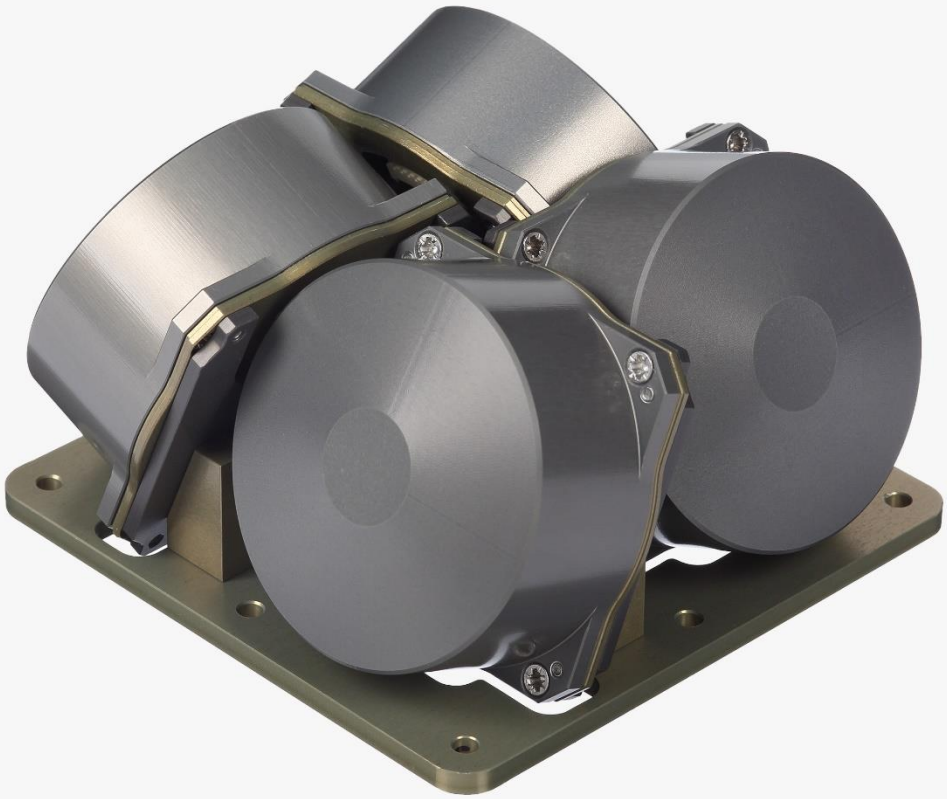
nano avionics



Reaction Wheels

Data Sheet

NA-4RW0-G0-R9

Address: 2901 Enterprise Lane Midland, Texas 79706
Mobile: +1 432 276 3966
E-mail: info@n-avionics.com

Document revisions traceability sheet

Rev. 0	**Date:** 2016-09-20
Changes: Original issue	
Rev. 1	**Date:** 2016-09-25
Changes: Major revision	
Rev. 2	**Date:** 2017-09-25
Changes: Software ICD revised. RW0 and 4RW0 electrical and mechanical properties updated.	
Rev. 3	**Date:** 2017-10-19
Changes: Speed set resolution and control accuracy added. Total residual disbalance added.	
Rev. 4	**Date:** 2018-02-06
Changes: Software configuration binary protocol, framing, command and CRC sections updated. Updated to 2.117 firmware version.	
Rev. 5	**Date:** 2018-03-08
Changes: Electrical parameters in Tables 1, 4, 5, 7 updated; mass updated.	
Rev. 6	**Date:** 2018-06-13
Changes: Speed units changed from cRPM to 0.1 RPM	
Rev. 7	**Date:** 2018-11-08
Changes: Data sheet new design	
Rev. 8	**Date:** 2019-02-15
Changes: Drawings updated	
Rev. 9	**Date:** 2019-09-19
Changes: Table 1, Table 2, Table 7 updated Table 12 (telemetry command structure) updated	

Contents

1 Introduction

NanoAvionics provides reaction/momentum wheels as a separate component (RW0) or an integral four- reaction wheels redundant 3-axis control system (4RW0) to enable precision pointing of the small satellite.

2 Feature Overview

- Interfaces: SPI / UART
- DC Brushless Motor in a sealed housing
- Sealed Design meaning no particle emissions or contamination of peripheral devices
- IPC – A600H class 3 assembly
- Mass of RW0: 137g, mass of 4RW0 system: 700g
- Operational Temperature: -40 °C to +85 °C
- 4RW0 System Mechanical Design Complies with Most CubeSat structures

3 Product Configuration

Below are the product configuration details for one separate reaction wheel (RW0) and a four reaction wheels system (4RW0).

3.1 RW0 Properties

Table 1. RW0 Product Properties and Performance.

Model No.	RW0
Maximum speed	6500 RPM
Maximum torque	3.2 mNm
Maximum momentum storage	20 mNms
Total residual disbalance	<50 mg*mm
Speed set resolution	0.1 RPM
Speed control accuracy 500-6500 RPM	±1 (including ripple) [1]
Speed control accuracy 100-500 RPM	±3 (including ripple)
DC Voltage	5.0 V
Power consumption (Idle)	45 mW
Power consumption (Steady state, 1000RPM)	150 mW
Power consumption (Peak)	3250 mW
Mass	137 g (typ)

[1] At 4.9 – 5.25 V supply voltage range.

3.2 4RW0 Properties

Table 2. 4RW0 Product Properties and Performance

Model No.	4RW0
Maximum Torque Around X axis	5.9 mNm
Maximum Torque Around Y axis	5.9 mNm
Maximum Torque Around Z axis	2.5 mNm (5 mNm when all 4 RWs are used)
Maximum Momentum Storage Around X axis	37 mNms
Maximum Momentum Storage Around Y axis	37 mNms
Maximum Momentum Storage Around Z axis	15.6 mNms (31.3 mNm when all 4 RWs are used)
DC Voltage	5.0 V
Power Consumption (Idle)	180 mW
Power Consumption (Steady state, 1000 RPM each)	600 mW
Mass	760 g (typ)

4 Electrical Configuration

4.1 Connector Pinout

A 7-pin Molex connector is provided for the reaction wheel connection. The receptacle connector part numbers are Molex 51021-0700 for the housing and Molex 50058-8000 for the crimp pins.

Pinout is detailed in Table 3 and Pin Numbering shown in Figure 1.

Table 3. Connector Pinout Details

Pin #	Signal	Type/Direction	Description; Voltage Level
1	+5V	Power	5 V Power Input
2	ENABLE	I/O, Input	HIGH = RW enabled
3	TXD / MOSI	I/O, Output	UART / SPI; 3.3V (5V tolerant)
4	RXD / MISO	I/O, Input	UART / SPI; 3.3V (5V tolerant)
5	SCK	-	SPI Clock Signal; 3.3V (3.6V tolerant)
6	NSS	-	SPI chip select signal; 3.3V (5V tolerant)
7	GND	Power	Ground

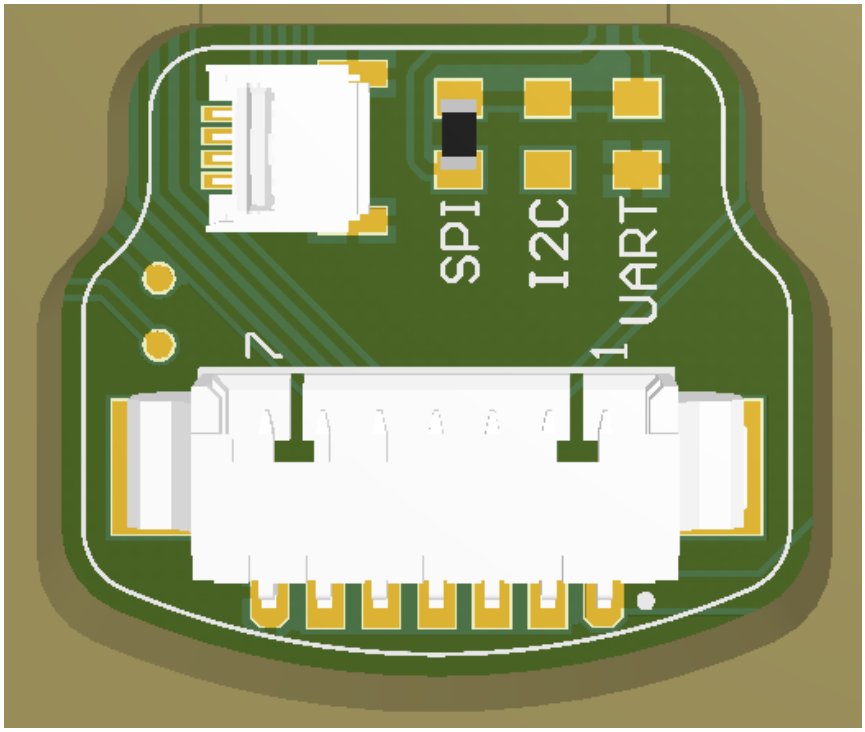


Figure 1. Pin Numbering

4.2 Interface Selection Using User Accessible Resistors

Interface type is selected using one 0 Ohm resistor. RW Firmware scans the resistors at power-up / enable and activates the corresponding interface. Only one resistor should be soldered.

4.3 Signals

4.3.1 Power Input

Table 4. Power Input Specification

Criteria	Value
Absolute Maximum Voltage	-0.3 ... +5.5 V
Operating Voltage	+4.75 ... +5.25 V
Overvoltage / Fault protection	5.6 V Power Zener, Poly-fuse

4.3.2 Enable Signal

LOW level - RW electronics disabled
HIGH level - RW enabled
The wheel is disabled, when ENABLE signal is not connected.

Table 5. Enable Signal Specification

Criteria	Value
Absolute Maximum Voltage	-10 ... +10 V
Input Low Voltage	0.7 V max
Input High Voltage	1.65 V min
Pull Down	47 kOhm

4.3.3 UART / SPI Signals

Interface I/O signals are 3.3V logic, but they are 5V tolerant (except SPI SCK signal).

Table 6. UART/SPI Signal Specification

Criteria	Value
Absolute Maximum Voltage (except SCK pin)	-0.3 ... +7.0 V
Absolute Maximum Voltage, SCK pin	-0.3 ... +4.0 V
Input Low Voltage	1.0 V Max
Input High Voltage	2.3 V Min
Output Low Voltage	0.45 V Max, 4 mA External Load
Output High Voltage	2.8 V Min, 4 mA External Load
Pull Up	25 ... 55 kOhm

4.3.4 Power Consumption

Power consumption at V power = 5.0 V.

Table 7. Power Consumption Specification of RW0

Criteria	Value
Idle State (0 rpm)	45 mW Max
Steady State, 1000 rpm	150 mW Max
Steady State, 6500 rpm	425 mW Max
At Maximum Torque (default current limit)	3250 mW Max
At Maximum Torque (reduced current limit)	1500 mW Max
Current Consumption at Power Down (ENABLE = LOW)	0.1 mA Max

4.3.5 Temperature Range

Table 8. Temperature Range Specification

Criteria	Value
Operating Temperature	-40 ... +85 °C
Storage Temperature	-40 ... +85 °C

4.3.6 Grounding

The wheel housing has no connection to GND signal. It should be grounded externally if required.

5 Software Configuration

This interface control document is valid for firmware version 3.129.

The device supports two types of interfaces:

Table 9. Interface Types

Interface type	Description	UART support	SPI support
Binary Protocol	For integration into other systems, default at system startup	✓	✓
Command Line Interface (CLI)	For fast prototyping and testing	✓	-

UART peripheral configuration:

Table 10. UART Peripheral Configurations

Parameter	Value
Baudrate	115200
Word Length	8 bits
Parity	None
Stop bits	1
Flow Control	None

SPI peripheral configuration:

Table 11. SPI Peripheral Configuration

Parameter	Value
Data Size	8 bits
Clock Polarity	0 (Low)
Clock Phase	0
First bit	MSB
Max Frequency	200 kHz

5.1 Switching from Binary Mode to CLI and Vice Versa

At system start-up device uses binary protocol. To switch to command line interface specific conditions must be met:

1. Idle timeout - no data in interface for more than 1 second.
2. After idle timeout specific byte sequence must be sent: cli\r\n. Hexadecimal representation: 0x63 0x6C 0x69 0x0D 0x0A.
3. Device responds with prompt >. After this all CLI commands can be executed.

In order to switch back to binary protocol exit command must be executed.

5.2 Binary Protocol

All packets are sent and received using HDLC-like framing using request-reply pattern. Device acts as slave and never sends unsolicited messages. Byte order - little endian. There are two types of packets: request and reply.

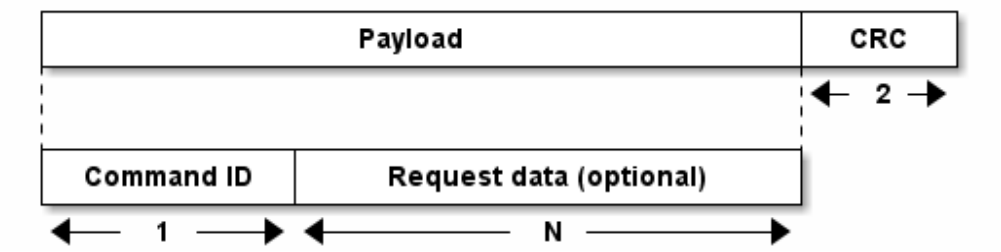


Figure 2. Request Packet Structure and Each Field Size in bytes

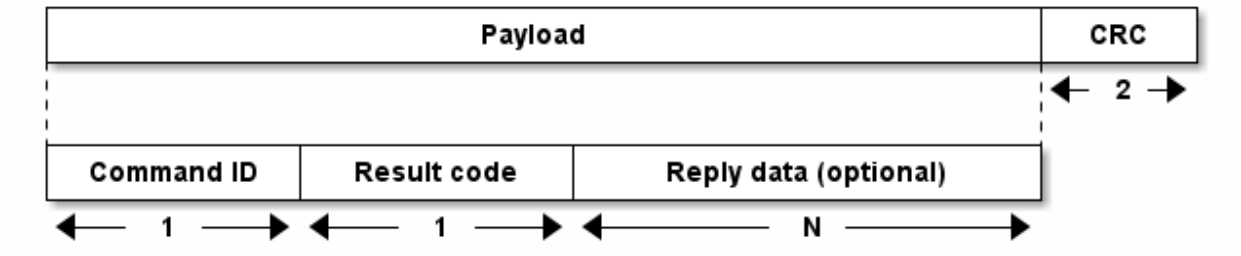


Figure 3. Reply Packet Structure and Each Field Size in bytes

*-request and reply data fields are optional and their size can be equal to zero. Replay wait timeout should be more than 20 ms for all commands and depends on peripheral frequency if SPI is used. Also, the SPI master has to read a reply from the device in no longer than 500 ms, otherwise an answer frame will be dropped.

5.2.1 Framing

Binary protocol uses HDLC-like framing. Each frame begins and ends with a flag byte, which is the binary sequence 01111110 (hexadecimal representation 0x7E). Device continuously checks for this flag, which is used for frame synchronization. Only one flag byte is required between two frames. Two consecutive flag bytes constitute an empty frame, which is silently discarded, and not counted as an error. Example:

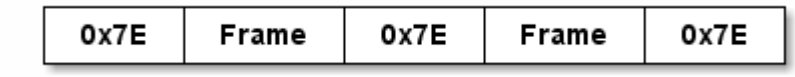


Figure 4. Example of HDLC-like Framing

The escape byte is 0x7D. Whenever a flag or escape byte appears in the message, it is escaped by 0x7D and the byte itself is XOR-ed with 0x20. So, for example 0x7E becomes 0x7D 0x5E. Similarly 0x7D becomes 0x7D 0x5D. The receiver unsuffs the escape byte and XORs the next byte with 0x20 again to get the original.

NOTE: Due to SPI specifics, one bit is transferred from slave to master, and one bit from master to slave, each clock cycle. When master is sending new request packet, slave will respond with sequence of frame bytes (empty frames) until reply is ready to transmit. Thus, master has to send empty frames to slave in order to receive replies from slave.

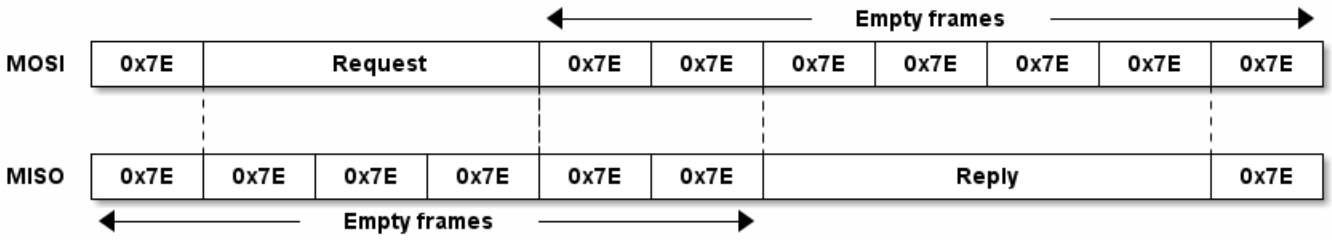


Figure 5. Example of HDLC-like Framing Over SPI

5.2.2 Commands

Command ID takes 1 byte. Possible values of some fields can be found in Section 5.4.

Table 12. Commands

ID	Description	Request data	Total length (bytes)	Reply data	Total length (bytes)
1	reset MCU	-	0	no reply	0
2	get last reset status*	-	0	uin8_t	1
3	clear last reset status*	-	0	-	0
4	get reaction wheel status	-	0	int32_t currSpeed int32_t referenceSpeed uint8_t state uint8_t clcMode	10
5	initialize reaction wheel controller	-	0	-	0
6	set reference speed	int32_t speed uint16_t rampTime	6	-	0
7	set current limit control mode	uint8_t value	1	-	0
8	get temperature	-	0	int32_t value	4
9	get telemetry	-	0	uint8_t lastResetStatus int32_t mcuTemperature float pressureSensorTemperature float pressure uint8_t rwState uint8_t rwClcMode int32_t rwCurrSpeed int32_t rwRefSpeed uint32_t numOfInvalidCrcPackets uint32_t numOfInvalidLenPackets uint32_t numOfInvalidCmdPackets uint32_t numOfCmdExecutedRequests uint32_t numOfCmdReplies uint32_t uartNumOfBytesWritten uint32_t uartNumOfBytesRead uint32_t uartNumOfParityErrors uint32_t uartNumOfNoiseErrors uint32_t uartNumOfFrameErrors uint32_t uartNumOfRegisterOverrunErrors uint32_t uartTotalNumOfErrors uint32_t spiNumOfBytesWritten uint32_t spiNumOfBytesRead uint32_t spiNumOfRegisterOverrunErrors uint32_t spiTotalNumOfErrors	87
10	ping	-	0	-	0
11	get system information	-	0	uint32_t versionMajor uint32_t versionBuildNumber uint32_t uid1 uint32_t uid2 uint32_t uid3	20

* - at startup user application should read last reset status and clear it. During operation this status must be read periodically in order to detect any unexpected system reset and restore previous system configuration immediately.

5.2.3 Result Code

Result code takes 1 byte and is equal to 0 (false) if command execution failed or is equal to 1 (true) if execution was successful.

5.2.4 CRC

CRC is calculated over payload before framing when transmitting and after framing on receiver side. CRC algorithm - CRC16-CCITT. Specifications:

- width - 16 bits;
- polynomial - 0x1021;
- reversed polynomial - 0x8408;
- initial value - 0xFFFF.

Some samples of CRC calculation for testing purpose:

Table 13 – CRC

Values	CRC
0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39	0x29B1
0x71 0x77 0x65 0x72 0x74 0x79	0x0CA2
0x00 0x00 0x00	0xCC9C

Code example for CRC calculation:

```
constexpr std::uint16_t crcTable[] =
{   0x0000,0x1021,0x2042,0x3063,0x4084,0x50a5,0x60c6,0x70e7,
    0x8108,0x9129,0xa14a,0xb16b,0xc18c,0xd1ad,0xe1ce,0xf1ef,
    0x1231,0x0210,0x3273,0x2252,0x52b5,0x4294,0x72f7,0x62d6,
    0x9339,0x8318,0xb37b,0xa35a,0xd3bd,0xc39c,0xf3ff,0xe3de,
    0x2462,0x3443,0x0420,0x1401,0x64e6,0x74c7,0x44a4,0x5485,
    0xa56a,0xb54b,0x8528,0x9509,0xe5ee,0xf5cf,0xc5ac,0xd58d,
    0x3653,0x2672,0x1611,0x0630,0x76d7,0x66f6,0x5695,0x46b4,
    0xb75b,0xa77a,0x9719,0x8738,0xf7df,0xe7fe,0xd79d,0xc7bc,
    0x48c4,0x58e5,0x6886,0x78a7,0x0840,0x1861,0x2802,0x3823,
    0xc9cc,0xd9ed,0xe98e,0xf9af,0x8948,0x9969,0xa90a,0xb92b,
    0x5af5,0x4ad4,0x7ab7,0x6a96,0x1a71,0x0a50,0x3a33,0x2a12,
    0xdbfd,0xcbdc,0xfbbf,0xeb9e,0x9b79,0x8b58,0xbb3b,0xab1a,
    0x6ca6,0x7c87,0x4ce4,0x5cc5,0x2c22,0x3c03,0x0c60,0x1c41,
    0xedae,0xfd8f,0xcdec,0xddcd,0xad2a,0xbd0b,0x8d68,0x9d49,
    0x7e97,0x6eb6,0x5ed5,0x4ef4,0x3e13,0x2e32,0x1e51,0x0e70,
```

```
    0xff9f,0xefbe,0xdfdd,0xcffc,0xbf1b,0xaf3a,0x9f59,0x8f78,
    0x9188,0x81a9,0xb1ca,0xa1eb,0xd10c,0xc12d,0xf14e,0xe16f,
    0x1080,0x00a1,0x30c2,0x20e3,0x5004,0x4025,0x7046,0x6067,
    0x83b9,0x9398,0xa3fb,0xb3da,0xc33d,0xd31c,0xe37f,0xf35e,
    0x02b1,0x1290,0x22f3,0x32d2,0x4235,0x5214,0x6277,0x7256,
    0xb5ea,0xa5cb,0x95a8,0x8589,0xf56e,0xe54f,0xd52c,0xc50d,
    0x34e2,0x24c3,0x14a0,0x0481,0x7466,0x6447,0x5424,0x4405,
    0xa7db,0xb7fa,0x8799,0x97b8,0xe75f,0xf77e,0xc71d,0xd73c,
    0x26d3,0x36f2,0x0691,0x16b0,0x6657,0x7676,0x4615,0x5634,
    0xd94c,0xc96d,0xf90e,0xe92f,0x99c8,0x89e9,0xb98a,0xa9ab,
    0x5844,0x4865,0x7806,0x6827,0x18c0,0x08e1,0x3882,0x28a3,
    0xcb7d,0xdb5c,0xeb3f,0xfb1e,0x8bf9,0x9bd8,0xabbb,0xbb9a,
    0x4a75,0x5a54,0x6a37,0x7a16,0x0af1,0x1ad0,0x2ab3,0x3a92,
    0xfd2e,0xed0f,0xdd6c,0xcd4d,0xbdaa,0xad8b,0x9de8,0x8dc9,
    0x7c26,0x6c07,0x5c64,0x4c45,0x3ca2,0x2c83,0x1ce0,0x0cc1,
    0xef1f,0xff3e,0xcf5d,0xdf7c,0xaf9b,0xbfba,0x8fd9,0x9ff8,
    0x6e17,0x7e36,0x4e55,0x5e74,0x2e93,0x3eb2,0x0ed1,0x1ef0
};

std::uint16_t crcValue = 0xFFFF;

void update(std::uint8_t byte)
{
    crcValue = (crcValue << 8) ^ crcTable[((crcValue >> 8) ^ byte) & 0x00FF];
}
```

5.3 Command Line Interface (CLI)

CLI is supported only via UART. Some basic CLI principles:

- Each line must end with sequence of \r\n symbols in order to start executing provided command.
- After the command is executed, the final result message of execution is returned on a new line. There are three possible results:
    - OK - command was executed successfully;
    - FAIL - command execution failed;
    - ERROR - internal command line interpreter error (ex., command was not found).
- After result message, prompt sign > is sent on a new line to inform that CLI is ready for the next command. Example:

```
>test
OK
>
```

- If command provides some extra information, that information is returned before final result message. Example:

```
>echo hello
hello
OK
>
```

- Some commands have optional or mandatory arguments/subcommands. See list of commands below. Possible values of some fields can be found in Section 5.4.

5.3.1 Commands

ping

Description

Dummy command for interface testing. Always responds with OK.

Example

```
>ping
OK
>
```

exit

Description

Exit CLI and return to binary protocol interface.

Example

```
>exit
OK
>
```

Note: even if prompt sign > is returned, CLI will not execute any commands sent and device will respond only to binary protocol requests.

reset

Description

Reset MCU. No answer is returned. Optional subcommands:

- status - get last reset status; (decimal representation, bit values are listed in Section 5.4)
- status clear - clear last reset status.

Example

```
>reset status
20
OK
>reset status clear
OK
>reset status
0
OK
>
```

temp

Description

Returns MCU temperature. Units - degrees Celsius.

Example

```
>temp
32
OK
>
```

rw

Description

Reaction wheel control. Subcommands:

- init - start initialization procedure. Needed to recover from error state.
- state - returns reaction wheel internal state. Find list of states below.
- status - get full reaction wheel status.
- speed - get current and reference speeds.
- speed <speed> [<ramp time>] - set new reference speed. Ramp time is optional, uses min value if not set.
- clc - get current limit control mode.
- clc [1|0] - set current limit control mode.

NOTE: if state is equal to 0 (error), no speed set command will take effect. To return to normal state user must send rw init command. Error state is entered when motor lock is detected.

Example

```
>rw status
rw state: 1
clc: 1
curr speed: 0
ref speed: 0
OK
>rw speed
curr speed: 0
ref speed: 0
OK
>rw clc
1
OK
>rw speed 5000
OK
>rw speed -5000 100
OK
>
```

telemetry

Description

Get full telemetry.

Example

```
>telemetry
reset status: 5
temp: 26
state: 1
clc: 1
curr speed: 0
ref speed: 0
cmd crc errs: 0
cmd len errs: 0
cmd id errs: 0
cmds executed: 0
cmd replies: 0
comm rx buffer overrun errs: 0
uart rx: 90
uart tx: 2484
uart frame errs: 0
uart noise errs: 0
uart parity errs: 0
uart rx register overrun errs: 0
uart total errs: 0
OK
>
```

help

Description

Print available commands and short help messages.

Example

```
>help
All commands and help messages:
exit - exit CLI, return to binary mode
temp - get MCU temperature (degree Celsius)
reset [status [clear]] - reset MCU, get last reset status or clear it
rw (init|state|status) - init, get state or status
rw speed [<speed> [<ramp time>]] - get or set speed
rw clc [1|0] - get or set current control limit mode
telemetry - get full telemetry
ping - ping (test) command
OK
>
```

5.4 Field Values

Field values are common for binary and command line interfaces.

5.4.1 Last Reset Status

Table 14. Last Reset Status

Bit Number	Description
7	-
6	-
5	Low Power Reset
4	Window Watchdog Reset
3	Independent Watchdog Reset
2	Software Reset
1	POR/PDR/BOR Reset
0	Pin Reset

5.4.2 Reaction Wheel State

Table 15. Reaction Wheel State

Value	Description
0	Error
1	Idle
2	Coasting
3	Running, Speed Stable
4	Running, Speed Changing

5.4.3 Speed

Table 16. Speed

	Min	Max	Unit
Clockwise	1000	65000	0.1 RPM
Counter-clockwise	-65000	-1000	0.1 RPM

5.4.4 Ramp Time

Table 17. Ramp Time

Min	Max	Unit
10	10000	ms

5.4.5 Current Limit Control Mode

Table 18. Current Limit Control Mode

Value	Description
0	Low Current Mode (0.3 A)
1	High Current Mode (0.6 A) – default value

6 Layout

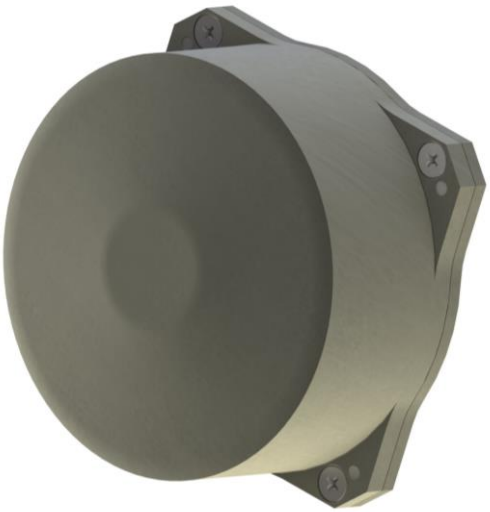


Figure 6. RW0 - General View

Figure 7. 4RW0 - General View

7 Mechanical interface

All dimensions are given in mm.

7.1 RW0

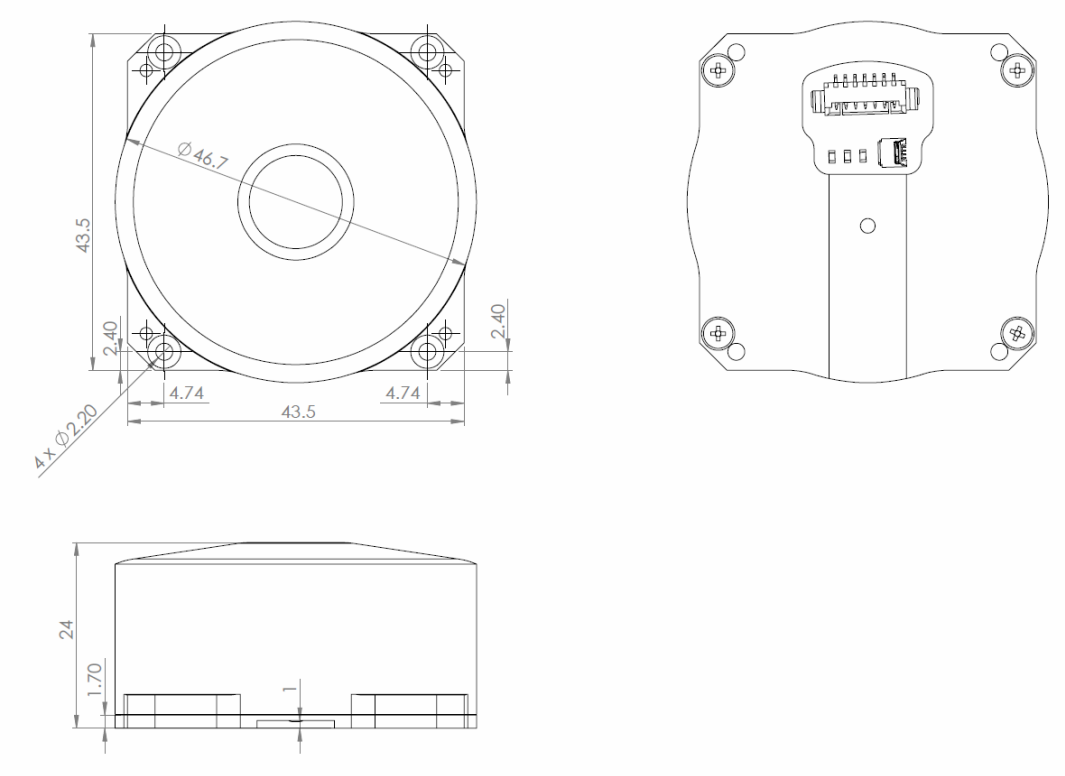


Figure 8. RW0 dimensions [2]

[2] Drawing is not to scale.

7.2 4RW0

4RW0 system is compatible with PC104 standard and is compliant with most cubesat structures.

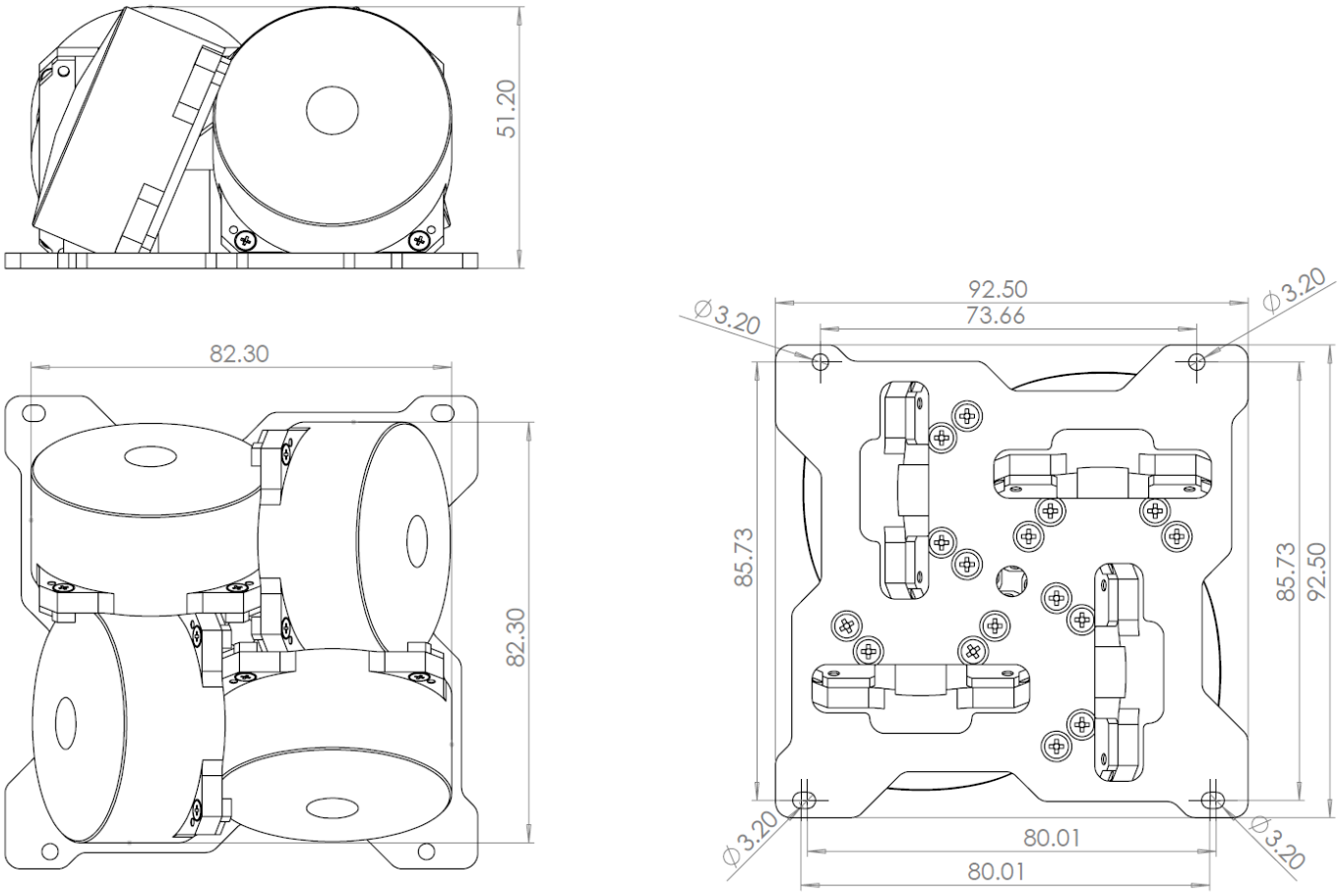


Figure 9 . 4RW0 dimensions [3]

[3] Drawing is not to scale.

8 Protection for Electrostatic Discharge Sensitive (ESDS) devices

1. Work area:

- It is essential to handle ESDS devices at static-safe workstations. This will prevent yield loss (through catastrophic damage) or worse, potential reliability failures in the field (through latent damage).
- Where it is impractical or impossible to use antistatic wrist-straps or remove items that are composed of insulative materials at a static-safe workstation, use an air ionizer designed to neutralize electrostatic charges or apply topical antistats to control generation and accumulation of static charges.
- When an air ionizer is utilized, it is vital that maintenance procedures and schedules are adhered to in order to ensure that ions generated by the ionizer are sufficiently balanced.
- Avoid bringing sources of static electricity within 1 meter of a static-safe work bench.
- Where it is necessary to use air-guns, use special models that do not generate static charges in the air stream.

2. Personnel:

- Any accumulated charge on the body of the human operator should be discharged first before opening the protective container with ESDS devices inside. The discharge can be accomplished by putting a hand on a grounded surface or, ideally, by wearing a grounded antistatic wrist-strap.
- The use of an antistatic smock for each worker is highly recommended.

8.1 General Handling

Gloves (ESD compliant) should be worn when handling all flight hardware.
The Reaction Wheels is robust and designed to withstand flight conditions. However, care must be taken when handling the device. Do not drop the device.

8.2 Shipping and Storage

The devices are shipped in anti-static packaging, enclosed in a Peli case. This case should be used for storage. All hardware should be stored in anti-static containers at temperatures between 20°C and 40°C and in a humidity-controlled environment of 40-60%rh.

9 Disclaimer

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