NanoTorque
GSW-600

Manual

High performance reaction wheel for 6U, 8U and 12U nanosatellites

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

Product name: NanoTorque GSW-600 Manual

Document No.: 1018204

Revision: 1.7

Author: KFUG

Approved by: KATU

Approval date: 11 Nov 2025

Confidentiality Notice

1 Table of Contents

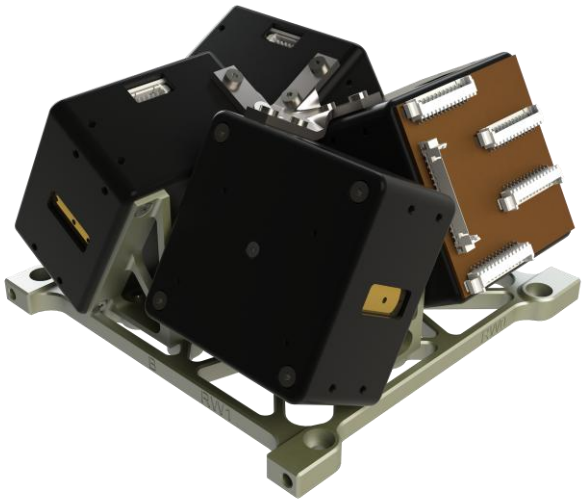
2 Changelog

Date	Revision	Author	Description
17 Oct 2017	1.1	JESM	First release
18 Oct 2017	1.2	JESM	Editing of doc to reflect changes in parameter tables + added I^2C chapter
05-10-2019	1.3	KLK	ISO9001.
21-12-2022	1.4	MBD	Added note on current measurement.
24-01-2025	1.5	KATU/TINI	Update of product renders and touch up
09-10-2025	1.6	KFUG	Added recommended baud rate for SPI and I^2C Changed wording for interface recommendation Added new parameter table (table 5: Debug) Added chapter on reaction wheel debugging
11-11-2025	1.7	KFUG	Added note on I2C being disabled by default Moved reaction wheel debugging chapter further down Added description on how to enable the I2C interface Added note specifying that the I2C interface is disabled if all storage locations are corrupted

3 Introduction

This manual gives an introduction on how to install, assemble and operate the GomSpace NanoTorque GSW-600 reaction wheel.

The GSW-600 reaction wheel can be ordered individually or four wheels in a pyramid setup. Below are shown a CAD of each.



4 Handling

Warnings:

The GSW-600 contains high performance bearings whose performance will be reduced if contaminated. Observe cleanliness precautions.

The GSW-600 system employs components based on FETs and therefore requires anti-static handling precautions to be observed. Do not touch or handle the product without proper grounding.

***Note**: GomSpace recommends spinning the wheels for a period pf time to redistribute any settled grease after shipping or extended time without use.*

---

5 Mounting

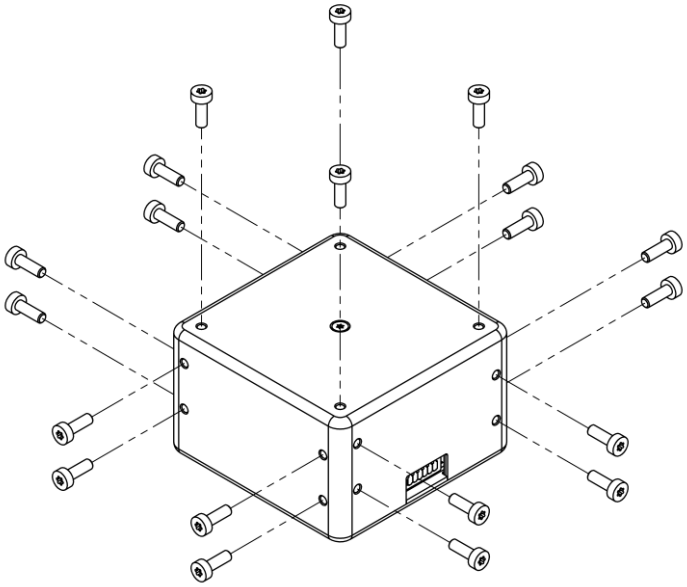
5.1 Standard Tightening Torque for Screws

Below is shown a table for GomSpace suggested standard tightening torque for screws.

Screw Diameter [mm]	Torque [Nm]
2	0.315
2.5	0.65
3	1.14

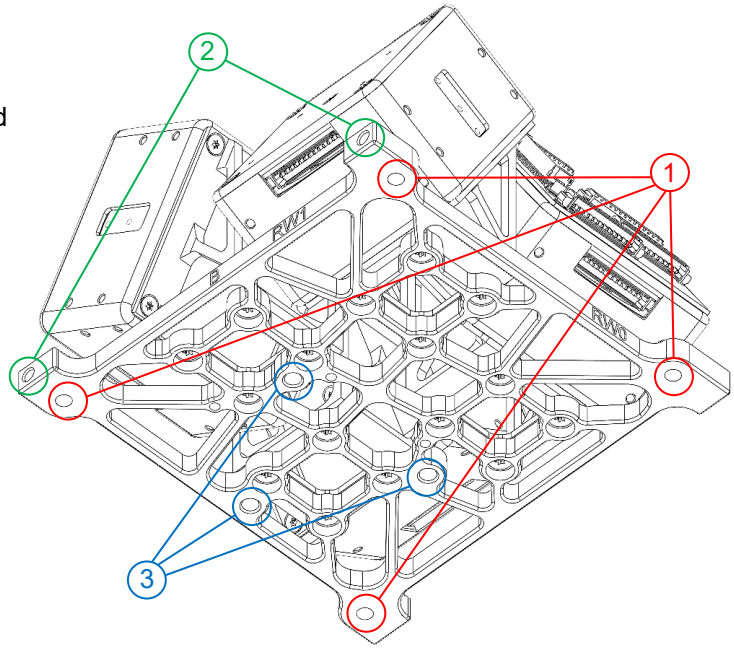
5.2 One Wheel

Five of the sides of the casing has threaded holes which can be used to mount the wheel on e.g. a frame. Use M2 screws.



5.3 Pyramid

This bracket is designed for the GomSpace 6U structure but can be used with other structures. See the drawing on the right.



Red 1: Are used to add the pyramid bracket to a standard PC104 stack. Hole diameter 3 mm.

Green 2: Used for attaching the pyramid bracket to a structure. Use M2.5 screws.

Blue 3: Designed for mounting a gyroscope on the GSW-600. Use M4 screws.

6 Hardware

The GSW-600 is equipped with a 13 pin Molex Picoblade which is the main connector (H2) used to connect the wheel to the NanoDock ADCS-6, NanoDock ADCS-8 or some other control unit.

In addition, there is a 5-pin connector (H1) mainly used by GomSpace for internal configuration.

6.1 Housekeeping

The GSW-600 provides several housekeeping points that enable monitoring of the condition of the system. These measurements are available through the SPI or the I^2C interface.

The telemetry parameters are listed in the parameter table in chapter 7.4.6.

7 Software

The GSW-600 supports two interfaces: SPI and I^2C. The two interfaces are described in the following chapters.

In addition, GomSpace has developed a console-like interface called GomSpace Shell (GOSH), which provides a simple but extensive human readable debug and configuration interface. GOSH is a general feature present on several of GomSpace's products. This is used as a debug interface, see Section 8.1.

The console provides a text-interface to a given input/output stream such as a serial port. GOSH is described in more detail in the GOSH manual. If this manual has not been provided upon purchase, please contact GomSpace support. It's possible to gain access to the GOSH console interface through the H1 connector.

7.1 Parameter Structure

The interface towards the GSW-600 uses a simple protocol like that of the Parameter System which may be known from other GomSpace products.

The parameter system gives access to variables onboard the GSW-600 with different datatypes such as bools, integers, floats or strings. These parameters can be used to configure and control the GSW-600, as well as to retrieve telemetry from it. The parameters have different behaviour depending on their implementation, such as read-on-boot, runtime-read, read-only etc. Configuration and command parameters can be set/read, whereas telemetry parameters should only be read.

The parameter system is a way of accessing the memory on the product, see Figure 1. Instead of specifying an absolute address in the memory space, the commanding interface specifies a parameter table and a parameter in this table to be set/read. This should prevent a user in accessing a wrong section of the memory unintentionally.

The parameter tables and the corresponding parameters are described in Section 7.4.

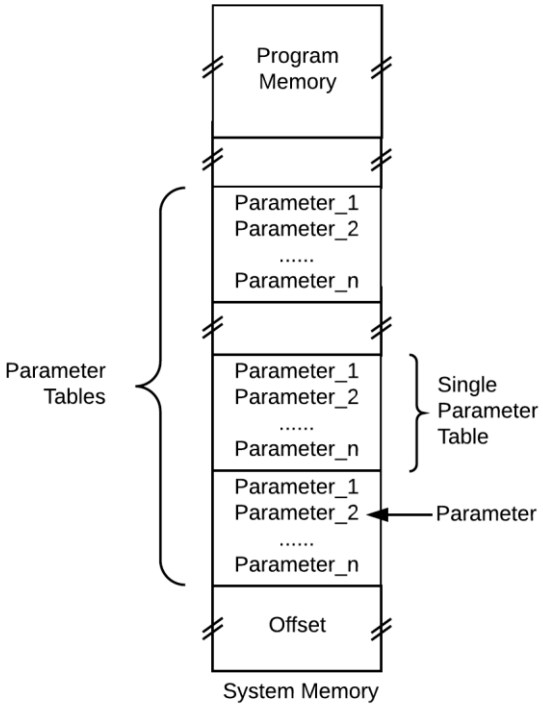


Figure 1: Parameter system overview

7.2 Special Parameters

The parameters from the GSW-600 are all a standard datatype, except for one. The parameter 'speed_torque' from the 'command' table consists of two parameters merged into one. This parameter is used by the GSW-600 as the reference setpoint for the internal controller, and both speed and torque references must be updated at the same time. To ensure an atomic write-process, two 16-bit variables have been merged into a single 32-bit one.
The parameter is constructed as:

speed_torque = (torque_ref << 16) | (speed_ref)

speed_torque	32 bit	
torque_ref	16 bit	Unit: 0.1 uNm (10^{-7} Nm)
speed_ref	16 bit	Unit: RPM

7.3 Interfaces

The GSW-600 supports two interfaces: SPI and I²C. Both interfaces give access to the parameters listed in Section 7.4. **Please note** that the SPI interface is the recommended interface for in-flight operation.

***Note**: The I2C interface has been disabled by default from production. If required, it must be enabled through the debug interface. Information on how to properly enable the I2C interface can be found in this document.*

***Note**: The interfaces give access to several more parameters than those listed in chapter 7.4. **Parameters not listed in the parameter tables in chapter 7.4 should not be modified**.*

7.3.1 SPI Interface

The implementation of SPI uses four lines:

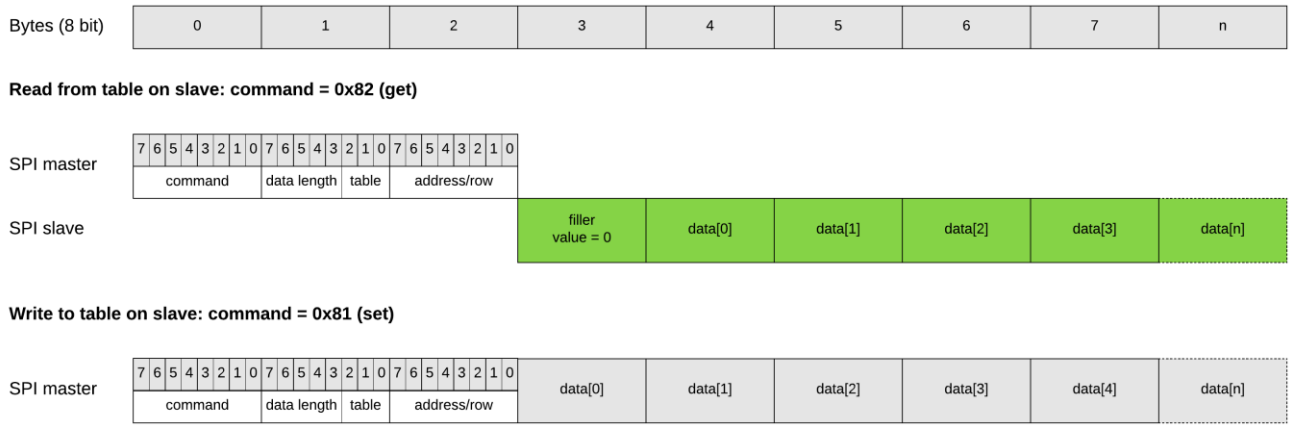
- SCLK (Serial CLocK)
- MOSI (Master Out Slave In)
- MISO (Master In Slave Out)
- CS (Chip Select)

Communication is initiated by the master pulling the CS line low. All data is encoded in **little endian**. The configuration is as follows:

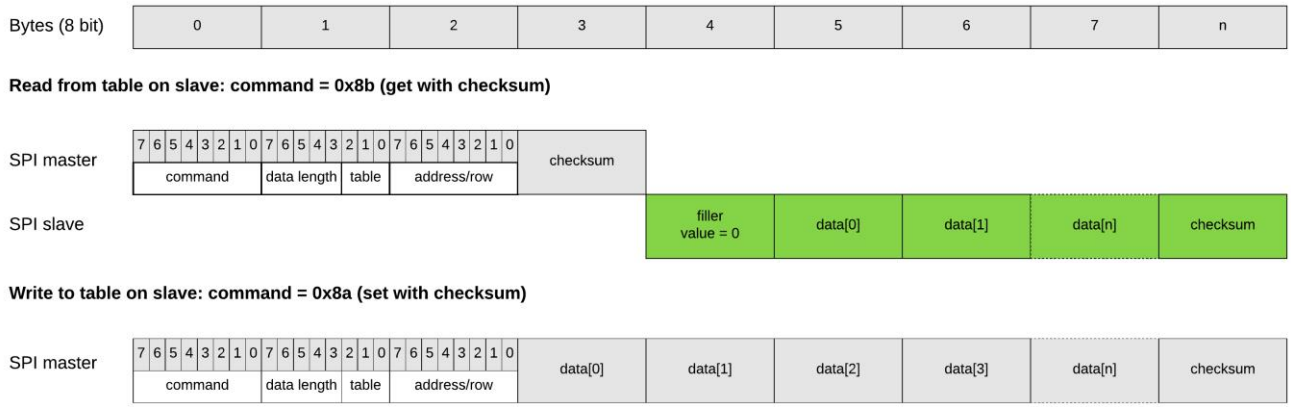
Data order:	MSB
Clock polarity:	0
Clock phase:	0
Recommended baud rate:	100 Kbps
Max baud rate:	130 Kbps

The following bytes are used by the SPI master and SPI slave, respectively:

Without checksum



With checksum



The first three bytes of the SPI master is divided into:

Command:	This is either a set or get command (see figures above)
Data length:	Number of data bytes. Depends on the parameter type
Table:	The ID of a specific table (memory id)
Address/row:	The address of a specific parameter inside table

Example of how to get a parameter value

The table below shows the bytes needed to get the speed_torque parameter value:

command	data length + table	address
0x82	0x26	0x0000

Example of how to set a parameter value

The table below shows the bytes needed to change the speed_torque parameter value in table 6. Speed is set to 2000 (0x07D0) RPM and torque is set to 10000 (0x2710) 10^{-7}Nm:

command	data length + table	address	data[0]	data[1]	data[2]	data[3]
0x81	0x26	0x0000	0xD0	0x07	0x10	0x27

Since the *speed_torque* parameter is of type uint32 the data length is 4 bytes – see chapter 7.4.7:

Hex	**Binary**	**Data length** (5 bits)	**Table** (3 bits)
0x26	00100110	00100 (4)	110 (6)

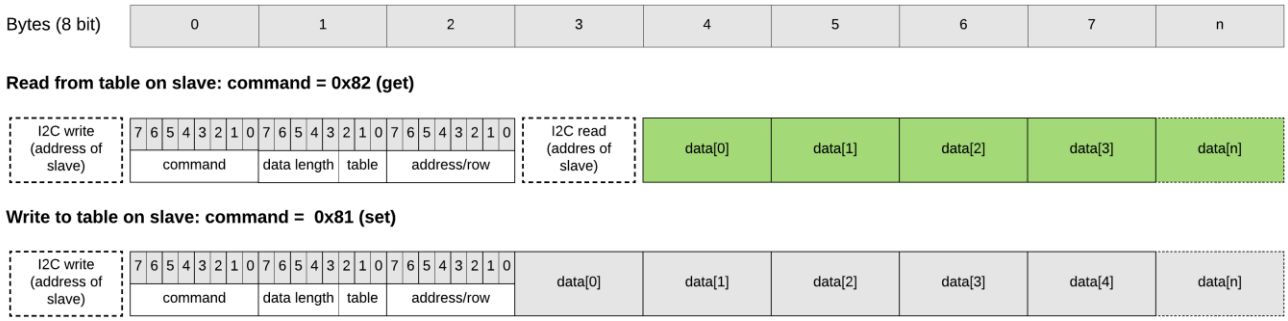
7.3.2 I²C Interface

The configuration is as follows:

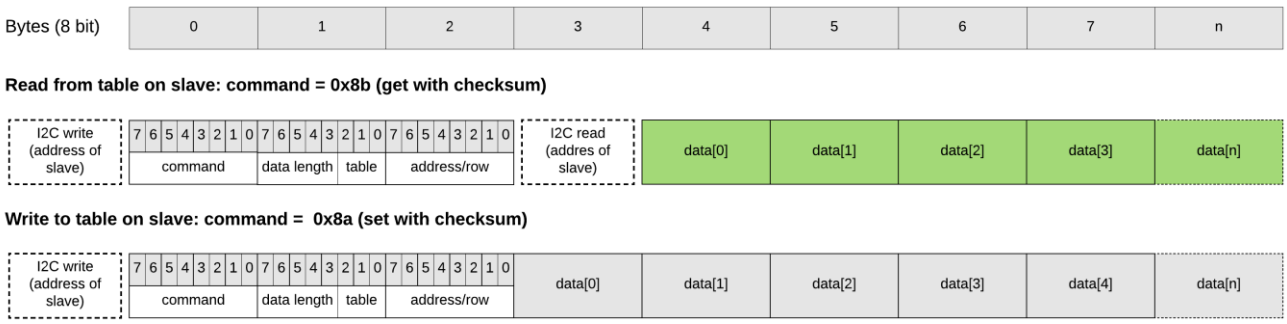
Data order:	MSB
Address:	7 bit
Recommend speed:	100 kHz
Speed (max):	400 kHz

Data transfer is **little endian**.
The following bytes are used by the I²C master and I²C slave, respectively:

Without checksum



With checksum
The first three bytes of the I²C master is divided into:



Command:	This is either a set or get command (see figures above)
Data length:	Number of data bytes. Depends on the data type
Table:	The memory address of a specific table (memory id)
Address/row:	The address of a specific parameter

Example of how to get a parameter value

The table below shows the bytes needed to get the *speed_torque* parameter value:

address of slave	domain + command	data length + table	address
09	0x82	0x26	0x0000

Example of how to set a parameter value

The table below shows the bytes needed to change the *speed_torque* parameter value in table 6. Speed is set to 2000 (0x07D0) RPM and torque is set to 10000 (0x2710) 10^{-7}Nm:

address of slave	domain + command	data length + table	address	data[0]	data[1]	data[2]	data[3]
09	0x81	0x26	0x0000	0xD0	0x07	0x10	0x27

Since the *speed_torque* parameter is of type uint32 the data length is 4 bytes – see table in chapter 7.4.7.

7.4 Parameters

The table below lists the parameter tables available for the GSW-600:

Table Number	Name	Description
0	Board	See Table 7-1: Board
1	Configuration	See Table 7-2: Config
2	Controller	See Table 7-3: Control
4	Telemetry	See Table 7-4: Telemetry
5	Debug	See Table 7-5: Debug
6	Command	See Table 7-6: Command

7.4.1 Parameter Modifiers

Writing to a parameter can have different consequences, determined by how the parameter is accessed by the firmware. This is expressed using parameter modifiers found next to the type.

(A) Active parameter
The types marked with an A are active parameters; this means they are checked by the firmware during execution. In practice, this means that for example, the 'mw_max_v' parameter is read for each time a packet is transmitted, and therefore the value will take effect on the next transmitted frame.

(B) Boot-up parameter
The types marked with an B can only be applied during system power-on. That means that after setting the parameter, nothing will happen. To apply the change, the table configuration must be stored and the system rebooted.

(R) Read-only parameter
The parameters marked with an R are read-only. That means that the firmware of the system will automatically update the value of the parameter, and that it can be used to readout a system state or value. Parameters marked with an R can be written to, but the value will most likely be overwritten by the firmware later. An exception to this is counter values, which is only incremented by the firmware. So, to reset the counters, it is possible to write a zero to the counter parameters.

7.4.2 Board Table: Id 0

Table 7-1: Board

Address	Parameter Name	Type	Default Value	Unit	Description
0x0000	uid	16-byte string (R)			Board id
0x0010	rev	uint8 (R)			Board revision
0x0011	en_i2c	bool (B)			Enable I^2C interface
0x0012	addr	uint8 (B)	9		Address, used for CSP and I^2C. Min: 1, Max: 31

7.4.3 Configuration Table: Id 1

Table 7-2: Config

Address	Parameter Name	Type	Default Value	Unit	Description
0x0000	mw_max_torque	uint16 (A)	20000	0.1x uNm	Maximum allowable motor torque
0x0002	mw_max_v	uint16 (A)	5000	mV	Maximum allowable control voltage
0x002e	cur_scale	uint16 (B)	1000	x1000	Current calibration scale
0x0030	cur_offset	int16 (B)	0	mA	Current calibration gain
0x0032	mw_max_cur	uint16 (A)	600	mA	Maximum allowable strator current
0x0034	temp_max	uint16	800	10x C	Maximum allowable internal or external temperature, before triggering a RW Idle
0x0036	temp_hyst	uint8	100	10x C	Re-enable RW when temperature is T_{max} - T_{hyst}

7.4.5 Controller Table: Id 2

Table 7-3: Control

Address	Parameter Name	Type	Default Value	Unit	Description
0x000b	en_ctrl	bool (A)	false		Enable/Disable control loop

7.4.6 Telemetry Table: Id 4

Table 7-4: Telemetry

Address	Parameter Name	Type	Default Value	Unit	Description
0x0000	uptime	uint32 (R)		seconds	How long the unit has been running, since last reset/boot
0x0004	bootcount	uint16 (R)			Number of times the unit has booted - NOT stored/persistent
0x0006	temp_ext	int16 (R)	INT16_MIN	10x C	Temperature on board
0x0008	temp_int	int16 (R)	INT16_MIN	10x C	Temperature inside MCU
0x000a	bootcause	uint16 (R)			Boot cause (from MCU)
0x000c	speed	int32 (R)		10x RPM	Reaction wheel speed
0x0010	current	int16 (R)		mA	Current consumption (motor only). *Please note that measurement is most accurate around motor nominal speed and not intended for use at low currents.*
0x0012	speed_ref	int16 (R)		RPM	Speed reference used by controller
0x0014	torque_ref	int16 (R)		mNcm	Torque reference used by controller
0x0016	status	int8 (R)			Current wheel status: *1: enabled* *0: disabled* *-1: error*
0x0017	input	uint8 (R)			Hardware input (SPI, I2C)

7.4.7 Debug Table: Id 5

Table 7-5: Debug

Address	Parameter Name	Type	Default Value	Unit	Description
0x0000	stall_count	uint32 (R)			Number of stall counts detected
0x0004	coil_ab_cur	int16 (R)			Current measured through coil A and B
0x0006	coil_ac_cur	int16 (R)		mA	Current measured through coil A and C
0x0008	coil_bc_cur	int16 (R)		mA	Current measured through coil B and C
0x000a	idle_cur_s	int16 (R)		mA	Idle current at start of test
0x000c	idle_cur_e	int16 (R)		mA	Idle current at end of test
0x000e	coil_tst_en	uint8 (R)			Enable coil test
0x000f	coil_tst_res	uint8 (R)			Coil test result

7.4.8 Command Table: Id 6

Table 7-6: Command

Address	Parameter Name	Type	Default Value	Unit	Description
0x0000	speed_torque	uint32 (A)	0	See Section 7.2	Set speed and torque. Upper 16 bit is torque. Lower 16 bit is speed (RPM).

7.5 Storing Parameters

The parameters of the GSW-600 can only be stored via the UART and rparam interface. Any non-persistent runtime parameters set via the SPI or I^2C interface will be reset during reboot of the wheel.

8 Interface Debug

This section contains a brief description of how to establish a connection with GSW-600.

8.1 Connecting via H1

DISCLAIMER

The H1 connector is primarily used by GomSpace for configuration and debugging. Be aware, that by connecting to the H1 UART you gain access to all commands and parameter settings. Many of these settings should not be altered by a user. If in doubt, please contact GomSpace support.

The following is a small example of how to verify that the GSW-600 is in working order (spin the wheel):

1) Connect to the H1 connector (UART) via FTDI cable
2) Your serial port communication program (minicom/tio) should be configured as follows:
   - baudrate: 500.000 baud, 8n1
   - disable flow-control
3) To spin the wheel, enter:

```
gsw #  dutycycle set <value>
```

where <value> is between -127 to 127. This will set the voltage at the coil of the motor.

To list all parameter tables:

```
gsw # param tableinfo
id  name
 0 board
 1 config
 2 control
 4 telemetry
 5 debug
 6 command
```

A specific table can be listed using:

```
gsw # param list board
Table board (0):
 0x0000 uid          STR "103486-340"
 0x0010 rev          U8  0
 0x0011 en_i2c       BL  false
 0x0012 addr         U8  9
 0x0014 brate_can    U32 1000000
```

To change a parameter the relevant table must be selected. This is done with the *param select* command:

```
gsw # param select board
```

To get a parameter inside selected table (using ‘addr’ in this example):

```
gsw # param get addr
  addr = 9
```

A parameter in the selected table can be changed with the *param set* command and the **entire** table can be saved with the *param save* command:

```
gsw #  param set addr 11
gsw #  param save board
```

Use the previously mentioned *param list board* to verify, that the parameter value has changed.

The command *param save* includes additional arguments to specify which storage to save the parameter table. The available storage locations can be listed using the *param storeinfo* command.

***Note**: The newest version of GOSH supports both table memory id and name (param list 0 / param list board). We recommend using table names over ids to reduce risk of accidentally choosing a wrong storage location.*

---

9 How to enable I2C interface

Follow the instructions in chapter 8.1 to open a GOSH shell through the debug connector. The I2C interface can then be enabled through the GOSH interface. Set the *en_i2c* parameter undet table 0 (Board) to true:

```
gsw # param select 0
gsw # param set en_i2c true
gsw # param save 0 persistent
gsw # param save 0 persistent_backup
```

The desired I2C address should also be set prior to saving the table e.g.:

```
gsw # param set addr 9
```

Reset the wheel and check that *en_i2c* is set to true:

```
gsw # reset

Welcome to nanotorque, version 2.1.0-0-gf43d80b, compiled 2025-10-02T13:59:07Z
gsw # param list 0
Table board (0):
  0x0000 uid          STR "103486-340"
  0x0010 rev          U8  0
  0x0011 en_i2c        BL  true
  0x0012 addr         U8  9
  0x0014 brate_can      U32 1000000
```

If set correctly the I2C interface will then be available for use.

***Note**: The en_i2c will be disabled by default if both storage positions (persistent and persistent_backup) are corrupted.*

10 Debugging the GSW-600 Reaction Wheel

10.1 The reaction wheel seems to be stuck!

Table 5 (debug) includes a parameter indicating whether the reaction wheel is stalled. The *stall_count* parameter is incremented every time a control voltage is applied while the reaction wheel speed is zero.

Notice that each time a stall is detected the wheel will automatically run a procedure to get the reaction wheel spinning again. In case the reaction wheel remains stuck the counter will keep increasing. If the reaction wheel is freed the counter will remain static.

Furthermore, the counter will increment once during startup and zero-crossings and as the parameter isn't persistent it will be cleared after a power cycle and software reset.

Hence, a stalled reaction wheel can be identified by continuously checking the counter for a period. If the counter is constantly increasing the reaction wheel is stuck. If the counter is static at any value the reaction wheel may have been stuck at some time, but the stall handling procedure was able to recover it.

10.2 One of the coils wires seems to be shorted or broken!

This scenario would typically be observed as a reaction wheel that will only spin up sometimes when command to, especially if a small rotational force is exerted on the wheel around the spin axis.

The GSW-600 reaction includes a function to analyse the three coil wires for both shorted and broken wires. On ground the function can be initiated through GOSH. Simply run the *test coil* command:

```
gsw # test coil
Idle current start: 0.002
AB current : 0.171
AC current : 0.215
BC current : 0.172
Idle current end: -0.003
Coil OK
```

In-orbit the function can be initiated by writing 1 to the *coil_tst_en* parameter under table 5 (debug). To do so use the param protocol interface from the host unit of the reaction wheel:

```
csp-client # pp spi_init 9
csp-client # pp checksum 1
Use CHECKSUM: 1
csp-client # pp set_uint8 5 0xe 1
csp-client # pp get_uint8 5 0xe 1
value(s): 2
csp-client # pp get_uint8 5 0xe 1
value(s): 0
csp-client # pp get_uint8 5 0xf 1
value(s): 0
```

The command *pp set_uint8 5 0xe 1* enables the coil test function. While the function is running the *coil_tst_en* parameter will return 2. When the parameter returns to zero the *coil_tst_res* parameter can be checked for the result. In the above example the coils where 0 (ok). For a non-zero result please contact GomSpace support for further information.

11 Disclaimer

The information in this document is subject to change without notice and should not be construed as a commitment by GomSpace. GomSpace assumes no responsibility for any errors that may appear in this document.

In no event shall GomSpace be liable for incidental or consequential damages arising from use of this document or the software and hardware described in this document.