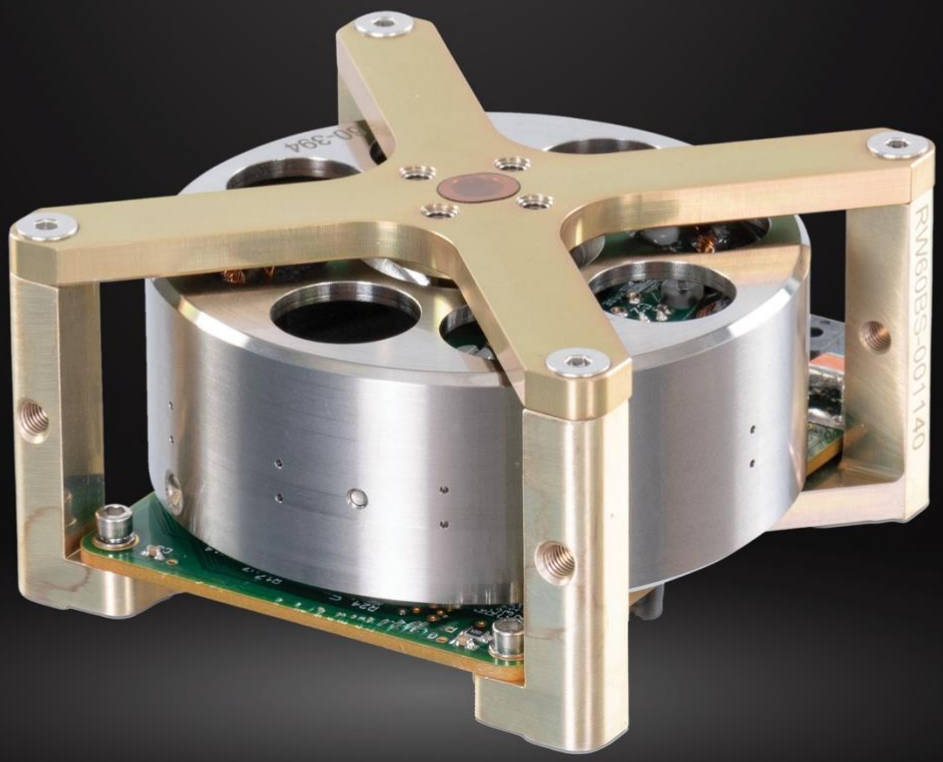

REACTION WHEEL

60 mNms RW-0.06

Interface Control Document

Rev. 2.0, June 28, 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 (1.14):

- Removed RW3-1.0 content as it has been superseded by the RW4-1.0
- Removed most references to electronics versions older than Rev 8.
- Ported addressing section (9.5) from “RW3-0.06 Addressing” document.

2 Scope

This document details the mechanical, electrical and software interfaces for the third generation Sinclair Interplanetary reaction wheels. At present these include:

- RW3-0.06-28-RS485
- RW3-0.06-28-RS485+CAN

3 Mechanical

3.1 RW3-0.06 Mechanicals

3.1.1 Mechanical Drawings

The following drawings refer to all units made with “Rev 8 RS485” electronics, and all units fabricated in 2016 and onwards. For earlier revisions, please consult the factory for the appropriate revisions of the ICD.

3.1.1 Top View

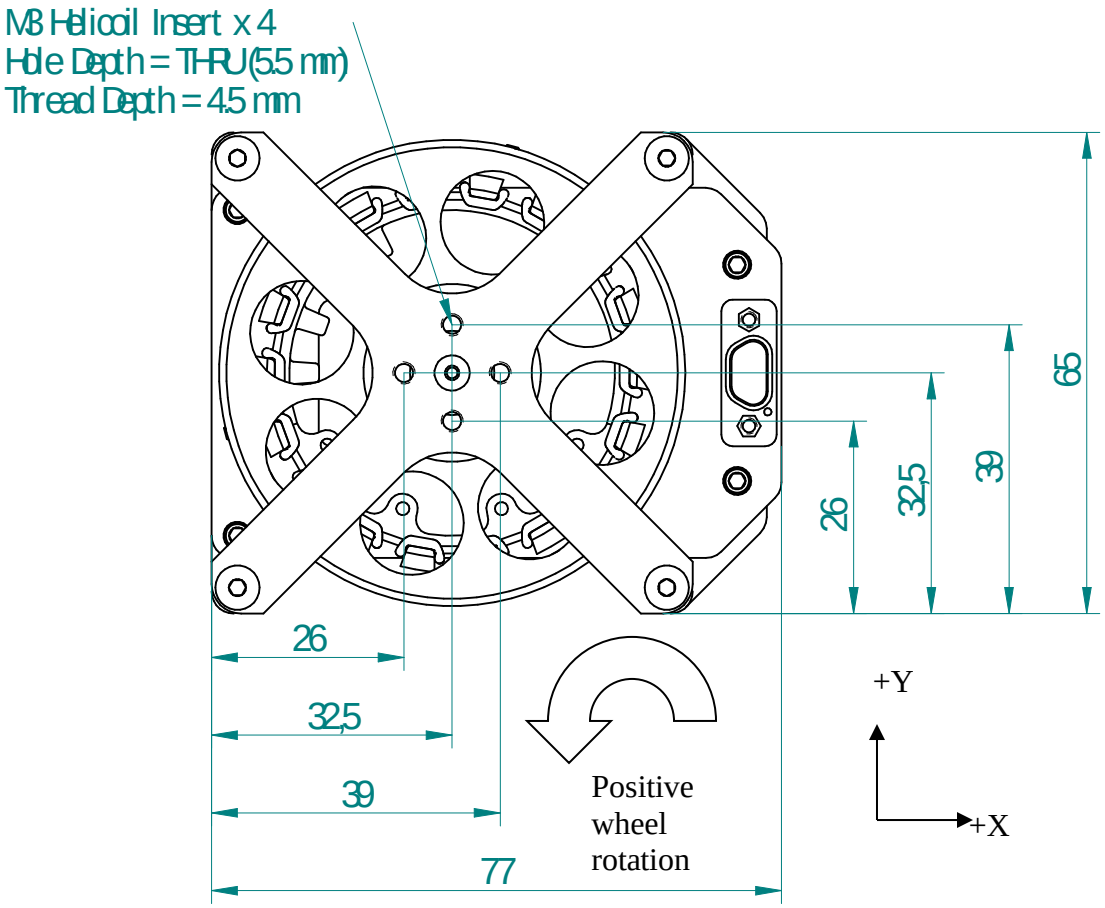

Figure 1: RW3-0.06 Top View

The X and Y axes are defined as shown in the figure. The Z-axis (illustrated in the following figures) completes the right-handed set. The rotation arrow shows the direction of wheel rotation that is considered positive speed. Rotation in the opposite direction is considered negative wheel speed.

The four holes in the top of the structure may be used by the customer to affix covers or other features. Take care not to use a fastener that is too long or it may protrude through the hole and contact the rotor.

3.1.1.1 Side View

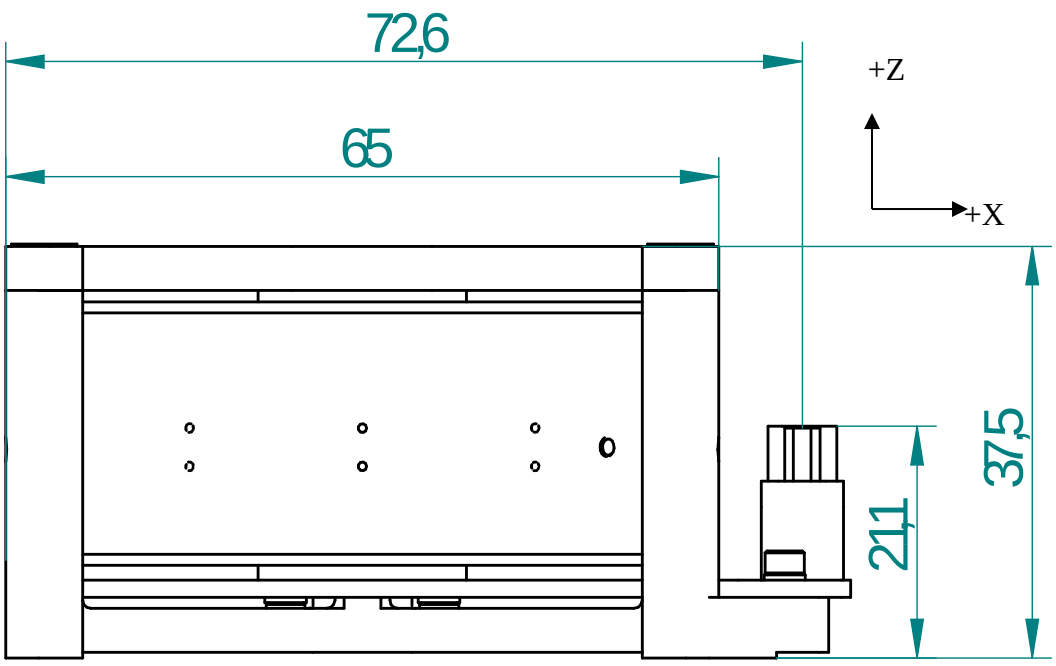

Figure 2: RW3-0.06 Side View

The overall height dimension (37.5 mm) is slightly variable from wheel to wheel due to the particular shimming of each part. When fitting a cover or other feature that must engage holes in both the top and front or back faces be sure that the mating holes are sufficiently oversized. Shimming tolerance is ±0.2 mm.

3.1.1.2 Front View

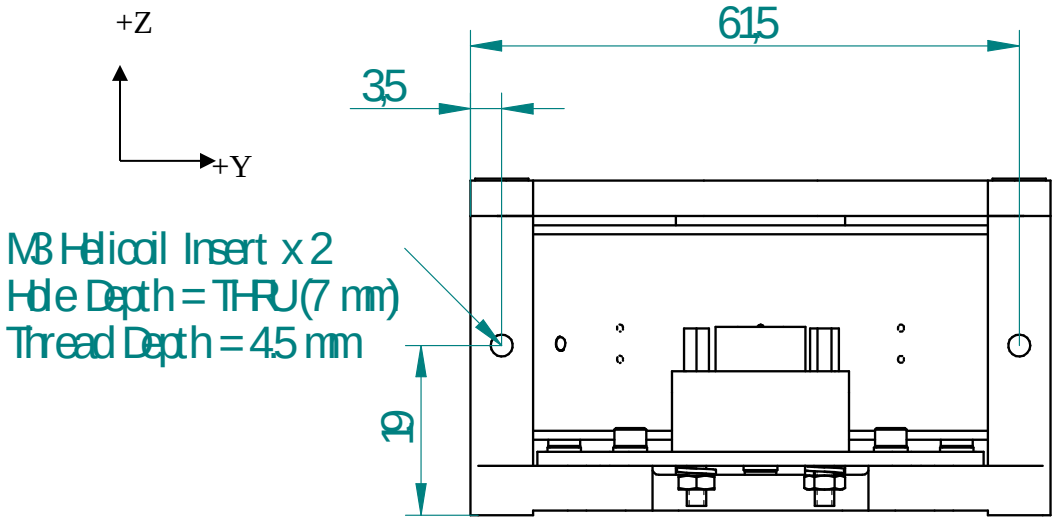

Figure 3: RW3-0.06 Front View

The two holes illustrated in this view may be used by the customer to affix covers or harness retaining features. Take care not to use a fastener that is too long or it may protrude through the hole and contact the rotor.

3.1.1.3 Bottom View

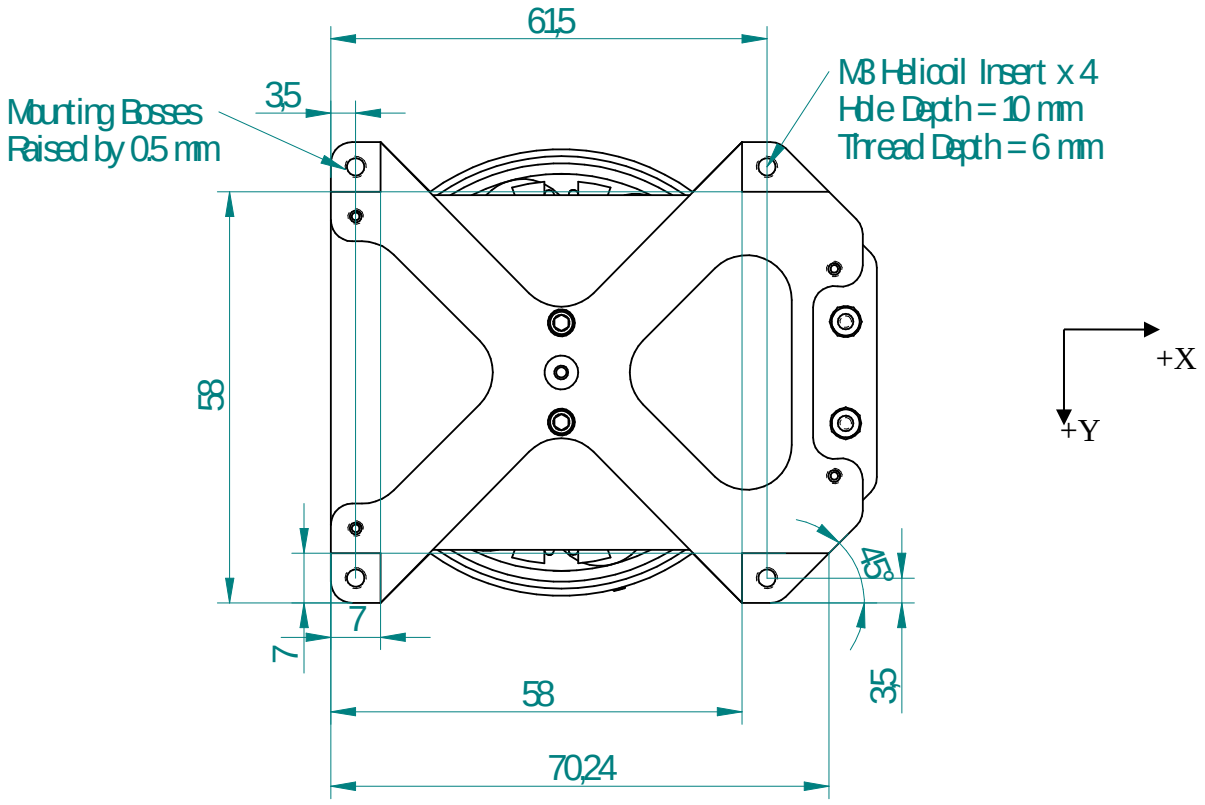

Figure 4: RW3-0.06 Bottom View

This view shows the bottom view of the wheel. This is the preferred mounting surface. There are four mounting bosses, each with a threaded hole to accept a fastener from below. The bosses are slightly taller than the rest of the wheel structure. This permits mounting on plates that may not be completely flat.

Be very careful not to over-torque the mounting screws. They thread into blind holes, and removing snapped fasteners is a difficult and expensive process.

3.1.1.4 Back View

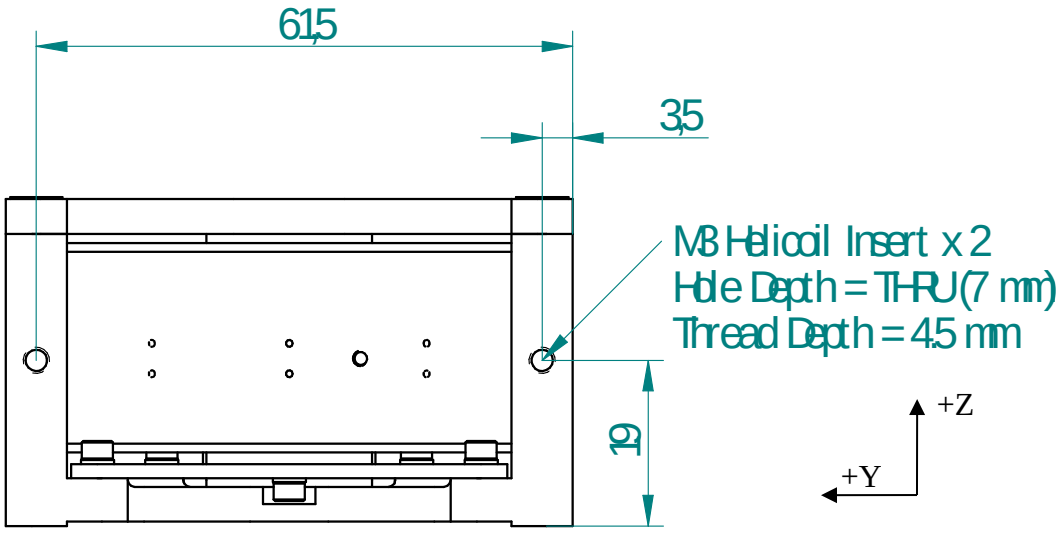

Figure 5: RW3-0.06 Back View

The two holes illustrated in this view may be used by the customer to affix covers or harness retaining features. Take care not to use a fastener that is too long or it may protrude through the hole and contact the rotor.

3.1.2 Mass Properties

Table 1: RW3-0.06 Mass Properties

Total Mass	< 0.235 kg
Rotating Mass	0.112 kg
CG Location	19.5 mm above mounting plane 2.1 mm forward (+X) of spin axis

Table 2: RW3-0.06 Moments of Inertia

	Ixx	Iyy	Izz
Rotor Inertia	5.98 x 10^{-5} kg-m^2	5.98 x 10^{-5} kg-m^2	8.66 x 10^{-5} kg-m^2
Inertia of Wheel Minus Rotor	7.55 x 10^{-5} kg-m^2	9.30 x 10^{-5} kg-m^2	8.86 x 10^{-5} kg-m^2
Inertia of Wheel with Rotor Locked	1.79 x 10^{-4} kg-m^2	1.97 x 10^{-4} kg-m^2	1.75 x 10^{-4} kg-m^2

3.1.3 Remove Before Flight

The following items may be removed before flight:

Table 3: RW-1.0 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
Kapton tape over bearings	May remove	If the spacecraft is clean, this may be removed to assist in quick venting on launch. If spacecraft is at all dusty, fly these covers.

4 Environmental

4.1 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. The wheel may be stored in a sealed bag with desiccant.

4.2 Thermal

Table 4: Allowable Temperature Range

Survival Temperature	-40°C to +125°C
Operating Temperature (short term)	-40°C to +100°C at interface
Operating Temperature (long term)	-20°C to +70°C at interface

Table 4 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.

4.3 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%.

4.4 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).

5 Electrical

5.1 Micro-D

The RW3-0.06 is fitted with a 9-socket micro-D connector.

Table 5: RW3-0.06 dual RS485 Micro-D Connector Pinout

Pin	Name
1	GND
2	RS485_1_A
3	RS485_0_A
4	GND
5	Address 0
6	V+A
7	RS485_1_B
8	RS485_0_B
9	V+A

Table 6: RW3-0.06 RS485+CAN Micro-D Connector Pinout

Pin	Name
1	GND
2	CAN_AH
3	RS485_0_A
4	GND
5	Address 0
6	V+A
7	CAN_AL
8	RS485_0_B
9	V+A

5.2 Programming Header

The RW3-0.06 has two holes in the PCB. For Rev 8 and earlier, these are fitted with silver-plated turret terminals. For Rev 9 and later they are simply gold-plated vias. These are for factory use, and allow the processor bootloader to be programmed. Customers should not use these without explicit factory advice.

6 Signals

6.1 Address [0|1]

Table 7: Address Input Electrical Specifications

Absolute Maximum, RW3-0.06	-50 V to +75 V, WRT Power Ground

The address input(s) allow the default network address of the wheel to be set. Depending on the exact wheel model, the pins may be strapped high, or low, or left open, or even connected to GND via resistors or diodes to set different behaviours. Consult the unit-specific EIDP for details.

6.2 CAN_A[H|L], CAN_B[H|L]

Table 8: CAN Electrical Specifications

Absolute Maximum	-36 V to +36 V, each signal, WRT Power Ground
ESD rating	±16 kV (Human-body model)
Input Common Mode Range	-7 V to +12 V
Dominant Input Voltage	VCANH – VCANL = 0.9 V to 3.3 V
Recessive Input Voltage	VCANH – VCANL = -1.0 V to +0.5 V
Input Hysteresis	0.1 V typ
Dominant Output Voltage	VCANH – VCANL > 2.3 V into 60 Ω load (28V)
Slew Rate	Configurable – Contact factory
Termination Resistor	Configurable – Contact factory

Each pair is CAN bus. Each pair features a common-mode filter. CAN buses may share pins with RS485 buses. The wheel has only one CAN controller, so even if two CAN buses are provided only one will be active at a time.

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

Table 9: RS485 Electrical Specifications

Absolute Maximum	-70 V to +70 V, each signal, WRT Power Ground
ESD rating	±16 kV (Human-body model)
Polarity	B > A in Mark (Idle) state A > B in Space (ON) state
Differential Output Voltage	> 1 V into 54 Ω termination.
Short-circuit Output Current	200 mA max
Input Resistance	> 96 kΩ each signal to Power Ground
Input Differential Threshold	-0.18 V to -0.035 V
Three-State Output Current	± 100 μA max

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. RS485 buses may share pins with CAN buses.

6.4 V+[A|B]

Table 10: V+ Electrical Specifications

Absolute Maximum, RW3-0.06	-60 V to +60 V, WRT Power Ground

Operating Range, RW3-0.06	+3.5 V to +36 V

All wheels have a power input line V+A. Some wheels also have a second power input V+B. Each power line may have multiple connector pins wired together for greater current carrying.

6.5 GND

There are multiple pins allocated to GND. This is the power ground signal. It is connected to chassis via 1 MΩ and 100 nF. GND is the return for the V+[A|B] signals, and is used as the reference for the Address[0|1] signals. All of the communications signals are differential, and do not use GND.

6.6 RW3-0.06 Power Architecture

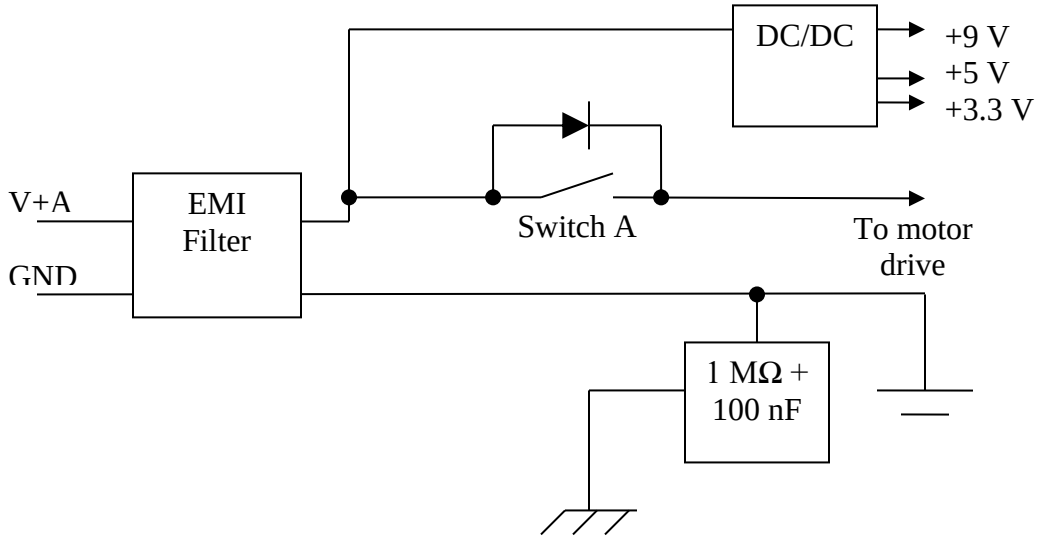

Figure 6: RW3-0.06 Rev 8 and Earlier Power Architecture

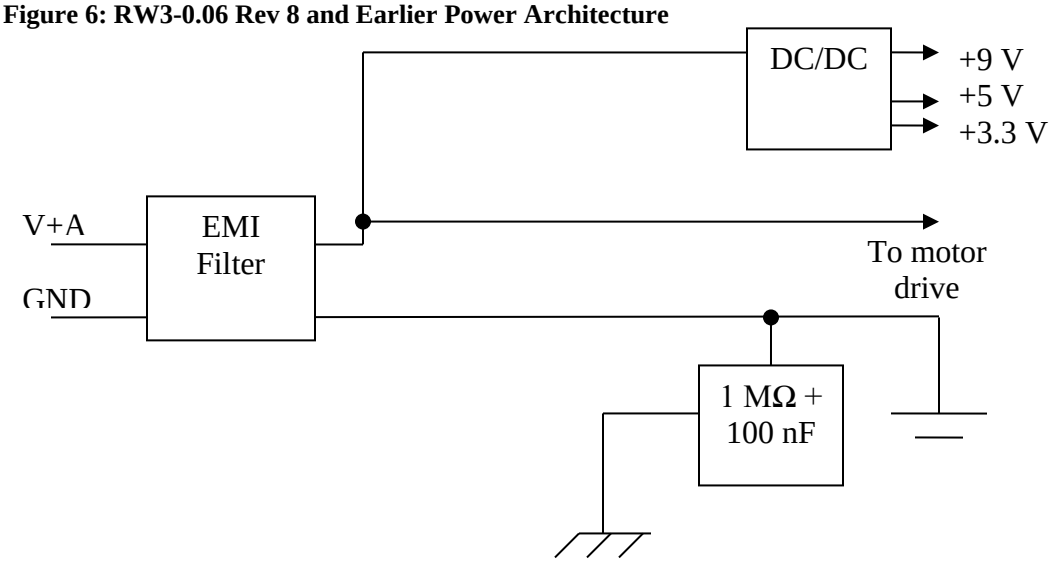

Figure 7: RW3-0.06 Rev 9 and Later Power Architecture

The RW3-0.06 has a simpler power architecture with only one power input. All of the voltage rails are produced by DC/DC conversion, giving greater efficiency.

The rev 8 and prior retains a switch in series with the motor drive, there is a parallel diode that ensures that the motor drive is always powered. Its purpose is to prevent unwanted regeneration when a quickly spinning wheel is turned off.

The rev 9 and later electronics dispense with the switch. The motor drive is always powered, and always able to regenerate.

6.7 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 ~52 V safety threshold is reached. This will cause the wheel to reset and cease regeneration.

The wheel will never regenerate when the internal power switches (if present) are turned off. Special software modes exist (BLEND, or CURRENT_BYPASS < 0.0) to minimize regeneration.

6.8 Bus Voltage Transient

Table 11: Allowed Bus Voltage Slope, Single Step

Electronics Revision	Maximum Bus Voltage Slope
RW3-0.06 of Rev 9B	Unlimited
RW3-0.06 of Rev 9	120000 V/sec
All other RW3	40000 V/sec

The maximum rate of bus voltage transient must be controlled to avoid damage to the reaction wheel. This primarily applies to turn-on steps, but could also manifest in steps while operating or in external short-circuit or crowbar events. Table 11 shows the allowable slope for a single-direction step, such as a transition from 0 to 28 V.

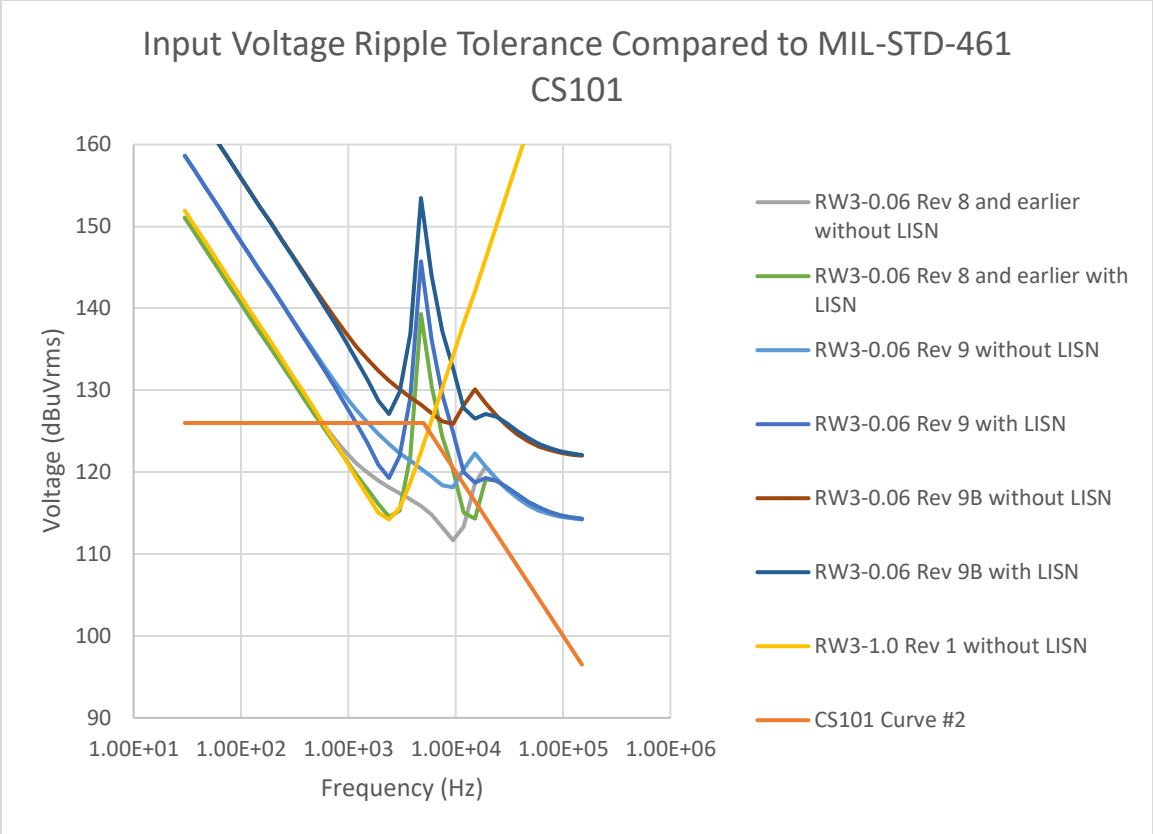

Figure 8: Allowed Bus Voltage Ripple, Sustained

Figure 8 shows the sustained ripple on the bus that is permitted. It is derived from a SPICE simulation, and makes the following assumptions:

- The limiting factor is heat dissipation in the damping resistors.
- Resistor nominal ratings are used without derating
- For continual operation, to derate power to 50%, subtract 3 dB from allowed limit.
- For < 5 seconds short-term overdrive operation, add 8 dB to allowed limit.
- For curves without LISN, a zero-impedance power supply is assumed.
- For curves with a LISN, the LISN from Figure 6 of MIL-STD-461F is assumed.

The RW3-0.06 revision 9B electronics pass the CS101 curve #2 requirements. The other revisions do not, though the revision 9 electronics are more robust than all prior versions.

The wheel has been tested to CS106, and is known to be tolerant to those short transients.

7 Protocol Layer 2 (Data Link Layer)

7.1 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 12: Default Asynchronous Serial Parameters

Nominal Baud Rate	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 actual output baud rate may deviate slightly from the nominal due to inaccuracies in the wheel master oscillator. Wheels typically use a silicon oscillator with 0.5% tolerance in bootloader mode. They may shift to a very accurate MEMS oscillator in application mode. See the EIDP to determine whether a MEMS oscillator is installed in your device.

7.2 CAN

Some wheels are fitted with a CAN communications port. The ISO 11898 CAN protocol is used. See the EIDP for information on CAN baud rate.

8 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.

8.1 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 13: 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.

8.2 CAN NSP Encapsulation

NSP messages are encapsulated for traffic over a CAN bus using a proprietary protocol which borrows heavily from CANopen. It is intended to coexist with other spacecraft CAN traffic on the bus which may be using CANopen, or any other protocol. It has the following features to accomplish this goal:

- Minimal usage of CAN Object IDs
- Controlled maximum CAN message length, to preserve bus latency for time critical applications.
- Bus throttling, to preserve bus bandwidth for rate critical applications.

8.2.1 Redundant CAN Bus Selection

Some hardware has redundant CAN bus physical layers. The logic for switching a single CAN controller between multiple physical layers is TBD.

8.2.2 CAN Object IDs

Table 14: CAN Receive Messages

Receive Message	CAN Object ID
Expedited Telecommand 1	0x200 + Unit NSP Address
Expedited Telecommand 2	0x300 + Unit NSP Address
Standard Receive	0x400 + Unit NSP Address

The table above shows the three CAN messages that a unit will receive. These map exactly to the default RPDO COB-IDs for CANopen.

It is not permitted for two messages with the same object ID to collide on the CAN network, so each message must have only a single source. A unit can receive expedited telecommands from two nodes, and standard telecommands from a third. This is sufficient for most spacecraft architectures.

Table 15: CAN Transmit Messages

Transmit Message	CAN Object ID
Expedited Telemetry	0x180 + Unit NSP Address
Standard Transmit	0x380 + Unit NSP Address

The table above shows the two CAN messages that a unit will transmit. These map exactly to the default TPDO COB-IDs for CANopen.

8.2.3 Expedited Telecommands

An expedited telecommand encapsulates an entire short NSP message within a single CAN message. Telecommands are sent from the spacecraft computer to Sinclair Interplanetary units.

Table 16: Expedited Telecommand Message Format

Message Type	Data frame, 11-bit identifier Length 2 to 8 bytes
CAN Object ID	0x200 + NSP Destination Address Or 0x300 + NSP Destination Address
Data Byte 0	NSP Source Address
Data Byte 1	NSP Message Control Field
Data Bytes 2 - 8	NSP Data Field

The expedited telecommand message is only as long as is required. The shortest possible message, carrying an NSP message with no data field, has a data length of 2. The longest possible message has a data length of 8, carrying an NSP message with 6 bytes of data field.

The NSP destination address is not carried within the message data, but can be inferred from the CAN object ID. The NSP message CRC is not carried – the CAN message carries its own CRC which is sufficient to protect against corruption.

8.2.4 Expedited Telemetry

Analogous to expedited telecommand messages, expedited telemetry encapsulates a single NSP message within a CAN message. Telemetry is sent from Sinclair Interplanetary units to the spacecraft computer.

Table 17: Expedited Telecommand Message Format

Message Type	Data frame, 11-bit identifier Length 2 to 8 bytes
CAN Object ID	0x180 + NSP Source Address
Data Byte 0	NSP Destination Address

Data Byte 1	NSP Message Control Field
Data Bytes 2 - 7	NSP Data Field

The expedited telemetry message is only as long as is required. The shortest possible message, carrying an NSP message with no data field, has a data length of 2. The longest possible message has a data length of 8, carrying an NSP message with 6 bytes of data field.

The NSP source address is not carried within the message data, but can be inferred from the CAN object ID. The NSP message CRC is not carried – the CAN message carries its own CRC which is sufficient to protect against corruption.

8.2.5 Standard Transfers

Standard transfers allow NSP messages of arbitrary sizes to be split into a sequence of CAN messages and later reassembled. Standard transfers are symmetric, in that the telemetry and telecommand interfaces are identical.

There are no explicit timing requirements. The acknowledgement mechanism allows for flow control. The receiving unit may delay sending acknowledgements until it is ready to receive more data. Either the transmitting or receiving unit may delay sending CAN messages to control the load on the spacecraft bus.

The message length used is configurable through the network management message. The entire protocol can be conducted with CAN messages of only 2 bytes of data if it is desired to minimize the worst-case latency of the spacecraft bus. Of course, 8 byte messages give the lowest protocol overhead.

Standard transfers can accommodate NSP messages with any source and destination address. This opens the possibility of future implementations conducting packet routing. Flow-control provisions anticipate wormhole routing to slower networks.

8.2.5.1 Start Message

Table 18: Start Message Format

Message Type	Data frame, 11-bit identifier Length 2 to 8 bytes	
CAN Object ID	0x400 + Unit NSP Address [into unit] 0x380 + Unit NSP Address [out of unit]	
Data Byte 0	Bits 6-7 (MSB)	0x00
	Bit 5	Final
	Bits 0-4 (LSB)	Reserved
Data Byte 1	NSP Destination Address	
Data Bytes 2 - 7	Subsequent bytes from NSP message	

A start message begins an NSP transfer. At a minimum it contains the NSP destination address. It may optionally contain 6 more bytes of NSP message.

The Final bit is set if the start message holds the entire NSP message. The longest NSP message that can be held has 2 bytes of data field, plus destination and source address, message control, and 2 bytes of NSP CRC. If the final bit is clear there is the expectation that the rest of the message will subsequently be sent in one or more continued messages.

A start message will be acknowledged with an acknowledge message.

8.2.5.2 Continuation Message

Table 19: Continuation Message Format

Message Type	Data frame, 11-bit identifier Length 2 to 8 bytes	
CAN Object ID	0x400 + Unit NSP Address [into unit] 0x380 + Unit NSP Address [out of unit]	
Data Byte 0	Bits 6-7 (MSB)	0x01
	Bit 5	Final
	Bits 0-4 (LSB)	Sequence Number
Data Bytes 1 - 7	Subsequent bytes from NSP message	

A continuation message continues an NSP transfer that was begun with a start message. Each continuation message can contain up to 7 bytes of the NSP message.

The Final bit is set if this is the last CAN message of the NSP message.

The first continuation message has a sequence number of 1. Each subsequent continuation message increments the sequence number, up to a maximum value of 31. Following this, the next continuation message has a sequence number of 1, and the next 2 and so-on. The sequence number must never be 0 – this is reserved as the implicit sequence number for a start message.

8.2.5.3 Acknowledge Message

Table 20: Acknowledge Message Format

Message Type	Data frame, 11-bit identifier Length 2 bytes	
CAN Object ID	0x400 + Unit NSP Address [into unit] 0x380 + Unit NSP Address [out of unit]	
Data Byte 0	Bits 6-7 (MSB)	0x02
	Bit 5	Management
	Bits 0-4 (LSB)	Sequence Number
Data Byte 1	Bits 3-7 (MSB)	Status
	Bits 0-2 (LSB)	Allowed count

An acknowledgement message may be sent in response to a start, continued or network management message.
In response to a start message, the Management bit will be cleared and the sequence number will be zero. In response to a continued message, the Management bit will be cleared and the Sequence Number will be equal to the sequence number of the last

continued message. In response to a network management message the Management bit will be set, and the Sequence Number will contain the current network settings in the same format as sent in the network management message.

The five-bit status field encodes the following possible results:

Table 21: Acknowledge Status Codes

Status Code	Meaning
0	Message successfully received, all is good
1	Start message received, aborting previous unfinished incoming message
2	Start message received while part-way through outgoing message. Unit is incapable of full-duplex. Outgoing message aborted, and incoming message can continue.
3	Continuation message received without start. Abort.
4	Continuation message received with sequence error. Abort.
5	Continuation message received after expedited telecommand received. Expedited telecommand has overwritten the unfinished incoming message. Abort.
6	I cannot handle messages with this destination address. Abort.
7	Final message received, but overall NSP CRC invalid. Abort.
8	Continuation message received. NSP message buffer overflow. Abort.
9	A message has been received with at least one byte of data, but not the number of data bytes expected.
10+	Reserved

The three-bit allowed count field encodes the following possible results:

Allowed Count	Meaning
0	Do not send any more continuation messages. This transaction is either complete, or aborted.
1	Send one continuation message and then wait for my acknowledgement.
2	Send up to 2 continuation messages and then wait for my acknowledgement. I will acknowledge the 2nd message, or the final message, or any error, whichever comes first.
3	Send up to 4 continuation messages and then wait for my acknowledgement. I will acknowledge the 4th message, or the final message, or any error, whichever comes first.
4	Send up to 8 continuation messages and then wait for my acknowledgement. I will acknowledge the 8th message, or the final message, or any error, whichever comes first.
5	Send up to 16 continuation messages and then wait for my acknowledgement. I will acknowledge the 16th message, or the final message, or any error, whichever comes first.
6	Send up to 32 continuation messages and then wait for my acknowledgement. I will acknowledge the 32nd message, or the final message, or any error, whichever comes first.

7	Send as many continuation messages as desired. I will acknowledge the final message, or any error, whichever comes first.

8.2.5.4 Network Management Message

Table 22: Acknowledge Message Format

Message Type	Data frame, 11-bit identifier Length 1 byte	
CAN Object ID	0x400 + Unit NSP Address [into unit] 0x380 + Unit NSP Address [out of unit]	
Data Byte 0	Bits 6-7 (MSB)	0x03
	Bit 5	Write
	Bit 4	Reserved
	Bit 3	Use Expedited
	Bits 0-2 (LSB)	Max DLC

The network management message is sent to control the network behaviour of a node. It does not contain any portion of an NSP message.

If the Write bit is set, the node's settings are updated based on the contents of this message. If the Write bit is clear the node's settings are unchanged. Clearing the Write bit can be used to read back the node's settings in the resulting acknowledge message, or just as a network level aliveness ping.

If the Use Expedited bit is set, then the node will encapsulate outgoing NSP messages in expedited telecommand/telemetry packets when possible, subject to the Max DLC setting. If the Use Expedited bit is clear, then the node will send all outgoing NSP messages using standard encapsulation.

The 3-bit Max DLC field controls the maximum length of CAN message that the unit is allowed to send.

Table 23: Max DLC Field Meaning

Max DLC Field	Meaning
0	The unit is prohibited from sending any CAN messages. There will be no acknowledgement.
1	The unit is allowed to send CAN messages with up to 2 bytes of data.
2	The unit is allowed to send CAN messages with up to 3 bytes of data.
3	The unit is allowed to send CAN messages with up to 4 bytes of data.
4	The unit is allowed to send CAN messages with up to 5 bytes of data.
5	The unit is allowed to send CAN messages with up to 6 bytes of data.

6	The unit is allowed to send CAN messages with up to 7 bytes of data
7	The unit is allowed to send CAN messages of any length

A unit is not required to obey a network management message. It should send back its actual settings in the acknowledge message.

9 Protocol Layer 4 (Transport Layer)

9.1 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.

Some commands may generate more data than can be fit into a single telemetry message. In this case a sequence of telemetry messages will be sent back-to-back to carry the required data. The last message will be indicated using the P/F bit.

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.

9.2 NSP Message Format

Table 24: 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 bootloader supports a maximum data length of 516 bytes, giving a total message length of 521 bytes. The wheel application program 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.

9.3 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.

9.4 Wheel Address and Port Selection

The wheel bootloader may support incoming communication on a number of communication ports: potentially up to two RS485 ports and two CAN 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.

When the bootloader starts the application program the configuration is frozen. The port that received the INIT command is now the only port that is monitored for commands, and all replies will be sent to the appropriate reply port. The application program will only respond to the NSP address that the INIT was sent to.

It is intended that the bootloader only receive commands on one port at a time. Multiple incoming commands will be serviced on a best-effort basis, but there may be insufficient realtime processing ability to handle them simultaneously.

9.5 Default Addressing

The NSP address is determined by the state of the Address 0 pin at boot. The user must configure the wire harness so that the pin is either:

- Tied to V+A
- Tied to GND
- Tied to GND through 100 kΩ
- Unconnected

The 100 kΩ resistor will be subjected to a maximum of +2.6 V, with an additional 11 kΩ in series, so it can be a low-power part. It is anticipated that a small through-hole resistor can be spliced right into the wire harness. This configuration is only needed if four wheels are flown. For three wheels, the other three options are recommended.

For RS485+CAN Configurations

Table 25: RW3-0.06 RS485+CAN Address Information

Address 0 Pin	NSP Address on RS485_0	NSP Address on CAN_A
Tied to V+A	0x03	0x03
Tied to GND	0x04	0x04
Tied to GND through 100 kΩ	0x05	0x05
Unconnected	0x06	0x06

In this configuration, telemetry is always sent to the command port. A NSP packet arriving on the RS485 port will be replied to on the RS485 port, while a packet arriving on the CAN port will be replied to on the CAN port.

For Dual RS485 Configurations

Table 26: RW3-0.06 Dual RS485 Address Information

Address 0 Pin	NSP Address	Command Port	Telemetry Port
Tied to V+A	0x03	RS485_1	RS485_1
Tied to V+A	0x13	RS485_0	RS485_1
Tied to V+A	0x23	RS485_1	RS485_0
Tied to V+A	0x33	RS485_0	RS485_0
Tied to GND	0x04	RS485_1	RS485_1
Tied to GND	0x14	RS485_0	RS485_1
Tied to GND	0x24	RS485_1	RS485_0
Tied to GND	0x34	RS485_0	RS485_0
Tied to GND through 100 kΩ	0x05	RS485_1	RS485_1
Tied to GND through 100 kΩ	0x15	RS485_0	RS485_1
Tied to GND through 100 kΩ	0x25	RS485_1	RS485_0
Tied to GND through 100 kΩ	0x35	RS485_0	RS485_0
Unconnected	0x06	RS485_1	RS485_1
Unconnected	0x16	RS485_0	RS485_1
Unconnected	0x26	RS485_1	RS485_0
Unconnected	0x36	RS485_0	RS485_0

For each Address 0 configuration, there are four NSP addresses that the wheel will recognize. Each corresponds to a different RS485 command and telemetry port. When the command and telemetry ports are different, the configuration is referred to as 4-wire RS485. When the command and telemetry ports are the same, the configuration is referred to as 2-wire RS485.

9.6 Message Control Field

Table 27: 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'.

9.7 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.

9.8 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;
      }
}
```

9.9 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'.

10Protocol Layer 5 (Session Layer)

10.1 Operating Modes

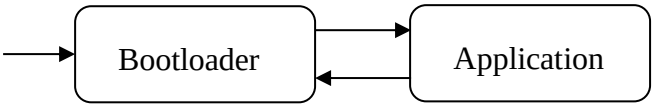

Figure 9: Mode Transition Diagram

Power-on starts the unit in bootloader mode.

10.1.1 Bootloader to Application Transition

The wheel will transition from bootloader to application mode upon receipt of an "INIT 0x00002000" command.

10.1.2 Application to Bootloader Transition

The wheel will transition from application mode to bootloader mode under the following conditions:

- An "INIT" command with no data is received.
- The bus voltage (VBUS) exceeds ~50 V.
- Any of the winding temperatures (TEMP0, TEMP1) exceeds 125 °C.

10.2 Power Switch Sequence

10.2.1 Power Switch Sequence, Rev 8 and Earlier

For power switch configuration in Rev 8 or earlier versions of the wheel, please consult the factory for the appropriate revision of the ICD.

10.2.2 Power Switch Sequence, Rev 9 and Later

The wheel has no power switch. Commands to turn the switch on and off are ignored. Telemetry will always show the A switch turned on, for compatibility with earlier interfaces.

The BUS_MIN_THRESHOLD and BUS_MAX_THRESHOLD parameters are ignored.

10.3 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.

The exact contents of the test scripts is not documented here, to avoid the danger that it might become out of sync with the actual software. The *rw-bit-term* program should be used to record and interpret test script output. It is automatically synced to the wheel onboard software.

10.4 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.

10.5 Command Codes

Table 28: 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.

10.6 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 name, and compile time and date of the software that is currently running on the target processor.

10.6.1 Command Format

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

10.6.2 Reply Format

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

10.7 INIT (0x01)

The INIT command is used to change the operating mode of a wheel. In general, and INIT with data is interpreted as an address to jump to. An init with no data is interpreted as a reset or exit command. 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. If it is in bootloader mode, it will respond to an INIT with 4 bytes of data by running an Application Module at the corresponding 32-bit start address. By convention, devices will ship from the factory with the processor primary application program stored at address 0x00002000. Thus, a command of INIT 0x00002000 will start the default behaviour.

10.7.1 Command Format

Reboot command:

No payload bytes

Application start command:

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

10.7.2 Reply Format

Reboot reply:

No payload bytes

Application start reply:

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

10.8 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.

The wheel processor has no restriction on the alignment or length of a peek.

10.8.1 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.

10.8.2 Long Command Format

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

10.8.3 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

10.9 POKE (0x03)

The POKE command is used to write the device memory. The wheel will only permit a POKE into flash memory when in bootloader mode. Each 512 byte block of flash memory

has a lifetime of only 20,000 write cycles. One cycle is consumed for each POKE command that accesses a particular block. This lifetime is more than sufficient for occasional software patches, but the user is cautioned that a looping sequence of POKE commands could easily wear out a block.

The wheel processor has no restriction on the alignment or length of a poke.

10.9.1 Command Format

Bytes 0 – 3	32-bit address to start poking data
Byte 4 – N	1 - 512 bytes to write to the target memory

10.9.2 Reply Format

Bytes 0 – 3	32-bit address where data write began
Bytes 4 – N	1 – 512 bytes written to the target memory

10.10 DIAGNOSTIC (0x04)

The DIAGNOSTIC command gathers error count data from the wheel.

10.10.1 Command Format

Byte 0	Address of the diagnostic channel to read, as an 8-bit integer

10.10.2 Reply Format

Byte 0	Address of the diagnostic channel read, as an 8-bit integer
Bytes 1 - 4	Diagnostic value from the addressed channel, as a 32-bit integer

10.11 CRC (0x06)

CRC command is used to calculate a checksum on an area of memory. Any of the memory spaces may be addressed, and the calculation window may be as large as desired provided that it does not contain any unimplemented memory.

The CRC uses the same 16-bit polynomial, with the same bit order, as is used for NSP messages.

The CRC command can potentially be used to request the CRC of the wheel's entire 128 kB flash memory. This can take a number of seconds, especially in bootloader mode where the system clock is much slower.

10.11.1.1 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

10.11.1.2 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

10.12 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.

10.12.1 Command Format

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

10.12.2 Reply Format

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

10.12.2.1 Mode Reply Structure

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

10.12.2.2 Normal Reply Structure

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

10.13 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. from 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.

10.13.1 Command Format

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

10.13.1.1 Mode Store Structure

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

10.13.1.2 Normal Store Structure

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

10.13.1 Reply Format

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

10.13.1.1 Mode Reply Structure

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

10.13.1.2 Normal Reply Structure

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

10.14 READ EDAC (0x09)

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

10.14.1 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.

10.14.2 Long Command Format

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

10.14.3 Reply Format

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

10.15 WRITE EDAC (0x0A)

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

10.15.1 Command Format

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

10.15.2 Reply Format

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

10.16GATHER EDAC (0X0B)

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

10.16.1 Command Format

Bytes 0 – N	List of gather command structures

10.16.1.1 Gather Command Structure

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

10.16.1 Result Format

Bytes 0 – N	List of gather result structures

10.16.1.1 Gather Result Structure

Bytes 0 – 1	EDAC address where reading started
Bytes 2 – 3	Number of bytes read
Bytes 4 – N	Data bytes

11Protocol Layer 6 (Presentation Layer)

11.1Memory Map

Table 29: Processor Memory Map

Address Range	Function
0x00000000 – 0x00001FFF	Bootloader program memory
0x00002000 – 0x0000F9FF	Program memory (flash)
0x0000FA00 – 0x0000FFFF	Stored parameters (flash)
0x00010000 – 0x0001F9FF	Extended program memory (flash)
0x0001FA00 – 0x0001FBFF	Bootloader program memory
0x01000000 – 0x010000FF	256 B IRAM (RAM)
0x02000000 – 0x02001FFF	8 kB XRAM (RAM)
0x03000080 – 0x030000FF	128 B SFR (RAM) Bank 00h
0x030C0080 – 0x030C00FF	128 B SFR (RAM) Bank 0Ch
0x030F0080 – 0x030F00FF	128 B SFR (RAM) Bank 0Fh
0x03100080 – 0x031000FF	128 B SFR (RAM) Bank 10h

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.

The first 8 kB of program memory contain the bootloader. These are protected against POKEs so that the bootloader cannot be accidentally changed. The next 56 kB contains the application program. A sequence of POKE commands in bootloader mode can be used to load new application programs.

The bootloader memory cannot be read by the application program, and so PEEK or CRC commands to those regions will fail if not in bootloader mode.

The processor has two RAM areas. There is little need for a user to touch these.

There are four banks of Special Function Registers (SFRs). These should not be POKEd without knowing exactly what is going on. Even PEEKing some of these registers can have unexpected side effects.

11.2Diagnostics

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

Table 30: Diagnostic Channels

Diagnostic Channel	Bootloader	Application
0x00	Reset Reason	
0x01	Reset Count	
0x02	RS485_0 Framing Errors	NSP Framing Errors
0x03	RS485_0 Runt Packets	NSP Runt Packets
0x04	RS485_0 Oversize Packets	NSP Oversize Packets
0x05	RS485_0 Bad CRC	NSP Bad CRC
0x06	RS485_0 Buffer Overflow	NSP Buffer Overflow
0x07	RS485_1 Framing Errors	Not Available

0x08	RS485_1 Runt Packets	
0x09	RS485_1 Oversize Packets	
0x0A	RS485_1 Bad CRC	
0x0B	RS485_1 Buffer Overflow	
0x0C	CAN_0 Framing Errors	
0x0D	CAN_0 Runt Packets	
0x0E	CAN_0 Oversize Packets	
0x0F	CAN_0 Bad CRC	
0x10	CAN_0 Buffer Overflow	

The bootloader contains error counters for two RS485 ports and one CAN port. The application program contains only one set of error counters, which apply to whichever port has been configured for command reception.

11.2.1 Reset Reason

The reset reason is an enumerated type, describing the reason for the most recent reset of the wheel processor.

Table 31: Reset Reason Codes

Reset Reason Code	Meaning
0	Power cycle. The wheel has either been freshly turned on, or the input voltage has drooped below approximately 3.5 V.
1	Flash error. An illegal attempt has been made to read or write flash memory.
2	Comparator reset. The on-chip comparators are not enabled, so this should never occur.
3	Watchdog reset. The default application program does not use the watchdog timer, but if it somehow does get turned on this is the reset that it would generate.
4	Missing clock. The clock source for the processor stopped ticking. This could be caused by failure of the MEMS oscillator (if fitted) or by a momentary failure of the internal silicon oscillator. The bootloader runs from the internal oscillator, so a permanent failure of that oscillator would disable the wheel.
5	Pin reset. The external /Reset signal has been pulled low. This is most likely caused by a stator winding over-temperature event.
6	Software reset. The most likely cause is that an INIT command has been received with no data, forcing a reset. This could also be caused if the software encounters an irrecoverable fault, such as a spurious interrupt.

11.2.2 Reset Count

The reset count contains the number of wheel processor resets since the last power cycle reset. Immediately after a power cycle the reset count will read as 0. After the first non-power-cycle reset it will read 1.

11.2.3 Framing Error

A framing error is declared if an NSP message is incorrectly encapsulated on the communications link. For RS485 links, this would be any time a FESC character is seen that is not immediately followed by TFESC or TFEND.

CAN framing errors are declared when an incoming standard continuation message is in error. This may be because there has been no preceding standard start, or the standard transfer has aborted. It may also be because the sequence number of the continuation message is unexpected.

11.2.4 Runt Packet

A runt packet is a NSP message that is less than 5 bytes long. Such a fragment cannot be a properly formed NSP message since it cannot contain a source and destination address, control field, and CRC.

Runts are counted only if the first byte is equal to the wheel's address on that port, which would normally indicate that the packet is addressed to this unit. A zero-length NSP message is not considered a runt. For example, on an RS485 link two FEND characters back-to-back is a valid bus condition and not a runt.

Using CAN, a runt packet is an expedited telecommand message with fewer than two data bytes, or an NSP message carried over standard telecommands that has fewer than five bytes.

11.2.5 Oversize Packet

An oversize packet is one that has too many bytes in the data field. Packets that are too long cannot fit into the allocated message buffers and so they must be rejected. See section 9.2 for the length constraints.

11.2.6 Bad CRC

This count is incremented every time a properly formatted (in length and framing) NSP message is received where the CRC field does not match with the computed CRC, and where the first byte is equal to the NSP address of the wheel.

Errors in the CRC internal to a CAN message are not counted.

11.2.7 FIFO Overflow

This count is incremented every time one or more incoming bytes are lost due to processor loading. Only UART0 is able to detect overflows, and due to the constraints of the hardware it is not guaranteed that all overflow events will be noticed.

11.3 EDAC Memory

The processor supports 1536 bytes of EDAC protected memory. These are implemented using software-based triple-redundant storage into conventional SRAM cells. EDAC memory can be read with READ 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 flash memory.

Files addresses marked with * indicate functionality not presently available on the RW3-0.06 hardware. These channels will read IEEE-754 NaN.

Table 32: EDAC Memory Contents

EDAC Address	File Address	Function	Format
0x000 – 0x003	0x00	Command Value	Command Dependent
0x004 – 0x007	0x01	VA	Volts (IEEE-754 float)
0x010 – 0x013	0x04	PHASE_COMMON	Volts (IEEE-754 float)
0x018 – 0x01B	0x06	8V	Volts (IEEE-754 float)
0x01C – 0x01F	0x07	VDD	Volts (IEEE-754 float)
0x020 – 0x023	0x08	VCC	Volts (IEEE-754 float)
0x028 – 0x02B	0x0A	CURRENT_PHASE0	Amps (IEEE-754 float)
0x02C – 0x02F	0x0B	CURRENT_PHASE1	Amps (IEEE-754 float)
0x030 – 0x033	0x0C	CURRENT_PHASE2	Amps (IEEE-754 float)
0x040 – 0x043	0x10	TEMP0	°C (IEEE-754 float)
0x048 – 0x04B	0x12	TEMP2	°C (IEEE-754 float)
0x04C – 0x04F	0x13	TEMP3	°C (IEEE-754 float)
0x050 – 0x053	0x14	TEMP4	°C (IEEE-754 float)
0x054 – 0x057	0x15	SPEED	Rad/sec (IEEE-754 float)
0x058 – 0x05B	0x16	MOMENTUM	N-m/sec (IEEE-754 float)
0x05C – 0x05F	0x17	SCRUB_INDEX	32-bit unsigned int
0x060 – 0x063	0x18	SEU_COUNT	counts (IEEE-754 float)
0x064 – 0x067	0x19	BUS_STATUS	Enum in IEEE-754 float
0x068 – 0x06B	0x1A	PWM	Duty cycle (IEEE-754 float)
0x06C – 0x06F	0x1B	HALL_DIGITAL	Enum in IEEE-754 float
0x070 – 0x073	0x1C	CONTROL_TIME	Timer ticks (IEEE-754 float)
0x074 – 0x077	0x1D	OSCILLATOR_CALIBRATE	Ratio IEEE-754 float

0x078 – 0x07B	0x1E	TARGET_CURRENT	Amps (IEEE-754 float)
0x07C – 0x07F	0x1F	MEASURED_CURRENT	Amps (IEEE-754 float)
0x080 – 0x083	0x20	SPEED_P_GAIN	Amps / rad / sec (IEEE-754 float)
0x084 – 0x087	0x21	SPEED_I_GAIN	Amps / rad (IEEE-754 float)
0x088 – 0x08B	0x22	SPEED_D_GAIN	Amps / rad / sec^2 (IEEE-754 float)
0x08C – 0x08F	0x23	ADC_I_GAIN	(IEEE-754 float)
0x090 – 0x093	0x24	ADC_P_GAIN	(IEEE-754 float)
0x094 – 0x097	0x25	MAX_GAIN_SPEED	Rad/sec (IEEE-754 float)
0x098 – 0x09B	0x26	MIN_GAIN_SPEED	Rad/sec (IEEE-754 float)
0x09C – 0x09F	0x27	TEST_TONE	(IEEE-754 float)
0x0A0 – 0x0A3	0x28	INERTIA	kg-m^2 (IEEE-754 float)
0x0A4 – 0x0A7	0x29	MOTOR_KT	N-m/A (IEEE-754 float)
0x0A8 – 0x0AB	0x2A	GAIN_SCHEDULE1	(IEEE-754 float)
0x0AC – 0x0AF	0x2B	GAIN_SCHEDULE2	(IEEE-754 float)
0x0B0 – 0x0B3	0x2C	GAIN_SCHEDULE3	(IEEE-754 float)
0x0B4 – 0x0B7	0x2D	GAIN_SCHEDULE4	(IEEE-754 float)
0x0B8 – 0x0BB	0x2E	PROPORTIONAL_OVERRIDE	(IEEE-754 float)
0x0BC – 0x0BF	0x2F	CONTROL_TYPE	(IEEE-754 float)
0x0C0 – 0x0C3	0x30	BUS_MIN_THRESHOLD	Volts (IEEE-754)
0x0C4 – 0x0C7	0x31	BUS_MAX_THRESHOLD	Volts (IEEE-754)
0x0C8 – 0x0CB	0x32	MAX_SPEED_AGE	sec (IEEE-754)
0x0CC – 0x0CF	0x33	LIMIT_SPEED1	Rad/sec (IEEE-754)
0x0D0 – 0x0D3	0x34	LIMIT_SPEED2	Rad/sec (IEEE-754)
0x0D4 – 0x0D7	0x35	LIMIT_CURRENT	Amps (IEEE-754)
0x0D8 – 0x0DB	0x36	TURNON_RATE	PWM counts per frame (IEEE-754)
0x0DC – 0x0DF	0x37	OSCILLATOR_TOLERANCE	IEEE-754
0x0E0 – 0x0E3	0x38	CURRENT_BYPASS	Boolean (IEEE-754 float)
0x0E4 – 0x0E7	0x39	BYPASS_GAIN	(IEEE-754 float)
0x0E8 – 0x0EB	0x3A	BYPASS_STEP	(IEEE-754 float)
0x0EC – 0x0EF	0x3B	SINUSOID_PHASE	Rad (IEEE-754)
0x0F0 – 0x0F3	0x3C	SINUSOID_FREQ	Hz (IEEE-754)
0x0F4 – 0x0F7	0x3D	SINUSOID_OFFSET	Rad/sec (IEEE-754)
0x0F8 – 0x0FB	0x3E	CURRENT_IIR_CONSTANT	(IEEE-754)

0x0FC – 0x0FF	0x3F	VOLTAGE_IIR_CONSTANT	(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)
0x128 – 0x12B	0x4A	HALL_TRANSITION	Counts (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)
0x140– 0x143	0x50	SFFT_STEP_NUMBER	32-bit integer
0x144 – 0x147	0x51	SFFT_STEP_TIMER	Sec (IEEE-754)
0x148 – 0x14B	0x52	SFFT_TELEM_COUNT	32-bit integer
0x14C – 0x5BF	0x53 – 0x16F	Self Test Results	
0x5C0 – 0x5C1		CRC	
0x5C2		LOAD_SOURCE	
0x5C3		MODE	8-bit enum
0x5C4 – 0x5C8		Startup I/O	40-bit binary
0x5C9 – 0x5CD		Floating I/O	40-bit binary
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
0x5D3		SMBUS_ABORT	8-bit unsigned int
0x5D4		SMBUS_TIMEOUT	8-bit unsigned int
0x5D5		SMBUS_STOP	8-bit unsigned int
0x5D6		IDLE_INHIBIT	8-bit boolean
0x5D7		REALTIME_DELAY	8-bit unsigned int
0x5D8 – 0x5DA		Reserved	
0x5DB		TEMPSENSE_INHIBIT	8-bit boolean
0x5DC		BUSOFF_REASON	8-bit enum
0x5DD		ANALOG_HALL_DISABLE	8-bit boolean
0x5DE		ADC_REGISTER_REFRESH	8-bit enum
0x5F0 – 0x5F3		CURRENT_THRESHOLD	Amps (IEEE-754)
0x5F4 – 0x5F7		CURRENT_FILTER	Amps (IEEE-754)
0x5F8 – 0x5FB		VOLTAGE_THRESHOLD	$Volts^2$ (IEEE-754)
0x5FC – 0x5FF		VOLTAGE_FILTER	$Volts^2$ (IEEE-754)

11.3.1 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 11.3.57.

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.

11.3.2 VA

This read-only parameters returns the voltage at the V+A signal. The maximum value that can be read is +69 V. The minimum value that can be read is 0 V. This is not a problem, since if V+A is negative the wheel will be turned off and not generating telemetry.

11.3.3 PHASE_COMMON

The motor is wound in a Y-configuration. This read-only parameter returns the voltage from the center of the motor windings. The maximum value that can be read is +64 V. On the RW-1.0 hardware, the minimum value that can be read is approximately -16 V. Accurate calibration of negative input voltages is not guaranteed, but a general indication of their polarity and magnitude is generated. The RW3-0.06 cannot read negative input voltages.

When the motor is being driven this will typically show half of the applied motor voltage. When the motor is idle the common point will float at a little over +4 V (RW-1.0) or +6V (RW3-0.06), due to leakage currents from current sensor and MOSFET driver circuits.

11.3.4 8V

This read-only parameter returns the voltage at the +8 V power supply rail. The "+8 V" rail can be measured from 0 V to +22.5 V. On Revision 6 hardware its nominal value is actually +9 V. Revision 7 hardware is nominally +8 V.

These rails should maintain their nominal voltages under all conditions.

11.3.5 VDD, VCC

These read-only parameters return the voltages at the low-voltage rails. VDD returns the processor's internal rail which is +2.2 V nominal.

VCC returns the processor's I/O rail. For the RW3-0.06 revision 6, this is +3.3 V nominal. For the RW3-0.06 revision 7, this is +2.6 V nominal.

Due to realtime resource conflicts, the processor cannot sample these telemetry points while running in a closed-loop motor current mode. IEEE-754 NaN is returned to indicate that data is unavailable.

11.3.6 CURRENT_PHASE[0|1|2]

These read-only parameters return the current in each of the three motor winding phases. Positive values indicate that current is flowing into the phase from the motor driver.

Negative values indicate that current is flowing out of the phase into the motor driver. The current range that can be measured is -3.75 to +3.75 A.

Current measurements are unreliable when PWM is being applied to the phase, or when the phase is undriven and PWM is applied to another phase. The measurement is only valid when either the phase is driven to ground, or all of the phases are undriven. Thus, in normal non-regenerative operation, all of the valid currents will be negative.

Due to realtime resource conflicts, the processor cannot sample these telemetry points while running in a closed-loop motor current mode. IEEE-754 NaN is returned to indicate that data is unavailable.

11.3.7 TEMP[0]

These read-only parameters return the temperature of the motor windings. The thermistor bead is bonded to the stator in contact with the windings.

The sensors are NTC devices, so an open-circuit failure causes an apparent low temperature reading. If the measured temperature of the sensor exceeds +125 °C the processor will reset. This behaviour is implemented in hardware and cannot be bypassed by software.

11.3.8 TEMP[2|3]

These read-only parameters return temperatures on the circuit board. There are two calibrated silicon temperature sensors. TEMP2 is located immediately next to the processor. TEMP3 is located adjacent to the motor drive transistors.

The maximum temperature that can be returned is +125 °C. The minimum temperature that can be reliably returned is -55 °C – lower temperatures may be returned but at lower accuracy.

11.3.9 TEMP4

This read-only parameter returns the temperature of the processor die. The return is nominally in °C, but the accuracy is poor! There is significant unit-to-unit variation, and no effort has been made to calibrate. This telemetry point should only be used for a general hot/cold indication. TEMP2 is a far better measure of the electronics temperature.

Due to realtime resource conflicts, the processor cannot sample this telemetry points while running in a closed-loop motor current mode. IEEE-754 NaN is returned to indicate that data is unavailable.

11.3.10 SPEED

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

11.3.11 MOMENTUM

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

11.3.12 SCRUB_INDEX

Each time through the control frame (approximately 93 Hz) one byte of EDAC memory is scrubbed for errors. This file contains a pointer to the last EDAC location scrubbed. It can be read to verify that scrubbing is occurring. It can also be written to force priority scrubbing of a particular area.

Note that this file is stored as an integer.

11.3.13 SEU_COUNT

This parameter records the number of errors that have been found during EDAC scrubbing. Any error in a byte is considered to be a single error – no attempt is made to determine how many bits were flipped.

This parameter can be read to determine the error count. It can also be written – typically to reset it to zero.

11.3.14 BUS_STATUS

This read-only parameter returns the state of the bus power switches. Wheels of revision 9 and later will always return a value of 5.0 indicating switch A is on and switch B is off.

11.3.15 PWM

This read-only parameter returns the PWM duty cycle of the motor drive. It has a range of 0.0 to +1.0. No motor drive direction is encoded. [Contrast this to the direction sign used in the PWM command mode.]

This is only valid when the motor is being driven. During idle mode non-zero PWM values may be seen which do not correspond to actual motor drive.

11.3.16 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 33: 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.

11.3.17 CONTROL_TIME

This read-only parameter provides an indication of the processor realtime margin. The free-running control frame timer counts from 0 to 65535 and then overflows to 0. At each overflow a new control frame is started. The contents of the timer are latched and stored in the CONTROL_TIME file when the control algorithm has completed.

Smaller values indicate greater realtime margin. Values that approach 65535 provide a caution that the processor may be overloaded and unable to reliably complete its control algorithm within the allotted time.

See the CONTROL_OVERFLOW file for indication of negative realtime margin.

11.3.18 OSCILLATOR_CALIBRATE

This read-only parameter compares the processor's internal 24 MHz (nominal) silicon oscillator to its external 48 MHz (nominal) MEMS oscillator. The result is a nominal ratio of 2.0.

This value is measured only once, when the application is started. It is judged for acceptability, based on the criterion in OSCILLATOR_TOLERANCE. If it is outside the acceptable bounds the MEMS oscillator is declared failed and is not used. The silicon oscillator will be used instead as the time base. If the ratio is acceptable then the MEMS oscillator will be used as the time base and the silicon oscillator shut down to save power.

Some models of reaction wheel may be provided without a MEMS oscillator, to save either cost or power. For these units the expected value of this telemetry is 0.0, and the silicon oscillator will always be used.

11.3.19 TARGET_CURRENT

This read-only parameter displays the desired motor winding current, if the wheel is in a mode which uses the closed-loop current controller (current, speed, torque, etc). Positive current makes the rotor speed more positive.

If the wheel is in a mode which does not use the closed-loop current controller (idle, pwm, etc) then this value will return as NaN.

11.3.20 MEASURED_CURRENT

If the wheel is in a mode which uses the closed-loop current controller (current, speed, torque, etc) this read-only parameter shows an instantaneous reading of the motor winding current. The motor winding current is measured at 187 kHz, so this telemetry is inherently undersampled. Positive current makes the rotor speed more positive.

If the wheel is in a mode which does not use the closed-loop current controller (idle, pwm, etc) then this value will return as NaN. The CURRENT_PHASEx telemetry channels can instead be used to look at the motor current.

11.3.21 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.

11.3.22 ADC_[I|P]_GAIN

These read/write parameters set the gains for the PI closed-loop motor current controller. This controller takes motor current as an input, and servos the drive PWM. It runs as an inner loop within the speed controller.

11.3.23 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.

11.3.24 TEST_TONE

TBD

11.3.25 INERTIA

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

11.3.26 MOTOR_KT

This read/write parameter sets the motor torque. This is <u>not</u> used in normal operation, as the normal controller is entirely closed-loop. This parameter is solely used when switching out of idle mode while the rotor is spinning. In this situation the wheel must make an initial guess as to the appropriate PWM duty cycle. Using an open-loop model to inform this guess minimizes the initial current spike.

11.3.27 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.

$$Kp = 0.6 \cdot Ku$$

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

$$Kd = 0.125 \cdot KpPu$$

11.3.28 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:

$$Kp = PROPORTIONAL\_OVERRIDE$$

$$Ki = 0.0$$

$$Kd = 0.0$$

11.3.29 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:

$$Kp = 0.45 \cdot Ku$$

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

$$Kd = 0.0$$

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

$$Kp = 0.6 \cdot Ku$$

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

$$Kd = 0.125 \cdot KpPu$$

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

$$Kp = 0.5 \cdot Ku$$

$$Ki = 0.0$$

$$Kd = 0.0$$

11.3.30 BUS_MIN_THRESHOLD, BUS_MAX_THRESHOLD

These read/write parameters determine the permissible range of voltages for a power switch to be on. They control the switch sequence behaviour detailed in 10.2.

11.3.31 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.

11.3.32 LIMIT_SPEED1

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.

11.3.33 LIMIT_SPEED2

This read/write parameter sets the absolute maximum speed that the wheel can reach. If the rotor exceeds this speed the drive will be set to idle. Once the rotor slows below this limit the drive is restored. Thus, the wheel may limit-cycle around this limit. LIMIT_SPEED2 is active in all modes, which is significant since LIMIT_SPEED1 is not effective in open-loop modes (PWM, CURRENT, etc).

11.3.34 LIMIT_CURRENT

This read/write parameter sets the greatest motor drive current used by closed-loop current modes. It has no effect in PWM mode. The largest current that can be sensed is +/- 3.0 A, and LIMIT_CURRENT must be smaller than this so that closed-loop control can be achieved. Reducing this value will limit the torque that the wheel can generate.

11.3.35 TURNON_RATE

This read/write parameter controls the turn-on speed of the power switches. The total turn-on time is 2.8 sec / TURNON_RATE. Only natural numbers should be used.

The parameter is important because it limits the inrush current into the EMI filter when a power switch is turned on. Smaller values give less inrush. Inrush is not a concern for the RW3-0.06 hardware, and a value of 255.0 should be used.

11.3.36 OSCILLATOR_TOLERANCE

The nominal value of OSCILLATOR_CALIBRATE is 2.0. OSCILLATOR_TOLERANCE is a read/write parameter that controls the accepted range of OSCILLATOR_CALIBRATE. At the default value of 0.025, a 2.5% variation is allowed. That is, OSCILLATOR_CALIBRATE is allowed to range between 1.95 and 2.05.

The decision as to whether to use the MEMS oscillator is made immediately after the application starts. Thus, the value of OSCILLATOR_TOLERANCE must be stored using the STORE_FILES command in order to influence the decision.

11.3.37 CURRENT_BYPASS, BYPASS_GAIN, BYPASS_STEP

Ideally this file should only be written with the wheel at rest. Changing the value while spinning will result in a small glitch. The CURRENT_BYPASS parameter is used to set one of four possible behaviours:

CURRENT_BYPASS < 0.0	The closed-loop motion control modes (SPEED, TORQUE, etc) generate BLEND commands. Regeneration is greatly reduced.
CURRENT_BYPASS = 0.0	The closed-loop motion control modes (SPEED, TORQUE, etc) generate CURRENT commands. Expect regeneration.

CURRENT_BYPASS > 0.0, CURRENT_BYPASS ≠ 1.0 [By convention, use 2.0]	The closed-loop motion control modes (SPEED, TORQUE, etc) generate PWM commands. Expect regeneration. Motor current is estimated from MOTOR_KT and BYPASS_GAIN (as a proxy for motor resistance). Bus voltage is as-measured by the analog telemetry. BYPASS_STEP is not used. LIMIT_CURRENT is respected. This configuration can be used if the current sensor is inoperative, or if there is a desire to read VDD, VCC or TEMP4 telemetry.
CURRENT_BYPASS = 1.0	The closed-loop motion control modes (SPEED, TORQUE, etc) generate PWM commands. Expect regeneration. Neither ADC is used, so both motor current and bus voltage are unavailable. Set BYPASS_GAIN to approximately motor resistance divided by nominal bus voltage. LIMIT_CURRENT is not respected, and large steps in speed target can result in large input currents. If set to a non-zero value, BYPASS_STEP sets the maximum allowed change of duty cycle ratio within a single control. It is suggested that this not be used unless absolutely necessary. It can destabilize the speed controller if used incorrectly. This configuration is not recommended.

11.3.38 SINUSOID_[PHASE, FREQ, OFFSET]

Please see SINUSOID mode for details.

11.3.39 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.

11.3.40 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.

11.3.41 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.

11.3.42 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.

11.3.43 HALL_TRANSITION

This read-only parameter contains the number of digital Hall-effect sensor transitions seen over the past frame. The value is signed, so transitions in the positive direction of rotation give positive counts and transitions in the negative direction of rotation give negative counts. This is used to help generate the HALL_SPEED measure.

11.3.44 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 (43 msec). The torque is computed as:

TORQUE = INERTIA * (SPEED – PREVIOUS_SPEED) * 93 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.

11.3.45 SFFT_STEP_NUMBER

This parameter contains the step number for the current script (SFFT, life, burn-in, etc). If a script is not running it is set to 0. It is technically a read/write parameter, but using it to jump forward or backwards within a script is discouraged.

11.3.46 SFFT_STEP_TIMER

This parameter contains the length of time, in seconds, that the current script step has been operating. It is updated upwards each frame until the step limit is reached at which time the script moves to the next step. It is technically a read/write parameter, but using it to adjust script timing is discouraged.

11.3.47 SFFT_TELEM_COUNT

This parameter starts counting at zero at the beginning of a script. Each time a self-test telemetry point is stored the count increments. It can be used to determine which self-test results are valid. As soon as the script ends the count is set to zero. It is technically a read/write parameter, but using it to adjust the storage location of self-test results is discouraged.

11.3.48 CRC

Not presently used.

11.3.49 LOAD_SOURCE

Not presently used.

11.3.50 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.

11.3.51 Startup I/O, Floating I/O

At initialization the processor will switch all of its I/O pins to high-impedance digital inputs with ~100 kΩ pull-up resistance on each. The digital value of each port (P0 .. P4) is stored in the Startup I/O bitmap. The pull-ups are then disabled, and the digital value of each port is stored in the Floating I/O bitmap.

These data can help to detect and debug hardware faults on the PCB, including missing pull-up/down resistors, short-circuits on logic lines, and open-circuit solder joints on the processor.

11.3.52 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.

11.3.53 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.

11.3.54 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.

11.3.55 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.

11.3.56 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.

11.3.57 SMBUS_ABORT

This value shows the number of times that the SMBus has aborted a transaction due to a NACK.

11.3.58 SMBUS_TIMEOUT

This value shows the number of times that the SMBus has timed out due to an overly long SCL clock stretch. At present the software does not support this.

11.3.59 SMBUS_STOP

This value shows the number of times that the processor has found SDA to be held low when it wanted to start a transaction, and has issued a STOP condition to free the SMBus.

11.3.60 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 (visible only through an external meter) and in TEMP4 telemetry.

11.3.61 REALTIME_DELAY

If this value is non-zero the processor will consume that many clock cycles per 256 executing NOP instructions. This feature can be used to intentionally degrade the realtime margins. Values greater than 32 will be replaced with 32.

11.3.62 TEMPSENSE_INHIBIT

If this value is non-zero then the TEMP2 and TEMP3 telemetry channels will read as NAN. The digital temperature sensors on the SMBus will not be used.

11.3.63 BUSOFF_REASON

This indicates the reason for the power switch being turned off:

Value	Meaning
0	No reason. Set at turn-on
1	Set upon command to SWITCH_A, SWITCH_B or SWITCH_HIGHEST
2	Set upon command to SWITCH_OFF
3	Switch turned off due to BUS_MIN_THRESHOLD or BUS_MAX_THRESHOLD
4	Switch turned off due to CURRENT_FILTER
5	Switch turned off due to VOLTAGE_FILTER

11.3.64 ANALOG_HALL_DISABLE

If non-zero, then the analog Hall effect sensors are powered down. The following parameters are set to NaN:

- HALL_ROTATION
- HALL_SPEED
- HALL_ANGLE
- HALL_PREVIOUS_ANGLE
- HALL_TRANSITION

This saves some power, and some processor realtime margin.

11.3.65 ADC_REGISTER_REFRESH

This can be used to refresh the ADC gain registers which may become corrupted by ESD events.

Value	Meaning
0	Do not refresh gain registers
1	Evidence that a one-shot refresh has occurred
2	When the wheel is next in a current-control drive mode, refresh the gain registers once and reset this byte to '1'.
3+	Whenever the wheel is in a current-controlled drive mode, refresh the gain registers once per control cycle.

11.3.66 CURRENT_FILTER, CURRENT_IIR_CONSTANT, CURRENT_THRESHOLD

At each control frame, occurring at 91.5 Hz, the IIR filter is updated:

$$CURRENT.FILTER \leftarrow MEASURED.CURRENT \times CURRENT.IIR.CONSTANT + CURRENT.FILTER \times (1 - CURRENT.IIR.CONSTANT)$$

If the absolute value of CURRENT_FILTER exceeds CURRENT_THRESHOLD, and if CURRENT_THRESHOLD is non-zero, then the bus switch is turned off and BUSOFF_REASON is set to 4.

CURRENT_FILTER is set to zero whenever MEASURED_CURRENT is NaN.

Note that some of the configuration parameters are above EDAC address 0x3FF, and so are inaccessible by the READ FILE and WRITE FILE commands.

11.3.67 VOLTAGE_FILTER, VOLTAGE_IIR_CONSTANT, VOLTAGE_THRESHOLD

At each control frame, occurring at 91.5 Hz, the IIR filter is updated:

$$VOLTAGE.FILTER \leftarrow (VA - VBUS)^2 \times VOLTAGE.IIR.CONSTANT + VOLTAGE.FILTER \times (1 - VOLTAGE.IIR.CONSTANT)$$

If the value of VOLTAGE_FILTER exceeds VOLTAGE_THRESHOLD, and if VOLTAGE_THRESHOLD is non-zero, then the bus switch is turned off and BUSOFF_REASON is set to 5.

VOLTAGE_FILTER is set to zero whenever VBUS is NaN.

Note that some of the configuration parameters are above EDAC address 0x3FF, and so are inaccessible by the READ FILE and WRITE FILE commands.

11.4 Command Modes

Command Number	Command Name	Command Value
0x00	IDLE	Ignored
0x01	PWM	Duty cycle (-1.0 to +1.0)
0x02	CURRENT	Amps (-2.5 to +2.5)
0x03	SPEED	Rads/sec
0x04	PWM_H1	Duty cycle (-1.0 to +1.0)
0x05	PWM_H2	Duty cycle (-1.0 to +1.0)
0x06	PWM_H3	Duty cycle (-1.0 to +1.0)
0x07	PWM_H4	Duty cycle (-1.0 to +1.0)
0x08	PWM_H5	Duty cycle (-1.0 to +1.0)
0x09	PWM_H6	Duty cycle (-1.0 to +1.0)
0x0A	CURRENT_H1	Amps (-2.5 to +2.5)
0x0B	CURRENT_H2	Amps (-2.5 to +2.5)
0x0C	CURRENT_H3	Amps (-2.5 to +2.5)
0x0D	CURRENT_H4	Amps (-2.5 to +2.5)
0x0E	CURRENT_H5	Amps (-2.5 to +2.5)
0x0F	CURRENT_H6	Amps (-2.5 to +2.5)
0x10	ACCEL	Rads/sec²
0x11	MOMENTUM	N-m-sec
0x12	TORQUE	N-m
0x13	BURNIN	Final test step #
0x14	SFFT	Final test step #
0x15	LIFE	Final test step #
0x16	STORE_FILES	0.0 or 1.0

0x17	DEFAULT_FILES	0.0 or 1.0
0x18	PWM_P0	Duty cycle (-1.0 to +1.0)
0x19	PWM_P1	Duty cycle (-1.0 to +1.0)
0x1A	PWM_P2	Duty cycle (-1.0 to +1.0)
0x1B	SWITCH_OFF	Ignored
0x1C	SWITCH_A	Ignored
0x1D	SWITCH_B	Ignored
0x1E	SWITCH_HIGHEST	Ignored
0x1F	SOAK	Final test step #
0x20	REPEAT	Ignored
0x21	COMPLETE	Ignored
0x22	TORQUE_TEST	Fraction of nominal torque
0x23	CURRENT_TEST	Fraction of nominal current
0x24	AUX1	Final test step #
0x25	AUX2	Final test step #
0x26	BRAKE	Amps (-2.5 to +2.5)
0x27	BRAKE_H1	Amps (-2.5 to +2.5)
0x28	BRAKE_H2	Amps (-2.5 to +2.5)
0x29	BRAKE_H3	Amps (-2.5 to +2.5)
0x2A	BRAKE_H4	Amps (-2.5 to +2.5)
0x2B	BRAKE_H5	Amps (-2.5 to +2.5)
0x2C	BRAKE_H6	Amps (-2.5 to +2.5)
0x2D	BLEND	Amps (-2.5 to +2.5)
0x2E	BLEND_H1	Amps (-2.5 to +2.5)
0x2F	BLEND_H2	Amps (-2.5 to +2.5)
0x30	BLEND_H3	Amps (-2.5 to +2.5)
0x31	BLEND_H4	Amps (-2.5 to +2.5)
0x32	BLEND_H5	Amps (-2.5 to +2.5)
0x33	BLEND_H6	Amps (-2.5 to +2.5)
0x34	SINUSOID	Rads/sec

11.4.1 IDLE

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

11.4.2 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.

11.4.3 CURRENT

In CURRENT mode the motor is driven with closed-loop current control. Positive values indicate current that will produce positive torque, while negative values indicate current that will produce negative torque.

Since motor torque is proportional to current, this mode can potentially be used for spacecraft fine control. There will be significant disturbances from bearing friction/stiction as the rotor passes through zero speed.

11.4.4 SPEED

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

11.4.5 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.

11.4.6 CURRENT_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 CURRENT mode. It allows a particular current 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.

11.4.7 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.

11.4.8 MOMENTUM

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

11.4.9 TORQUE

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

11.4.10 BURNIN

The BURNIN mode starts a test script intended to bring the bearing lubricant to a steady-state initial condition. Details are TBD.

11.4.11 SFFT

The SFFT mode starts a test script to fully evaluate the health of an integrated reaction wheel. The test will run for a number of minutes before terminating. All of the result data is stored in the parameter file. Consult the factory for automated software that will generate pass/fail reports.

11.4.12 LIFE

The LIFE mode starts a test script intended for long-term operation on a life-test reaction wheel. Details are TBD.

11.4.13 STORE_FILES

If the STORE_FILES mode is entered with a value of exactly 1.0, all of the parameters will be stored to non-volatile flash memory. 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.

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

11.4.14 DEFAULT_FILES

If the DEFAULT_FILES mode is entered with a value of exactly 1.0 the stored parameters in non-volatile flash memory 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. 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.

11.4.15 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. This allows the motor phase voltage to be read on PHASE_COMMON telemetry, while no current flows in the motor.

11.4.16 SWITCH_OFF

This mode turns off both power switches. The switches turn off immediately, and stay off until a further SWITCH command. This mode does not drive the motor, and is equivalent in that way to IDLE.

11.4.17 SWITCH_[A|B]

This mode attempts to turn on one of the power switches. The switch turn-on rate is governed by TURNON_RATE. This mode does not drive the motor, and is equivalent in that way to IDLE.

11.4.18 SWITCH_HIGHEST

If neither power switch is on, this mode compares the VA and VB telemetry. The bus with the highest voltage is turned on. Once this decision is made there are no further

comparisons, so there is no danger of chatter if VA and VB are almost equal. This mode does not drive the motor, and is equivalent in that way to IDLE.

11.4.19 SOAK

The SOAK mode starts a test script intended to facilitate the 120 hour high-temperature burn-in test. It is expected that the electronics unit will be connected to a stator, but there will not be a rotor. The mode drives current through each of the motor phases in turn and logs analog telemetry.

11.4.20 REPEAT

This mode exists as a flag within a script, indicating that the script should return to the first step. It is not a mode that can be usefully commanded by a user.

11.4.21 COMPLETE

This mode exists as a flag within a script, indicating that the script should terminate.. It is not a mode that can be usefully commanded by a user.

11.4.22 TORQUE_TEST

This mode is intended to be used within a script, so that wheels with large and small nominal torques can share the same script files. It is not a mode that should be commanded by a user.

11.4.23 CURRENT_TEST

This mode is intended to be used within a script, so that wheels with large and small nominal torques can share the same script files. It is not a mode that should be commanded by a user.

11.4.24 AUX1, AUX2

These are special test modes, which make customer-specific measurements.

11.4.25 BRAKE

This mode is usually not what the user wants. See BLEND before using BRAKE.

BRAKE mode controls the motor winding current in a closed-loop manner similar to CURRENT mode. However it uses 732 Hz PWM instead of 187.5 kHz. Furthermore, only the low-side MOSFETs are switched. This has the effect of slowing the motor without regenerating significant power back to the spacecraft.

BRAKE should be used with a current sign consistent with slowing the motor. Thus, when the rotor speed is positive the BRAKE sign should be negative. When the rotor speed is negative the BRAKE sign should be positive. A BRAKE command with the wrong sign will result in a maximum braking effort which is probably not what is intended.

11.4.26 BRAKE_H[1..6]

Similar to CURRENT_H[1..6], these modes are the same as BRAKE except that the Hall-effect sensors are overridden.

11.4.27 BLEND

This mode controls the motor winding current in a closed-loop manner. If a gentle braking is required, this emulates BRAKE. If acceleration or rapid deceleration is required, this emulates CURRENT.

This can be thought of as a CURRENT mode with minimal regeneration.

11.4.28 BLEND_H[1..6]

Similar to CURRENT_H[1..6], these modes are the same as BLEND except that the Hall-effect sensors are overridden.

11.4.29 SINUSOID

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.

$$Setpoint = amplitude \cdot sin(SINUSOID_{PHASE}) + SINUSOID_{OFFSET}$$

$$SINUSOID_{PHASE} \leftarrow \left(SINUSOID_{PHASE} + \Delta t \cdot SINUSOID_{FREQ}\right) modulo\ 2\pi$$

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.