REACTION WHEEL

400 mNms RW-0.4

Interface Control Document

Rev. 1.1, July 6, 2021

Doug Sinclair, Cordell Grant

components@rocketlabusa.com
rocketlabusa.com

ROCKETLAB

1 Revision Notes

This revision of the document contains the following changes relative to the previously released version (1.0):

- Added this Revision Notes section. All subsequent sections renumbered and respaced.
- Added Init address to Section 11.1.1

2 Scope

This document details the mechanical, electrical and software interfaces for the fourth generation Sinclair Interplanetary reaction wheels of size 200mNms and 400mNms. Part numbers available at present include:

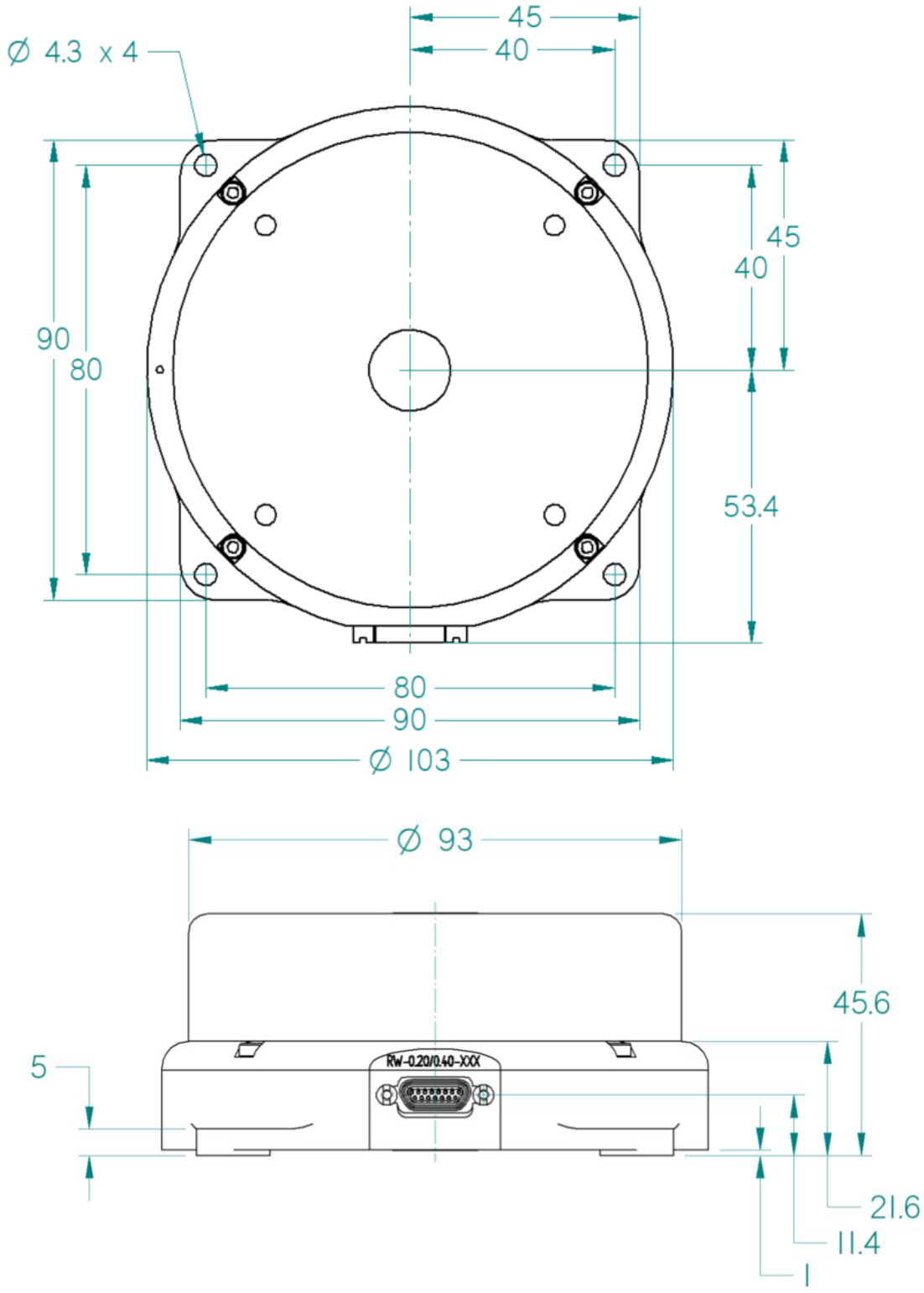
- RW4-0.2-28-RS485
- RW4-0.4-28-RS485

3 Cautions

None

4 Mechanical

4.1 Dimensions



The reaction wheel attaches to the host spacecraft through four mounting holes. Each is 4.3 mm in diameter. They are sized to accept M4 or #8 hardware at the customer's option. The flange thickness is 5 mm.

4.2 Mass Properties

Table 1: Mass Properties

Total Mass (RW4-0.2)	600g
CG Location (RW4-0.2)	20.6mm above mounting plane on rotation axis
Total Mass (RW4-0.4)	770g
CG Location (RW4-0.4)	25.4mm above mounting plane on rotation axis

4.3 Remove Before Flight

The following items may be removed before flight:

Table 2: Remove Before Flight Items

Item	Remove?	Notes
Connector dust cover	Must remove	
Connector saver	Should remove if fitted	Remove with 1/8" wrench, or specially modified 1/8" nut driver

5 Environmental

5.1 Storage

The wheel must be stored in a clean environment to keep dust out of the bearings. The humidity must be kept low to prevent corrosion of the steel rotor.

5.2 Thermal

Table 3: Allowable Temperature Range

Survival Temperature	-40°C to +100°C
Operating Temperature (short term)	-40°C to +90°C at interface
Operating Temperature (long term)	-20°C to +70°C at interface

Table 3 shows the allowed temperature range for the wheel. Short term operating temperatures are permitted for periods of hours to days, while long term operating temperatures are permitted for the many years of a mission.

5.3 Pressure

The wheel will operate in sea-level atmosphere and in hard vacuum. It has not been qualified to operate at high altitude atmospheres, and should not be powered during ascent unless additional testing is performed to show that there is no danger of arcing.

All materials meet the standard outgassing requirements of TML < 1%, CVCM < 0.1%.

5.4 Vibration

The wheel is designed to survive typical launch environments. It has been qualified to NASA GEVS levels (14.1 Grms for 2 mins/axis).

6 Electrical

6.1 Micro-D

The wheel is fitted with a 15-socket micro-D connector.

Table 4: Micro-D Connector Pinout

Pin	Name
1	Address 1
2	Address 2
3	Secondary Return
4	RS485_0_A
5	Secondary Return
6	Primary Return
7	Primary Return
8	Primary Power
9	Address 0
10	RS485_1_B
11	RS485_1_A
12	RS485_0_B
13	Primary Return
14	Primary Power
15	Primary Power

6.2 Programming Header

Each wheel has a programming header on the PCA. These are for factory use, and allow the processor bootloader to be programmed. Customers should not use these without explicit factory advice.

7 Signals

7.1 Address [0|1|2]

Table 5: Address Input Electrical Specifications

Absolute Maximum	±30 V, WRT Secondary Return
Input Resistance	4.75 kΩ
Pull-up	40 kΩ nominal to +3.3 V

The address inputs allow the network address of the wheel to be set. Inputs must either be left open-circuit (digital '1') or connected to Secondary Return (digital '0').

The pull-up may only be active a turn-on. Once the wheel has determined its address, it may disable the pull-up.

7.2 RS485_0[A|B], RS485_1[A|B]

Table 6: RS485 Electrical Specifications

Absolute Maximum	-9 V to +13 V, each signal WRT Secondary Return
ESD rating	±20 kV (Human-body model)
Polarity	B > A in Mark (Idle) state A > B in Space (ON) state
Differential Output Voltage	> 1.5 V into 54 Ω termination.
Short-circuit Output Current	250 mA max
Input Resistance	> 48 kΩ each signal
Input Differential Threshold	-0.20 V to -0.01 V

Each pair is an RS485 signal. They may be used as two half-duplex 2-wire buses, or used together as a 4-wire bus. As a 4-wire bus, the wheel is interoperable with legacy RS422 signals.

7.3 Primary Power, Primary Return

Table 7: Primary Power Electrical Specifications

Absolute Maximum	-65 V to +65 V, WRT Primary Return Time above overvoltage threshold shall be limited to 0.5 sec pulses at no more than 10% duty-cycle
Operating Range	+22 V to +34 V, WRT Primary Return
Isolation from Chassis	100 nF, in parallel with 1 MΩ 100 V Hipot test permissible

All wheels have a Primary Power input, which is returned on the Primary Return.

7.4 Secondary Return

Table 8: Secondary Return Electrical Specifications

Connection to Chassis	Direct connection

The wheel has a galvanically isolated DC/DC converter, and makes its Secondary Return available on the connector. Secondary Return is also locally connected to Chassis.

7.5 Power Architecture

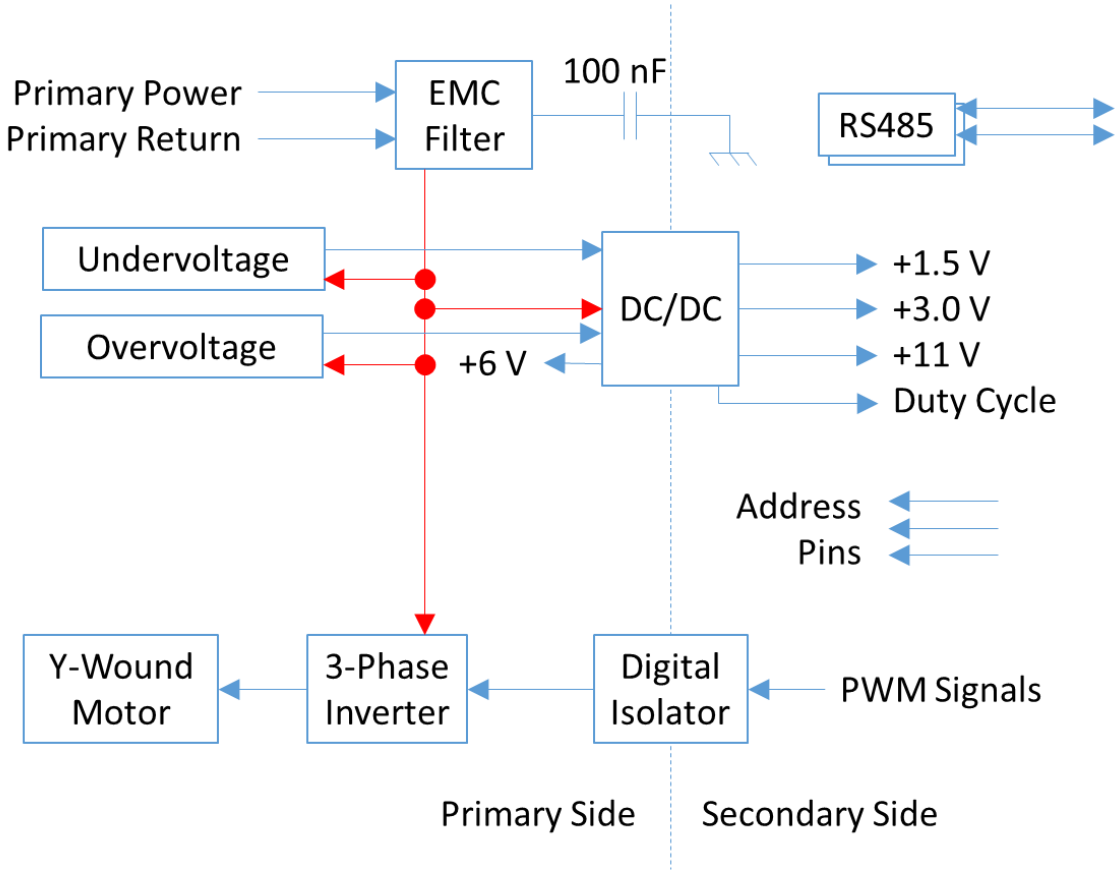


Figure 1: Power Block Diagram

7.6 Undervoltage Limit

Turn-on Voltage	+20.7 V max
Turn-off Voltage	+14.5 V min

The power supply uses an undervoltage lockout (UVLO) circuit to prevent brownout behaviour at low input voltages. There is a large hysteresis on the limit to prevent oscillation.
When the input voltage is too low, the DC/DC converter is inhibited and no power will reach the secondary circuits. The startup regulator and oscillator will remain active, so a modest input current will remain.

7.7 Overvoltage Limit

Overvoltage Limit	+52.6 V typ @ 25 C
Temperature Variation	+46.6 to +57.2 mV/K

The power supply uses an overvoltage limit circuit to prevent damage from very high bus voltages. This circuit is implemented with a simple Zener diode, and thus has a significant temperature variation – the limit voltage increases with increasing temperature.
There is no hysteresis on the limit, and so chatter may be possible. Activation of this circuit should only happen in an emergency, so this is considered acceptable.
The startup regulator and oscillator remain active in an overvoltage condition. In this case the power dissipated in the startup linear regulator can become significant: maybe 10 mA across a 45 V drop. The startup regulator does not have overtemperature protection, so the time spent in the overvoltage condition must be strictly limited.

7.8 Regenerative Braking

The wheel makes use of regenerative braking when slowing the rotor under moderate torque. This will result in the wheel consuming a net negative amount of power, pushing current back out onto the spacecraft power bus. The spacecraft power system design must be able to deal with this.

In an emergency, if the power line becomes disconnected from the power system (such as if turned off via a relay switch) regeneration will increase the voltage at the wheel until the overvoltage threshold is reached. This will cause the wheel to reset and cease regeneration.

8 Protocol Layer 2 (Data Link Layer)

8.1 Asynchronous Serial

The RS485 communications ports use an asynchronous serial protocol. The parameters are programmed into the unit bootloader at the factory, and special-order units with different parameters are available.

Table 9: Default Asynchronous Serial Parameters

Nominal Baud Rate	115200 bps, unless special order (See EIDP)
Data bits per byte	8
Parity	None
Stop bits	1

Each word begins with a start bit with space (0) value. Eight data bits follow, with the LSB sent first and the MSB last. Finally, a stop bit is sent with mark (1) value. Once the stop bit has concluded the output transmitter may be disabled if there are no further words to follow.

The baud rate is driven from a crystal oscillator, and should be highly accurate and stable over the lifetime of the unit.

9 Protocol Layer 3 (Network Layer)

NSP is the Nanosatellite Protocol, originally developed at UTIAS/SFL for use on the CanX nanosatellites. This in turn is descended from the Simple Serial Protocol (SSP) used by UTIAS/SFL and Dynacon on the MOST and CHIPSAT spacecraft as well as the Dynacon reaction wheels in the wider market.

The reaction wheel uses NSP messages for all communication.

9.1 Asynchronous Serial NSP Encapsulation

NSP messages are encapsulated for transmission on an asynchronous serial channel using SLIP framing, as described in RFC 1055. This is required in order to indicate the beginning and end of NSP messages.

Table 10: SLIP Framing Special Characters

FEND	0xC0
FESC	0xDB
TFEND	0xDC
TFSEC	0xDD

Each NSP message is transmitted with a FEND character added to the beginning and end. Whenever FEND would occur within the message it is replaced by two bytes: FESC TFEND. Whenever FESC would occur within the message it is replaced by FESC TFESC.

10Protocol Layer 4 (Transport Layer)

10.1 Command and Reply

The wheel generates telemetry messages in response to command messages received. In the usual case, a single telemetry message will be sent as quickly as possible after reception of the command.

Some commands will take a period of time to execute, and will only generate a telemetry message when they are complete. The wheel should be considered to own the communications bus while such a command is executed, so do not send additional commands to it or any other unit until the reply is complete.

While some NSP devices can generate multiple telemetry messages in response to a single command, the wheel will not. The P/F bit will be set for all replies, indicating that they are stand-alone final messages.

Nonwithstanding the above, the wheel will not generate messages that are not linked to a command. The host spacecraft must poll it to determine its status and to read telemetry.

10.2 NSP Message Format

Table 11: NSP Message Fields

Length	Field
1 byte	Destination Address
1 byte	Source Address
1 byte	Message Control Field
0 or more bytes	Data Field
2 bytes	Message CRC

Each NSP message has the format shown above. The shortest possible messages are 5 bytes (with zero data, not counting framing).

The wheel supports a maximum data length of 1028 bytes, giving a total message length of 1033 bytes. Note that network-layer framing may add additional bytes to the message as it is transmitted.

10.3 NSP Addresses

All NSP messages contain a destination and a source address. A reply message will be sent with a destination address equal to the source address of its command message. Similarly, the source address will be set equal to the destination address from the command.

The user is free to pick one or more NSP addresses for flight computers and other units that may talk to the wheel. Avoid choosing the SLIP framing characters FEND (0xC0) and FESC (0xDB), as well as the reserved address 0x00. By convention the flight computer would normally use NSP address 0x11.

The wheel pays no particular attention to the source address of commands, and will accept commands from any unit on the bus.

10.4 Wheel Address and Port Selection

The wheel bootloader may support incoming communication on a number of communication ports: potentially up to two RS485 ports. In addition, there may be cases where outgoing reply packets are sent on a different port from the command packet. For example, a 4-wire RS485 link can be implemented by receiving commands on one 2-wire RS485 port and sending replies on a different 2-wire RS485 port.

A wheel may respond to different NSP addresses on different ports. On any given port, incoming commands with different NSP addresses may cause replies to be issued on different ports. See the unit-specific EIDP for full information on the available ports and their addresses.

The wheel has two NSP state machines, and can handle simultaneous interactions on both ports. There are two possible contention situations:

- If a reply is generated on a port that is currently receiving an incoming message, the incoming message is abandoned and the reply is sent out. This probably results in an RS485 bus contention.
- If a reply is generated on a port that is already sending out a reply, the reply in progress is not interrupted. The new reply is abandoned – there is no attempt to queue it.

Contention should not occur in a well-managed spacecraft.

10.5 Default Addressing

Unless otherwise specified in the procurement documentation and the unit EIDP, the wheel uses the following NSP addresses:

Table 12: Default Addressing

Address 0	Address 1	Address 2	NSP Address	Command Port	Telemetry Port
Short	Short	Short	0x40	RS485_0	RS485_1
Short	Short	Short	0x50	RS485_1	RS485_1
Short	Short	Short	0x60	RS485_0	RS485_0
Short	Short	Short	0x70	RS485_1	RS485_0
Open	Short	Short	0x41	RS485_0	RS485_1
Open	Short	Short	0x51	RS485_1	RS485_1
Open	Short	Short	0x61	RS485_0	RS485_0
Open	Short	Short	0x71	RS485_1	RS485_0
Short	Open	Short	0x42	RS485_0	RS485_1
Short	Open	Short	0x52	RS485_1	RS485_1
Short	Open	Short	0x62	RS485_0	RS485_0
Short	Open	Short	0x72	RS485_1	RS485_0
Open	Open	Short	0x43	RS485_0	RS485_1
Open	Open	Short	0x53	RS485_1	RS485_1
Open	Open	Short	0x63	RS485_0	RS485_0
Open	Open	Short	0x73	RS485_1	RS485_0
Short	Short	Open	0x44	RS485_0	RS485_1

Short	Short	Open	0x54	RS485_1	RS485_1
Short	Short	Open	0x64	RS485_0	RS485_0
Short	Short	Open	0x74	RS485_1	RS485_0
Open	Short	Open	0x45	RS485_0	RS485_1
Open	Short	Open	0x55	RS485_1	RS485_1
Open	Short	Open	0x65	RS485_0	RS485_0
Open	Short	Open	0x75	RS485_1	RS485_0
Short	Open	Open	0x46	RS485_0	RS485_1
Short	Open	Open	0x56	RS485_1	RS485_1
Short	Open	Open	0x66	RS485_0	RS485_0
Short	Open	Open	0x76	RS485_1	RS485_0
Open	Open	Open	0x47	RS485_0	RS485_1
Open	Open	Open	0x57	RS485_1	RS485_1
Open	Open	Open	0x67	RS485_0	RS485_0
Open	Open	Open	0x77	RS485_1	RS485_0

Short = Address pin shorted to Secondary Return in the harness
Open = Address pin left unconnected

In the normal spacecraft configuration with four reaction wheels, it is advised that the following addresses be used:

- X1
- X2
- X4
- X7

In the event of a failure of any one address input pin or harness straps the wheel will go to one of these addresses:

- X0
- X3
- X5
- X6

Thus, a single failure will never place two wheels at identical addresses. [This is a fancy way of saying that the recommended addresses have a Hamming distance of >1 bit between them.]

10.6 Message Control Field

Table 13: Message Control Field

Bit 7 (MSB)	“Poll/Final” Bit
Bit 6	“B” Bit
Bit 5	“ACK” Bit
Bits 4 – 0	Command code

The message control field packs four values into a single byte. The command code is an enumerated value between 0x00 and 0x1F that determines how the data field should be interpreted.

The “ACK” bit is ignored on commands coming into the wheel. On telemetry reply messages sent by the wheel it is set to indicate successful execution of the command, or cleared to indicate that the command cannot be executed.

The “B” bit is copied unchanged from a command message into its reply message. The wheel does not use it internally.

The “Poll/Final” bit is interpreted differently for command and telemetry messages. For a command, the bit is “Poll”. If it is set to ‘1’ then the wheel will generate a telemetry message in reply. If it is cleared to ‘0’ then the command will be executed, but no response telemetry message will be sent.

For a telemetry message, the bit is “Final”. If a reply consists of a single telemetry message, then the bit is set to ‘1’. If a reply is too large to fit into a single message then the final message has the bit set to ‘1’ and the others have the bit cleared to ‘0’. This reaction wheel will never send out messages with the final bit cleared to ‘0’.

10.7 Data Field

The interpretation of the data field is dependent on the command code in the message control field. Some command codes may have no data, some may require a certain fixed number of data bytes, and some can accept a variable data length.

10.8 Message CRC

Each NSP message contains a 2 byte (16-bit) CRC to guard against errors in transmission. The 16-bit CCITT polynomial is used: x^16 + x^12 + x^5 + 1. The initial shift register value is 0xFFFF. Bytes are fed into the CRC computation starting with the destination address, and concluding with the last byte of the data field. Within a byte, bits are fed in LSB first.

The following fragment of C code, courtesy of Henry Spencer, illustrates how the CRC can be computed.

```
#define POLY 0x8408 /* bits reversed for LSB-first */
unsigned short crc = 0xffff;
while (len-- > 0) {
      unsigned char ch = *bufp++;
      for (i = 0; i < 8; i++) {
            crc = (crc >> 1) ^ ( ((ch ^ crc) & 0x01) ? POLY : 0
            );
            ch >>= 1;
      }
}
```

10.9 Error Conditions

The wheel will ignore NSP command messages where the destination address does not correspond to its own NSP address. NSP messages with invalid CRC, invalid encapsulation, too short or too long are also ignored. In none of these cases will any reply message be generated.

If an NSP command message is in error due to an unknown command code, or if the data field is not consistent with the requirements of the command code, and if the “Poll” bit is

set, then a NACK reply message will be generated. This message will be the same length as the command message, and contain the same data field. The command code will be the same, as will the “B” bit. The “ACK” bit will be cleared to ‘0’.

11Protocol Layer 5 (Session Layer)

11.1 Operating Modes

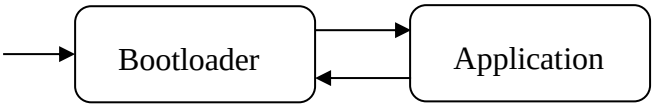


Figure 2: Mode Transition Diagram

Power-on starts the unit in bootloader mode.

11.1.1 Bootloader to Application Transition

The wheel will transition from bootloader to application mode upon receipt of an "INIT 0x20050000" command.

11.1.2 Application to Bootloader Transition

Any transition from application mode to bootloader mode is accomplished through a processor reset. Reset is caused by the following conditions:

- An undervoltage is detected on Primary Power
- An overvoltage is detected on Primary Power
- An "INIT" command with no data is received.
- An exception or unexpected interrupt occurs

The DIAGNOSTIC command can be used to get information on the most recent reset.

11.2 Test Scripts

The reaction wheel contains a number of preprogrammed test scripts. These are used in the factory for initial characterization and pass/fail acceptance testing. They can also be used by customers to verify the health of the wheel during integration and on-orbit.

Test script details are TBC.

11.3 Byte Order

All multi-byte values transported in the data field of NSP messages are in little-endian format. That is, the least-significant byte is stored first, and the most-significant byte is stored last.

11.4 Command Codes

Table 14: Command Codes

Command Code	Command	Bootloader	Application
0x00	PING	Yes	Yes
0x01	INIT	Yes	Yes
0x02	PEEK	Yes	Yes
0x03	POKE	Yes	Yes
0x04	DIAGNOSTIC	Yes	Yes
0x06	CRC	Yes	Yes
0x07	READ FILE	No	Yes

0x08	WRITE FILE	No	Yes
0x09	READ EDAC	No	Yes
0x0A	WRITE EDAC	No	Yes
0x0B	GATHER EDAC	No	Yes

The table above shows the command codes that can be used by the host spacecraft to communicate with the wheel.

11.5 PING (0x00)

The PING command is typically used during testing to verify communications. Incoming data is ignored. The reply packet contains a human-readable text string containing:

- The type of device and the manufacturer
- The compile time and date of the bootloader software
- The compile time and date, and the identity, of the application software if running

11.5.1 Reply Format

Bytes 0 – N	Human-readable ASCII string. No NULL termination.

11.6 INIT (0x01)

The INIT command is used to change the operating mode of a wheel. In all cases, if a reply has been requested ("Poll" bit set to '1') then the reply will be sent before the processor state is changed.

The wheel will respond to an INIT with no data by completely resetting the device, returning to bootloader mode. Otherwise, an INIT command should have 4 bytes of data.

As a special case, the command INIT 0xDEADBEEF is used once in the factory to write-protect the bootloader FRAM. The bootloader FRAM can only be unlocked by issuing a POKE while the /WP pin on the test connector is held high.

An INIT command with an address will be NACKed if the device is not in bootloader mode.

An INIT command with an address in the bootloader or user FRAM will cause an application image to be loaded from FRAM into RAM (see application image section for format). Once the application has been loaded, the processor will branch to the specified starting point and run the program. In the special case where the starting point is zero, no jump will be made and the processor will remain in bootloader mode.

An INIT command with an address in either RAM area will cause a program branch to that address. It is assumed that a program resides at that address: either copied from FRAM by a previous command, or loaded from the first 128 kB of bootloader FRAM on a reset, or loaded directly from a series of POKEs.

11.6.1 Command Format

Bytes 0 – N	Zero or more bytes, ignored by the NSP module

11.6.2 Command Format

Reboot command:

No payload bytes

Application start command:

Bytes 0 – 3	32-bit integer address of program to start

11.6.3 Reply Format

Reboot reply:

No payload bytes

Application start reply:

Bytes 0 – 3	32-bit integer address of program to be started

11.7 PEEK (0x02)

The PEEK command is used to read the device memory. Short and long formats of this command are available for historical reasons. Short commands can be distinguished from long commands by their lengths.

PEEK access to FRAM can be of any length, and without alignment restriction, provided it does not cross the boundary from bootloader FRAM to user FRAM. PEEKs to other memory areas must obey one of the following restrictions:

- Length = 1
- Length = 2, aligned to even address
- Length = 4*N, aligned to multiple of 4

PEEKs to unimplemented memory locations may be expected to generate hardfault exceptions.

11.7.1 Short Command Format

Bytes 0 – 3	32-bit address to start peeking data
Byte 4	Number of bytes to read. A value of 0 indicates that 256 bytes should be read.

11.7.2 Long Command Format

Bytes 0 – 3	32-bit address to start peeking data
Byte 4 - 5	Number of bytes to read.

11.7.3 Reply Format

Bytes 0 – 3	32-bit address of the start of data
Bytes 4 – N	One or more bytes read from the target memory

11.8 POKE (0x03)

The POKE command is used to write the device memory.

POKE access to FRAM can be of any length, and without alignment restriction, provided it does not cross the boundary from bootloader FRAM to user FRAM. POKEs to other memory areas must obey one of the following restrictions:

- Length = 1
- Length = 2, aligned to even address
- Length = 4*N, aligned to multiple of 4

POKEs to unimplemented memory locations may be expected to generate hardfault exceptions. POKEs to write-protected FRAM will not generate any sort of error return, but the target memory will not be altered.

11.8.1 Command Format

Bytes 0 – 3	32-bit address to start poking data
Byte 4 – N	1 - 1024 bytes to write to the target memory

11.8.2 Reply Format

Bytes 0 – 3	32-bit address where data write began
Bytes 4 – N	1 – 1024 bytes written to the target memory

11.9 DIAGNOSTIC (0x04)

The DIAGNOSTIC command gathers error counts and other useful data from the wheel.

11.9.1 Command Format

Byte 0 – N	List of one or more address of the diagnostic channel to read, as 8-bit integers

11.9.2 Reply Format

Byte 0 – N	List of one or more diagnostic result structures

11.9.2.1 Diagnostic Result Structure

Bytes 0	Diagnostic channel read
Bytes 1 – 4	Diagnostic value, generally as 32-bit integer

11.10 CRC (0x06)

CRC command is used to calculate a checksum on an area of memory. The CRC uses the same 16-bit polynomial, with the same bit order, as is used for NSP messages.

CRC addresses are not restricted as to alignment, for any of the memory areas. A CRC must not cover both bootloader FRAM and user FRAM. CRC access to unimplemented memory can be expected to generate hardfault exceptions.

The CRC command can potentially be used to request the CRC of the wheel's entire memory. The worst case would be the CRC of a 256 kB FRAM chip which takes 340 msec from command to reply (assuming no additional realtime processor load from the application program). The application control loop may be stalled during this time, preventing fine speed control.

11.10.1.1 Command Format

Bytes 0 – 3	Address of the first byte to CRC as 32-bit integer
Bytes 4 – 7	Address of the last byte to CRC as 32-bit integer

11.10.1.2 Reply Format

Bytes 0 – 3	Address of the first byte in CRC as 32-bit integer
Bytes 4 – 7	Address of the last byte in CRC as 32-bit integer
Bytes 8 – 9	CRC result as 16-bit integer

11.11 READ FILE (0x07)

The Read File command returns one or more "files", which are four consecutive bytes of EDAC protected memory. A read from address 0 is a special case, and an additional mode byte is returned.

Note that because of the single byte of addressing, not all of the EDAC memory can be accessed by this command.

When multiple telemetry files are read by a single command, they are guaranteed to be internally consistent (i.e. from the same control frame). Files can be read in any order, and a single file can be read multiple times.

11.11.1 Command Format

Bytes 0-N	List of EDAC addresses divided by 4 (0 – 255). 0 for mode, 1 – 255 for normal.

11.11.2 Reply Format

Bytes 0-N	List of File Reply structures. The first byte of each structure determines its type.

11.11.2.1 Mode Reply Structure

Byte 0	0
Byte 1	Command type read from EDAC
Bytes 2 - 5	Command value read from EDAC

11.11.2.2 Normal Reply Structure

Byte 0	Non-zero EDAC address divided by 4 (1 – 255)
Bytes 1 - 4	EDAC data bytes read from memory

11.12 WRITE FILE (0x08)

The Write File command stores one or more "files", which are four consecutive bytes of EDAC protected memory. A write to address 0 is a special case, and an additional mode byte is stored.

Note that because of the single byte of addressing, not all of the EDAC memory can be accessed by this command.

When multiple parameter files are written by a single command, they are guaranteed to be internally consistent (i.e. into the same control frame). Files can be written in any order, and a single file can be written multiple times.

If a Write File command fails due to improper formatting then no modification to EDAC memory is made.

11.12.1 Command Format

Bytes 0-N	List of File Store structures. The first byte of each structure determines its type.

11.12.1.1 Mode Store Structure

Byte 0	0
Byte 1	Command type to store
Bytes 2 - 5	Command value to store

11.12.1.2 Normal Store Structure

Byte 0	Non-zero EDAC address divided by 4 (1 – 255)
Bytes 1 - 4	Data bytes to write to EDAC memory

11.12.1 Reply Format

Bytes 0-N	List of File Reply structures. The first byte of each structure determines its type.

11.12.1.1 Mode Reply Structure

Byte 0	0
Byte 1	Command type read from EDAC
Bytes 2 - 5	Command value read from EDAC

11.12.1.2 Normal Reply Structure

Byte 0	Non-zero EDAC address divided by 4 (1 – 255)
Bytes 1 - 4	EDAC data bytes read from memory

11.13 READ EDAC (0x09)

The Read EDAC command returns bytes from EDAC memory. The read process is atomic. Long and short command formats are available.

11.13.1 Short Command Format

Bytes 0 – 1	EDAC address to start reading
Byte 2	Number of bytes to read. A value of 0 indicates that 256 bytes should be read.

11.13.2 Long Command Format

Bytes 0 – 1	EDAC address to start reading
Bytes 2 - 3	Number of bytes to read.

11.13.3 Reply Format

Bytes 0 – 1	EDAC address where reading started
Bytes 2 – N	The data bytes read from EDAC memory

11.14 WRITE EDAC (0x0A)

The Write EDAC command writes bytes into EDAC memory. The write process is atomic.

11.14.1 Command Format

Bytes 0 – 1	EDAC address to start writing
Bytes 2 – N	Data bytes to write to EDAC memory

11.14.2 Reply Format

Bytes 0 – 1	EDAC address where writing started
Bytes 2 – N	The data bytes written to EDAC memory

11.15GATHER EDAC (0X0B)

The Gather EDAC command allows multiple separate ranges of EDAC memory to be read in an atomic manner.

11.15.1 Command Format

Bytes 0 – N	List of gather command structures

11.15.1.1 Gather Command Structure

Bytes 0 – 1	EDAC address to start reading
Bytes 2 – 3	Number of bytes to read

11.15.1 Result Format

Bytes 0 – N	List of gather result structures

11.15.1.1 Gather Result Structure

Bytes 0 – 1	EDAC address where reading started
Bytes 2 – 3	Number of bytes read

Bytes 4 – N	Data bytes

12Protocol Layer 6 (Presentation Layer)

12.1Fault Handling

The application program can detect 7 different fault conditions:

- Motor windings overtemperature
- Processor undertemperature
- Drive transistors overtemperature
- PCA temperature difference too large
- Rotor overspeed
- Motor overcurrent
- Hall sensor error

For the first 6, the fault threshold (temperature, speed, current) can be adjusted. A Hall sensor error is declared when the HALL_IMPOSSIBLE or HALL_SKIP count increments. Each of these fault conditions sets a unique fault flag. Each fault can also be masked.

If an unmasked fault occurs, the motor drive is turned off. The rotor will slowly coast to a halt under friction alone.

The user can detect a fault by reading FLAGS_ACTIVE. The user can clear a fault by writing '0' to the corresponding fault flag. Once all fault flags are clear or masked, the wheel will resume normal function.

Users are cautioned against clearing or masking faults without understanding the cause. The fault logic exists to protect the hardware. For example, running at high torque with the overtemperature faults masked risks damaging the motor from overheat.

12.2Memory Map

Table 15: Processor Memory Map

Address Range	Function
0x00000000 – 0x0001FFFB	Program RAM
0x0001FFFC – 0x0001FFFF	Program RAM ECC trap word
0x10000000 – 0x10007FFB	Data RAM
0x10007FFC – 0x10007FFF	Data RAM ECC trap word
0x20000000 – 0x2003FFFF	Bootloader FRAM
0x20040000 – 0x2007FFFF	User FRAM
0x40000000 – 0xFFFFFFFF	Hardware registers

The processor memory can be directly accessed with PEEK and POKE commands, and CRCs calculated with CRC commands. It is represented as a single 32-bit memory space, sparsely populated.

12.2.1 Program RAM

Code is executed from this memory. At reset the first 128 kB of bootloader FRAM is copied into program RAM.

12.2.2 Data RAM

This memory is used as software scratchpad. At power-up it is cleared to '0'. It is not cleared by other resets.

12.2.3 Bootloader FRAM

This non-volatile memory is programmed at the factory, and then write-protected. Starting at the lowest address it stores the bootloader. It may also store application programs and/or special test programs. The bootloader FRAM cannot be changed by the user.

12.2.4 User FRAM

This non-volatile memory will be delivered from the factory with the application program loaded. The program can be patched or replaced at any time using a sequence of POKE commands. This permits software updates after delivery, and even on-orbit.

12.2.5 Application Images

Application image structures can be stored in FRAM, to be loaded and executed by INIT commands. The format is:

Table 16: Application Image Structure

Offset	Length	Function
0x00	4	RAM target address (32-bit unsigned integer)
0x04	4	Data length (32-bit unsigned integer)
0x08	4	Start address (32-bit unsigned integer)
0x0C	Variable	1 or more bytes of image data

When an INIT command is executed on an application image the processor will copy a number of bytes equal to "Data length" from the "image data" field to the "RAM target address". It will then branch to the start address, which should be a pointer into RAM. In the special case that the start address is zero, no branch is made.
No integrity checks are made on the image format. Bad data (target address outside of RAM, data length too long, etc) will result in faults.

12.2.6 Error Mitigation

The RAM areas are protected from Single Event Upset (SEU) by hardware Error Correcting Codes (ECC). Each 8-bit byte is stored along with a 5-bit syndrome code. When the byte is read, the syndrome is re-calculated. Single-bit errors are automatically and transparently corrected and fixed. Multi-bit errors cannot be fixed. Both RAM areas are hardware scrubbed on a timescale of seconds.
The FRAM areas are protected by a hardware power switch. Whenever FRAM is not being used it is powered down. When FRAM is required the switch is turned ON and a 1 msec delay is observed before access. In typical orbital use the FRAM is turned ON for 380 msec after every reset, and is then OFF for normal operation.

12.2.7 ECC Trap Words

The last words in both of the RAM areas have their ECC compromised, as the stored syndrome code is always '0'. When the stored word is '0', the ECC shows no errors. If a byte with a single '1' bit is written, a single-bit error is simulated. ECC hardware will correct the error. If a byte with two '1' bits is written, a multi-bit error is simulated. ECC hardware will count the error, but will be unable to correct it.

The trap words can be used on the ground for self-test purposes. These words should not be used for actual data storage.

12.3 Diagnostics

The diagnostics contain a series of read-only integers that relate to the health of the wheel.

Table 17: Diagnostic Channels

Diagnostic Channel	Function	Format
0x00	Reset reason	32-bit enum: 0: Power-on 1: Reset pin 2: Lockup 3: Watchdog 4: Memerr 5: INIT command 6: Hardfault exception 7: Unknown
0x01	Reset count	32-bit unsigned integer Value = 1 after power-on
0x02	Data RAM SEU count	16-bit unsigned integer x 2 Low bytes: Single bit error count High bytes: Multi bit error count
0x03	Program RAM SEU count	16-bit unsigned integer x 2 Low 2 bytes: Single bit error count High 2 bytes: Multi bit error count
0x04	Bootloader retries count (from most recent reset)	Byte 0: 8-bit unsigned integer Bytes 1-3: ‘0’
0x05	Serial number	32-bit unsigned integer Unique, but maybe not sequential
0x06	FRAM status bytes	Byte 0: Bootloader FRAM status Byte 1: User FRAM status Bytes 2-3: ‘0’
0x07	RS485-0 NSP Framing error count	32-bit unsigned integer
0x08	RS485-0 Runt count	32-bit unsigned integer
0x09	RS485-0 Oversize count	32-bit unsigned integer
0x0A	RS485-0 Bad CRC count	32-bit unsigned integer
0x0B	RS485-0 FIFO overflow count	32-bit unsigned integer
0x0C	RS485-0 Incoming discarded count	32-bit unsigned integer
0x0D	RS485-0 Outgoing discarded count	32-bit unsigned integer
0x0E	RS485-1 NSP Framing error count	32-bit unsigned integer

0x0F	RS485-1 Runt count	32-bit unsigned integer
0x10	RS485-1 Oversize count	32-bit unsigned integer
0x11	RS485-1 Bad CRC count	32-bit unsigned integer
0x12	RS485-1 FIFO overflow count	32-bit unsigned integer
0x13	RS485-1 Incoming discarded count	32-bit unsigned integer
0x14	RS485-1 Outgoing discarded count	32-bit unsigned integer
0x15	Fault register R0	32-bit unsigned integer
0x16	Fault register R1	32-bit unsigned integer
0x17	Fault register R2	32-bit unsigned integer
0x18	Fault register R3	32-bit unsigned integer
0x19	Fault register R12	32-bit unsigned integer
0x1A	Fault register LR	32-bit unsigned integer
0x1B	Fault register PC	32-bit unsigned integer
0x1C	Fault register xPSR	32-bit unsigned integer
0x1D	Application restart address	32-bit pointer
0x1E	Application restart mask	32-bit bitmap: Bit 0: Reserved Bit 1: Restart on reset pin Bit 2: Restart on lockup Bit 3: Restart on watchdog Bit 4: Reserved Bit 5: Reserved Bit 6: Restart on hardfault Bit 7: Restart on unknown reset

FRAM status bytes can be expected as:

Byte value	Meaning
0x40	FRAM chip is unlocked
0xCC	FRAM chip is write-protected

RS485 port error counters are zeroed at power-on, but are not cleared at subsequent resets.

The fault registers store the contents of the key processor registers at the moment of hardfault, memmanage, busfault and usagefault exceptions. They may be used to help debug the cause of the fault. Note that since the Data RAM is not cleared on an exception reset, it may be possible to use subsequent PEEKs to get more information about the machine state when the exception occurred.

The application restart registers are zeroed when an application is called, and written by the application as desired. After a reset, the bootloader looks at the restart mask to determine whether an autonomous application restart should be attempted. If so, it jumps to the restart address.

12.4 EDAC Memory

For historical reasons, the memory area used to store commands and telemetry is referred to as "EDAC Memory". In older wheels, this special area was subject to software-based triple-voting and scrubbing. In this wheel every single flip-flop is protected against upset. There is no more or less protection for the EDAC Memory, but the name is preserved.

The memory area is 1536 bytes long. EDAC memory can be read with READ EDAC, GATHER EDAC and READ FILE commands, and written with WRITE EDAC and WRITE FILE commands. The MODE_STORE command will save EDAC memory into non-volatile User FRAM.

Table 18: EDAC Memory Contents

EDAC Address	File Address	Function	Format
0x000 – 0x003	0x00	Command Value	Command Dependent
0x00C – 0x00F	0x03	VBUS	Volts (IEEE-754)
0x020 – 0x023	0x08	VCC	Volts (IEEE-754)
0x040 – 0x043	0x10	TEMP0	°C (IEEE-754)
0x048 – 0x04B	0x12	TEMP2	°C (IEEE-754)
0x04C – 0x04F	0x13	TEMP3	°C (IEEE-754)
0x054 – 0x057	0x15	SPEED	Rad/sec (IEEE-754)
0x058 – 0x05B	0x16	MOMENTUM	N-m/sec (IEEE-754)
0x068 – 0x06B	0x1A	PWM	Duty cycle (IEEE-754)
0x06C – 0x06F	0x1B	HALL_DIGITAL	Enum in IEEE-754 float
0x074 – 0x077	0x1D	ADC_CALIBRATE	Ratio (IEEE-754)
0x080 – 0x083	0x20	SPEED_P_GAIN	Amps / rad / sec (IEEE-754)
0x084 – 0x087	0x21	SPEED_I_GAIN	Amps / rad (IEEE-754)
0x088 – 0x08B	0x22	SPEED_D_GAIN	Amps / rad / sec^2 (IEEE-754)
0x094 – 0x097	0x25	MAX_GAIN_SPEED	Rad/sec (IEEE-754)
0x098 – 0x09B	0x26	MIN_GAIN_SPEED	Rad/sec (IEEE-754)
0x0A0 – 0x0A3	0x28	INERTIA	kg-m^2 (IEEE-754)
0x0A4 – 0x0A7	0x29	MOTOR_KT	N-m/A (IEEE-754)
0x0A8 – 0x0AB	0x2A	GAIN_SCHEDULE1	(IEEE-754)
0x0AC – 0x0AF	0x2B	GAIN_SCHEDULE2	(IEEE-754)
0x0B0 – 0x0B3	0x2C	GAIN_SCHEDULE3	(IEEE-754)
0x0B4 – 0x0B7	0x2D	GAIN_SCHEDULE4	(IEEE-754)
0x0B8 – 0x0BB	0x2E	PROPORTIONAL_OVERRIDE	(IEEE-754)
0x0BC – 0x0BF	0x2F	CONTROL_TYPE	(IEEE-754)
0x0C8 – 0x0CB	0x32	MAX_SPEED_AGE	sec (IEEE-754)

0x0CC – 0x0CF	0x33	LIMIT_SPEED	Rad/sec (IEEE-754)
0x0D4 – 0x0D7	0x35	LIMIT_CURRENT	Amps (IEEE-754)
0x0E4 – 0x0E7	0x39	MOTOR_RESISTANCE	Ohms (IEEE-754)
0x0EC – 0x0EF	0x3B	SINUSOID_PHASE	Rad (IEEE-754)
0x0F0 – 0x0F3	0x3C	SINUSOID_FREQ	Hz (IEEE-754)
0x0F4 – 0x0F7	0x3D	SINUSOID_OFFSET	Rad/sec or Volts (IEEE-754)
0x100 – 0x103	0x40	PREVIOUS_SPEED	Rad/sec (IEEE-754)
0x104 – 0x107	0x41	SPEED_INTEGRATOR	Amps (IEEE-754)
0x108 – 0x10B	0x42	SPEED_LAST_ERROR	Rad/sec (IEEE-754)
0x10C – 0x10F	0x43	ACCEL_TARGET	Rad/sec (IEEE-754)
0x12C – 0x12F	0x4B	TORQUE_T0	Nm (IEEE-754)
0x130 – 0x133	0x4C	TORQUE_T1	Nm (IEEE-754)
0x134 – 0x137	0x4D	TORQUE_T2	Nm (IEEE-754)
0x138 – 0x13B	0x4E	TORQUE_T3	Nm (IEEE-754)
0x13C – 0x13F	0x4F	TORQUE_T4	Nm (IEEE-754)
0x168 – 0x16B	0x5A	SLEEP_DUTY	Fraction (IEEE-754)
0x16C – 0x16F	0x5B	DCDC_FREQ	Hz (IEEE-754)
0x178 – 0x17B	0x5E	DRIVE_FREQ	Hz (IEEE-754)
0x184 – 0x187	0x61	RESPONSE_AMPLITUDE	Rad/sec (IEEE-754)
0x188 – 0x18B	0x62	RESPONSE_PHASE	Rad (IEEE-754)
0x190 – 0x193	0x64	KT_ESTIMATE	N-m/A (IEEE-754)
0x194 – 0x197	0x65	R_ESTIMATE	Ohms (IEEE-754)
0x198 – 0x19B	0x66	DV_ESTIMATE	Volts (IEEE-754)
0x19C – 0x19F	0x67	DRY_FRICTION_ESTIMATE	Nm (IEEE-754)
0x1A0 – 0x1A3	0x68	WET_FRICTION_ESTIMATE	N-m/rad/sec (IEEE-754)
0x1A4 – 0x1A7	0x69	AERO_FRICTION_ESTIMATE	$N\text{-}m/rad^2/sec^2$ (IEEE-754)
0x1A8 – 0x1AB	0x6A	RUNDOWN_TIME	sec (IEEE-754)
0x1C0 – 0x1C3	0x70	FAULT_OVERTEMP0	°C (IEEE-754)
0x1C4 – 0x1C7	0x71	FAULT_UNDERTEMP2	°C (IEEE-754)
0x1C8 – 0x1CB	0x72	FAULT_OVERTEMP3	°C (IEEE-754)
0x1CC – 0x1CF	0x73	FAULT_TEMP_DELTA	°C (IEEE-754)
0x1D0 – 0x1D3	0x74	FAULT_OVERSPEED	Rad/sec (IEEE-754)
0x1D4 – 0x1D4	0x75	FAULT_OVERCURRENT	Amps (IEEE-754)
0x200 – 0x203	0x80	TEMP_R0	Ohms (IEEE-754)
0x204 – 0x207	0x81	TEMP_R2	Ohms (IEEE-754)
0x208 – 0x20B	0x82	TEMP_R4	Ohms (IEEE-754)
0x20C – 0x20F	0x83	ADC_RAW_VBUS	Ratio (IEEE-754)
0x210 – 0x213	0x84	ADC_RAW_VCC	Ratio (IEEE-754)

0x214 – 0x217	0x85	ADC_RAW_TEMP0	Ratio (IEEE-754)
0x218 – 0x21B	0x86	ADC_RAW_TEMP2	Ratio (IEEE-754)
0x21C – 0x21F	0x87	ADC_RAW_TEMP3	Ratio (IEEE-754)
0x220 – 0x223	0x88	ADC_RAW_CALIBRATE	Ratio (IEEE-754)
0x5C3		MODE	8-bit enum
0x5CE		HALL_IMPOSSIBLE	8-bit unsigned int
0x5CF		HALL_SKIP	8-bit unsigned int
0x5D0		CONTROL_OVERFLOW	8-bit unsigned int
0x5D1		SPEED_TABLE_SIZE	8-bit unsigned int
0x5D2		USED_TABLE_SIZE	8-bit unsigned int
0x5D6		IDLE_INHIBIT	8-bit boolean
0x5D7		FLAGS_ACTIVE	8-bit bitmap
0x5D8		FAULTS_MASK	8-bit bitmap
0x5D9		FLAG_OVERTEMP0	8-bit boolean
0x5DA		FLAG_UNDERTEMP2	8-bit boolean
0x5DB		FLAG_OVERTEMP3	8-bit boolean
0x5DC		FLAG_TEMP_DELTA	8-bit boolean
0x5DD		FLAG_OVERSPEED	8-bit boolean
0x5DE		FLAG_OVERCURRENT	8-bit boolean
0x5DF		FLAG_HALL_ERROR	8-bit boolean
0x5E0		HALT	8-bit boolean
0x5E1		RESET_ENABLE	8-bit bitmap
0x5E2		RESTART_MASK	8-bit bitmap
0x5E3		STARTUP_DELAY	8-bit integer

12.4.1 Command Value

Accessing file 0 causes an extra mode byte to be transferred. By writing to this file the mode of the wheel can be commanded. By reading this file the current mode can be determined. The modes are enumerated in section 12.4.45.

If this parameter is accessed through EDAC writes and reads instead of file reads and writes there is no explicit mode byte transferred. It is possible to read and write the number associated with the command, but this is not advised.

12.4.2 VBUS

This read-only parameter returns the voltage on the primary power bus. It is inferred by measuring the duty cycle of the DC/DC converter.

12.4.3 VCC

This read-only parameter returns the voltage on the +3.3 V nominal secondary output from the DC/DC converter.

12.4.4 TEMP0

ThIS read-only parameters return the temperature of the motor windings.

The sensors are NTC devices, so an open-circuit failure causes an apparent low temperature reading.

12.4.5 TEMP[2|3]

These read-only parameters return temperatures on the circuit board. TEMP2 is located immediately next to the processor. TEMP3 is located adjacent to the motor drive transistors.

12.4.6 TEMP_R[0|2|3]

These read-only parameters return the calculated thermistor temperatures, corresponding to TEMP0, TEMP2 and TEMP3. The thermistors are specified at 10 kΩ at +25 C. These parameters are intended primarily for initial factory calibration.

12.4.7 SPEED

This read-only parameter returns the speed of the rotor.

12.4.8 MOMENTUM

This read-only parameter returns the angular momentum of the rotor. It is derived from the SPEED multiplied by INERTIA.

12.4.9 HALL_DIGITAL

This read-only parameter returns the state of the three digital Hall-effect sensors. Each switch can be in one of two states: '0' and '1'. The state can be decoded from the following table:

Table 19: Digital Hall-effect sensor status codes

HALL_DIGITAL	Hall 0	Hall 1	Hall 2
0.0	0	0	0
1.0	1	0	0
2.0	0	1	0
3.0	1	1	0
4.0	0	0	1
5.0	1	0	1
6.0	0	1	1
7.0	1	1	1

Note that codes 0.0 and 7.0 should not be mechanically possible.

12.4.10 ADC_CALIBRATE

A read-only parameter, updated frequently from the self-calibrate cycle of the ADC. It is roughly analogous to the input offset voltage of the ADC. It will be a value, positive or negative, on the order of 0.01. It varies with temperature.

12.4.11 SPEED_[P|I|D]_GAIN

These read-only parameters set the gains for the PID closed-loop speed controller. See CONTROL_TYPE for the formula to determine the gains.

12.4.12 MIN_GAIN_SPEED, MAX_GAIN_SPEED

These read/write parameters bound the speed used as an input to the speed controller gain formula.

By setting these two parameters to the same value the speed dependence of the gains can be effectively disabled.

12.4.13 INERTIA

This read/write parameter sets the rotor inertia. It is used to scale between acceleration and torque, and momentum and speed.

12.4.14 MOTOR_KT

This read/write parameter sets the motor torque constant.

12.4.15 GAIN_SCHEDULE[1..4]

These four read/write parameters are used to set the speed control gains, in those cases when PROPORTIONAL_OVERRIDE is zero. First, the characteristic speed ω is determined based on the actual and setpoint speeds and on MAX_GAIN_SPEED and MIN_GAIN_SPEED.

$$\omega = MIN\left(MAX\left(\left|\omega_{actual}\right|,\left|\omega_{target}\right|,\omega_{MIN}\right),\omega_{MAX}\right)$$

The critical gain and period are modeled as a function of the characteristic speed. The four GAIN_SCHEDULE parameters are written as G1..G4.

$$Ku = G2 \cdot \omega^{G1}$$

$$Pu = 91.5Hz \cdot G4 \cdot \omega^{G3}$$

The gains are then set according to the Ziegler-Nichols method.

$$Kp = 0.6 \cdot Ku$$

$$Ki = {2.0 \cdot Kp}/{Pu}$$

$$Kd = 0.125 \cdot KpPu$$

12.4.16 PROPORTIONAL_OVERRIDE

This read/write parameter is used to override the gain settings, usually in a factory gain tuning context. When non-zero, the gains are set accordingly:

$$Kp = PROPORTIONAL\_OVERRIDE$$

$$Ki = 0.0$$

$$Kd = 0.0$$

12.4.17 CONTROL_TYPE

This read/write parameter is used to determine the control type, using the Ziegler-Nichols method.
The value stored in CONTROL_TYPE is truncated to an integer. If the value is 1, a PI controller is used:

$$Kp = 0.45 \cdot Ku$$

$$Ki = {1.2 \cdot Kp}/{Pu}$$

$$Kd = 0.0$$

If the value is 2, a PID controller is used:

$$Kp = 0.6 \cdot Ku$$

$$Ki = {2.0 \cdot Kp}/{Pu}$$

$$Kd = 0.125 \cdot KpPu$$

In the case of any other value, a P controller is used:

$$Kp = 0.5 \cdot Ku$$

$$Ki = 0.0$$

$$Kd = 0.0$$

12.4.18 MAX_SPEED_AGE

This read/write parameter determines which digital Hall sensor transitions are used to determine the SPEED telemetry. Transitions are discarded if they are older than MAX_SPEED_AGE in time, if a complete rotor revolution has occurred since them, or if a rotor direction reversal is detected.

MAX_SPEED_AGE is relevant at very low rotor speeds. A larger value will allow more Hall sensor transitions to be used, giving a less noisy speed estimate. However, it will also increase the latency in speed measurements which may cause closed-loop speed control modes to become unstable.

12.4.19 LIMIT_SPEED

This read/write parameter sets the maximum speed that closed-loop modes will target. The magnitude of the speed target used in speed, torque, momentum and acceleration modes is clamped to this value. This is particularly significant in torque and acceleration modes – if communication with the flight computer is lost for any reason the rotor will slowly accelerate until this limit is reached.

12.4.20 LIMIT_CURRENT

This read/write parameter sets the greatest motor drive current used by closed-loop modes. Reducing this value will limit the torque that the wheel can generate.

12.4.21 MOTOR_RESISTANCE

This read/write parameter sets the motor nominal resistance.

12.4.22 SINUSOID_[PHASE, FREQ, OFFSET]

Please see SINUSOID mode for details.

12.4.23 PREVIOUS_SPEED

This read-only parameter contains the SPEED file from the previous control frame. It is expected that it might be used in the future to generate torque telemetry, but at present it is unused.

12.4.24 SPEED_INTEGRATOR

This parameter contains the closed-loop controller integrator, scaled in amps of actuation. It is technically a read/write parameter, and it is possible for the user to write this for test purposes.

12.4.25 SPEED_LAST_ERROR

This read-only parameter contains the controller error from the previous control frame. It is used with the differential gain term of the closed-loop controller.

12.4.26 ACCEL_TARGET

This parameter contains the speed setpoint used by the acceleration controller. The controller will add the acceleration to this file each frame. It is technically a read/write parameter, and it is possible for the user to write this as a way to force a new speed while remaining in acceleration/torque mode.

12.4.27 TORQUE_[T0..T4]

These five read-only parameters record the instantaneous torques measured in the last five control frames. T0 is the result of the most recent control frame. T4 is four frames old (40 msec). The torque is computed as:

TORQUE = INERTIA * (SPEED – PREVIOUS_SPEED) * 100 Hz

Torque telemetry at low speed should be used with caution. The speed estimate is only updated when new hall sensor pulses are seen (or a very long period elapses). If there has been no hall sensor pulse in the previous control frame then SPEED == PREVIOUS_SPEED and so TORQUE == 0.

12.4.28 SLEEP_DUTY

This read-only parameter shows the fraction of the last 10 msec control frame that the processor has spent in the sleep mode. It provides a conservative measure of realtime margin.
A value of 1.0 would indicate continual sleep. A value of 0.0 would indicate that the processor is never sleeping, possibly due to the IDLE_INHIBIT parameter.

12.4.29 DCDC_FREQ

This read-only parameter shows the frequency of the internal DC/DC converter, averaged over the last 10 cycles. The expected value is on the order of 100 kHz.

12.4.30 DRIVE_FREQ

This read/write parameter sets the switching frequency of the motor drive inverter. Permissible values are in the range of 100 kHz to 300 kHz.
As a special case, a value of 0.0 will synchronize the motor drive inverter to the DC/DC converter.

12.4.31 RESPONSE_AMPLITUDE, RESPONSE_PHASE

These read-only parameters are updated every time SINUSOID_PHASE wraps around from 2π to 0. They measure the speed oscillation that has resulted from a SINUSOID_SPEED or SINUSOID_VOLTAGE command.
RESPONSE_AMPLITUDE is the amplitude of the speed sinusoid.
RESPONSE_PHASE is the phase of the speed sinusoid with respect to the excitation sinusoid – expect a negative number which indicates a phase lag.

12.4.32 FAULT_OVERTEMP[0|3]

These read/write parameters set the fault limits for TEMP0 and TEMP3. If the temperature exceeds this threshold, even for a single measurement cycle, the corresponding fault flag will be set.
These limits detect overtemperature in the motor windings (channel 0) and the drive transistors (channel 3). These are areas that could be expected to get hot from continual high-torque operation.

12.4.33 FAULT_UNDERTEMP2

This read/write parameter set the fault limit for TEMP2. If the temperature is below this this threshold, even for a single measurement cycle, the corresponding fault flag will be set.
The limit detects undertemperature in the processor (channel 2). This is a proxy for bearing temperature, and can be used to prevent very low temperature startup.

12.4.34 FAULT_TEMP_DELTA

This read/write parameter set the fault limit for the difference between TEMP2 and TEMP3. If the absolute value of this difference exceeds the threshold, even for a single measurement cycle, the corresponding fault flag will be set.
Both TEMP2 and TEMP3 are mounted to the PCA. A large temperature split between them suggests large power dissipation on the board.

12.4.35 FAULT_OVERSPEED

This read/write parameter set the fault limit for SPEED. If the absolute value of SPEED exceeds the threshold, even for a single measurement cycle, the corresponding fault flag will be set. This parameter is complementary to LIMIT_SPEED. While LIMIT_SPEED only works in closed-loop modes, FAULT_OVERSPEED works in all modes. By setting one limit higher than the other, the behavior of closed-loop modes can be changed – they can hold at LIMIT_SPEED, or they can fault immediately upon reaching a limit.

12.4.36 FAULT_OVERCURRENT

This read/write parameter set the fault limit for current. If the computed absolute value of motor current exceeds the threshold, even for a single measurement cycle, the corresponding fault flag will be set. This parameter is complementary to LIMIT_CURRENT. While LIMIT_CURRENT only works in closed-loop modes, FAULT_OVERCURRENT works in all modes.
For this limit to be effective, MOTOR_KT and MOTOR_RESISTANCE must be set correctly.

12.4.37 KT_ESTIMATE, R_ESTIMATE

Every time that SINUSOID_VOLTAGE updates RESPONSE_AMPLITUDE and RESPONSE_PHASE, these two read-only parameters are also updated. They are the values of motor KT and R that best fit the response sinusoid, given the INERTIA parameter that is assumed to be known.
The user must copy these values over to MOTOR_KT and MOTOR_RESISTANCE as part of an initial calibration operation – this is not done automatically.

12.4.38 DRY_FRICTION_ESTIMATE, WET_FRICTION_ESTIMATE, AERO_FRICTION_ESTIMATE, RUNDOWN_TIME

When the RUNDOWN mode completes, these read-only parameters are updated. The friction estimates are the best fit to the function:

$$TotalFriction = DryFriction + WetFriction * speed + AeroFriction * speed^2$$

The sign of DRY_FRICTION_ESTIMATE is the same as the sign of the starting speed for the rundown. A rundown from a positive speed will result in a positive dry friction estimate, and a negative starting speed will result in a negative dry friction estimate. The wet friction estimate should always be positive.
RUNDOWN_TIME measures the time, in seconds, for the wheel speed to reach zero.

12.4.39 MODE

This parameter stores the wheel's current mode. It is more often accessed through file 0, where the mode and command value can be read or written simultaneously.

12.4.40 HALL_IMPOSSIBLE

This value counts the number of times that a transition to an "impossible" digital Hall-effect sensor configuration is seen. Impossible configurations are all "0", or all "1". This is an error condition, and would normally indicate failure of a sensor or loss of a rotor magnet. It is read/write, and can be written as zero to reset the count. The count range is 0..255. If an impossible configuration occurs with the count at 255 it will cycle back to 0.

12.4.41 HALL_SKIP

This value counts the number of times that a Hall-effect sensor pattern transitions to another pattern that should not be immediately adjacent. Adjacent sensor patterns are those that differ by only one bit.

12.4.42 CONTROL_OVERFLOW

This value counts the number of control frames where the control algorithm has not finished processing before the start of the next frame. This is an error condition, and would be expected to result in poor control. It is read/write, and can be written as zero to reset the count. The count range is 0..255. If a control overflow occurs with the count at 255 it will cycle back to 0.

Issuing a CRC command over a large memory range is one sure-fire way to cause the count to increase.

12.4.43 SPEED_TABLE_SIZE

This value shows the number of digital Hall sensor transitions that are held in the transition table. Transitions are discarded if they are older than MAX_SPEED_AGE in time, if a complete rotor revolution has occurred since them, or if a rotor direction reversal is detected.

12.4.44 USED_TABLE_SIZE

This value shows the number of digital Hall sensor transitions that are being used to compute the SPEED estimate. The speed estimator will attempt to use the following number of transitions, in order of decreasing preference:

- A number of transitions equal to a full revolution, plus one. This is 3P+1, where P is the number of magnetic poles in the rotor. This is the most accurate estimate.
- A number of transitions in the form 6N+1, where N is as large a natural number as possible. This number nulls error from Hall sensor orientation and offset, but incurs error from uneven magnet spacing.
- Four transitions. This number nulls error from Hall sensor orientation. Hall sensor magnetic offset and uneven magnet spacing will introduce noise.
- Three transitions. This is suitable for very slow rotor speeds. All noise sources apply.
- Two transitions. This is suitable for even slower rotor speeds. All noise sources apply.
- If two transitions are not available, the speed is declared to be 0.0 rad/sec.

12.4.45 IDLE_INHIBIT

If this value is zero then the processor will go into a power-saving idle mode when not needed. It wakes immediately when interrupted, and there is no performance penalty. If this value is non-zero then the processor will stay on continually. Changing this parameter will show a modest change in power consumption.

12.4.46 FLAGS_ACTIVE

This read-only bitmap shows the present state of the fault flag bits, regardless of their mask state. It is provided as a convenience, so that 7 individual flag registers do not have to be read.

Table 20: FLAGS_ACTIVE bitfield

Bit	Function
0 (lsb)	FLAG_OVERTEMP0 value

1	FLAG_UNDERTEMP2 value
2	FLAG_OVERTEMP3 value
3	FLAG_TEMP_DELTA value
4	FLAG_OVERSPEED value
5	FLAG_OVERCURRENT value
6	FLAG_HALL_ERROR value
7	One or more unmasked faults are active

12.4.47 FAULTS_MASK

This read/write bitmap shows which flag bits can generate a fault. If a bit is set to '1', then the corresponding flag is masked and will not generate faults. If a bit is set to '0' then the flag is not masked and can generate faults.

Table 21: FAULTS_MASK bitfield

Bit	Function
0 (lsb)	FLAG_OVERTEMP0 masked
1	FLAG_UNDERTEMP2 masked
2	FLAG_OVERTEMP3 masked
3	FLAG_TEMP_DELTA masked
4	FLAG_OVERSPEED masked
5	FLAG_OVERCURRENT masked
6	FLAG_HALL_ERROR masked
7	Unused

12.4.48 FLAG_[OVERTEMP0|UNDERTEMP2|OVERTEMP3|OVERSPEED|OVERCURRENT|HALL_ERROR]

These read/write Boolean values show that a particular type of event has occurred. When the event happens, the flag is set to '1'. It can be cleared by the user writing '0'. The user can also write '1' to intentionally simulate an event.

12.4.49 ADC_RAW_[VBUS|VCC|TEMP0|TEMP2|TEMP3|CALIBRATE]

These read-only parameters show the raw ADC ratio for the corresponding telemetry channel. They are intended for factory calibration.

12.4.50 HALT

Writing a non-zero value to this parameter will cause the processor to intentionally become unresponsive (turn-off interrupts and busy-loop). It is intended as a mechanism to test watchdogs.

12.4.51 RESET_ENABLE

This read/write bitmap shows which processor errors can generate resets. If a bit is set to '1', then the corresponding reset is enabled.

Table 22: RESET_ENABLE bitfield

Bit	Function
0 (lsb)	Watchdog reset enabled
1	Data RAM multi-bit error reset enabled
2	Program RAM multi-bit error reset enabled
3-7	Unused

The watchdog feature piggy-backs on the DC/DC converter frequency and voltage estimation process. A counter counts DC/DC converter pulses. After 10 pulses are counted, an interrupt is triggered. The interrupt clears the counter. If the timer reaches a count of 15, and if the watchdog is enabled, then a processor reset is triggered.

Multi-bit RAM errors, which cannot be corrected by internal ECC, can be configured to cause resets. Single-bit errors are immediately corrected by hardware.

12.4.52 RESTART_MASK

This value is copied into the "application restart mask" field of the diagnostics. It determines which sorts of errors should lead to an autonomous application restart.

12.4.53 STARTUP_DELAY

When this value is non-zero, the effective control mode is IDLE and the fault comparators are inhibited. It is decremented once every control frame (i.e. at 100 Hz) until zero.
At application start or restart it is set to a value of 5, for a 50 msec delay. The intent is to let the rotor speed estimator settle before making any control actuation. This is most relevant if the application starts while the rotor is still spinning, as might be the case for an autonomous restart.

12.5 Command Modes

Command Number	Command Name	Command Value
0x00	IDLE	Ignored
0x01	PWM	Duty cycle (-1.0 to +1.0)
0x02	VOLTAGE	Volts (-Vbus to +Vbus)
0x03	SPEED	Rads/sec
0x04	PWM_H1	Duty cycle (0 to +1.0)
0x05	PWM_H2	Duty cycle (0 to +1.0)
0x06	PWM_H3	Duty cycle (0 to +1.0)
0x07	PWM_H4	Duty cycle (0 to +1.0)
0x08	PWM_H5	Duty cycle (0 to +1.0)
0x09	PWM_H6	Duty cycle (0 to +1.0)

0x0A	VOLTAGE_H1	Volts (0 to +Vbus)
0x0B	VOLTAGE_H2	Volts (0 to +Vbus)
0x0C	VOLTAGE_H3	Volts (0 to +Vbus)
0x0D	VOLTAGE_H4	Volts (0 to +Vbus)
0x0E	VOLTAGE_H5	Volts (0 to +Vbus)
0x0F	VOLTAGE_H6	Volts (0 to +Vbus)
0x10	ACCEL	Rads/sec^2
0x11	MOMENTUM	N-m-sec
0x12	TORQUE	N-m
0x16	STORE_FILES	0.0 or 1.0 or 2.0
0x17	DEFAULT_FILES	0.0 or 1.0
0x18	PWM_P0	Duty cycle (0 to +1.0)
0x19	PWM_P1	Duty cycle (0 to +1.0)
0x1A	PWM_P2	Duty cycle (0 to +1.0)
0x34	SINUSOID_SPEED	Rads/sec
0x35	SINUSOID_VOLTAGE	Volts
0x36	RUNDOWN	0.0 or 1.0

12.5.1 IDLE

In IDLE mode the motor drive is turned off. If it is spinning, the rotor is free to slow down under friction.

12.5.2 PWM

In PWM mode the motor is driven with a constant duty cycle. The command may be between -1.0 and 1.0. This is interpreted as a duty cycle between 0.0 and 1.0, in either the positive or negative direction.

PWM mode does not use closed-loop current or speed control, so it is not of great use in spacecraft fine control. However it does allow for extremely high torques (and very high power consumption!), so it may be used open-loop during slew maneuvers.

12.5.3 VOLTAGE

In VOLTAGE mode the motor is driven with closed-loop voltage control. Positive values indicate voltage that will produce positive speed, while negative values indicate voltage that will produce negative speed.

This mode can potentially be used for spacecraft fine control at high bandwidth.

12.5.4 SPEED

In SPEED mode the rotor speed is servoed to the command value. The closed-loop speed controller outputs a voltage setpoint, which is in turn used by the closed-loop voltage controller.

12.5.5 PWM_H[1..6]

In these modes the digital Hall-effect sensors are overridden, and the binary code is set to the H1..H6 value. Other than that, the mode is identical to PWM mode. It allows a

particular PWM duty cycle to be driven onto a particular motor phase regardless of the rotor position. The rotor will typically not spin in these modes, but will oscillate about a particular electrical angle.

12.5.6 VOLTAGE_H[1..6]

In these modes the digital Hall-effect sensors are overridden, and the binary code is set to the H1..H6 value. Other than that, the mode is identical to VOLTAGE mode. It allows a particular voltage to be driven onto a particular motor phase regardless of the rotor position. The rotor will typically not spin in these modes, but will oscillate about a particular electrical angle.

12.5.7 ACCEL

When not in ACCEL mode, the ACCEL_TARGET file is set to SPEED. In ACCEL mode, the acceleration command is added to ACCEL_TARGET each control frame. ACCEL_TARGET is then used as the setpoint for the speed mode controller.

12.5.8 MOMENTUM

In MOMENTUM mode, the SPEED controller is used with a setpoint equal to the commanded MOMENTUM divided by the INERTIA file.

12.5.9 TORQUE

In TORQUE mode, the ACCEL controller is used with a setpoint equal to the commanded TORQUE divided by the INERTIA file.

12.5.10 STORE_FILES

If the STORE_FILES mode is entered with a value of exactly 1.0, all of the parameters will be stored to user FRAM. The mode value will be set to 0.0, to indicate that the write has occurred and to prevent multiple writes. Whenever the wheel resets it will start with the stored parameters.

If the STORE_FILES mode is entered with a value of exactly 2.0, all of the parameters will be stored to bootloader FRAM. The mode value will be set to 0.0, to indicate that the write has occurred and to prevent multiple writes.

This operation is used at the factory, while the bootloader FRAM is still unlocked, to store hardware-specific calibration parameters. After the bootloader FRAM has been locked, subsequent commands will fail silently.

This mode does not drive the motor, and is equivalent in that way to IDLE.

12.5.11 DEFAULT_FILES

If the DEFAULT_FILES mode is entered with a value of exactly 1.0 the stored parameters in user FRAM are erased. The mode value will be set to 0.0, to indicate that the erasure has occurred and to prevent multiple erasures. Whenever the wheel resets it will start with default parameters from bootloader FRAM. This command has no effect on the parameters currently in the wheel parameter file, only on the parameters after the next reset.

This mode does not drive the motor, and is equivalent in that way to IDLE.

12.5.12 PWM_P[0..2]

The PWM_P[0..2] modes allow the duty cycle of a particular motor phase (0..2) to be set. Only the one phase is driven, and none of the phases is connected to ground.

12.5.13 SINUSOID_SPEED

This mode puts the wheel in a closed-loop state, tracking a sinusoidal speed profile. The amplitude of the sinusoid is given by the mode command, while the frequency and offset are given by the SINUSOID_FREQ and SINUSOID_OFFSET files.

$$Setpoint = amplitude \cdot sin(SINUSOID_{PHASE}) + SINUSOID_{OFFSET}$$
$$SINUSOID_{PHASE} \leftarrow (SINUSOID_{PHASE} + \Delta t \cdot SINUSOID_{FREQ}) modulo\ 2\pi$$

This mode can be used to characterize the closed-loop frequency response of the wheel. Be careful not to use too large an amplitude, as overheating can occur. A 100 rad/sec amplitude 1 Hz sinusoid is used in the factory to test the overtemperature shutdown.

12.5.14 SINUSOID_VOLTAGE

This mode is equivalent to SINUSOID_SPEED, except that it applies a sinusoidal voltage profile to the motor.

12.5.15 RUNDOWN

This mode is used to estimate the friction in the reaction wheel. To use it:

1. Spin the wheel to some starting speed, usually using a SPEED command.
2. Send the command RUNDOWN 1.0. The motor will turn off, and the wheel will coast to a stop.
3. Poll the COMMAND variable, until it turns to 0.0.
4. Read the DRY_FRICTION_ESTIMATE, WET_FRICTION_ESTIMATE parameters.
5. Read FRICTION_R, and determine whether the estimates should be trusted.
6. Read RUNDOWN_TIME.

RUNDOWN will only collect data when COMMAND is exactly 1.0. When the wheel speed hits zero, it clears COMMAND to 0.0.

13 Special Features

13.1 Virtual Oscilloscope

Various internal waveforms can be drive out of one of the RS485 ports, while the other port remains in normal operation for commands and telemetry. This is of no use on-orbit, but can be useful to help debug issues in an integrated system where it is hard to use an oscilloscope probe.
The RS485 port must be resistively terminated (120 ohms recommended), since the modulated signal is the drive-enable, not the output data.

To output a waveform on the RS485-0 port, perform the following sequence while in application mode:

- POKE 0x00006200 to 0x4000207C
- POKE 0x00000013 to 0x40020000 [Must be full 32-bit write]
- POKE 0x00000001 to 0x40020010
- POKE [Channel Number] to 0x40020014

To output a waveform on the RS485-1 port, perform the following sequence while in application mode:

- POKE 0x00006200 to 0x4000206C
- POKE 0x00000013 to 0x40031000 [Must be full 32-bit write]
- POKE 0x00000001 to 0x40031010
- POKE [Channel Number] to 0x40031014

Channel Number	Function	Notes
0x09	Processor awake/sleep state	
0x26	Hall sensor A	These can be used as an external speed tach output
0x28	Hall sensor B	
0x35	Hall sensor C	
0x41	DC/DC converter waveform	
0x4A	Motor phase B PWM	
0x4B	Motor phase A Enable	
0x4C	Motor phase A PWM	
0x4D	Motor phase B Enable	
0x4E	Motor phase C PWM	
0x4F	Motor phase C Enable	
0x54	Motor drive PWM	
0x7F	Clock divider 7	POKE 0x00000FA0 to 0x40000064 This generates a 10.00 kHz pulse train. Can be used to look for crystal oscillator drift over temperature.

A hardware reset is the easiest way to cancel the virtual oscilloscope configuration.

13.2 Inducing Processor Faults

It is useful to intentionally trigger various processor fault states, to verify the fault handler behavior.

Fault Type	Trigger Command	State
Hardfault	POKE 0x01 to 0xCAFEBABE	Bootloader or application

Note that these faults are executed immediately upon receipt of the POKE command. No NSP reply will be sent.

13.3 Autonomous Restart

The wheel processor is rad-hard, with ECC on all memories and registers. We might think from this that it would be guaranteed to operate correctly in all circumstances, assuming no software bugs. However, testing has shown that there is an ESD sensitivity. ESD events can cause processor hardfault and lockup events, and increment ECC counters. We hypothesize that this is due to spurious short clock pulses which leave an instruction half-executed.

The wheel can be configured to recover from these events with minimal user impact. To do this, set the RESTART_MASK byte. When an error occurs, the processor will reset and then immediately restart the application program. The EDAC memory will be untouched, including the command mode and value. Motor drive will resume after a 50-60 msec period of IDLE. The only signal to the user that this has occurred is an incremented reset count visible in the diagnostics.

Multi-bit memory errors can cause a reset if configured through RESET_ENABLE. Such an error never results in an autonomous restart – the wheel will instead remain in bootloader mode until commanded.