchdogs

The P60 PDU-200 module has two CSP watchdogs that allows to actively monitor two different nodes on the CSP satellite bus.

If a CSP address is configured for `wdt_csp_addr[]`, the P60 PDU-200 will actively monitor the connectivity to each of these systems with a ping packet each `wdt_csp[]` seconds. If `wdt_csp_ping[]` consecutive pings are lost (maximum allowed response time is 30 ms), the corresponding power channel `wdt_csp_chan[]` is power cycled with a 5 second off time.

The CSP watchdogs will only run if the configured output channel `wdt_csp_chan[]` is on.

6. Client API

The client API consists of a set of wrapper functions that simplify the CSP interface to the P60 PDU-200. These functions are implemented in the `p60pdu_client.c` file and can be integrated in custom code by including the `p60pdu.h` header file. The file `p60pdu_cmd.c` implements the GOSH commands for the P60 PDU-200 and can be used as an additional reference for the use of the client API.

All the client functions specify a `timeout` argument that is used to specify the maximum number of milliseconds to wait for a reply. The client interface automatically performs endian conversion to network byte order on all arguments.

6.1 CSP Port Numbers

The P60 PDU-200 listens on the following CSP port numbers

Table 6.1: P60 PDU-200 CSP port numbers

Port	Name	Description
0	CSP_CMP	Control Port
1	CSP_PING	Returns a copy of the packet received
2	CSP_PS	Returns process list
3	CSP_MEMFREE	Returns memory free
4	CSP_REBOOT	Reboots subsystem
5	CSP_BUF_FREE	Returns number of free buffers
6	CSP_UPTIME	Returns subsystem uptime
7	P60_PORT_RPARAM	Controls P60 PDU with parameter system (see below)
9	P60_PORT_GNDWDT_RESET	Used for ground watchdog reset
10	P60_PORT_CMDCONTROL	Reserved for future use

For a description on how to use the CSP ports, please see libcsp manual. For a description on the parameter system, please see libparam manual.

6.2 Get and Set Configuration Parameters

Getting and setting configuration parameters is done by using the remote parameter system API provided by libparam.

Get a single configuration parameter from P60 PDU-200:

```
#include <p60pdu.h>

uint8_t output_enable;
int res = rparam_get_single(&output_enable,        /* Pointer to variable */
                            P60PDU_OUT_EN(0),      /* Logical address of parameter */
                            PARAM_UINT8,           /* Parameter type */
                            1,                     /* Parameter size */
                            P60PDU_PARAM,          /* Parameter table id */
                            p60pdu_node,           /* P60 PDU csp node address */
                            P60_PORT_RPARAM,       /* CSP port number */
                            1000);                 /* Timeout in milliseconds */
if (res > 0) {
    printf("Get output enable: %u\n", output_enable);
}
```

Set a single configuration parameter on P60 PDU-200:

```
#include <p60pdu.h>

uint8_t output_enable = 1;
int res = rparam_set_single(&output_enable,        /* Pointer to variable */
                            P60PDU_OUT_EN(0),      /* Logical address of parameter */
                            PARAM_UINT8,           /* Parameter type */
                            1,                     /* Parameter size */
                            P60PDU_PARAM,          /* Parameter table id */
                            p60pdu_node,           /* P60 PDU csp node address */
                            P60_PORT_RPARAM,       /* CSP port number */
                            1000);                 /* Timeout in milliseconds */
if (res > 0) {
    printf("Set output enable: %u\n", output_enable);
}
```

6.3 Retriving Housekeeping Parameters

Retrieving Housekeeping Parameters from the P60 PDU-200 is provided by the `p60pdu_get_hk` function.

```
int p60pdu_get_hk(param_index_t * mem, uint8_t node, uint32_t timeout);
```

This function is used to retrieve housekeeping parameters from the P60 PDU-200. The housekeeping parameters are retrieved over CSP using the remote parameter system API provided by libparam. The function `p60pdu_get_hk` is basically a wrapper around the remote parameter system API. The input parameter `mem` must be provided with a pointer to memory to hold the housekeeping parameters. The input parameter `node` must provide the CSP node address of the P60 PDU-200 and the `timeout` must provide the timeout in milliseconds. The following shows an example of this.

```
#include <p60pdu.h>

uint8_t p60pdu_node = 3;
uint8_t hk_mem[P60PDU_HK_SIZE];
param_index_t p60pdu_hk = {0};
p60pdu_hk.physaddr = hk_mem;
if (!p60pdu_get_hk(&p60pdu_hk, p60pdu_node, 1000)) {
    printf("Error getting p60pdu hk\n");
} else {
    param_list(&p60pdu_hk, 1);
}
```

Housekeeping parameters can also be retrieved directly from P60 PDU-200 using the remote parameter system API provided by libparam:

```
#include <p60pdu.h>

uint8_t p60pdu_node = 3;
uint8_t out_en;
int res = rparam_get_single(&out_en,               /* Pointer to variable */
                            P60PDU_HK_OUT_EN(0),   /* Logical address of parameter */
                            PARAM_UINT8,           /* Parameter type */
                            1,                     /* Parameter size */
                            P60PDU_HK,             /* Parameter table id */
                            p60pdu_node,           /* P60 PDU csp node address */
                            P60_PORT_RPARAM,       /* CSP port number */
                            1000);                 /* Timeout in milliseconds */
if (res > 0) {
    printf("Get out_en[0]: %u\n", out_en);
}
```

The previous example retrieves a single parameter. It is possible to retrieve all Housekeeping parameters by retrieving the entire table:

```
#include <p60pdu.h>

uint8_t hk_mem[P60PDU_HK_SIZE];
param_index_t node_hk = {0};
uint8_t p60pdu_node = 3;

node_hk.physaddr = hk_mem;
node_hk.table = p60pdu_hk;
node_hk.mem_id = P60PDU_HK;
node_hk.count = P60PDU_HK_COUNT;
node_hk.size = P60PDU_HK_SIZE;
int result = rparam_get_full_table(&node_hk,
                                   p60pdu_node,
                                   P60_PORT_RPARAM,
                                   node_hk.mem_id,
                                   1000);
if (result != 0) {
   printf("Error retrieving P60 PDU housekeeping\n");
} else {
   printf("Retrieved P60 PDU housekeeping\n");
   /* List all out_en[] values, using parameter name */
   const param_table_t * param = param_find_name(node_hk.table,
                                                node_hk.count,
                                                "out_en");
   if (param != NULL) {
      for (uint8_t index = 0; index < 9; index++) {
         /* Read parameter using name */
         uint8_t *out_en = param_read_addr(param->addr + param->size * index,
                                           &node_hk,
                                           param->size);
         printf("out_en[%d]: %u\n", index, *out_en);
      }
   }
   /* List all c_out[] values, using parameter address */
   param = param_find_addr(node_hk.table, node_hk.count, 0x0000);
   if (param != NULL) {
      for (uint8_t index = 0; index < 9; index++) {
         /* Read parameter using address */
         int16_t *c_out = param_read_addr(param->addr + param->size * index,
                                          &node_hk,
                                          param->size);
         printf("c_out[%d]: %d mA\n", index, *c_out);
      }
   }
}
```

6.4 Reset Ground Watchdog Timer

The Ground Watchdog Timer is reset by a dedicated command to the P60 PDU-200. This can for example be used as a ground communication watch dog, i.e. this command is issued to P60 PDU-200 on each connection with the ground station. If no communication has been received for a period of 48 hours P60 PDU-200 will switch off and do a reset back to default configuration.

The Ground Watchdog Timer on the P60 PDU-200 is reset by sending a single byte with the value 0x78 to CSP port `P60_PORT_GNDWDT_RESET`.

The following code will reset the Ground Watchdog Timer on the P60 PDU-200:

```c
#include <p60pdu.h>

int p60pdu_gndwdt_clear(uint8_t node, uint32_t timeout) {
   uint8_t magic = 0x78;
   return csp_transaction(CSP_PRIO_HIGH,
                          node,
                          P60_PORT_GNDWDT_RESET,
                          timeout,
                          &magic,
                          1,
                          NULL,
                          0);
}
```

7. Parameter Tables

A number of parameters on the P60 PDU-200 can be adjusted through the GomSpace parameter system. The parameters are divided across four tables depending on the parameter types.

Table 0 is used for system level parameters that are expected to stay fixed for the entire mission. Table 1 is used for adjusting the operation and performance parameters. Table 2 is used for system level calibrartion parameters that are expected to stay fixed for the entire mission. Table 4 is used to store telemetry data.

Table 7.1: P60 PDU-200 Parameter tables

Table	File	Default	Description
0	0	4	Board Configuration
1	1	5	Module Configuration
2	2	6	Calibration parameters
4	-	-	Telemetry Data

Through GOSH it is possible to list all parameters in a table by running the `param list <table>` command. Here it shows the default configuration parameters:

```
p60-pdu # param list 1
Parameter list 1:
  0x0000 out_name          STR "ch0" "ch1" "ch2" "ch3" "ch4" "ch5" "ch6" "ch7" "ch8"
  0x0048 out_en            U8  0 0 0 0 0 0 0 0 0
  0x0052 out_on_cnt        U16 0 0 0 0 0 0 0 0 0
  0x0064 out_off_cnt       U16 0 0 0 0 0 0 0 0 0
  0x0076 init_out_norm     U8  0 0 0 0 0 0 0 0 0
  0x0080 init_out_safe     U8  0 0 0 0 0 0 0 0 0
  0x008A init_on_dly       U16 0 0 0 0 0 0 0 0 0
  0x009C init_off_dly      U16 0 0 0 0 0 0 0 0 0
  0x00AE safe_off_dly      U8  0 0 0 0 0 0 0 0 0
  0x00B8 cur_lu_lim        U16 2500 2500 2500 2500 2500 2500 2500 2500 2500
  0x00CA cur_lim           U16 2500 2500 2500 2500 2500 2500 2500 2500 2500
  0x00DC cur_ema           U16 0 0 0 0 0 0 0 0 0
  0x00EE out_link          U8  0 0 0 0 0 0 0 0 0
  0x00F8 out_conv          U8  1 1 0 0 2 2 1 0 2
  0x0102 out_voltage       U16 3300 3300 5000 5000 3300 3300 3300 5000 3300
  0x0114 conv_en           U8  0 1 0
  0x0118 cur_ema_gain      FLT 0.500000
  0x011C batt_hwmax        U16 16000
  0x011E batt_max          U16 9000
  0x0120 batt_norm         U16 8000
  0x0122 batt_safe         U16 7000
  0x0124 batt_crit         U16 6000
  0x0126 wdt_i2c_rst       U8  0
  0x0127 wdt_can_rst       U8  0
  0x0128 wdt_i2c           U32 0
  0x012C wdt_can           U32 0
  0x0130 wdt_csp           U32 60 60
  0x0138 wdt_csp_ping      U8  5 5
  0x013A wdt_csp_chan      U8  0 0
  0x013C wdt_csp_addr      U8  0 0
```

A parameter is modified by first switching to its table with `param mem <table>` and then running `param set <parameter> <value>`. E.g., to set the `out_en` value for channel 0 to 1 we run the commands:

```
p60-pdu # param mem 1
p60-pdu # param set out_en[0] 1
```

```
p60-pdu # param get out_en[0]
  GET out_en[0] = 1
```

The updated parameter values are only valid until next reboot. To store a parameter permanently, the parameter table must be stored to its matching file number (listed in the table above). To make our change to the out_en value permanent, we then need to run `param save <table> <file>`:

```
p60-pdu # param save 1 1
Table CRC 58821
Data CRC 57628
```

The parameter system protects the stored table with two checksums: one to protect the data itself against corruption and one to ensure that the stored data matches the table structure.

System parameters in tables 0, 1 and 2 are saved in FRAM file numbers 0, 1 and 2 respectively.

The P60 PDU-200 also has a default factory setting version of table 0, 1 and 2. The default factory settings are stored in locked FRAM sectors and must be unlocked prior to saving the parameter table. This is done using the `config unlock` command. After e.g. the `param save 0 4` command has been executed, the FRAM sectors can be relocked by calling `config lock` or by rebooting the system.

There is a convenience function for saving the default configuration for each configuration table 0, 1 and 2 or for all of them at once:

config update_default 0 config update_default 1 config update_default 2 config update_default all

---

Note: As a safety measure, the `config` commands are only available through the GOSH interface and can not be executed via CSP. It is therefore not possible to unlock and modify the table 0 parameters remotely. This prevents accidental modification of e.g. the CSP address in orbit.

Telemetry data in table 4 is not saved to permanent storage and thus not backed by a file number or a default factory setting.

7.1 Table 0: Board Parameters

Table 0 holds system level configuration. Modification of these parameters require a reboot of the system, before they take effect.

Table 7.2: P60 PDU-200 Parameter table 0: Board parameters

Name	Address	Type	Index	Default Value	Unit	Comment
uid	0x0000	STR (16)		0123456789ABCD		Board UID
type	0x0010	U8		1		Board type
rev	0x0011	U8		0		Board revision
csp_addr	0x0012	U8		2		CSP address
i2c_addr	0x0013	U8		2		I^2C address
i2c_speed	0x0014	U16		400	kbps	I^2C bitrate in kbps
can_speed	0x0016	U16		1000	kbps	CAN bitrate in kbps
kiss_en	0x0018	U8		0		Enable KISS
rs422_mode	0x0019	U8		0		RS422 mode (not used)
rs422_speed	0x001C	U32		115200	kbps	RS422 bitrate in kbps (not used)
csp_rtable	0x0020	STR (96)		*Empty*		CSP routing table

7.2 Table 1: Configuration Parameters

Table 1 contains P60 PDU-200 runtime configuration parameters.

Table 7.3: P60 PDU-200 Parameter table 1: Configuration parameters

Name	Address	Type	Index	Default Value	Unit	Comment
out_name	0x0000	STR	[0..8]	“ch0” .. “ch8”		Output channel name (max 8 characters)
out_en	0x0048	U8	[0..8]	0 .. 0		Output channel enable (on/off)
out_on_cnt	0x0052	U16	[0..8]	0 .. 0	sec	Output channel on delay in seconds
out_off_cnt	0x0064	U16	[0..8]	0 .. 0	sec	Output channel off delay in seconds
init_out_norm	0x0076	U8	[0..8]	0 .. 0		Output channel on/off in battery normal mode
init_out_safe	0x0080	U8	[0..8]	0 .. 0		Output channel on/off in battery safe mode
init_on_dly	0x008A	U16	[0..8]	0 .. 0	sec	Output channel on initial (boot) delay
init_off_dly	0x009C	U16	[0..8]	0 .. 0	sec	Output channel off initial (boot) delay
safe_off_dly	0x00AE	U8	[0..8]	0 .. 0	sec	Output channel off delay when entering safe mode
cur_lu_lim	0x00B8	U16	[0..8]	2500 .. 2500	mA	Output channel instant latchup current limit (mA)
cur_lim	0x00CA	U16	[0..8]	2500 .. 2500	mA	Output channel ema latchup current limit (mA)
cur_ema	0x00DC	U16	[0..8]	0 .. 0	mA	Output channel latest current ema value (mA)
out_link	0x00EE	U8	[0..8]	0 .. 0		Output channel linked channel [0..8]
out_conv	0x00F8	U8	[0..8]	1 1 0 0 2 2 1 0 2		Output converter id [0..2]
out_voltage	0x0102	U16	[0..8]	0 .. 0	mV	Output converter voltage (mV)
conv_en	0x0114	U8	[0..2]	0 0 0		Output Converter enable
cur_ema_gain	0x0118	FLT		0.5		Output current ema gain
batt_hwmax	0x011C	U16		16000	mV	Battery HW max value (mV)
batt_max	0x011E	U16		9000	mV	Battery full mode threshold value (mV)
batt_norm	0x0120	U16		8000	mV	Battery normal mode threshold value (mV)
batt_safe	0x0122	U16		7000	mV	Battery safe mode threshold value (mV)
batt_crit	0x0124	U16		6000	mV	Battery critical mode threshold value (mV)
wdt_i2c_rst	0x0126	U8		0		Reboot if I^2C WDT expires (0/1)
wdt_can_rst	0x0127	U8		0		Reboot if CAN WDT expires (0/1)
wdt_i2c	0x0128	U32		0	sec	I^2C WDT timeout value in seconds
wdt_can	0x012C	U32		0	sec	CAN WDT timeout value in seconds
wdt_csp	0x0130	U32	[0..1]	10 60		CSP watchdog ping interval in seconds
wdt_csp_ping	0x0138	U8	[0..1]	5 5		CSP watchdog failed ping limit
wdt_csp_chan	0x013A	U8	[0..1]	0 0		Output channel to monitor via CSP
wdt_csp_addr	0x013C	U8	[0..1]	0 0		CSP address to ping

7.3 Table 2: Calibration Parameters

Table 2 contains P60 PDU-200 calibration parameters.

Table 7.4: P60 PDU-200 Parameter table 2: Calibration parameters

Name	Address	Type	Index	Default Value	Unit	Comment
vref	0x0000	U16		2500	mV	Reference voltage (mV)
gain_vcc	0x0004	FLT		3.2		Gain for VCC measurement
gain_vbat	0x0008	FLT		25.0		Gain for VBAT measurement
gain_v_out	0x000C	FLT	[0..8]	25.0 .. 25.0		Gain for output voltage measurement
gain_c_out	0x0030	FLT	[0..8]	1.0 .. 1.0		Gain for input current meas.
offs_c_out	0x0054	I16	[0..8]	0 .. 0		Offset for output current meas.

7.4 Table 4: Telemetry

This table contains P60 PDU-200 telemetry data. The values are automatically updated once per second by the housekeeping task. The `bootcnt`, `resetcause`, `wdt_cnt_i2c`, `wdt_cnt_can` and `wdt_cntcsp` parameters are stored to persistent memory.

Table 7.5: P60 PDU-200 Parameter table 4: Telemetry parameters

Name	Address	Type	Index	Default Value	Unit	Comment
c_out	0x0000	I16	[0..8]	0 .. 0	mA	Measured output current (mA)
v_out	0x0012	I16	[0..8]	0 .. 0	mV	Measured output voltage (mV)
vcc	0x0024	I16		3300	mV	Measured VCC voltage (mV)
vbat	0x0026	I16		0	mV	Measured VBAT voltage (mV)
temp	0x0028	I16		255	0.1 ℃	Measured temperature (0.1 ℃)
conv_en	0x002A	U8		0 0 0		Output converter enable status
out_en	0x002E	U8	[0..8]	0 .. 0		Output channel enable status
bootcause	0x0038	U32		0		Boot cause
bootcnt	0x003C	U32		0		Boot count
uptime	0x0040	U32		0	sec	Uptime in seconds
resetcause	0x0044	U16		0		Reset cause
batt_mode	0x0046	U8		3		Battery mode (1=Critical, 2=Safe, 3=Normal, 4=Full)
latchup	0x0048	U16	[0..8]	0 .. 0		Latchup (ema) count for output channel
device_type	0x005A	U8	[0..7]	0 .. 0		Device type (see below)
device_status	0x0062	U8	[0..7]	0 .. 0		Device status (0=None, 1=OK, 2=Error, 3=Not found)
wdt_cnt_gnd	0x006C	U32		0		Ground WDT reboots
wdt_cnt_i2c	0x0070	U32		0		I^2C WDT reboots
wdt_cnt_can	0x0074	U32		0		CAN WDT reboots
wdt_cnt_csp	0x0078	U32	[0..1]	0 0		CSP WDT reboots
wdt_gnd_left	0x0080	U32		172800	sec	Ground WDT value (remaining seconds before reboot)
wdt_i2c_left	0x0084	U32		0	sec	I^2C WDT value (remaining seconds before reboot)
wdt_can_left	0x0088	U32		0	sec	CAN WDT value (remaining seconds before reboot)
wdt_csp_left	0x008C	U8	[0..1]	5 5		CSP WDT value (remaining pings before power cycle)

The following table shows the device types for the P60 PDU-200:

Table 7.6: P60 PDU-200 Device type overview

Index	Type	Description
0	7	FRAM
1	2	ADC
2	2	ADC
3	4	Temperature sensor
4	0	Reserved
5	0	Reserved
6	0	Reserved
7	0	Reserved