Nanopower
SADA-50

Manual

Software Documentation

Release 3.0.1

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

Document Details

Title: Nanopower SADA-50 Manual
Reference: 1028361
Revision: 3.0.1
Date: 16 April 2021

Confidentiality Notice

Table of Contents

1. Introduction

The Solar Array Drive Assembly (SADA-5) consists of a stepper motor, gearing, and a driver mechanism to control the motor. The product delivers slow rotational speed on the output shaft. The motor and driver is also able to determine the position of the shaft. In combination with knowledge of the position of the sun, the SADA unit is suitable for controlling suntracking solar panels.

The SADA-50 is not designed to be an autonomous unit. It requires an external entity to update the requested angle of the attached solar panels.

1.1 Unpacking and handling

1.1.1 ESD Precautions

Warning: The SADA-50 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!

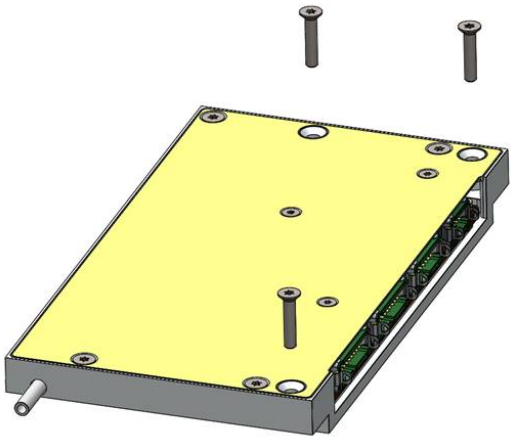
1.1.2 Integration

Electrical integration:

The SADA-50 must be connected to the main communication bus through connector J1. J2 can be used as pass through for the main bus. Power must be connected to J3. Refer to *[sada-datasheet]* for pinouts and location of the connectors.

Mechanical integration:

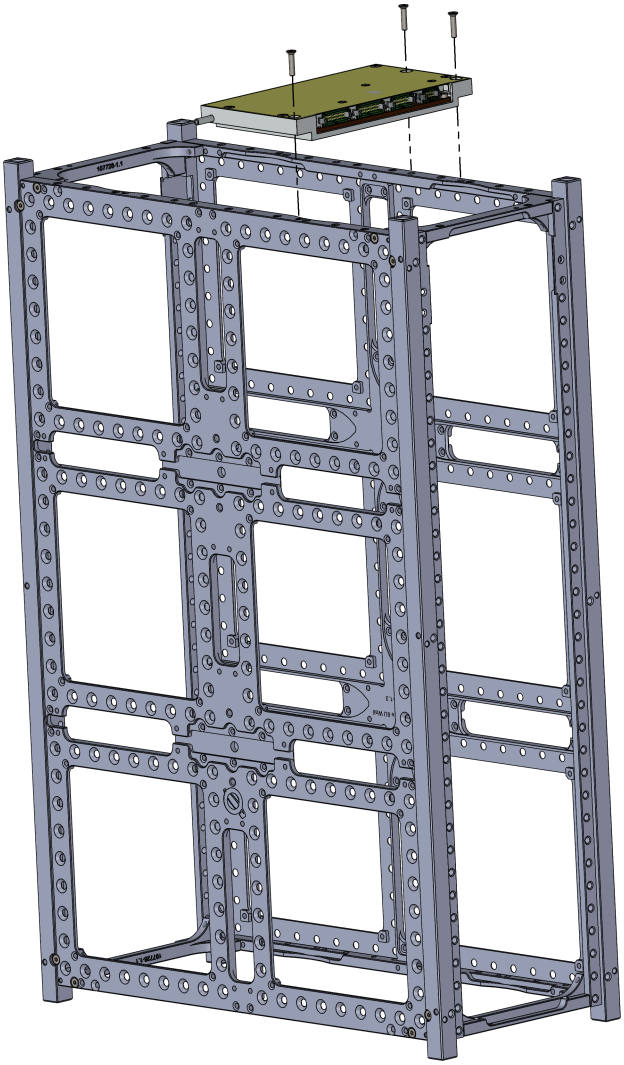
The SADA-50 shall be mounted to the structure using three M2 CSK screws (GomSpace recommends A4-70 screws) as shown in below figure:



The SADA-50 interfaces to both the positive and negative Z end of the Gomspace 6U structure (the 2U face). Here the screw length shall be 10mm, and additional threadlocking (such as Loctite is not necessary since the structure has screwlock helicoils inserted. The tightening torque shall be 0.51Nm.

Remark that the SADA-50 output shaft may not be manually backdriven unless the supply voltage is switched off.

The figure below shows how the SADA-50 is integrated in the GomSpace 6U structure.



1.2 Getting started

This section describes how to get started with the initial set up and configuration of the SADA-50. In addition to the SADA-50 itself, the following items are required (not included in the delivery).

- A LAB power supply capable of generating 5V at min. 200mA.
- A Linux PC (Ubuntu is recommended).
- A CAN communication dongle to connect the PC to the SADA-50 (GomSpace recommends https://www.peak-system.com/PCAN-USB.199.0.html?&L=1).
- Harness to connect the power supply to connector J3 of SADA-50.
- Harness to connect the CAN dongle to connector J1 of SADA-50. This harness must include a 120 Ohm termination resistor.

1.2.1 Hardware setup

First setup the linux PC with the platform interface application as described in *[csp-client-man]* and connect the CAN dongle to the PC. Connect the CAN dongle to J1 of SADA-50 and connect the power supply to connector J3.

Warning: 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.

When the hardware is setup, the csp-client program is started as described in *[csp-client-man]*. A prompt like this should appear:

```
csp-client #
```

To test if there is connection to the SADA-50, perform a ping test:

```
csp-client # ping 13
Ping node 13, timeout 1000, size 1: options: 0x0 ... reply in 2.038 ms
```

After successful ping test, the hardware is now setup and ready to be configured.

2. Configuration and Operation

The SADA-50, as many other GomSpace products, is configured and operated through the GomSpace parameter system. A description of the GOSH commands for the parameter system is found in Section 4.2, the parameters in the SADA-50 are described in Section 3.

This chapter describes how to configure the SADA-50 before flight, and how to operate the SADA-50 in flight.

The SADA-50 application is in charge of the following:

- Rotation of the attached panels (as per request).
- Monitoring of the EPS and switching into safe panel orientation if battery level becomes too low.
- Going to a pre-configured angle when the spacecraft is in eclipse.
- Handling re-calibration of motor and encoder based on physical end stop switches (either based on time or command).
- Rotation lockout until the panels are safely released.

2.1 Interfaces and protocols

This section describes the different interfaces and their respective protocols. For the pin out please refer to *[sada-datasheet]*.

The SADA-50 has two physical communication interfaces:

- CAN
  - ISO 11898-2 compliant
  - External termination resistors needed (120 Ohm between CAN_H and CAN_L at each end of the bus)
  - Used for configuration and operation
- I^2C
  - SDA and SCL at 3.3 V
  - Multimaster
  - 7 bit addressing
  - External pull-up required
  - Used for configuration and operation

The CubeSat Space Protocol (CSP), is available over both I^2C and CAN. The communication stack is sketched in Fig.2.1.

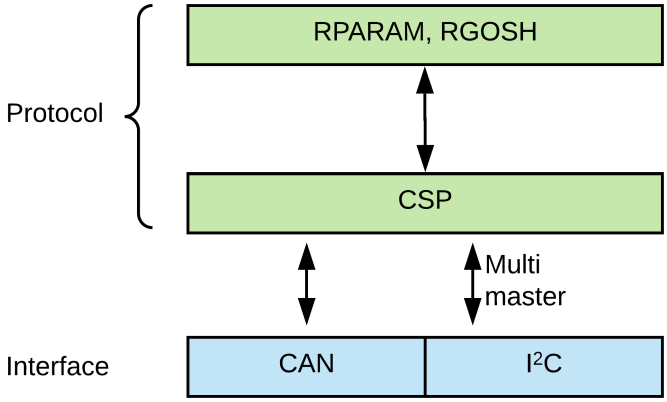


Fig. 2.1: Communication stack.

All setup of the communication is done through the board table Table 3.3.

2.1.1 CSP, RGOSH and RPARAM

The CSP protocol is a small universal network layer delivery protocol similar to TCP/IP which is more suited for smaller networks without as much overhead. Most of the GomSpace subsystems utilize this protocol. Refer to *[csp-client-man]* for further information.

On top of CSP there are two main protocols for controlling the SADA-50, RGOSH and RPARAM.

The RGOSH protocol can be used to execute every GOSH command remotely, and is described in more detail in *[csp-client-man]*.

The protocol RPARAM can be used to read and/or change parameters remotely, and is described in more detail in the *[csp-client-man]*.

For further information on implementation of RGOSH and RPARAM over CSP please refer to *[csp-client-man]*.

2.2 Configuration

The unit has a number of configuration that should be configured according to the mission. The parameters to be considered here are:

- *addr* Set the CSP address of the node.
- *csp_rtable* Routing table for CSP.
- *panel_limit* Set the negative and positive limit angles for the panel.
- *safe_angle* Set the panel angle in safe mode.
- *eclip_angle* Set the panel angle in eclipse mode.
- *norm_speed* Set the speed used in normal mode.
- *norm_acc* Set The acceleration used in normal mode.
- *fast_speed* Set the speed used in homing, safe and eclipse mode.
- *fast_acc* Set The acceleration used in homing, safe and eclipse mode.
- *rot_orient* Set the rotation orientation.

- *ctrl_en* Enable the control.
- *hom_int_time* Set time between homing.
- *hom_int_ecl* Number of eclipses between homing.
- *aa_wdt* Number of seconds before the aa watchdog times out.
- *eps_type* Type of EPS.
- *eps_addr* Address of the eps dock / pmu.
- *pvw_temp_addr* Addresses of the temp sensors on the panels.

Note: ctrl_en must be set to true while doing the calibration homing. It *must* be set to False before launch to prevent rotation before the panels are unfolded.

See chapter Section 3 for details on the individual parameters.

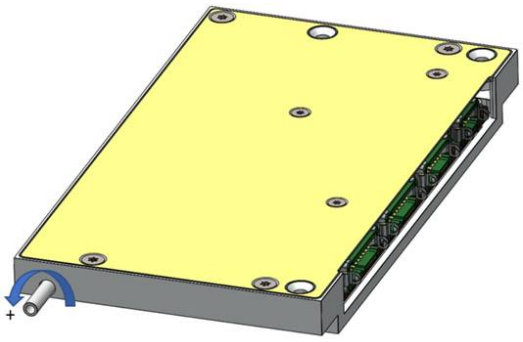


Fig. 2.2: Rotational direction

The figure Fig.2.2 shows the direction for positive rotation when the parameter *rot_orient* is set to true.

When the parameters above have been configured according to mission, they should be saved as described in *Save configuration table in every storage*. Now the unit shall be restarted.

Note: For the next steps, the panels must be unfolded and control enabled.

The last step in the configuration is to calibrate the location of the stop switches and the physical end stops. The procedure for this is:

- Set the parameter *do_homing* to 3.
- Wait for the homing to complete (*aa_mode* returns to 2) and the rotation stops.
- Set the parameter *do_homing* to 4.
- Wait for the homing to complete (*aa_mode* returns to 2) and the rotation stops.
- Now the parameters must be saved as described in *Save configuration table in every storage*.

2.3 Operation

The SADA-50 is not an autonomous unit. An external unit is required to control the actuation angle. In case of a suntracking solar panel, it could be the Attitude Determination and Control System (ADCS) unit which has knowledge of the position of the sun and is needed to control the rotational movement. The connection to the controlling unit is secured with a watchdog. Furthermore, the operational mode is dependent on the power mode of the power management system (EPS). The SADA-50 can be operated in 4 modes as depicted in

figure Fig.2.3. There is also a fifth mode (Error) that is not shown in the diagram, to not clutter the overview. The parameters used to control the SADA-50 can be found in the control table *Parameter Table 'control'* of the parameter system.

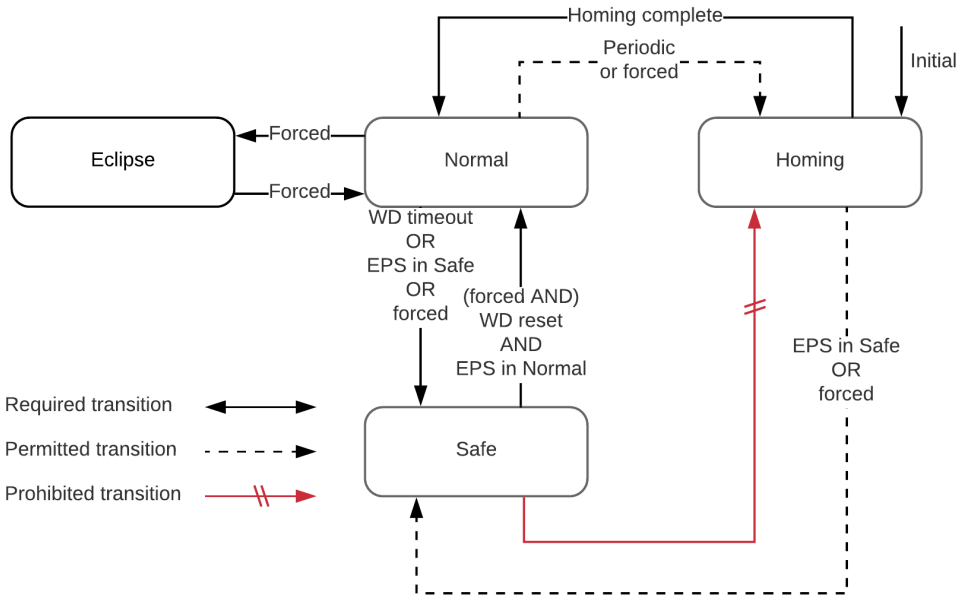


Fig. 2.3: Software modes

2.3.1 Normal mode

In normal mode the SADA-50 reacts to changes on the angle parameter *target_ang*. If the difference between the current and desired motor angle is larger than the threshold configured in *d_angle*, the motor will start to rotate to the desired angle. The set angle is remembered even after reboots. During operation, the SADA-50 detects motor stalls by comparing the angle difference of the motor and encoder. If the difference is above the configured threshold, the motor run current is increased and the motion is restarted.

The aa-watchdog must be serviced during operation by writing a 1 to the *reset_aa_wdt* parameter. Failure to refresh the watchdog in time will transition the unit into safe mode. The maximum interval is configured in *aa_wdt*.

2.3.2 Homing mode

During operation, the motor and encoder angles might deviate from the real angles. The SADA-50 implements a functionality to realign the angles. This process is called homing.

Homing is initiated in 3 ways.

- **Time based homing** is automatically performed at every *hom_int_time* seconds if the value is > 0.
- **Eclipse based homing** is automatically performed at every *hom_int_ecl* eclipses if the value is > 0. This mode can not be used in case of earth tracking during eclipse.
- **Manual homing** is performed when writing a 1 to the parameter *do_homing*.

In addition to the normal homing modes, there are a few homing modes used for checkout calibration. These are:

- **Full range check (2)** Make a full rotation of the panel in each direction.
- **Calibration (3)** Detects the angles of the end stop switches and sets the zero angle in the middle of these.

- **Physical stop (4)** Detects the physical stall angles.
- **Backlash (5)** Determines the backlash.

In all cases homing can be inhibited by setting the parameter *home_allow* to False.

2.3.3 Safe mode

The SADA-50 can be configured to rotate the panels to a pre-configured angle in case it enters safe mode. The unit will enter safe mode in these cases:

- The EPS is in safe mode, if configured.
- The aa_watchdog is not refreshed in time.
- By writing true to *safe_mode*.

Normal operation is resumed whenever all of the above conditions are removed again.

2.3.4 Eclipse mode

In eclipse mode, the panels are rotated to a pre-configured angle. Eclipse mode is entered by setting the parameter *eclipse* to true. Eclipse mode is left when writing false to the parameter.

This functionality should not be used if earth tracking during eclipse is wanted. In this case the SADA-50 should be left in normal mode and tracking should be performed externally.

2.3.5 Error mode

The SADA-50 will enter an error mode in case the unit detects any error it can't handle by itself. The unit will stay in error mode until acknowledged by writing true to the parameter *clear_error*.

3. Parameter system

The parameter system on the SADA-50 controls all of the functionality. It consists of six different tables, each containing a number of variables. The tables are stored as different copies in different stores.

3.1 Stores

Each table is stored in different dynamic versions in various stores with different levels of accessibility. The table store matrix is shown in Table 3.1.

Table 3.1: Table store matrix

Table	FRAM (writable)	FRAM (write protected)	MCU Flash (write protected)
board	yes	yes	yes
configuration	yes	yes	yes
control	no	no	no
drv_settings	yes	yes	yes
drv_readings	no	no	no
hw_readings	no	no	no
telemetry	no	no	no

Note that the actual naming used for the stores is shown in Table 3.2.

Table 3.2: Table store naming

Medium	Name in parameter system
FRAM (w)	persistent
FRAM (wp)	protected
MCU Flash (wp)	flash

Upon boot, the parameter system will load each table from the leftmost available store (referring to Table 3.1). Each table is stored with a CRC value. If the CRC fails while loading, the parameter system will try to load the table from the next store. If all dynamic table versions get corrupted, then the table obtains non changeable default values, which will be the same for every SADA-50.

For instance, if the writable version of the configuration table, stored in the external FRAM, gets corrupted it will fall back to the write protected version, stored in the external FRAM. If this one also 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 default version.

The write protected versions should be changed only on ground.

Refer to Section 4.2 for information regarding changing parameters and saving to different stores.

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 the telemetry table are also persistently stored, such as boot count, but as they change dynamically there is no fall back versions.

3.2 Parameters

The content of the six different tables is described in the following subsections.

3.2.1 Board

Table 3.3 holds all critical configuration of the SADA-50. It should not be changed in flight.

Table 3.3: Parameter Table 'board'

Name	Addr.	Type	
uid	0x00	string	Hardware UID string
type	0x10	uint8	Hardware type/variant Default: `0`
rev	0x11	uint8	Hardware revision. Default: `0`
hostname	0x12	string	Hostname for CSP and GOSH prompt Default: `sada-50` Reboot Required: `Yes`
addr	0x22	uint8	Address, used for CSP Default: `13` Reboot Required: `Yes` Valid Range: `1 <= addr <= 31`
csp_rtable	0x23	string	CSP routing table Default: `0/0 CAN` Reboot Required: `Yes`
can_brate	0x84	uint32	CAN bitrate Default: `1000000` Unit: `baud` Reboot Required: `Yes` Valid Range: `125000 <= can_brate <= 1000000`
i2c_brate	0x88	uint32	I2C bitrate Default: `400000` Unit: `baud` Reboot Required: `Yes` Valid Range: `100000 <= i2c_brate <= 400000`
teststate	0x8c	uint8	Test state Bitfield to indicate which test step has been passed in production Default: `0` Bits: **0:** PCBA passed **1:** PCBA failed **2:** Final/Calib passed **3:** Final/Calib failed

3.2.2 Configuration

Table 3.4 holds the runtime configuration of the SADA-50. Only the version in the writable FRAM should be changed in flight.

Table 3.4: Parameter Table 'configuration'

Name	Addr.	Type	
stop_switch	0x00	int16[2]	The angle of the detector switches. Index 0 is the negative switch. Default: `[-1220, 1220]` Unit: `decidegree` Valid Range: `-3600 <= stop_switch <= 3600`
stop_phys	0x04	int16[2]	The angle of the physical stops. Index 0 is the negative stop. Default: `[-1260, 1260]` Unit: `decidegree` Valid Range: `-3600 <= stop_phys <= 3600`

Continued on next page

Table 3.4 – continued from previous page

Name	Addr.	Type	
panel_limit	0x08	int16[2]	The configured software angle limits. Index 0 is the negative direction. Default: `[-1180, 1180]` Unit: `decidegree` Valid Range: `-3600 <= panel_limit <= 3600`
full_range	0x0c	uint16	Full angular range between the detector switches. Default: `2440` Unit: `decidegree` Valid Range: `0 <= full_range <= 3600`
backlash	0x0e	int16	Backlash angle span. Default: `0` Unit: `decidegree` Valid Range: `0 <= backlash <= 3600`
safe_angle	0x10	int16	The angle set to safe mode. Default: `900` Unit: `decidegree` Valid Range: `-1800 <= safe_angle <= 1800`
eclip_angle	0x12	int16	The angle set to eclipse mode. Default: `0` Unit: `decidegree` Valid Range: `-1800 <= eclip_angle <= 1800`
norm_speed	0x14	int16	Normal operation angular speed of the solar panel. Used in normal mode. Default: `100` Unit: `centidegree/second`
norm_acc	0x16	int16	Normal operation acceleration of the solar panel. Used in normal mode. Default: `10` Unit: $\texttt{decidegree/second}^2$
fast_speed	0x18	int16	Fast operation angular speed of the solar panel. Used when switching mode, e.g. when going to safe mode. Default: `100` Unit: `centidegree/second`
fast_acc	0x1a	int16	Fast operation acceleration of the solar panel. Used when switching mode, e.g. when going to safe mode. Default: `10` Unit: $\texttt{decidegree/second}^2$
ctrl_en	0x1c	bool	Flag to enable control, i.e. rotation is possible. Used to prevent accidental rotation until solar panels are fully deployed in orbit. Default: `False`
boot_home	0x1d	bool	Perform homing after boot when control has been enabled by setting ctrl_en true. Default: `False`
rot_orient	0x1e	bool	Defines if positive angles result in counter-clockwise orientation. This parameter needs to be set differently in case of two actuators placed on opposite sides of a spacecraft. Default: `True` Reboot Required: `Yes`
d_angle	0x20	uint16	Delta angle required before actuator will rotates. Default: `20` Unit: `decidegree` Valid Range: `0 <= d_angle <= 3600`
stall_thrs	0x22	uint16	Angular difference threshold for stall detection. Default: `30`
det_stop_en	0x24	bool	When enabled, motor stops when detector switch is enabled. Default: `True`
sw_r_lim_en	0x25	bool	Enable software rotation limit. Default: `True`
hom_int_time	0x28	uint32	Time between periodic homing procedures. 0 means periodic homing is disabled. Default: `0` Unit: `second`

Continued on next page

Table 3.4 – continued from previous page

Name	Addr.	Type	
hom_int_ecl	0x2c	uint32	Eclipse count between homing procedures. 0 means eclipse count triggered homing is disabled. Default: `0`
min_hom_int	0x30	uint32	Minimum time interval between 2 consecutive homing procedures. 0 means no limit. Default: `600` Unit: `second`
max_hom_time	0x34	uint32	Maximum duration for one homing procedure. Default: `600` Unit: `second`
vcc_mot_en	0x38	bool	Enable motor VCC. Default: `True`
vcc_pvw_en	0x39	bool	Enable PVW VCC. Default: `True`
vcc_5v0_en	0x3a	bool	Enable 5V0 VCC. Default: `True`
vcc_3V3_en	0x3b	bool	Enable 3V3 VCC. Default: `True`
drv_en	0x3c	bool	Enable TMC5130 driver. Default: `True`
gssb_i2c_en	0x3d	bool	Enable GSSB I2C. Default: `True`
csp_i2c_en	0x3e	bool	Enable CSP I2C. Default: `True`
boot_delay	0x3f	uint8	Delay after booting before switching on supplies for the motor and subsequently supply for PVW. Default: `2` Unit: `second`
aa_wdt	0x40	uint32	Actuator watchdog start value. The Purpose is to ensure that the system is triggered from master (normally ADCS). In case of watchdog timeout the panels will go to safe position. Default: `3600` Unit: `second`
eps_type	0x44	uint8	Type of EPS to query for the POWER mode. Default: `0` Reboot Required: `Yes` Valid Values: **0:** none **6:** nanopower_p60 **8:** nanopower_p80
eps_addr	0x45	uint8	CSP address of EPS dock / pmu. Used to check the power mode, 0 means feature is disabled. Default: `0` Reboot Required: `Yes` Valid Range: `0 <= eps_addr <= 31`
pvw_temp_addr	0x46	uint8[4]	I2C address of temperature sensor on PVW's. 0 means no sensor. Default: `[0, 0, 0, 0]` Reboot Required: `Yes` Valid Range: `1 <= pvw_temp_addr <= 127`

Continued on next page

Table 3.4 – continued from previous page

Name	Addr.	Type	
log_mask	0x4a	string	Log mask/level settings Settings applied to log groups and appenders during boot, overwriting the default levels set in the code. Format: <group|appender>=<level> (CSV list). Level: 0 (off), e (error), w (warning), n (notice), d (debug), t (trace) Example: `i2c=0,csp=t` Default: " " Reboot Required: `Yes`

3.2.3 Control

Table 3.5 holds the parameters which will control the application. This table can not be saved.

Table 3.5: Parameter Table 'control'

Name	Addr.	Type	
target_ang	0x00	int16	Requested angle of the solar panels. Default: `0` Unit: `decidegree` Auto Persist: `Yes` Valid Range: `-3600 <= target_ang <= 3600`
set_mot_ang	0x02	int16	Write the angle of the motor into the driver register without rotating. This is used for calibration or homing. Default: `0` Unit: `decidegree` Valid Range: `-3600 <= set_mot_ang <= 3600`
set_enc_ang	0x04	int16	Write the angle of the encoder into the driver register without rotating. This is used for calibration or homing. Default: `0` Unit: `decidegree` Valid Range: `-3600 <= set_enc_ang <= 3600`
home_allow	0x06	bool	Flag to allow homing of the panels. Default: `True` Auto Persist: `Yes`
eclipse	0x07	bool	Flag to activate eclipse mode. The panel will go to eclipse position. Default: `False` Auto Persist: `Yes`
safe_mode	0x08	bool	Flag to activate safe mode. The panel will go to safe position. Default: `False` Auto Persist: `Yes`
do_homing	0x09	uint8	Command the actuator to perform a homing sequence. Will reset to 0 when command is executed Valid Values: **0:** None **1:** Simple_Homing **2:** Full_range_Check **3:** Calibration **4:** Physical_stop **5:** Backlash
reset_aa_wdt	0x0a	bool	Reset the Actuator watchdog to avoid going to safe position. Returns to false when command is executed.
clear_error	0x0b	bool	Clear the error condition and go to normal mode. Returns to false when command is executed.

3.2.4 drv_settings

Table 3.6 holds the configuration register values in the TMC5130A-TA that are used to control the stepper motor.

Table 3.6: Parameter Table 'drv_settings'

Name	Addr.	Type	
i_hold	0x00	uint8	Ihold field of IHOLD_RUN register Default: `5` Valid Range: `0 <= i_hold <= 31`
i_holddelay	0x01	uint8	Ihold delay field of IHOLD_RUN register Default: `6` Valid Range: `0 <= i_holddelay <= 15`
i_run	0x02	uint8	Irun filed of IHOLD_RUN register Default: `22` Auto Persist: `Yes` Valid Range: `0 <= i_run <= 31`
gconf	0x04	uint32	Raw GCONF register. Note that bit 4 will be set based on direction setting in configuration Default: `4`
tpowerdown	0x08	int32	TPOWERDOWN Default: `0x0a`
tpwmthrs	0x0c	uint32	Raw TPWMTHRS register Default: `0x01f4`
tcoolthrs	0x10	int32	Raw TCOOLTHRS register Default: `0`
thigh	0x14	int32	Raw THIGH register Default: `0`
chopconf	0x18	uint32	Raw CHOPCONF register Default: `0x7f5`
pwmconf	0x1c	uint32	Raw PWMCONF register Default: `0x401c8`
vsense	0x20	bool	State of the Vsense bit in CHOPCONF register Default: `True`

3.2.5 drv_readings

Table 3.7 holds values read from the TMC5130A-TA registers. This table cannot be saved.

Table 3.7: Parameter Table 'drv_readings'

Name	Addr.	Type	
vzero	0x00	bool	vzero ramp status bits from motor driver
pos_reach	0x01	bool	position reached ramp status bits from motor driver
vel_reach	0x02	bool	velocity reached ramp status bits from motor driver
stallguard	0x03	bool	stall guard driver status bits from motor driver
standstill	0x04	bool	standstill driver status bits from motor driver
over_temp_war	0x05	bool	over temperature warning from motor driver
motor_speed	0x06	int16	last sampled motor speed. Raw value of the VACTUAL register
motor_current	0x08	int16	last sampled motor current Unit: `milliampere`
gstat	0x0c	uint32	Raw value of the GSTAT register
ioin	0x10	uint32	Raw value of the IOIN register
ramp_stat	0x14	uint32	Raw value of the RAMP_STAT register
mscuract	0x18	uint32	Raw value of the MSURACT register
drv_status	0x1c	uint32	Raw value of the DRV_STATUS register

Continued on next page

Table 3.7 – continued from previous page

Name	Addr.	Type	
pwm_scale	0x20	uint32	Raw value of the PWM_SCALE register
lost_steps	0x24	uint32	Raw value of the LOST_STEPS register
tstep	0x28	int32	Raw value of the TSTEP register
rampmode	0x2c	int32	Raw value of the RAMPMODE register

3.2.6 hw_readings

Table 3.8 holds values read from the load switch readings. This table cannot be saved.

Table 3.8: Parameter Table 'hw_readings'

Name	Addr.	Type	
pwr_status	0x00	uint16	Bit mask to show the status signals from the pmu. Bits: **0:** PMU_FLT_RAIL_5V **1:** PMU_PG_RAIL_5V **2:** PMU_FLT_VCC_3V3 **3:** PMU_PG_VCC_3V3 **4:** PMU_FLT_VCC_MOTOR **5:** PMU_PG_VCC_MOTOR **6:** PMU_PG_VCC_5V0 **7:** PMU_FLT_VCC_PVW **8:** PMU_PG_VCC_PVW
vcc_3v3_i	0x02	int16	Current measurement on 3V3 supply. Unit: `milliampere`
vcc_mot_i	0x04	int16	Current measurement on motor. Unit: `milliampere`
rail_5v_i	0x06	int16	Current measurement on 5V rail. Unit: `milliampere`
vcc_pvw_i	0x08	int16	Current measurement on PVW. Unit: `milliampere`
vcc_5v0_i	0x0a	int16	Current measurement on 5V supply. Unit: `milliampere`

3.2.7 Telemetry

Table 3.9 holds all telemetry collected by the application. Parameters that are auto persistent should not be changed externally.

Refer to Section 4.1 for clearing telemetry.

Table 3.9: Parameter Table 'telemetry'

Name	Addr.	Type	
uptime	0x00	uint32	How long the unit has been running, since last reset/boot. Unit: `second`
bootcount	0x04	uint16	Number of times the unit has booted. Auto Persist: `Yes`

Continued on next page

Table 3.9 – continued from previous page

Name	Addr.	Type	
bootcause	0x06	uint16	Boot cause, exctracted from the MCU. Valid Values: **0:** Unknown **1:** Brownout **2:** Power on **3:** Watchdog **4:** Software **5:** External reset pin **8:** JTAG
eclipse_cnt	0x08	uint32	Number of eclipses in unit lifetime. Auto Persist: `Yes`
motor_temp	0x0c	int16	Temperature of the motor. Default: `-10000` Unit: `decidegC`
int_temp	0x0e	int16	Temperature inside MCU. Default: `-10000` Unit: `decidegC`
panel_temp	0x10	int16[4]	Temperature of the solar panels Default: `[-10000, -10000, -10000, -10000]` Unit: `decidegC`
encoder_angle	0x18	int16	Angle measured on encoder. Unit: `decidegree` Auto Persist: `Yes`
motor_angle	0x1a	int16	Angle measured on motor. Unit: `decidegree` Auto Persist: `Yes`
encoder_pos	0x1c	int32	Raw encoder value from motor driver. Auto Persist: `Yes`
motor_pos	0x20	int32	Raw position value from motor driver. Auto Persist: `Yes`
motor_target	0x24	int32	Raw target value from motor driver.
l_enc_angle	0x28	int16	Latched angle measured on encoder when reaching stop switch. Unit: `decidegree`
l_mot_angle	0x2a	int16	Latched angle measured on motor when reaching stop switch. Unit: `decidegree`
l_enc_pos	0x2c	int32	Latched raw encoder value from motor driver when reaching stop switch.
l_mot_pos	0x30	int32	Latched raw position value from motor driver when reaching stop switch.
aa_wdt_cnt	0x34	uint16	Actuator watchdog timeouts that has triggered safe mode. Auto Persist: `Yes`
aa_wdt_left	0x38	uint32	Time left on Actuator watchdog. Unit: `second` Auto Persist: `Yes`
err_flags	0x3c	uint8	Error flags. Bitfield signalling misc errors. Bits: **0:** positive stall **1:** negative stall **2:** positive homing error **3:** negative homing error
last_ang_err	0x3e	int16	Last angle error. Unit: `decidegree`
max_ang_err	0x40	int16	Maximum angle error. Unit: `decidegree` Auto Persist: `Yes`

Continued on next page

Table 3.9 – continued from previous page

Name	Addr.	Type	
aa_mode	0x42	uint8	Angular Actuator mode. Auto Persist: `Yes` Valid Values: **0:** Initial **1:** Homing **2:** Normal **3:** Safe **4:** Eclipse **5:** Error
last_home	0x44	uint32	Time since last homing. Unit: `second` Auto Persist: `Yes`
last_range	0x48	uint16	Range measured at last check. Unit: `decidegree` Auto Persist: `Yes`
sec_to_home	0x4c	uint32	Seconds to next periodic homing. 0 means periodic homing is disabled. Unit: `second` Auto Persist: `Yes`
ecl_to_home	0x50	uint32	Number of eclipses to next periodic homing. 0 means periodic homing is disabled. Auto Persist: `Yes`
i_run_delta	0x54	uint8	Delta I_RUN caused by stall. Auto Persist: `Yes`

4. Commands

The SADA-50 features a command interface to configure and checkout the SADA-50. The commands are executed trough rGOSH protocol - see *[csp-client-man]*. This chapter describes the SADA-50 specific GOSH commands and the GOSH commands needed to operate the parameter system.

4.1 SADA-50 commands

Every normal operation and configuration of the SADA-50 is done through the parameter system. There is therefore only one command group, which is specific to the SADA-50. This group is described in Table 4.1.

Table 4.1: Command Table ‘aa_cmd’

Command	
ap_clear	Reset ap data to default values.

4.2 Parameter system commands

Setting and getting individual parameters should be done through the rparam interface. Functions like storing to backup stores cannot be done through rparam and must be done through rgosh. see *Save configuration table in every storage*.

4.2.1 Setting target angle

Enter “rparam download 13 3” to select the control table to work on. Entering “rparam list” will now show the control table:

```
csp-term # rparam download 13 3
csp-term # rparam list
  Table control (3):
  0x0000 target_ang      I16 0
  0x0002 set_mot_ang     I16 0
  0x0004 set_enc_ang     I16 0
  0x0006 home_allow      BL  true
  0x0007 eclipse         BL  false
  0x0008 safe_mode       BL  false
  0x0009 do_homing       U8  0
  0x000A reset_aa_wdt    BL  false
  0x000B clear_error     BL  false
```

The content is the volatile working copy of the table (only version for the control table).

Entering “rparam set target_ang 900” followed by “rparam send” will set the target angle to 90 degrees and start moving the panel, if control is allowed.

4.2.2 Save configuration table in every storage

To change this configuration in all stores, the following procedure should be followed:

- Configure rgosh server “rgosh server 13”.
- Unlock the MCU FLASH storage by entering “rgosh run “param unlock flash””.

- Select the configuration table after boot by entering “rgosh run “param select configuration””.
- Unlock the protected FRAM storage by entering “rgosh run “param unlock protected”” (note that it automatically enables lock upon boot).
- Save the changed working table to the different stores:
  - Enter “rgosh run “param save configuration flash”” to save in MCU flash.
  - Enter “rgosh run “param save configuration protected”” to save in protected FRAM.
  - Enter “rgosh run “param save configuration persistent”” to save in unprotected FRAM.

Note: This procedure shall be followed for all the stores board, configuration and drv_settings.

The above example is shown in the following console printout:

```
csp-term # rgosh server 13
server: 13 (port: 12, timeout 30000 mS)
csp-term # rgosh run "param unlock flash"
csp-term # rgosh run "param select configuration"
csp-term # rgosh run "param save configuration flash"
csp-term # rgosh run "param save configuration protected"
csp-term # rgosh run "param save configuration persistent"
```

5. References

[csp-client-man] csp-client manual - gs-man-nanosoft-product-interface-application-2.6.1.pdf

[sada-datasheet] nanopower sada datasheet - gs-ds-nanopower-sada.pdf

6. Disclaimer

Information contained in this document is up-to-date and correct as at the date of issue. As GomSpace A/S cannot control or anticipate the conditions under which this information may be used, each user should review the information in specific context of the planned use. To the maximum extent permitted by law, GomSpace A/S will not be responsible for damages of any nature resulting from the use or reliance upon the information contained in this document. No express or implied warranties are given other than those implied mandatory by law.