GOMSPACE

User Manual

NanoPower P80

Configuration and Interface Description

MAN 1055446
September 23, 2024
Revision 1.1.0

User Manual NanoPower P80
Configuration and Interface Description

Document reference: MAN 1055446
Source reference: doc-nanopower-p80-user-manual
Date: September 23, 2024
Revision: 1.1.0

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

ACU Array Conditioning Unit.

CSP Cubesat Space Protocol.

EPS Electrical Power System.

GOSH GomSpace Shell.

LUP Latchup Protection.

MPPT Maximum Power Point Tracking.

P80 NanoPower P80.

PDU Power Distribution Unit.

PMU Power Management Unit.

1 Introduction

1.1 Overview

The NanoPower P80 is an Electrical Power System (EPS) for small satellites, based on GomSpaces earlier EPS, the NanoPower P60 system. The P80 consists of a Power Management Unit (PMU), Array Conditioning Unit (ACU) and a Power Distribution Unit (PDU), stacked on top of each other and fitted into an enclosure with mounting brackets that functions as shield and provides thermal dissipation. The unit fits in a standard PC104 form, but does not support the PC104 stack connector.

1.2 Highlighted features

- Power Management Unit (PMU)
  - EPS master
  - Handling of battery modes
  - Killswitch (KS) logic
  - Deploy device control
  - Four power modes depending on battery voltage

- Array Conditioning Unit (ACU)
  - 2x6 Maximum Power Point Tracking (MPPT) boost converters
  - KS/Remove Before Flight (RBF) inhibit switch
  - Software and hardware Latchup Protection (LUP)

- Power Distribution Unit (PDU)
  - 12 low voltage LUP channels, fed by 4 converters. All low voltage channels can be configured to an arbitrary converter in hardware.
  - 12 high voltage LUP channels - raw battery channels.
  - Handling of battery modes.
  - Software and hardware LUP.
  - Four power modes depending on battery voltage.

1.3 Functional description

The P80 is made up of three submodules; the PMU, PDU and ACU. The submodules are stacked on top of each-other, allowing power and communication to flow between the individual boards. An overview of the P80 system is illustrated in Figure 1.1.

NOTE: The battery (NanoPower BP8) is not part of the EPS, but a seperate product.

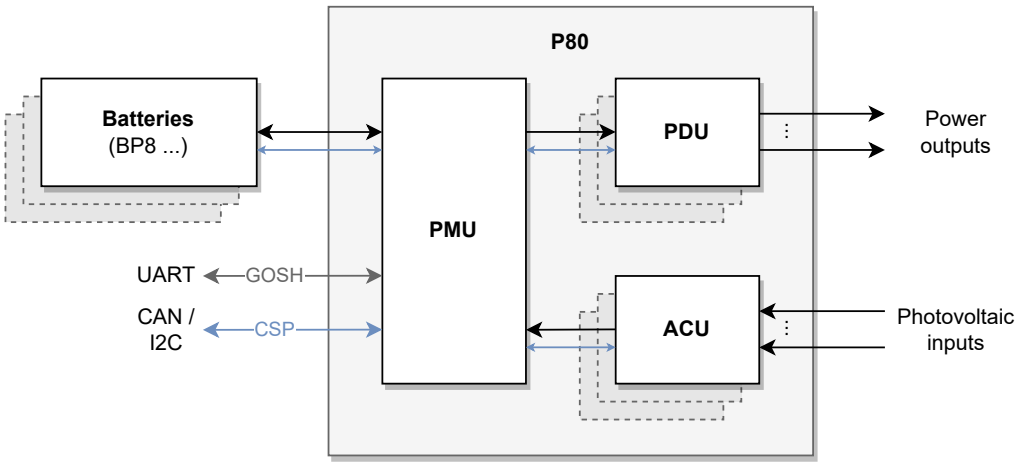

Figure 1.1: Block diagram of the NanoPower P80 system. The black arrows show the power paths, while the blue shows communication.

1.3.1 Unpacking and Handling

The NanoPower P80 (P80) EPS must be properly ESD grounded before connecting it to a subsystem. It is very important that the P80 EPS has the same ground potential as the system it connects to.

> ⚠ **WARNING:** The P80 EPS system employs components based on FETs and therefore requires anti-static handling precautions to be taken. 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 is 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!

1.4 Getting Started

This section describes how to easily connect to the P80 modules. Both the P80 PMU, P80 PDU and P80 ACU contain a debug and configuration interface called GOSH. This requires a PC and power supply along with the following items from the delivery:

- The P80 submodule itself (PMU, PDU or ACU)
- Debug adapter print
- Custom FTDI USB/Serial

The debug interface (GOSH) is accessed through the 13-pin combined programming and debugging connector. It has JTAG, GND, VCC and UART RX/TX. Use the supplied debug adapter to connect and attach the custom FTDI USB/SERIAL cable to the GOSH1 connector. The VCC pin is driven by the power supply and not connected in the cable.

> ⚠ **CAUTION:** Please ensure that any PC/Laptop connected to the USB cable has a properly grounded AC-plug. The external power supply ground and PC/Laptop ground must be the same.

1.4.1 Debug and Configuration Interface GOSH

GomSpace has developed a console-like interface called GomSpace Shell (GOSH), which provides a simple but extensive debug and configuration interface through UART. GOSH is a general feature present on several GomSpace products. The console provides a human readable text-interface via a serial connection.

The console on the P80 PMU, P80 PDU and P80 ACU is running at 500000 baud, 8n1, 3.3 V, TTL levels, no flow-control. The serial console can for instance be reached by using programs like 'minicom', 'tio' or 'putty'.

When connected correctly the prompt will update by the press of <Enter>, looking like this:

```
p80 #
p80 #
p80 #
```

The availabe commands are visible by pressing the <Tab> key.

The GomSpace Parameter System is also available via GOSH. The available Parameter Tables can be found by entering: 'param tableinfo'. The parameters can be listed by entering: 'param list <tablename>', for instance 'param list telemetry'. The parameter tables and their content is different for the P80 PMU, P80 PDU and P80 ACU modules.

2 P80 Interfaces

The P80 contain several electrical and software interfaces.

2.1 Electrical Interfaces

The P80 provides the following physical interfaces:

2.1.1 UART

The UART was briefly introduced in 1.4.1. The UART connection is not intented to be used for integration towards the satellite bus, but is used as Human-Machine-Interface to debug and configure the P80 units.

NOTE: The commands available in GOSH is to be used for configuration and interfacing during testing and integration. Use the CSP interface to integrate the product towards the mission software.

The UART interface is implemented as a 500.000 baud, 8n1, 3.3 V, TTL levels, no flow-control connection.

2.1.2 CAN

A CAN connection is available on the Main bus connector. The CAN connection complies to ISO 11898-2. The baudrate of the connection is default 1 MBaud.

A termination resistor of 120 Ohm between CAN_H and CAN_L is needed in order for the connection to funciton. It is possible to fit the P80 with a termination resistor as a part of the optionsheet configuration.

The CAN connection implements the CSP protocol and is used as the main interface towards the satellite bus.

2.1.3 i2c

An i2c connection is available at the Main Bus connector. This is used as an alternative or backup to the CAN connection as the main interface towards the satellite bus. The i2c connection is implemented as:

- SDA and SCL at 3.3 V
- Multimaster (for CSP)
- 7 bit addressing

A secondary i2c is available to interface the GomSpace Deployment Modules (AR6 devices). This is implemented as a Master-Slave configuration and is not intended to be used for other purposes than AR6 devices.

NOTE: The pinout and location details of the connections are available in the P80 Datasheet.

2.2 Software Interfaces

The P80 implements two software interfaces. Cubesat Space Protocol (CSP) and GOSH. The interfaces give access to the P80 software functionalities and commands described in section 4

2.2.1 GOSH

The GomSpace Shell (GOSH) is used to interface the P80 directly from a PC with a standard TTL UART cable, with nothing but a terminal needed on the PC.

The GOSH interface is used as a Human-Machine-Interface where the user can access the functionalities of the P80 system in order to operate and configure it. Functionalities which is available via CSP is typically also available via GOSH.

2.2.2 CSP

The CSP interface is used to implement the P80 into the satellite. The protocol is a small universal network layer delivery protocol similar to TCP/IP, just more suited for smaller networks without as much overhead. Most of the GomSpace subsystems utilizes this protocol. Refer to the Product Interface Application documentation for further information.

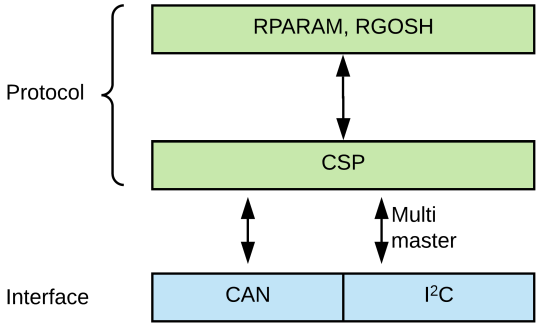

Figure 2.1: P80 Protocol Stack

On top of CSP there are two main protocols for controlling the P80 EPS: RGOSH and RPARAM.

The RGOSH protocol can be used to execute every GOSH command remotely, and is described in more detail in the Product Interface Application documentation.

The protocol RPARAM can be used to read, change and save parameters remotely, and is described in more detail in the Product Interface Application documentation.

For further information on implementation of RGOSH and RPARAM over CSP please refer to Product Interface Application documentation.

CSP Port Numbers

The P80 PMU, P80 PDU and P80 ACU listens on the following CSP port numbers:

Port	Name	Description
0	GS_CSP_CMP	Control Port
1	GS_CSP_PING	Returns a copy of the packet received
2	GS_CSP_PS	Returns process list
3	GS_CSP_MEMFREE	Returns memory free
4	GS_CSP_REBOOT	Reboots subsystem
5	GS_CSP_BUF_FREE	Returns number of free buffers
6	GS_CSP_UPTIME	Returns subsystem uptime
7	GS_CSP_PORT_RPARAM	Controls P80 PMU with parameter system
9	P80_PORT_GNDWDT_RESET	Used for ground watchdog reset
10	P80_PORT_CMDCONTROL	Used for power interface commands over CSP
12	GS_CSP_PORT_RGOSH	Remote Gosh interface
16	GS_CSP_PORT_GSSB	Remote gssb interface
22	GS_CSP_PORT_GSCRIPT	Gosh scripting interface

Details regarding the CSP ports, please see the Product Interface Application documentation.

Details regarding the Param ports, please see the Product Interface Application documentation.

2.3 Parameter System

The parameter system on the P80 EPS controls all of the functionalities. The P80 system uses the content of the parameter system to defermine configuration, setup and calibration, as well as to expose its telemetry parameters. It is common for the PMU, PDU and ACU that it consists of four different tables, each containing a number of variables. The tables are stored as different copies in different stores.

2.3.1 Stores

Each table is stored in different dynamic versions in different stores with different levels of accessibility.

Table	FRAM (Writable)	FRAM (Write Protected)	MCU Flash (Write Protected)
Board	Yes	Yes	Yes
Configuration	Yes	Yes	Yes
Calibration	Yes	Yes	Yes
Telemetry	No	No	No

Table 2.1: Table Store Matrix

The parameter system will upon boot load each table from the leftmost available store. This is either the persistent or protected 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 the default values defined at the software compile time without any specific configuration or calibration values.

For instance, if the writable version of the configuration table, stored in the FRAM, gets corrupted it will fall back to the write protected version, stored in the FRAM. If also this one fails, it falls back to the write protected version stored in the MCU flash. If this is also corrupt, the last fall back is to the hard coded values defined for the software when it was compiled.

NOTE: The write protected parameters should be changed only on ground.

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

Note, that some of the variables in telemetry table are also persistently stored, such as bootcount, but as they change dynamically there is no fall back versions.

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

Board Parameters

This table holds all critical configuration of the P80 EPS. It should be write protected before flight.

The default CSP address of the P80 PMU is configured to 1.

The default CSP address of the P80 ACU is configured to 2 and 3.

The default CSP address of the P80 PDU is configured to 4.

Configuration Parameters

This table holds run time configuration of the P80 EPS, which can be changed in orbit. These parameters can be saved persistent.

Calibration Parameters

This table holds the values used for calibration of the sensing circuitry. It should not be changed.

⚠ **WARNING:** The factory calibration will be lost by saving different calibration values.

Telemetry Parameters

This table holds all telemetry collected by the application. Only the parameters marked as auto persist is saved to persistent storage. The remaining parameters have no backing store, so they will not contain any meaningfull value right after reboot.

2.3.2 Get and Set Remote Parameters

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

An example on retrieving a single parameter from the Configuration Table on the P80 PMU is done by:

```
#include <p80_pmu.h>

/* Get enable status for Stack 3.3 V */
bool enable;
gs_error_t res = gs_rparam_get(p80_pmu_node,           /* P80 PMU csp node address */,
                   GS_P80_PMU_CONFIFURATION_TABLE_ID,  /* Parameter table id */
                   GS_P80_PMU_CONFIGURATION_OUT_EN(0), /* Logical address of parameter */
                   GS_PARAM_BOOL,                      /* Parameter type */
                   GS_RPARAM_MAGIC_CHECKSUM,           /* Table checksum */
                   1000,                               /* Timeout in milliseconds */
                   &enable,                            /* Pointer to variable */
                   sizeof(enable));                    /* Parameter size */
if (res == GS_OK) {
    printf("Get Stack 3.3 V enable: %u\n", enable);
}
```

An example on setting a single parameter in the Configuration Table on the P80 PMU is done by:

```
#include <p80_pmu.h>

/* Enable Stack 3.3 V */
bool enable = true;
gs_error_t res = gs_rparam_set(p80_pmu_node,           /* P80 PMU csp node address */,
                   GS_P80_PMU_CONFIFURATION_TABLE_ID,  /* Parameter table id */
                   GS_P80_PMU_CONFIGURATION_OUT_EN(0), /* Logical address of parameter */
                   GS_PARAM_BOOL,                      /* Parameter type */
                   GS_RPARAM_MAGIC_CHECKSUM,           /* Table checksum */
                   1000,                               /* Timeout in milliseconds */
                   &enable,                            /* Pointer to variable */
                   sizeof(enable));                    /* Parameter size */
if (res == GS_OK) {
    printf("Set Stack 3.3 V enable: %u\r\n", enable);
}
```

Getting or setting parameters in other parameter tables, or on the P80 ACU or P80 PDU, is done in a similar way according to the above examples.

2.3.3 Get and Set Local Parameters

Entering *param* in GOSH shows the following sub commands:

```
p80 # 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
```

The main functions to introduce are:

```
param select [table]         // Select working table. Arguments: Board, Config, Calib, Telem
param list                   // List parameters and values of the selected table
param set [param] [value(s)] // Set parameter value
param get [param]            // Get parameter value
param save [table]           // Save table
```

With these commands it is possible to control the P80.

2.4 Software Client

The client API consists of a set of wrapper functions that simplify the CSP interface to the P80 submodules. These functions are implemented in the *p80_pmu_client.c*, *p80_pdu_client.c* and *p80_acu_client.c* files and can be integrated in custom code by including the *p80_pmu.h*, *p80_pdu.h* or *p80_acu.h* header files.

The files ending with **_cmd.c* implements the GOSH commands for the submodules 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.

3 Client Interface

3.0.1 Retrieving Housekeeping Parameters

Retrieving Housekeeping Parameters from a P80 module is provided by the *p80_pmu_get_hk*, *p80_pdu_get_hk* or *p80_acu_get_hk* functions:

```
gs_error_t p80_pmu_get_hk(gs_param_table_instance_t *tinst, uint8_t node, uint32_t timeout
)
```

This example retrieves housekeeping parameters from the P80 PMU. The housekeeping parameters are retrieved over CSP using the remote parameter system API provided by libparam. The function *p80_pmu_get_hk* is basically a wrapper around the remote parameter system API. The input parameter *tinst* 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 P80 PMU and the *timeout* must provide the timeout in milliseconds. An example is shown below.

```
 #include <p80_pmu.h>

uint8_t p80_pmu_node = 1;
GS_PARAM_TINST_VAR(node_hk);
gs_error_t err = p80_pmu_get_hk(node_hk, p80_pmu_node, 1000);
if (err) {
    return err;
}
gs_param_list(node_hk, 1);
```

Housekeeping parameters can also be retrieved directly from a P80 submodule using the remote parameter system API provided by libparam:

```
#include <p80_pmu.h>

uint8_t p80_pmu_node = 1;
bool out_en;
gs_error_t res = gs_rparam_get(p80_pmu_node,       /* P80 PMU csp node address */,
                  GS_P80_PMU_TELEMETRY_TABLE_ID,  /* Parameter table id */
                  GS_P80_PMU_TELEMETRY_OUT_EN(0), /* Logical address of parameter */
                  GS_PARAM_BOOL,                  /* Parameter type */
                  GS_RPARAM_MAGIC_CHECKSUM,       /* Table checksum */
                  1000,                           /* Timeout in milliseconds */
                  &enable,                        /* Pointer to variable */
                  sizeof(enable));                /* Parameter size */
if (res == GS_OK) {
    printf("Get out_en[0]: %u\n", out\_en);
}
```

The above example retrieves a single parameter from the P80 PMU.

3.0.2 Reset Ground Watchdog Timer

The Ground Watchdog Timer is reset by a dedicated command to the P80 PMU. This can for example be used as a ground communication watchdog, i.e. this command is issued to P80 PMU on each connection with the ground station. If no communication has been received for a period of 48 hours the P80 PMU will switch off and do a reset back to default configuration. More details on Watchdog can be found in section 4.5

The Ground Watchdog Timer on the P80 PMU is reset by sending a single byte with the value 0x78 to CSP port *P80_PORT_GNDWDT_RESET*.

The following code will reset the Ground Watchdog Timer on the P80 PMU:

```
#include <p80_pmu.h>

uint8_t p80_pmu_node = 1;
gs_error_t err = p80_pmu_gndwdt_clear(p80_pmu_node, 1000);
```

Similar Ground Watch Dog functions is implemented on the P80 PDU and P80 ACU. The P80 PMU can be configured to propagate the Watch Dog Reset to these modules, see section 4.5.1

3.0.3 P80 Power Command

The output channels on the P80 PDU and P80 PMU can be controlled locally or remotely by using the p80_power command. The p80_power commands are implemented in GOSH on the PMU, PDU and ACU, and can be included in custom mission software by using the libp80_client software.

The p80_power command line interface has a few parameters that needs to be defined. By default the parameters are set as listed below, but can be changed by using the *p80_power port* and *p80_power timeout* commands.

- port is set to 10
- timeout is set to 5000 ms

NOTE: All of the *p80_power* commands run over loopback CSP using GOSH command wrappers around the client interface.

p80_power status <node> <channel> Can be used to fetch the status of a specific output channel. The *<node>* corresponds to the CSP Address of the module. The *<channel>* is the output channel of the module (0-5 for PMU, 0-23 for PDU).

```
p80 # p80_power status 1 0
Node 1, Output channel 'st_3v3' (0) is ON
  ch_idx:     0
  mode:       1
  on_cnt:     0
  off_cnt:    0
  cur_lu_lim: 2500
  cur_lim:    2500
  voltage:    3277
  current:    47
  latchup:    0
```

p80_power on <node> <channel> [<on_cnt> <off_cnt>] Can be used to turn a specific channel on.

```
p80 # p80_power on 1 1
Node 1, Output channel 'st_5v' (1) is ON
  ch_idx:     1
  mode:       1
  on_cnt:     0
  off_cnt:    0
  cur_lu_lim: 2500
  cur_lim:    2500
  voltage:    5011
  current:    16
  latchup:    0
```

p80_power off <node> <channel> [<on_cnt> <off_cnt>] Can be used to turn a specific channel off.

```
p80 # p80_power off 1 1
Node 1, Output channel 'st_5v' (1) is OFF
  ch_idx:     1
  mode:       0
  on_cnt:     0
  off_cnt:    0
  cur_lu_lim: 2500
  cur_lim:    2500
  voltage:    5011
  current:    16
  latchup:    0
```

The *[<on_cnt> <off_cnt>]* are the delay in seconds before turning on or off a channel. For instance *[p80_power on 1 1 10 100]* will turn on node 1 channel 1 after 10 seconds and off after 100 seconds.

NOTE: The interval between issuing an automatic on and off (see above) should be greater than 5 seconds.

Finally *p80_power list* can be used to list the status of all power channels.

```
p80 # p80_power list 1
ch name     status
 0 st_3v3   ON
 1 st_5v    OFF
 2 g1_3v3   OFF
 3 g1_vbat  OFF
 4 g2_3v3   OFF
 5 g2_vbat  OFF
```

4 Configuration and Operation

The P80 EPS can via the GOSH or CSP interface be configured and operated via the parameter system and specific client software.

This chapter describes how to configure the P80 EPS before flight, and how to operate the P80 EPS in flight.

The P80 EPS handles the following:

- Power state handling of attached submodules.
- Power state handling of attached batteries.
- Optimal charging control of solar panel inputs.
- Watchdogs for attached submodules and batteries.
- Latchup protection of output channels.
- Telemetry sampling.
- Handling of attached release devices.

In the following chapter the commands are mainly shown with examples in GOSH, but can also used with the Client or Param interface via CSP.

4.1 P80 Setup

The P80 has some generic functions, besides handling power and watchdogs, which is described in the following section.

4.1.1 Submodules

The P80 must contain a single PMU. This PMU has support for controlling the state of up to 8 submodules. These can either be ACU's or PDU's. Each submodule can be configured to be turned on or off based on the battery state - like the power channels. There are no support for delayed switching.

The PDU and ACU is each configured in hardware to have an input enable signal amongst 6 options. These are the submodule ID's.

The mapping between submodule ID's and enable signals must be setup in the configuration. This is due to the fact that the ACU has 2 MCU's and as such are treated as two submodules, but are controlled by one enable signal. The parameter that controls this mapping is *sm_output[]* from the configuration table. A *-1* indicates no submodule configured. Setting the first three elements of *sm_output[]* to be *[0 0 1]* tells the PMU that submodule 0 uses enable signal 0, submodule 1 uses enable signal 0 (the two ACU blocks), and submodule 2 uses enable signal 1.

Depending on the battery states (see section 4.2) the submodules can be turned on or off. The parameters that controls the battery dependent output states are: *sm_init_norm[]* and *sm_init_safe[]*. When entering "normal" or "full" battery mode (upon startup or from "safe" battery mode) the submodules defined in *sm_init_norm[]* will be enabled. Similar behaviour ocurs for *sm_init_safe[]* but with the battery being in "safe" mode.

A submodule can be controlled manually by setting the corresponding *sm_en[]* parameter.

4.1.2 Automatic Deployment

The P80 PMU module has support for 12 AR6 compatible deployment devices.

To enable deployment, it is required to set a number of parameters. These are:

- *depl_en* Global switch to enable / disable deployment
- *depl_delay* Number of seconds before deployment procedure starts (starting from PMU boot complete)
- *ar6_addr[]* GSSB address of the deployment device
- *ar6_delay[]* Delay on top of depl_delay before sequence starts
- *ar6_burns[]* Number of burns to perform
- *ar6_burn_time[]* Time for each burn.

The P80 PMU will start the deploy sequence when *depl_en* is True and the system uptime is larger than *depl_delay*. Initially the power for the two GSSB busses are turned on (3.3 V and Vbatt for both GSSB1 and GSSB2).

Then, for all the devices configured in the ar6-parameters shown above, the sequence is as follows:

After a delay of *ar6_delay[]* seconds, the AR6 on the address of *ar6_addr[]* is commanded with a burn of *ar6_burn_time[]* seconds duration. This is done *ar6_burns[]* amount of times. Within the AR6 device there are two burn channels which will be used alternating for each commanded burn. The actual number of burn attempts are stored in the telemetry parameter *ar6_burn_try[]*.

NOTE: The release device will switch between the 2 burn channels for each burn command. It is recommended to configure *ar6_burns[]* to a value of 2 or more.

NOTE: It is adviced to configure an *ar6_delay[]* to make sure the release device has booted before being commanded.

The deployment status of each release device is available in the telemetry parameter *ar6_status[]*.

NOTE: It is advisable to reconfigure the *depl_en* to 0 when successful deployment is confirmed. This is to avoid repeating the deploy procedure when the P80 PMU restarts.

The deployment procedure is paused whenever the battery level becomes critical.

4.1.3 Status LEDs

The P80 PMU and the P80 PDU contain status LEDs. These are only visible with the shielding removed, and is hence not of interest for the end user.

The LEDs can be completely disabled by setting the parameter led_enable in the board table to false. This should be done as part of the preparation for flight procedure.

4.2 Battery Interface

4.2.1 Battery Pack

The P80 PMU must be configured with the battery pack type. In the Configuration table, the parameter *batt_type[]* must be set with the appropriate value for the battery pack. To configure a NanoPower BP8 battery pack, the value must be set to 8. For any other option the values should be set to 0.

If the CSP watchdog feature for the battery pack is wanted then *batt_addr[]* must be configured with the CSP address of the battery pack. See section 4.5 for configuration of the watchdog.

> **NOTE:** The enabling/disabling of battery packs only affects the digital part of the battery - e.g. telemetry and heater control. The power path is not affected

Battery packs can be enabled or disabled by setting the configuration parameter *batt_en[]*.

4.2.2 Battery Voltage Level

The P80 PMU and P80 PDU module has a software implemented 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.

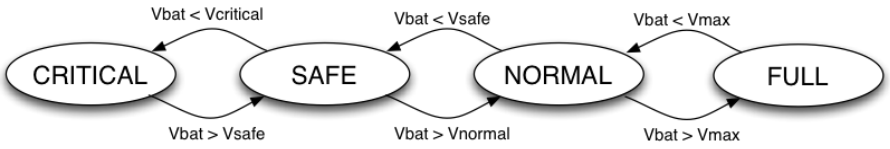

Figure 4.1: P80 Battery States

In normal mode everything is nominal but should the battery voltage drop below batt_safe, the P80 PMU and P80 PDU modules will change its output switch configuration to a safe mode configuration. This allows the modules to switch off all non-essential systems and leave a simple low power mode running.

the P80 PMU and P80 PDU modules will switch off all channels, should the battery voltage continue to drop below *batt_crit*.

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 must be set accordingly to ensure that outputs are turned on/off as required by mission.

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

The battery voltage measured by the P80 PMU is provided in the parameter *batt_v*, the battery mode is provided in the parameter *batt_mode*. The battery current is also measured and is provided in the parameter *batt_i*, all located in the Telemetry table.

Battery Pack Voltage	32 V
batt_max	32000 mV
batt_norm	28000 mV
batt_safe	26800 mV
batt_crit	21000 mV

Table 4.1: Recommended Settings for Nanopower BP8 Battery Levels

4.3 Power Input Management

4.3.1 Solar Panel Input Management

The P80 ACU Power Point Tracking can either be Fixed or MPPT mode. This is controlled by the parameter in the Configuration table *mppt_mode*. To use Fixed mode, set *mppt_mode* to 2. To use Tracking mode, set *mppt_mode* to 1.

Each side of the P80 ACU has 6 input channels. Hence there is a total of 12 channels for the P80 ACU module.

For each input channel, the nominal power point voltage level used for charging the battery can be set by the parameter *pv_setpoint[]*.

In Fixed mode, the voltage is set to the value of parameter *pv_setpoint[]* and is not changed, except if battery charge is completed.

In Tracking mode, the voltage is initialized to the value of parameter *pv_setpoint[]*. A control loop uptimizes the power input by regulating the input voltage. It is possible to limit how much the voltage is changed from the initial *pv_setpoint* value. This is controlled by the parameter *max_dv*. E.g. if the *pv_setpoint[]* is set to 10000 mV and the *max_dv* is set to 1000 mV, the minimum voltage allowed will be 9000 mV and the maximum will be 11000 mV.

In both modes, upper limit for *pv_setpoint[]* is 25000 mV.

4.3.2 Charging Control

When the battery is almost fully charged, the charging will be regulated to avoid over-charging the battery. This is controlled by two parameters, *vbat_max_hi* and *vbat_max_lo*. When the battery voltage exceeds *vbat_max_lo*, the P80 ACU will regulate the voltage down slowly to reduce the charging power. If the charging continues, even with reduced voltage, and the *vbat_max_hi* limit is reached, the charging will be reduced to a minimum. Charging will not be resumed until the battery voltage drops below *vbat_max_lo*. The following table lists the recommended values for *vbat_max_hi* and *vbat_max_lo*.

Battery Pack Voltage	32 V
vbat_max_hi	33200 mV
vbat_max_lo	32000 mV

Table 4.2: Recommended settings for *vbat_max_hi* and*vbat_max_lo*

When the battery voltage is above *vbat_max_hi*, the charging is minimised by setting the pv_setpoint voltage level to the minimum value.

If the battery is charged further, the NanoPower BP8 will disconnect the battery charge input as a final protection method until the voltage is within safe operation again.

4.3.3 Hardware protection

Both the P80 ACU and the NanoPower BP8 has hardware protection circuits to protect the battery. The P80 ACU takes this into account when controlling the charging of the battery. Charging will be stopped if the battery voltage exceeds the hardware protection high threshold and will not resume until the battery voltage level has dropped below the hardware protection low threshold. For security reasons, the hardware protection low and high threshold are not configurable, but are listed here for reference:

Battery Pack Voltage	**32 V**
High Threshold	34600 mV
Low Threshold	30500 mV

Table 4.3: Hardware protection thresholds for battery voltage

4.4 Power Output Management

4.4.1 Manual Output Channel Control

The P80 PMU has a total of 6 latchup protected output channels and the P80 PDU has a total of 24 latchup protected output channels. First 12 channels (0-11) are high voltage outputs (full battery voltage), the remaining channels (12-23) are low voltage outputs delivering one of the voltages from the converters. The converter voltages are decided in hardware configuration via option sheet.

These can all be turned on and off individually.

Table 4.4: PMU and PDU output channels

No.	**Default Name**	**Default State**	**Channel Type**
PMU 0	st_3v3	ON	-
PMU 1	st_5v	ON	-
PMU 2	g1_3v3	OFF	-
PMU 3	g1_vbat	OFF	-
PMU 4	g2_3v3	OFF	-
PMU 5	g3_vbat	OFF	-
-	-	-	-
PDU 0	ch00	OFF	High Voltage
PDU 1	ch01	OFF	High Voltage
PDU 2	ch02	OFF	High Voltage

continued on next page ...

Table 4.4 continued: PMU and PDU output channels

No.	Default Name	Default State	Channel Type
PDU 3	ch03	OFF	High Voltage
PDU 4	ch04	OFF	High Voltage
PDU 5	ch05	OFF	High Voltage
PDU 6	ch06	OFF	High Voltage
PDU 7	ch07	OFF	High Voltage
PDU 8	ch08	OFF	High Voltage
PDU 9	ch09	OFF	High Voltage
PDU 10	ch10	OFF	High Voltage
PDU 11	ch11	OFF	High Voltage
PDU 12	ch12	OFF	Low Voltage
PDU 13	ch13	OFF	Low Voltage
PDU 14	ch14	OFF	Low Voltage
PDU 15	ch15	OFF	Low Voltage
PDU 16	ch16	OFF	Low Voltage
PDU 17	ch17	OFF	Low Voltage
PDU 18	ch18	OFF	Low Voltage
PDU 19	ch19	OFF	Low Voltage
PDU 20	ch20	OFF	Low Voltage
PDU 21	ch21	OFF	Low Voltage
PDU 22	ch22	OFF	Low Voltage
PDU 23	ch23	OFF	Low Voltage

Each channel can be configured with a name by the parameter *out_name* in the Configuration parameter table. Names can be no longer than 7 characters.

The ouptut channels can all be turned on and off manually and/or automatically. From factory the P80 PDU will have its output channels disabled. The P80 PMU will be configured to turn on the power channels in the following way:

- Stack 3.3V and Stack 5V are turned on automatically when the P80 PMU has completed boot sequence
- All other channels are not turned on automatically

The P80 PDU and PMU has parameters to control the power channels manually, they are located in the Configuration parameter table:

```
p80 # param list configuration
Table configuration (1):
  0x0000 out_name        STR "st_3v3" "st_5v" "g1_3v3" "g1_vbat" "g2_3v3" "g2_vbat"
  0x0030 out_en          BL  true true false false false false
```

```
0x0036 out_on_cnt      U16 0 0 0 0 0 0
0x0042 out_off_cnt     U16 0 0 0 0 0 0
0x004E out_init_norm   BL  true true false false false false
0x0054 out_init_safe   BL  true true false false false false
0x005A init_on_dly     U16 0 0 0 0 0 0
0x0066 init_off_dly    U16 0 0 0 0 0 0
0x0072 cur_lu_lim      U16 2500 2500 2500 2500 2500 2500
0x007E cur_lim         U16 2500 2500 2500 2500 2500 2500
0x008A out_lu_dly      U16 5 5 5 5 5 5
[...P80 PMU TABLE TRUNCATED...]
```

To turn a channel on or off manually, set the appropriate *out_en[]* parameter.The array-index corresponds to the channel number (the array is indexed from 0).

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.

For the P80 PDU, it is ensured that there will be a 20 ms delay between each channel when setting multiple channels on at the same time.

Before a channel is turned on, it is verified that the associated converter is on - if not it will automatically turn on. Then there is a 20 ms delay before the channel is turned on.

When turning a channel off, it is checked if the associated converter is no longer used by other channels. If no one is using the converter, it is turned off immediately.

The output status of each output channel is available by listing the *out_en[]* parameter in the Telemetry parameter table. The measured voltage and current for each output channel is available in the parameters *out_v[]* and *out_i[]* respectively from the Telemetry parameter table.

4.4.2 Automatic Output Control (Battery State Dependant)

The P80 PMU output channels can be turned on and off automatically depending on the battery state. Four states are defined: *Full*, *Normal*, *Safe* and *Critical*.

Battery mode FULL:
When transiting into battery mode FULL, all output channels are left unchanged.

Battery mode NORMAL:
To turn a channel on automatically without delay when battery mode becomes NORMAL, set the appropriate *out_init_norm[]* parameter to true.

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 *out_init_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. The channel

must be turned on before it can be turned off - in this case the *out_init_norm[]* parameter should be true 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 *out_init_safe[]* parameter to true 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.

For the P80 PDU 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.

> **NOTE:** If the output channels have been turned on/off manually by the paramters *out_en[]*, *out_on_cnt[]* or *out_off_cnt[]*, these are void after a battery mode change.

> **NOTE:** *out_on_cnt[]* is void after a latchup condition.

4.4.3 Output Channel Current Limitation

The channels on the P80 PDU 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 *out_lu_dly[]* seconds.

The parameter *cur_lim[]* is an averaged limit, meaning that if the channel over a period of time on average consumes more than this setpoint 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 *out_lu_dly[]* seconds.

In case of a hard- or averaged over-current event on an output channel, the Telemetry parameter *latchup[]* will increase by 1 each time an over-current is detected and the output channel is turned off by the current monitoring function.

> **NOTE:** The P80 PDU channels has a hardware implemented latchup protection which can not be circumvented.

4.4.4 Linking PDU Output Channels

Output channels in the P80 PDU can be linked together so that changing the state on one of them, will automatically change the state of the linked channel. Linking channels is performed by setting the parameter *out_link[]*. The value of this must be the channel number of the linked channel. This must be a two way link to work.

Linking channel 2 and 4 can be accomplished by setting *out_link[2]* to 4 and *out_link[4]* to 2.

4.5 Watchdogs

The P80 system utilizes different watchdog features in order to recover in case of a non-operational satellite. The different watchdogs are:

- Ground watchdog
- Bus watchdog
- Submodule watchdog
- Battery watchdog
- CSP watchdog

4.5.1 Ground Watchdog

The P80 modules all have a ground watchdog feature that needs to be reset at least every 48 hours if enabled. The purpose of the ground watchdog is to enable the P80 module to revert to the default configuration if for some reason the running configuration prevents it from operating, ie. if the power for the communication has been turned off unintentional.

If the ground watchdog is triggered, the P80 module will erase the **persistent** storage for all tables then do a reset of itself to reload configuration from the **protected** storage.

To enable the ground watchdog and set the timeout value, run the following command from the GOSH interface:

config gnd_wdt <timeout in seconds>

The minimum value is 172800 seconds (48 hours). The maximum value 7776000 seconds (90 days). To disable the ground watchdog, set the timeout to 0 seconds.

> **NOTE:** The P80 PMU will do a reset of all mounted submodules (PDU, ACU) if the ground watchdog is triggered.

Propagating the Ground Watchdog Reset
The P80 PMU has an optional feature that allows to propagate the ground watchdog reset message to all P80 submodules.

This is achieved by setting the *sm_addr[]* parameter to the CSP addresses of the P80 submodules, and to set *sm_gnd_wdt_en[]* to true E.g if the P80 PMU has two P80 ACU and a P80 PDU daughter boards with CSP addresses 2, 3 and 4 respectively, the *sm_addr[]* should be set to [2 3 4], and *sm_gnd_wdt_en[]* to [true true true]. When the ground watchdog is reset on the P80 PMU, the P80 PMU will forward the ground watchdog reset message to the two P80 ACUs and the P80 PDU.

Using the ground watchdog reset propagation, it is only necessary to send the ground watchdog reset message to the P80 PMU.

4.5.2 Bus Watchdog

The P80 modules has a satellite bus watchdogs that monitor the satellite bus for CSP traffic. If there hasn't been any traffic on the main bus for a period, it could indicate an error somewhere is inhibiting communication on the satellite bus. The bus watchdog has a configurable timeout that can be set with the parameter *bus_wdt*. The value set here is the number of seconds before the watchdog times out. A value of 0 will disable the watchdog. The P80 module can be configured to reboot if the satellite bus watchdog is triggered, this is controlled by the *bus_wdt_rst* parameter.

NOTE: The satellite bus watchdog does not run when the battery mode is CRITICAL.

4.5.3 Submodule Watchdogs

The P80 PMU module has a submodule watchdog for each submodule attached. This allows to actively monitor the attached submodules.

If a CSP address is configured for *sm_addr[]* and *sm_wdt_en[]* is set, the P80 PMU will actively monitor the connectivity to each of these systems with a CSP ping packet each *sm_wdt[]* seconds. If *sm_wdt_ping[]* consecutive pings are lost (maximum allowed response time is 30 ms), the corresponding submodule is power cycled with a 5 second off time.

The submodule watchdogs will only run if the configured submodule *sm_en[]* is on.

4.5.4 Battery Pack Watchdogs

The P80 PMU module has a battery pack watchdog for each battery pack attached. This allows to actively monitor the attached battery packs.

If a CSP address is configured for *batt_addr[]* the P80 PMU will actively monitor the connectivity to each of the battery packs with a ping packet each *batt_wdt[]* seconds. If *batt_wdt_ping[]* consecutive pings are lost (maximum allowed response time is 30 ms), the corresponding battery pack is power cycled with a 5 second off time.

The battery pack watchdogs will only run if the configured battery pack *batt_en[]* is on.

4.5.5 CSP Watchdog

The P80 PDU module has 12 CSP watchdogs that can be can be used to monitor the connectivity to attached devices.

If a CSP address is configured (> 0) for *csp_wdt_addr[]* the P80 PDU will actively monitor the connectivity to each of these systems with a CSP ping packet each *csp_wdt[]* seconds. If *csp_wdt_ping[]* consecutive pings are lost (maximum allowed response time is 30 ms), the corresponding power channel *csp_wdt_chan[]* is power cycled with a 5 second off time.

The CSP watchdogs will only run if the configured output channel is turned on.

4.6 Saving Protected Parameter Tables

To save the parameter tables in all available stores, the following procedure is to be followed:

- Make sure all tables are configured as wanted.
- Unlock the MCU FLASH storage by entering *param unlock flash*
- Unlock the protected FRAM storage by entering *param unlock protected* (note, that it automatically enables lock upon boot).
- Execute the GOSH command *config update_default all* to save all tables to all stores.
- Reboot

The above example is shown in the following console printout:

```
p80 # param unlock flash
p80 # param unlock  protected
p80 # config update_default all
Updated settings in FRAM for table 0, saving to file persistent: OK
Updated default factory settings in FRAM for table 0, saving to file protected: OK
Updated default factory settings in FLASH for table 0, saving to file flash: OK
Updated settings in FRAM for table 1, saving to file persistent: OK
Updated default factory settings in FRAM for table 1, saving to file protected: OK
Updated default factory settings in FLASH for table 1, saving to file flash: OK
Updated settings in FRAM for table 2, saving to file persistent: OK
Updated default factory settings in FRAM for table 2, saving to file protected: OK
Updated default factory settings in FLASH for table 2, saving to file flash: OK
p80 # 0000.002540 N default: boot cause: power on (2), platform boot cause: 1, reset
cause: unknown (0)
```

5 Application Examples

This chapter provides examples on the use of the NanoPower P80 system

5.1 Configuring Submodules [PMU]

From section 4.1.

The following submodules configuration is used in this example:

- PMU: CSP Address 1
- ACU: CSP Address 2 & 3 - Enable Signal 0
- PDU1: CSP Address 4 - Enable Signal 1
- PDU2: CSP Address 5 - Enable Signal 2

The 3.3 V and 5 V supply from the PMU must be turned on. By default they are.

The ACU uses the same Enable Signal for both parts of the ACU (Addr 2 and 3).

The array index on the parameter *sm_output* corresponds to the Submodule ID. Here we will configure *sm_output* to have ACU (part one) at ID0, ACU (part two) at ID1, PDU1 as ID2 and PDU2 as ID3:

```
p80 # param select configuration
p80 # param set sm_output[0 0 1 2 -1 -1]
```

To use the submodule watchdog feature, the submodule addresses needs to be configured. Enter the CSP address of the submodule at the corresponding ID array index:

```
  p80 # param set sm_addr[2 3 4 5 0 0]
```

5.2 Setting MPPT Mode [ACU]

From section 4.3.

From the P80 ACU configuration table:

```
p80 # param list configuration
  Table configuration (1):
  0x0000 mppt_mode       U8  2
  0x0002 pv_setpoint     U16 10000 10000 10000 10000 10000 10000
  0x000E vbat_max_hi     U16 33600
  0x0010 vbat_max_lo     U16 32000
  0x0014 mppt_period     U32 50
  0x0018 mppt_dv_max     U16 1000
```

To change the Maximum Power Point Tracking mode from 'Tracking' (1) to 'Fixed' (2), set the *mppt_mode* to 2:

```
p80 # param set mppt_mode 2
```

5.3 Changing Solar Panel Power Point [ACU]

From section 4.3.

In Fixed mode the Power Point is set to the value of *pv_setpoint[]*. For instance, changing the setpoint on channel 3 to 9500 mV is done by:

```
p80 # param select configuration
p80 # param set pv_setpoint[3] 9500
```

In Tracking mode the MPPT center-point is configured by *pv_setpoint[]* and the deviation is set by *mppt_dv_max*. For instance, changing channel 4 to have a center-point in 11200 mV and a deviation of 1500 mV (to each side) is done by:

```
p80 # param select configuration
p80 # param set pv_setpoint[4] 11200
p80 # param set mppt_dv_max 1500
```

5.4 Enabling an output channel [PMU, PDU]

The output channels can be configured from the Configuration table:

```
p80 # param list configuration
Table configuration (1):
  0x0000 out_name        STR "st_3v3" "st_5v" "g1_3v3" "g1_vbat" "g2_3v3" "g2_vbat"
  0x0030 out_en          BL  true true false false false false
  0x0036 out_on_cnt      U16 0 0 0 0 0 0
  0x0042 out_off_cnt     U16 0 0 0 0 0 0
  0x004E out_init_norm   BL  true true false false false false
  0x0054 out_init_safe   BL  true true false false false false
  0x005A init_on_dly     U16 0 0 0 0 0 0
  0x0066 init_off_dly    U16 0 0 0 0 0 0
  0x0072 cur_lu_lim      U16 2500 2500 2500 2500 2500 2500
  0x007E cur_lim         U16 2500 2500 2500 2500 2500 2500
  0x008A out_lu_dly      U16 5 5 5 5 5 5
  0x0098 cur_ema_gain    FLT 0.500000
  [...TRUNCATED...]
```

Here the P80 PMU configuration table is shown. The principle is the same for the P80 PDU.

Enter *param set out_en[2] true* to change the *out_en* parameter to true. This enables output channel 2

Entering *param get out_en[2]* shows the value of the parameter.

```
p80 # param get out_en[2]
  out_en[2] = true
```

5.5 Linking Output Channels [PDU]

From section 4.4.4.

Output channels can be configured to follow each other. The number of the linked channel has to be entered in *out_link[]* on the paired channels index. For instance:

In order to pair channel 0 and channel 1 as well as channel 5 and channel 6, the *out_link[]* parameter must be set as *out_link[**1 0** 0 0 0 **6 5** 0 (...)]*

5.6 Current Limitation [PDU]

From section 4.4.3.

Current limitation on output channels is done with the *cur_lim* and *cur_la_lim* parameters in the configuration table. For instance, in order to configure channel 8 to have a filtered current limitation at 1500 mA and a instantanious current limit at 2000 mA, the following parameters are set:

```
p80 # param select configuration
p80 # param set cur_lim[8] 1500
p80 # param set cur_la_lim[8] 2000
```

5.7 Configuring Watchdogs

From section 4.5.

5.7.1 Ground Watchdog

To enable the ground watchdog, use the *config gnd_wdt* command GOSH terminal. To set the ground watchdog to expire in 2 days:

```
p80 # config gnd_wdt 172800
```

Minimum value is 172800 seconds, maximum is 777600 seconds. To disable the GND WDT, set it to a value of 0.

5.7.2 Bus Watchdog

To enable the bus watchdog feature, configure the *bus_wdt* from the configuration parameter table with a value equal to the desired timeout in seconds. For instance:

```
p80 # param select configuration
p80 # param set bus_wdt 1800
```

If the module should reset itself when the watchdog is triggered, set the parameter *bus_wdt_rst* to *True*.

5.7.3 Submodule Watchdog

An example on configuring the parameters *sm_addr[]* and *sm_output[]* is given in section 5.1.

To enable the submodule watchdog feature, set the parameter *sm_wdt_en[]* to *True*.

The parameters *sm_wdt[]* and *sm_wdt_ping[]* can be left as default to have a ping interval (*sm_wdt[]*) of 60 seconds and 5 failed pings (*sm_wdt_ping[]*) in a row before issuing a reset.

5.7.4 Battery Watchdog

The battery packs are connected to the PMU with individual enable signals. The CSP addresses of the battery packs should be configured in the parameter *batt_addr[]*, with the array index corresponding to the Battery Enable signal ID (0-3). For instance a single battery pack, CSP address 10, Battery Enable signal 0 becomes:

```
p80 # param set batt_addr[10 0 0 0]
```

Every *batt_wdt[]* seconds the battery pack will recieve a CSP ping. If the ping times out *batt_wdt_ping[]*-amount of times the battery pack will be reset.

5.7.5 CSP Watchdog

As an example the CSP watchdog is configured to monitor the response of three CSP systems on PDU output channel 2, 14 and 18 with CSP address 14, 15, 16 respectively:

```
p80 # param set csp_wdt_addr[14 15 16 0 0 0 0 0 0 0 0 0]
p80 # param set csp_wdt_chan[2 14 18 0 0 0 0 0 0 0 0 0]
```

The parameters *csp_wdt[]* and *csp_wdt_ping[]* has been left default, to have an interval of 60 seconds between pings and 5 timeouts before a reset is issued.

6 Parameters

6.1 P80 PMU

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

6.1.1 Configuration

Table 6.1: Parameter table 1: 'configuration'.

Name	Address	Type	Description
out_name	0x0	string[6]	Output channel name. **Max string length:** 8 **Default:** ['st_3v3', 'st_5v', 'g1_3v3', 'g1_vbat', 'g2_3v3', 'g2_vbat']
out_en	0x30	bool[6]	Output channel enable **Default:** [False, False, False, False, False, False]
out_on_cnt	0x36	uint16[6]	Output channel on delay **Unit:** sec **Default:** [0, 0, 0, 0, 0, 0]
out_off_cnt	0x42	uint16[6]	Output channel off delay **Unit:** sec **Default:** [0, 0, 0, 0, 0, 0]
out_init_norm	0x4e	bool[6]	Output channel enabled in battery normal mode **Default:** [True, True, False, False, False, False]
out_init_safe	0x54	bool[6]	Output channel enabled in battery safe mode **Default:** [True, True, False, False, False, False]
init_on_dly	0x5a	uint16[6]	Output channel on initial delay **Unit:** sec **Default:** [0, 0, 0, 0, 0, 0]
init_off_dly	0x66	uint16[6]	Output channel off initial delay **Unit:** sec **Default:** [0, 0, 0, 0, 0, 0]
cur_lu_lim	0x72	uint16[6]	Output channel instant latchup current limit **Unit:** mA **Default:** [2500, 2500, 400, 400, 400, 400]

Continued on next page

Table 6.1: Parameter table 1: 'configuration'. (Continued)

Name	Address	Type	Description
cur_lim	0x7e	uint16[6]	Output channel ema (exponential moving average) latchup current limit **Unit:** mA **Default:** [2500, 2500, 400, 400, 400, 400]
out_lu_dly	0x8a	uint16[6]	Time before re-enabling output in case of latchup **Unit:** sec **Default:** [5, 5, 5, 5, 5, 5]
cur_ema_gain	0x98	float	Output current ema (exponential moving average) gain. **Default:** 0.5
batt_type	0x9c	uint8[4]	Battery pack type **Default:** [8, 8, 8, 8] **Valid values:** '0': None '8': BP8
batt_en	0xa0	bool[4]	Battery pack enable **Default:** [True, True, True, True]
batt_addr	0xa4	uint8[4]	CSP address of the battery pack (used battery watchdog) **Valid range:** 0 <= batt_addr <= 31
batt_wdt	0xa8	uint32[4]	Battery watchdog ping interval in seconds (0 disables the watchdog) **Unit:** sec **Default:** [60, 60, 60, 60]
batt_wdt_ping	0xb8	uint8[4]	Battery watchdog failed ping limit **Default:** [5, 5, 5, 5]
batt_hwmax	0xbc	uint16	Battery HW max value **Unit:** mV **Default:** 33200
batt_max	0xbe	uint16	Battery full threshold **Unit:** mV **Default:** 32000

Continued on next page

Table 6.1: Parameter table 1: 'configuration'. (Continued)

Name	Address	Type	Description
batt_norm	0xc0	uint16	Battery normal mode threshold **Unit:** mV **Default:** 28000
batt_safe	0xc2	uint16	Battery safe mode threshold **Unit:** mV **Default:** 26000
batt_crit	0xc4	uint16	Battery critical mode threshold **Unit:** mV **Default:** 21000
bus_wdt_rst	0xc6	bool	Reboot if BUS watchdog expires **Default:** False
bus_wdt	0xc8	uint32	BUS watchdog timeout value **Unit:** sec **Default:** 0
sm_wdt	0xcc	uint32[8]	Submodule watchdog ping interval **Unit:** sec **Default:** [60, 60, 60, 60, 60, 60, 60, 60]
sm_wdt_ping	0xec	uint8[8]	Submodule watchdog failed ping limit **Default:** [5, 5, 5, 5, 5, 5, 5, 5]
sm_name	0xf4	string[8]	Name of attached submodule **Max string length:** 8 **Default:** ['acu_l', 'acu_r', 'pdu', '', '', '', '', '']
sm_addr	0x134	uint8[8]	CSP address of submodule (used for GND watchdog propagation & watchdog) **Default:** [0, 0, 0, 0, 0, 0, 0, 0] **Valid range:** 0 <= sm_addr <= 31
sm_output	0x13c	int8[8]	Control output of submodule, -1 to disable **Default:** [0, 0, 1, -1, -1, -1, -1, -1] **Valid range:** -1 <= sm_output <= 6
sm_wdt_en	0x144	bool[8]	Enable submodule watchdog **Default:** [False, False, False, False, False, False, False, False]

Continued on next page

Table 6.1: Parameter table 1: 'configuration'. (Continued)

Name	Address	Type	Description
sm_gnd_wdt_en	0x14c	bool[8]	Enable ground watchdog propagation to submodule **Default:** [False, False, False, False, False, False, False, False]
sm_init_norm	0x154	bool[8]	Submodule enabled in battery normal mode **Default:** [True, True, True, False, False, False, False, False]
sm_init_safe	0x15c	bool[8]	Submodule enabled in battery safe mode **Default:** [True, True, True, False, False, False, False, False]
sm_en	0x164	bool[8]	submodule enable **Default:** [False, False, False, False, False, False, False, False]
conv_5v_en	0x16c	bool	Enable 5 V converter manually (for test) **Default:** False
depl_en	0x16d	bool	Enable deployment **Default:** False
depl_delay	0x170	uint32	Deployment delay after boot **Unit:** seconds **Default:** 0
ar6_addr	0x174	uint8[12]	Address of the AR6 deploy devices **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
ar6_burns	0x180	uint8[12]	Number of burns **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
ar6_burn_time	0x18c	uint8[12]	Duration of each burn **Unit:** seconds **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
ar6_delay	0x198	uint32[12]	Delay on top depl_delay before starting burn sequence **Unit:** seconds **Default:** [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]

6.1.2 Calibration

Table 6.2: Parameter table 2: 'calibration'.

Name	Address	Type	Description
out_v_gain	0x0	float[6]	Gain for output voltage measurement. **Default:** [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
out_i_gain	0x18	float[6]	Gain for output current measurement. **Default:** [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
out_i_offs	0x30	int16[6]	Offset for output current measurement. **Unit:** mA
vref	0x3c	uint16	Reference voltage. **Unit:** mV **Default:** 2500
vbat_v_gain	0x40	float	Gain for VBAT PMU Voltage measurement. **Default:** 1.0
vbat_i_gain	0x44	float	Gain for VBAT PMU current measurement. **Default:** 1.0
vbat_i_offs	0x48	int16	Offset for VBAT PMU current measurement. **Unit:** mA **Default:** 0
vcc_v_gain	0x4c	float	Gain for VCC PMU Voltage measurement. **Default:** 1.0
vcc_i_gain	0x50	float	Gain for VCC PMU current measurement. **Default:** 1.0
vcc_i_offs	0x54	int16	Offset for VCC PMU current measurement. **Unit:** mA **Default:** 0
batt_v_gain	0x58	float	Gain for battery voltage measurement. **Default:** 1.0
batt_i_gain	0x5c	float	Gain for battery current measurement. **Default:** 1.0
batt_i_offs	0x60	int16	Offset for battery current measurement. **Unit:** mA **Default:** 0

6.1.3 Telemetry

Table 6.3: Parameter table 4: 'telemetry'.

Name	Address	Type	Description
uptime	0x0	uint32	How long the unit has been running, since last reset/boot **Unit:** seconds
bootcause	0x4	uint32	Boot cause (from MCU, see below) **Valid values:** '0': Unknown '1': Brownout '2': Power on '3': Watchdog '4': Software '5': External reset pin '6': Wake from sleep '7': CPU error (exception) '8': JTAG
resetcause	0x8	uint16	Cause of last reset (which watchdog triggered the reset) **Valid values:** '0': Unknown '1': BUS watchdog '2': Ground watchdog '3': Stack overflow '4': Exception '5': Gosh command '6': CSP request '8': Out of memory
bootcount	0xa	uint16	Number of times the unit has booted
out_i	0xc	int16[6]	Measured output current **Unit:** mA
out_v	0x18	uint16[6]	Measured output voltage **Unit:** mV
out_en	0x24	bool[6]	Output channel enable status
temp	0x2a	int16[2]	Measured temperature **Unit:** decidegC

Continued on next page

Table 6.3: Parameter table 4: 'telemetry'. (Continued)

Name	Address	Type	Description
batt_mode	0x2e	uint8	Battery mode **Valid values:** '1': Critical '2': Safe '3': Normal '4': Full
conv_5v_en	0x2f	bool	5V converter enabled status
latchup	0x30	uint16[6]	Latchup count for output channel
cur_ema	0x3c	uint16[6]	Output channel ema (exponential moving average) current **Unit:** mA
vbat_v	0x48	uint16	PMU VBAT voltage **Unit:** mV
vbat_i	0x4a	int16	PMU VBAT current **Unit:** mA
vcc_v	0x4c	uint16	PMU VCC voltage **Unit:** mV
vcc_i	0x4e	int16	PMU VCC current **Unit:** mA
batt_i	0x50	int16	Battery current **Unit:** mA
batt_v	0x52	uint16	Battery voltage **Unit:** mV
sm_en	0x54	bool[8]	Enable status of submodule
batt_en	0x5c	bool[4]	Enable status of battery pack

Continued on next page

Table 6.3: Parameter table 4: 'telemetry'. (Continued)

Name	Address	Type	Description
device_type	0x60	uint8[8]	Device type **Valid values:** '0': N/A '1': ADC_ADS1118 '2': ADC_ADS7952 '4': Temperature sensor '6': RTC '4': FRAM
device_status	0x68	uint8[8]	Device status **Valid values:** '0': No device '1': OK '2': Error '3': Not found
dep_inhbt	0x70	bool	True if deployment is inhibited by HW
gnd_wdt_cnt	0x72	uint16	Ground watchdog reboots
bus_wdt_cnt	0x74	uint16	BUS watchdog reboots
sm_wdt_cnt	0x76	uint16[8]	Submodule watchdog reboots
batt_wdt_cnt	0x86	uint16[4]	Battery watchdog reboots
gnd_wdt_left	0x90	uint32	Ground watchdog value (remaining seconds before reboot) **Unit:** sec
bus_wdt_left	0x94	uint32	BUS watchdog value (remaining seconds before reboot) **Unit:** sec
sm_wdt_left	0x98	uint8[8]	Submodule watchdog value (remaining pings before power cycle)
batt_wdt_left	0xa0	uint8[4]	Battery watchdog value (remaining pings before power cycle)

Continued on next page

Table 6.3: Parameter table 4: 'telemetry'. (Continued)

Name	Address	Type	Description
ar6_status	0xa4	int32[12]	State of the AR6 deployment **Valid values:** '0': Not released '1': Released '2': Not configured '-5': IO error
ar6_burn_try	0xd4	int8[12]	Number of attempted burns

6.2 P80 PDU

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

6.2.1 Configuration

Table 6.4: Parameter table 1: 'configuration'.

Name	Address	Type	Description
out_name1	0x0	string[12]	Output channel name for channel 0..11 **Max string length:** 8 **Default:** ['ch00', 'ch01', 'ch02', 'ch03', 'ch04', 'ch05', 'ch06', 'ch07', 'ch08', 'ch09', 'ch10', 'ch11']
out_name2	0x60	string[12]	Output channel name for channel 11..23 **Max string length:** 8 **Default:** ['ch12', 'ch13', 'ch14', 'ch15', 'ch16', 'ch17', 'ch18', 'ch19', 'ch20', 'ch21', 'ch22', 'ch23']
out_en	0xc0	bool[24]	Output channel enable (on/off) **Default:** [False, False]
out_on_cnt	0xd8	uint16[24]	Output channel on delay in seconds **Unit:** sec **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Continued on next page

Table 6.4: Parameter table 1: 'configuration'. (Continued)

Name	Address	Type	Description
out_off_cnt	0x108	uint16[24]	Output channel off delay in seconds **Unit:** sec **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
out_init_norm	0x138	bool[24]	Output channel on/off in battery normal mode **Default:** [False, False]
out_init_safe	0x150	bool[24]	Output channel on/off in battery safe mode **Default:** [False, False]
init_on_dly	0x168	uint16[24]	Output channel on initial delay **Unit:** sec **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
init_off_dly	0x198	uint16[24]	Output channel off initial delay **Unit:** sec **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
safe_off_dly	0x1c8	uint16[24]	Output channel off safe delay **Unit:** sec **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
cur_lu_lim	0x1f8	uint16[24]	Output channel instant latchup current limit **Unit:** mA **Default:** [2500, 2500]

Continued on next page

Table 6.4: Parameter table 1: 'configuration'. (Continued)

Name	Address	Type	Description
cur_lim	0x228	uint16[24]	Output channel ema (exponential moving average) latchup current limit **Unit:** mA **Default:** [2500, 2500]
out_lu_dly	0x258	uint16[24]	Time before re-enabling output in case of latchup **Unit:** sec **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
out_lu_cnt	0x288	int16[24]	Max re-activation count, -1 to try forever **Default:** [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
out_link	0x2b8	uint8[24]	Output channel linked channel **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
out_conv	0x2d0	uint8[24]	Output converter ID **Default:** [255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255]
out_voltage	0x2e8	uint16[24]	Output converter voltage **Unit:** mV **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
conv_en	0x318	bool[4]	Output converter enable **Default:** [False, False, False, False]
cur_ema_gain	0x31c	float	Output current ema (exponential moving average) gain. **Default:** 0.5
batt_hwmax	0x320	uint16	Battery HW max value **Unit:** mV **Default:** 33600

Continued on next page

Table 6.4: Parameter table 1: 'configuration'. (Continued)

Name	Address	Type	Description
batt_max	0x322	uint16	Battery full threshold **Unit:** mV **Default:** 33200
batt_norm	0x324	uint16	Battery normal mode threshold **Unit:** mV **Default:** 28000
batt_safe	0x326	uint16	Battery safe mode threshold **Unit:** mV **Default:** 26800
batt_crit	0x328	uint16	Battery critical mode threshold **Unit:** mV **Default:** 26000
bus_wdt_rst	0x32a	bool	Reboot if BUS watchdog expires **Default:** False
bus_wdt	0x32c	uint32	BUS watchdog timeout value **Unit:** sec **Default:** 0
csp_wdt	0x330	uint32[12]	CSP watchdog ping interval **Unit:** sec **Default:** [60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60]
csp_wdt_ping	0x360	uint8[12]	CSP watchdog failed ping limit **Default:** [5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
csp_wdt_chan	0x36c	uint8[12]	Output channel to monitor via CSP **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
csp_wdt_addr	0x378	uint8[12]	CSP address to ping when monitoring output channel via CSP **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

6.2.2 Calibration

Table 6.5: Parameter table 2: 'calibration'.

Name	Address	Type	Description
vref	0x0	uint16	Reference voltage. **Unit:** mV **Default:** 2500
vcc_v_gain	0x4	float	Gain for VCC measurement. **Default:** 1.0
vcc_i_gain	0x8	float	Gain for VCC_i measurement. **Default:** 1.0
vcc_i_offs	0xc	int16	Offset for VCC_i measurement. **Default:** 0
vbat_v_gain	0x10	float	Gain for VBAT measurement. **Default:** 1.0
out_v_gain	0x14	float[24]	Gain for output voltage measurement. **Default:** [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
out_i_gain	0x74	float[24]	Gain for output current measurement. **Default:** [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
out_i_offs	0xd4	int16[24]	Offset for output current measurement. **Default:** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
conv_v_gain	0x104	float[4]	Gain for converter voltage measurement. **Default:** [1.0, 1.0, 1.0, 1.0]

6.2.3 Telemetry

Table 6.6: Parameter table 4: 'telemetry'.

Name	Address	Type	Description
uptime	0x0	uint32	Uptime. **Unit:** seconds

Continued on next page

Table 6.6: Parameter table 4: 'telemetry'. (Continued)

Name	Address	Type	Description
bootcause	0x4	uint32	Boot cause. **Valid values:** '0': Unknown '1': Brownout '2': Power on '3': Watchdog '4': Software '5': External reset pin '6': Wake from sleep '7': CPU error (exception) '8': JTAG
resetcause	0x8	uint16	Reset cause. **Valid values:** '0': Unknown '1': BUS watchdog '2': Ground watchdog '3': Stack overflow '4': Exception '5': Gosh command '6': CSP request '8': Out of memory
bootcount	0xc	uint32	Boot count.
out_i	0x10	int16[24]	Measured output current. **Unit:** mA
out_v	0x40	uint16[24]	Measured output voltage. **Unit:** mV
out_en	0x70	bool[24]	Output channel enable status.
latchup	0x88	uint16[24]	Latchup count for output channel.
cur_ema	0xb8	uint16[24]	Output channel ema (exponential moving average) current **Unit:** mA

Continued on next page

Table 6.6: Parameter table 4: 'telemetry'. (Continued)

Name	Address	Type	Description
vcc_v	0xe8	uint16	PDU VCC voltage. **Unit:** mV
vcc_i	0xea	uint16	PDU VCC current. **Unit:** mA
vbat_v	0xec	uint16	PDU VBAT voltage. **Unit:** mV
temp	0xee	int16	PDU temperature. **Unit:** decicelsius
conv_v	0xf0	uint16[4]	Measured Converter voltage). **Unit:** mV
conv_en	0xf8	bool[4]	Output converter enable status.
batt_mode	0xfc	uint8	Battery mode. **Valid values:** '1': Critical '2': Safe '3': Normal '4': Full
device_type	0xfd	uint8[8]	Device type.
device_status	0x105	uint8[8]	Device status. **Valid values:** '0': No device '1': OK '2': Error '3': Not found
gnd_wdt_cnt	0x110	uint32	Ground watchdog reboots.
bus_wdt_cnt	0x114	uint32	BUS watchdog reboots.
csp_wdt_cnt	0x118	uint32[12]	CSP watchdog reboots.
gnd_wdt_left	0x148	uint32	Ground watchdog value (remaining seconds before reboot). **Unit:** sec

Continued on next page

Table 6.6: Parameter table 4: 'telemetry'. (Continued)

Name	Address	Type	Description
bus_wdt_left	0x14c	uint32	BUS watchdog value (remaining seconds before reboot) **Unit:** sec
csp_wdt_left	0x150	uint8[12]	CSP watchdog value (remaining pings before power cycle).

6.3 P80 ACU

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

6.3.1 Configuration

Table 6.7: Parameter table 1: 'configuration'.

Name	Address	Type	Description
mppt_mode	0x0	uint8	MPPT mode **Default:** 2 **Valid values:** '1': Tracking '2': Fixed
pv_setpoint	0x2	uint16[6]	Configured setpoint for panel voltage for input channels. **Unit:** mV **Default:** [10000, 10000, 10000, 10000, 10000, 10000]
vbat_max_hi	0xe	uint16	Maximum Battery value high value. **Unit:** mV **Default:** 33600
vbat_max_lo	0x10	uint16	Maximum Battery value low value. **Unit:** mV **Default:** 32000
mppt_period	0x14	uint32	MPPT control loop period. **Unit:** msec **Default:** 50
mppt_dv_max	0x18	uint16	MPPT max deviation from configured pv_setpoint. **Unit:** mV **Default:** 1000

6.3.2 Calibration

Table 6.8: Parameter table 2: 'calibration'.

Name	Address	Type	Description
vref	0x0	uint16	Reference voltage. **Unit:** mV **Default:** 2500
vcc_v_gain	0x4	float	Gain for VCC Voltage measurement. **Default:** 1.0
vcc_i_gain	0x8	float	Gain for VCC Current measurement. **Default:** 1.0
vcc_i_offs	0xc	int16	Offset for VCC Current measurement. **Default:** 0
st5_v_gain	0x10	float	Gain for ST5 Voltage measurement. **Default:** 1.0
st5_i_gain	0x14	float	Gain for ST5 Current measurement. **Default:** 1.0
st5_i_offs	0x18	int16	Offset for ST5 Current measurement. **Default:** 0
vbat_v_gain	0x1c	float	Gain for VBAT voltage measurement. **Default:** 1.0
in_v_gain	0x20	float[6]	Gain for input voltage measurement. **Default:** [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
in_i_gain	0x38	float[6]	Gain for input current measurement. **Default:** [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
in_i_offs	0x50	int16[6]	Offset for input current measurement. **Default:** [0, 0, 0, 0, 0, 0]
dac_gain	0x5c	float[6]	Gain for dac. **Default:** [11.0, 11.0, 11.0, 11.0, 11.0, 11.0]

6.3.3 Telemetry

Table 6.9: Parameter table 4: 'telemetry'.

Name	Address	Type	Description
uptime	0x0	uint32	Uptime. **Unit:** sec
bootcause	0x4	uint32	Boot cause. **Valid values:** '0': Unknown '1': Brownout '2': Power on '3': Watchdog '4': Software '5': External reset pin '6': Wake from sleep '7': CPU error (exception) '8': JTAG
resetcause	0x8	uint16	Reset cause. **Valid values:** '0': Unknown '1': BUS watchdog '2': Ground watchdog '3': Stack overflow '4': Exception '5': Gosh command '6': CSP request '8': Out of memory
bootcount	0xc	uint32	Boot count.
input_i	0x10	int16[6]	Measured input current (mA). **Unit:** mA
input_v	0x1c	uint16[6]	Measured input voltage (mA). **Unit:** mV
vcc_v	0x28	uint16	ACU VCC voltage. **Unit:** mV

Continued on next page

Table 6.9: Parameter table 4: 'telemetry'. (Continued)

Name	Address	Type	Description
vcc_i	0x2a	int16	ACU VCC current. **Unit:** mA
st5_v	0x2c	uint16	Stack 5V voltage. **Unit:** mV
st5_i	0x2e	int16	Stack 5V current. **Unit:** mA
vbat_v	0x30	uint16	ACU VBAT voltage. **Unit:** mV
temp	0x32	int16[3]	ACU temperature. **Unit:** decidegC
mppt_mode	0x38	uint8	MPPT mode. **Valid values:** '1': Tracking '2': Fixed
mppt_limit	0x3a	uint16	MPPT limit set by battery charge level. **Unit:** mV
pv_setpoint	0x3c	uint16[6]	Actual setpoint for panel voltage for input channels. **Unit:** mV
power	0x48	uint16[6]	Power (mW). **Unit:** mW
channel_en	0x54	bool[6]	Channel enable status.
dac_val	0x5a	uint16[6]	DAC raw channel value.
mppt_time	0x66	uint16	MPPT processing time. **Unit:** msec
mppt_period	0x68	uint16	MPPT loop time. **Unit:** msec
device_type	0x6a	uint8[8]	Device type.

Continued on next page

Table 6.9: Parameter table 4: 'telemetry'. (Continued)

Name	Address	Type	Description
device_status	0x72	uint8[8]	Device status. **Valid values:** '0': No device '1': OK '2': Error '3': Not found
gnd_wdt_cnt	0x7c	uint32	Ground watchdog reboots.
gnd_wdt_left	0x80	uint32	Ground watchdog value (remaining seconds before reboot). **Unit:** sec

7 References