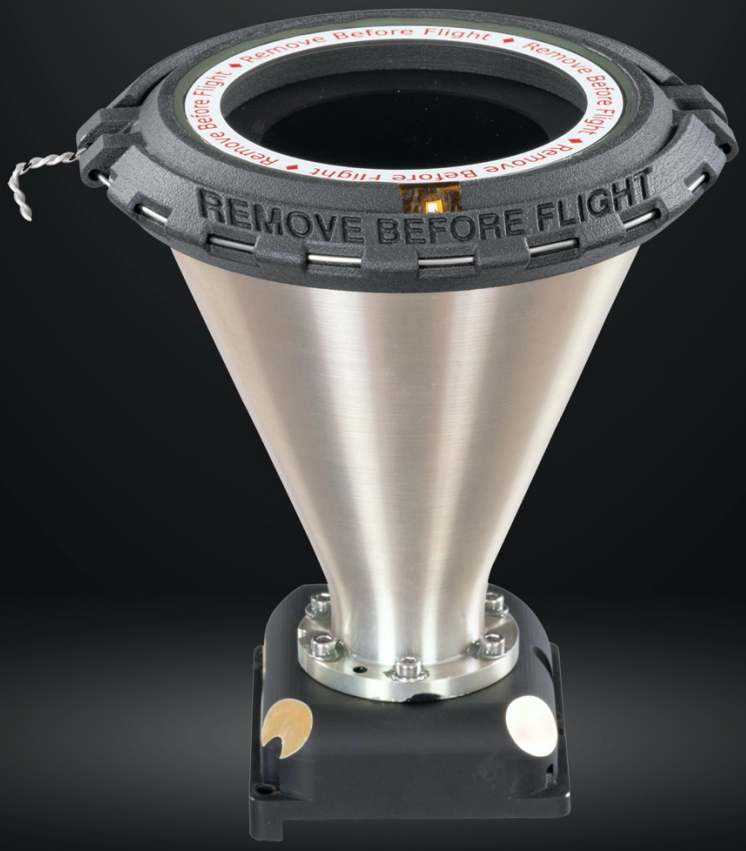


STAR TRACKER

ST-16RT2

Interface Control Document

REV 1.32, APRIL 22, 2021

Doug Sinclair, Cordell Grant

components@rocketlabusa.com

SINCLAIR INTERPLANETARY
BY ROCKET LAB

ROCKET LAB

1. Revision Notes

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

- Fixed typo in the caption of Table 49

2. Markings

ST-16 star trackers are laser-engraved with the text “ST-16” and a six-digital serial number beginning with 001. The lens does not have any markings.

ST-16RT star trackers are laser-engraved with the text “ST-16RT”. The lens has a four-digital serial number beginning with 01. ST-16RT units are not authorized for space-flight use, and are intended as engineering models only.

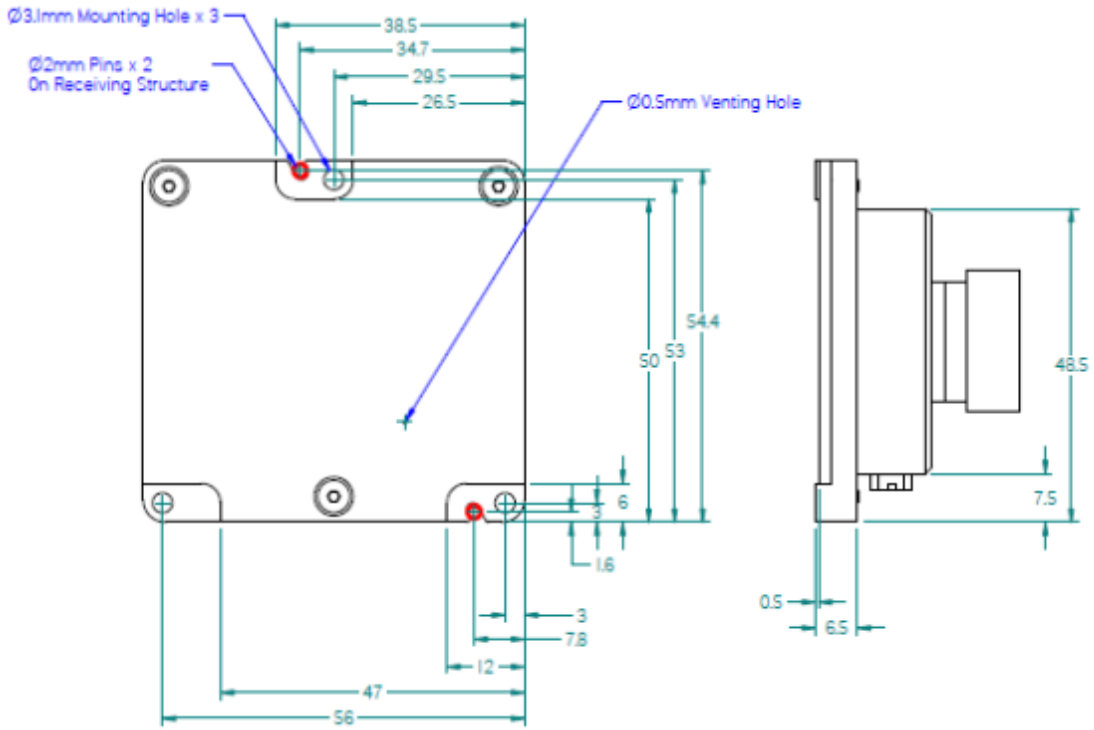
ST-16RT2 star trackers are laser-engraved with the text “ST-16RT”. The lens has a four-digital serial number beginning with 1. Thus, in order to tell the difference between a ST-16RT and a ST-16RT2 the user must inspect the lens serial number.

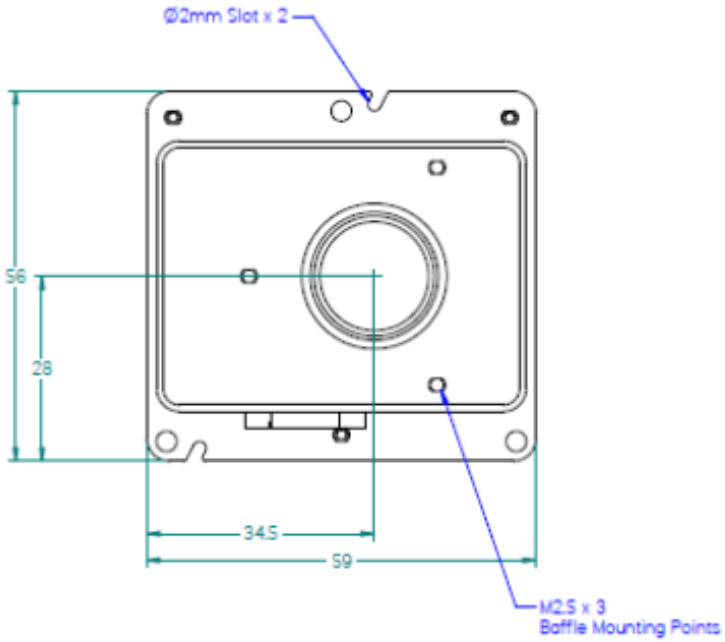
To determine the electronics revision, and thus the pinout, the end-item data package must be consulted. There is no externally observable marking denoting the electronics revision.

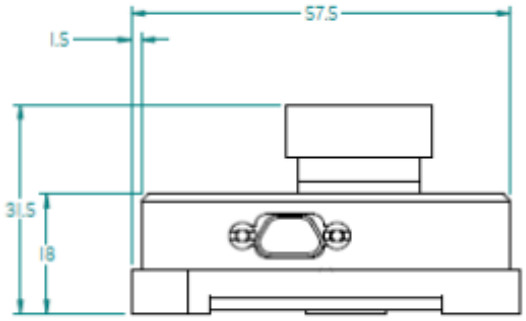
3. Mechanical

3.1. Interface Drawing

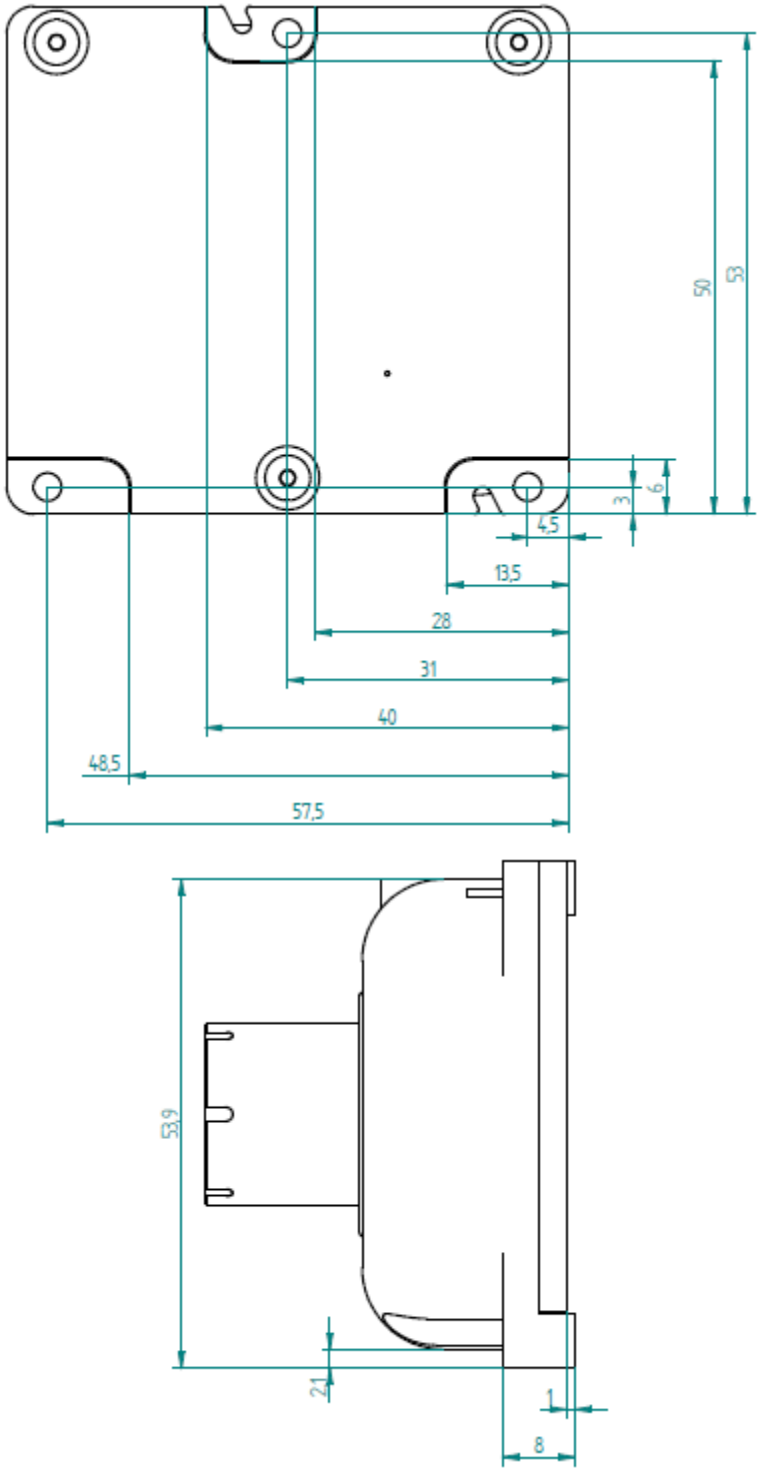
3.1.1. ST-16 Dimensions without Baffle

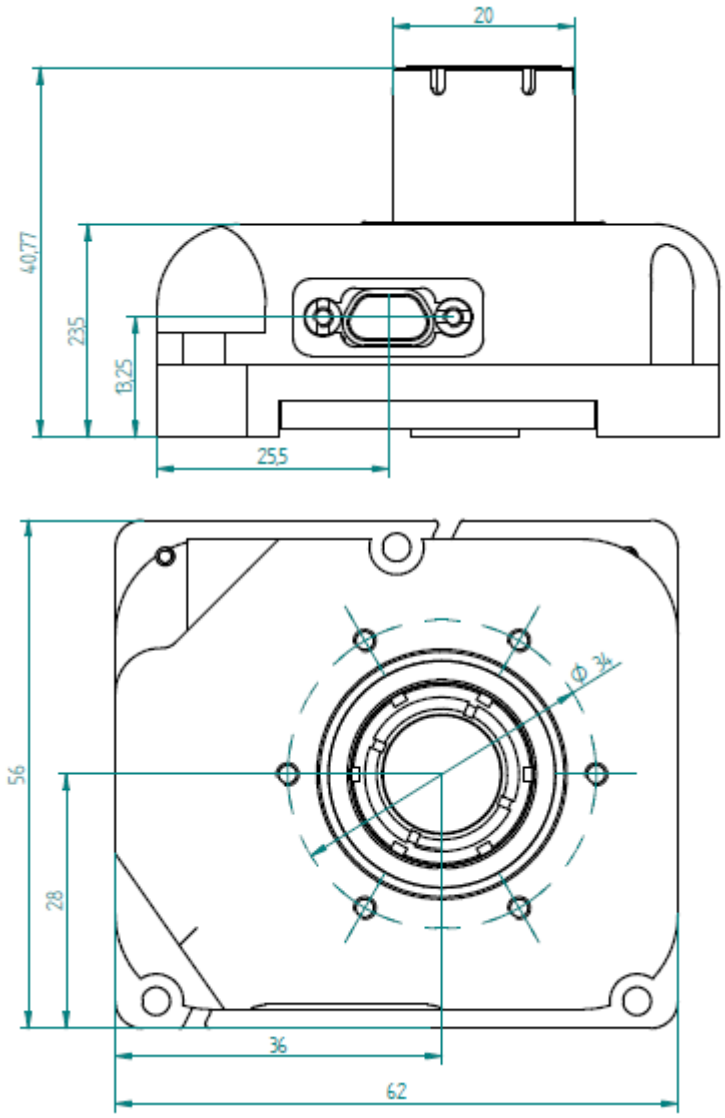



Ø2mm Slot x 2
56
28
34.5
59
M2.5 x 3
Baffle Mounting Points

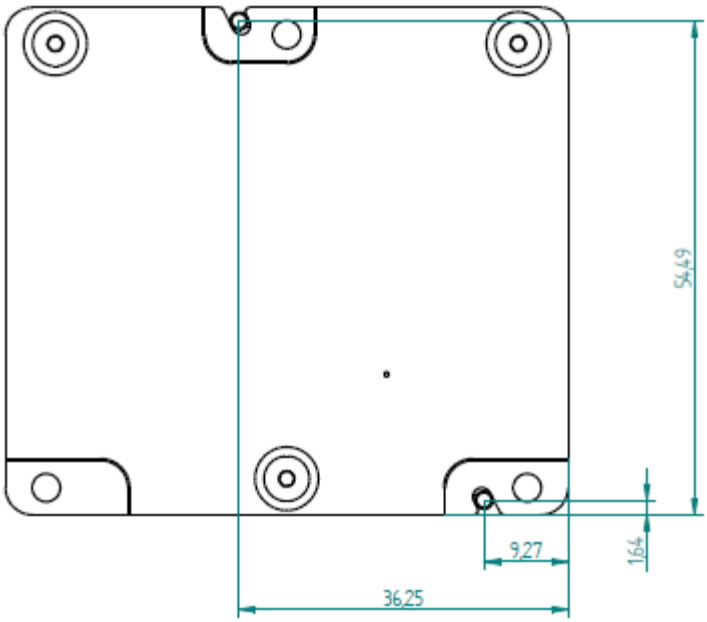

57.5
1.5
31.5
18

3.1.2. ST-16RT (Prism) Dimensions without Baffle

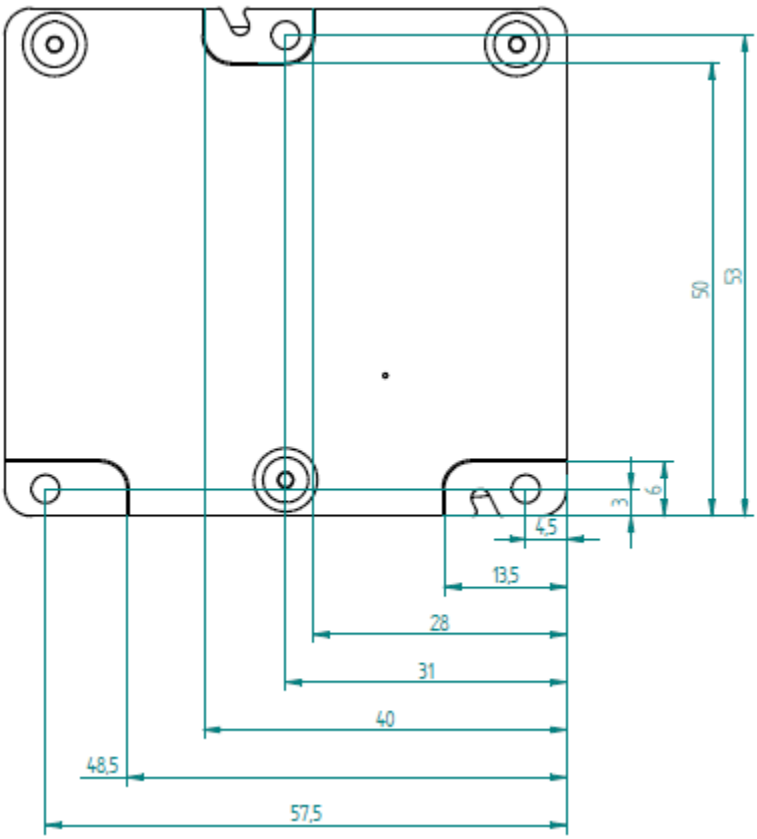


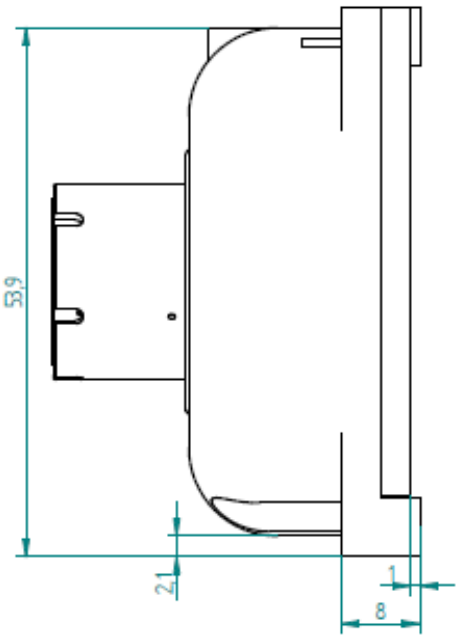


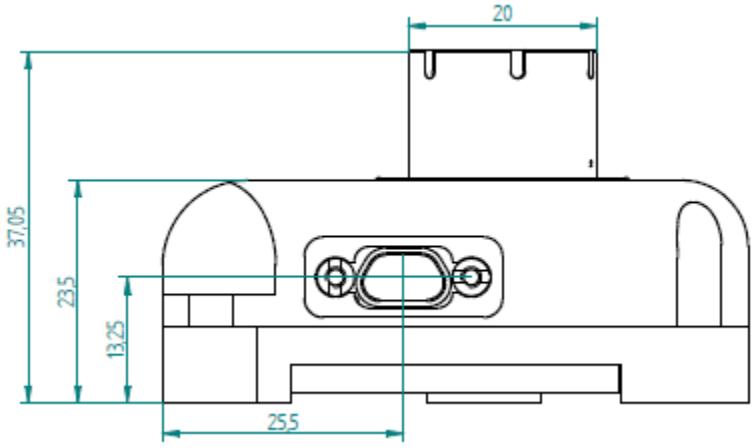
Optionally the spacecraft may use two 2 mm dowel pins to locate the star tracker. They should be located as shown.

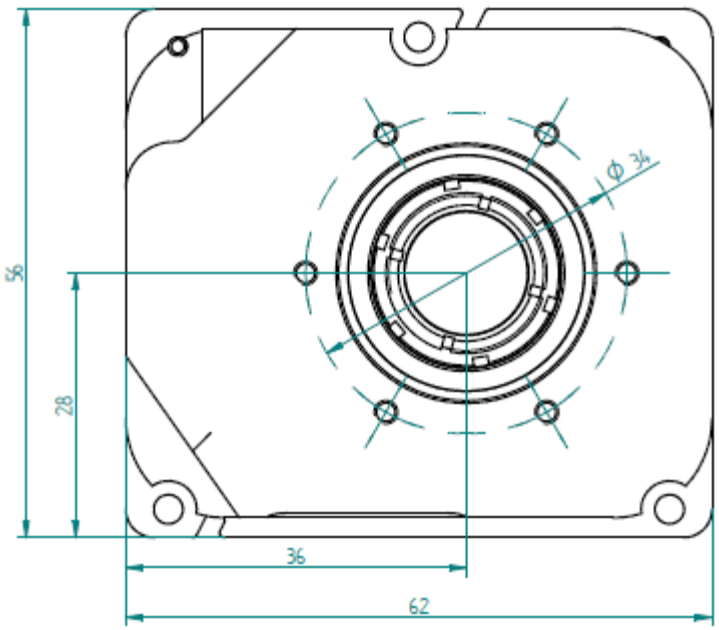


3.1.3. ST-16RT2 (Prism) Dimensions without Baffle

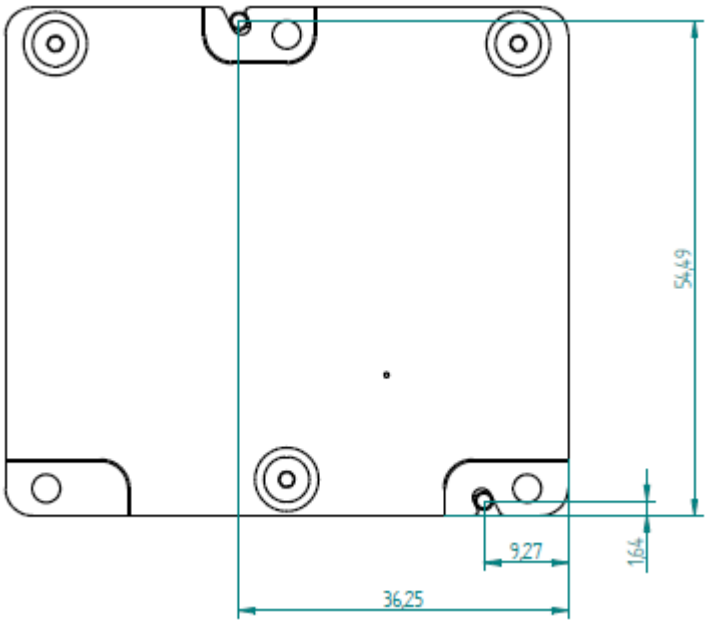



53,9
2,1
1
8

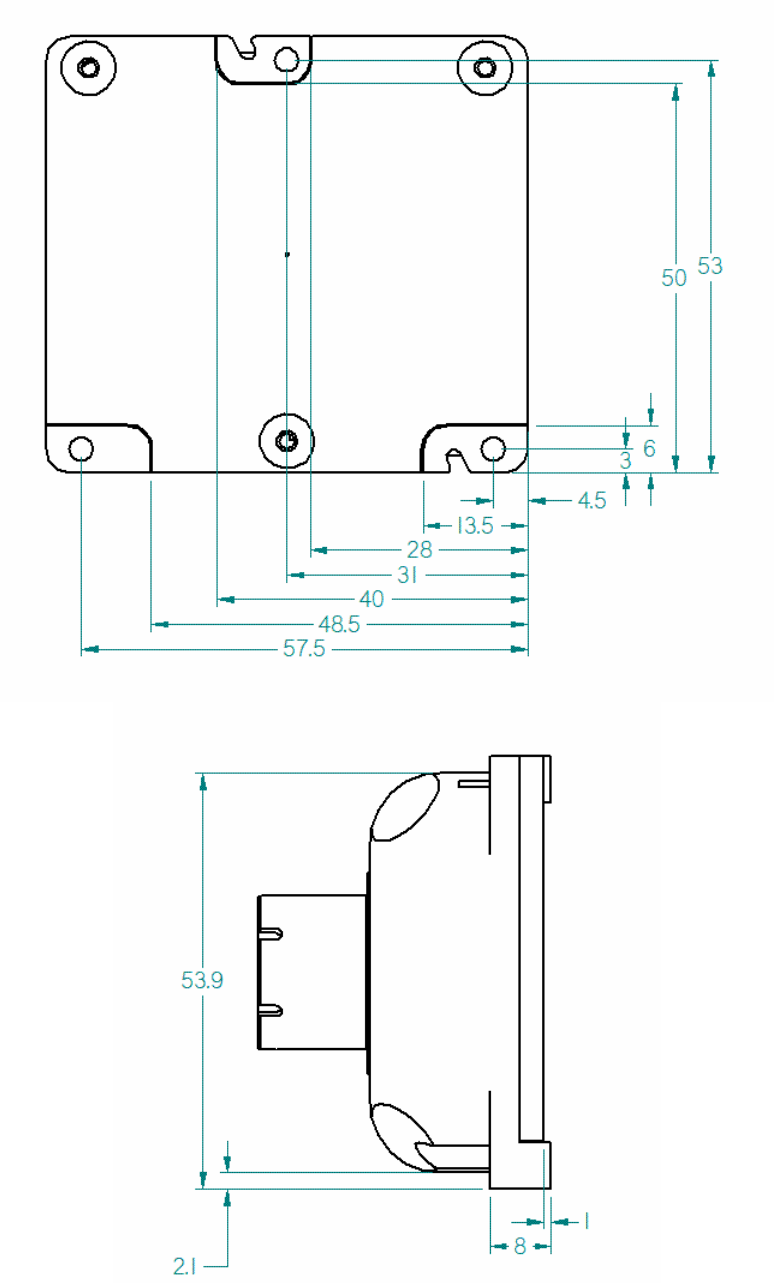

20
37,05
23,5
13,25
25,5

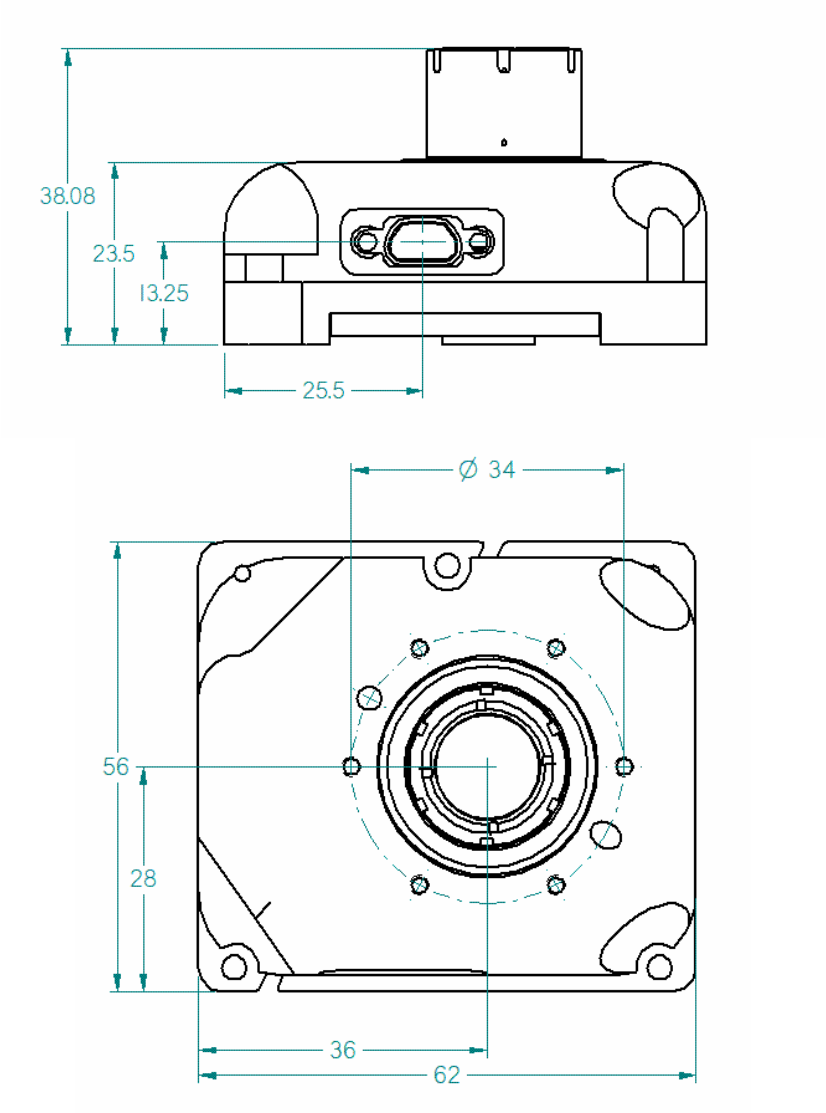


Optionally the spacecraft may use two 2 mm dowel pins to locate the star tracker. They should be located as shown.

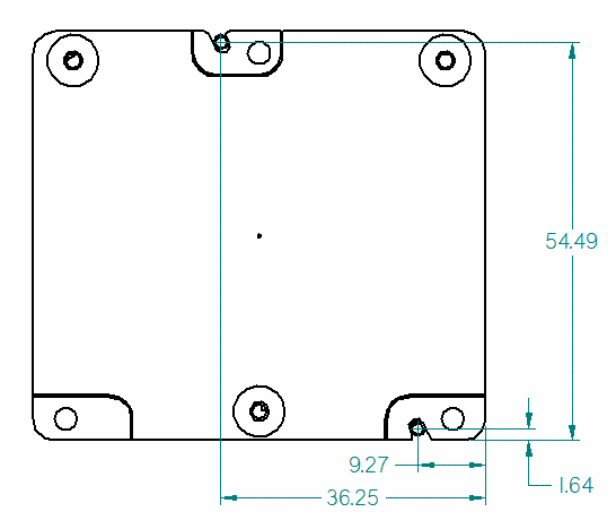


3.1.4. ST-16RT2 (Polished Corners) Dimensions without Baffle



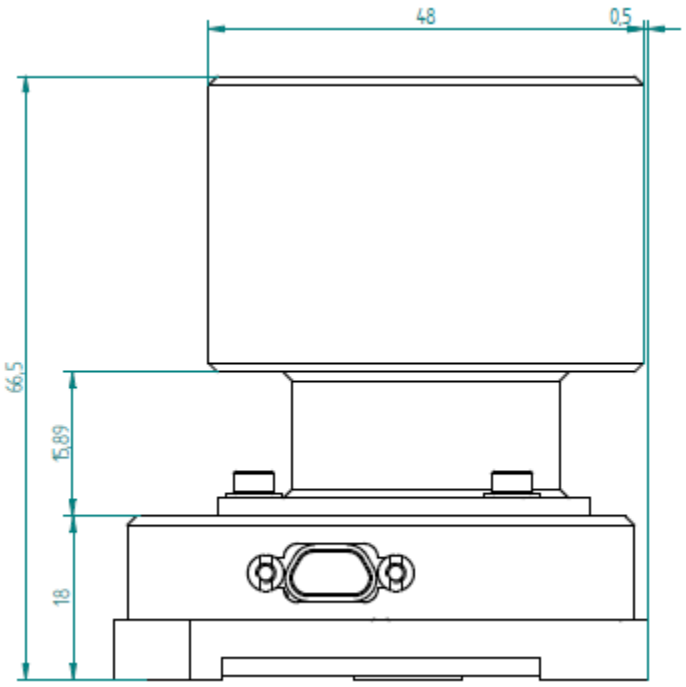


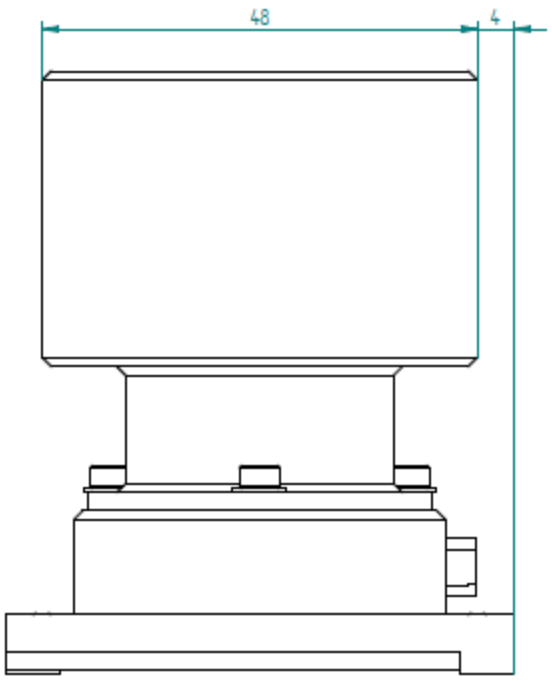
Optionally the spacecraft may use two 2 mm dowel pins to locate the star tracker. They should be located as shown.

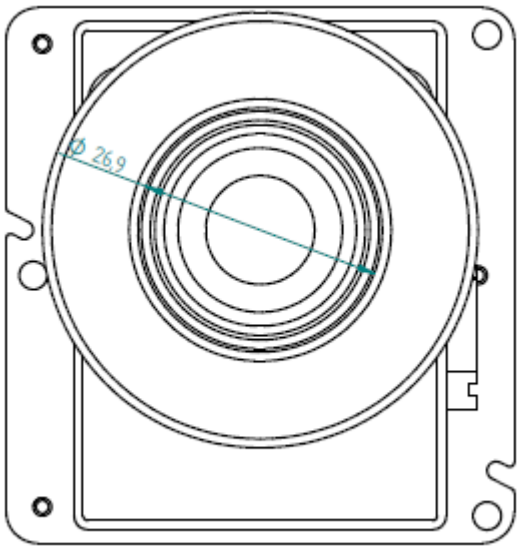


3.1.5. ST-16 Dimensions with Short Rigid Baffle

The baffle shown here is the third generation “lathed baffle” design. Previous machined and electroformed baffle styles for the ST-16 are now obsolete.

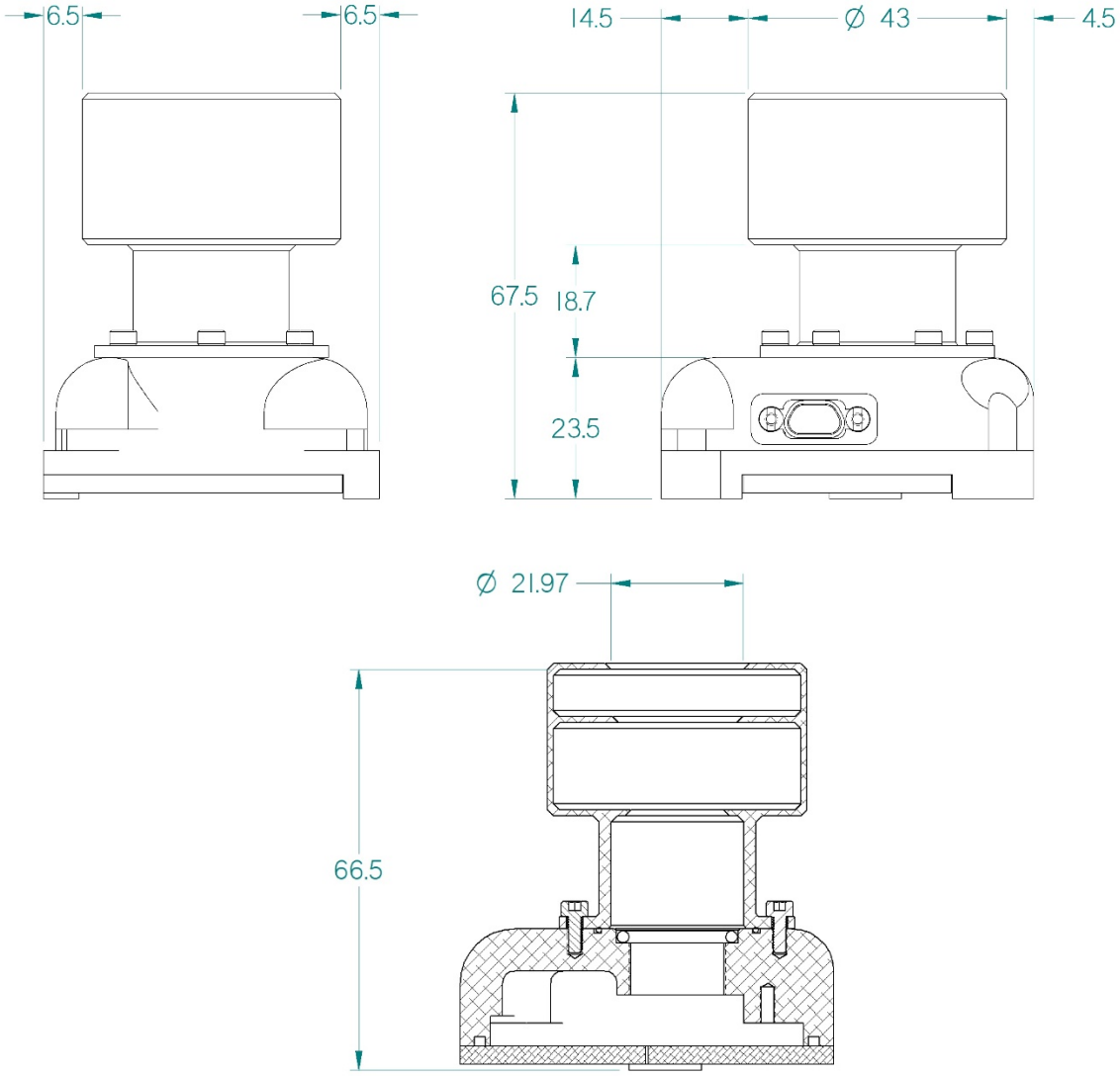



48
4

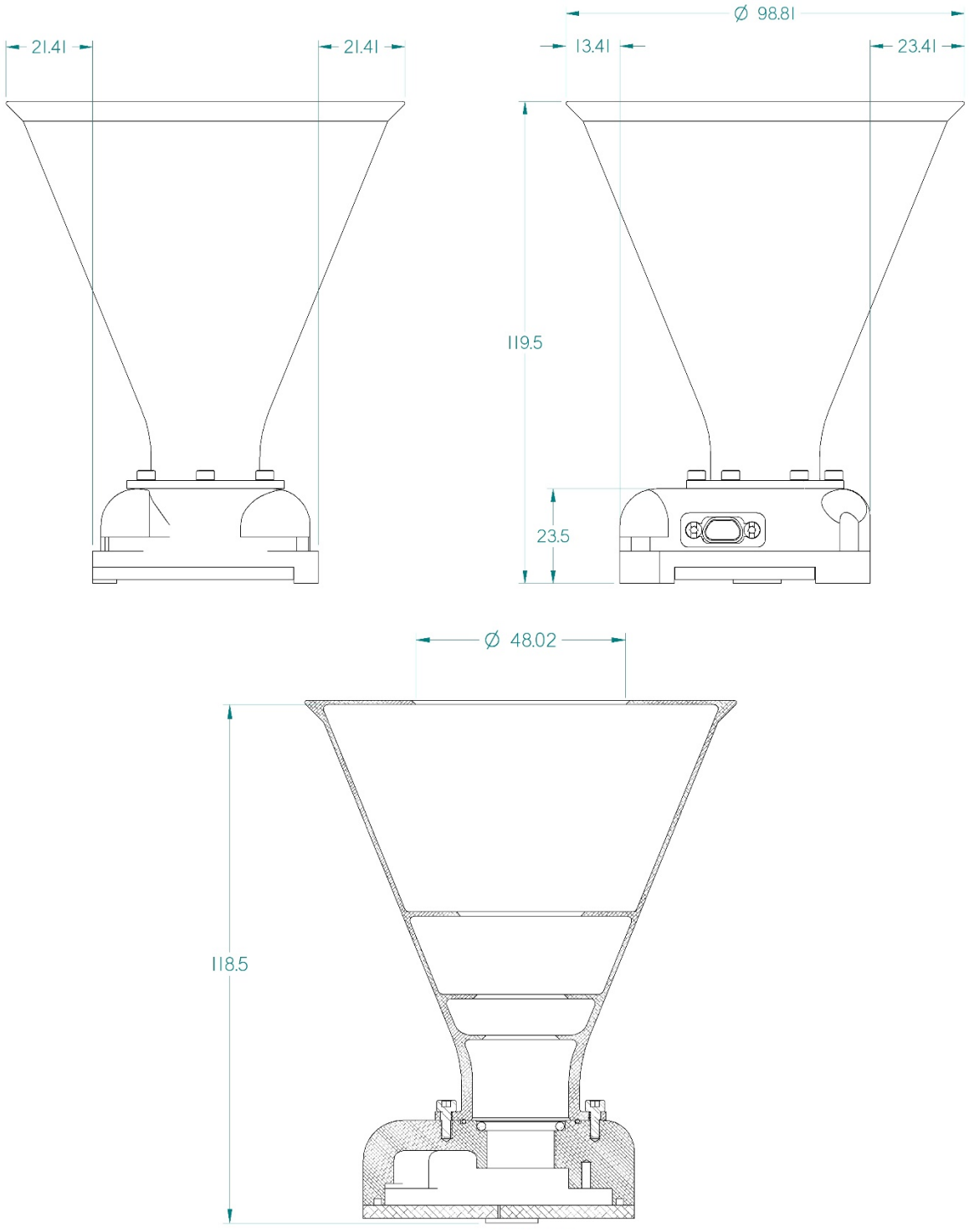

Ø 26.9

3.1.6. ST-16RT2 Dimensions with Short Rigid Baffle

Note, the Short Rigid Baffle for the ST-16RT2 is not equivalent to the Short Rigid Baffle for the ST-16 (see Section 3.1.5).



3.1.7. ST-16RT2 Dimensions with Long Rigid Baffle



3.2. Mass Properties

Table 1: Mass Properties

ST-16 with No Baffle , including dust cap	85.0 g ± 1.0
ST-16 with Short Rigid Baffle	128.5 g ± 1.0
ST-16RT with Prism, no Baffle	153 g
ST-16RT2 with Prism, no Baffle	150 g
ST-16RT2 with Polished Corners, no Baffle	147 g
ST-16RT2 with Polished Corners, Short Rigid Baffle	180 g
ST-16RT2 with Polished Corners, Long Rigid Baffle	230 g

The mass does not include any connector saver, mating connector, or mounting hardware.

3.3. Star Tracker Mounting Points

The star tracker attaches to the host spacecraft with a stable three-point mount. Three through-hole mounting points are provided. These are 3.1 mm in diameter, and are intended to accept M3 hardware. #4 hardware is also acceptable if preferred by the customer. Washers or screw heads should be no more than 8 mm in diameter. The use of soft plastic washers is recommended for non-flight mounting to avoid marring the star tracker surface.

The body of the star tracker is made from aluminum. The bulk of it is black anodized, but the bottoms of the three mounting feet have yellow chemical conversion coating. This material is electrically conductive, and is used as the grounding path for the device. Note that the mounting screws will bear on the anodized top surface, and so the screws are not guaranteed to be grounded on the star tracker side.

In addition to the three screws there is provision to accept two alignment pins. These pins should be precision M2. The flat mounting surface on the spacecraft defines a plane, and the pins then constrain the rotation of the star tracker within this plane.

The star tracker structure is a snug fit on the pins, as is necessary for tight tolerance. The pins may cause some marring of the anodize coating in the star tracker alignment grooves. This is normal. When removing the star tracker from a pinned mounting be careful to withdraw it without any rotation which could cause it to bind. When building GSE mounting plates it may be useful to add holes to the bottom to allow the star tracker to be gently ejected.

3.4. Alignment Reference Surface

The ST-16RT and ST-16RT2 star trackers have either a precision prism installed or have two integral polished corners. The polished corners have replaced the bonded prism. Both features can be used to more accurately survey the alignment of the star tracker relative to the spacecraft payload. See Section 11 on frames on reference for further details.

3.5. Baffle Mounting Points

There are points on the top surface of the star tracker intended to receive a baffle. If a baffle has been provided by Sinclair Interplanetary, then these points will already be

occupied. Otherwise, the customer may use these to attach a rigid or deployed baffle or other structure.

The ST-16 has three points are located on a 38 mm diameter bolt circle. Each will accept an M2.5x0.45 screw to a maximum depth of 4 mm. Stainless steel anti-seizing helical inserts are installed in these holes to prevent damage to the aluminum structure. The screw threads may or may not be grounded – external grounding wires are recommended for solid baffle grounding.

The ST-16RT and ST-16RT2 have six points located on a 34 mm diameter bolt circle. Each will accept an M2.5x0.45 screw to a maximum depth of 4 mm. Stainless steel anti-seizing helical inserts are installed in these holes to prevent damage to the aluminum structure. The screw threads are grounded. An O-ring groove allows the baffle to seal tightly to the star tracker chassis.

Table 2: Entrance Pupil Geometry

Star Tracker Model	Entrance Pupil Diameter	Entrance Pupil Location
ST-16RT	9.96 mm	27.47 mm above mounting plane
ST-16RT2	10.23 mm	27.44 mm above mounting plane

If the baffle is customer-designed, the engineer will need to know the location of the star tracker entrance pupil. This is given in the table above. The mounting plane is the interface plane between the star tracker and the host spacecraft, not the star tracker and the baffle.

3.6. Vent

The star tracker is fitted with a filtered vent located on the bottom cover. This allows internal gasses to be released upon ascent on the launch vehicle. The vent is covered with a porous Teflon filter which will trap all particles 1.0 μm and larger. It will also prevent gross intrusion of liquids, making the sensor somewhat "splashproof". Note that it will not prevent water vapour from entering the sensor and condensing inside.

User cautions associated with the vent are:

1. Ensure that the vent is not covered on the satellite, and in storage. If the star tracker is mounted to a flat plate there is adequate clearance for the vent.
2. Do not submerge the star tracker in cleaning solvents. Use great caution when cleaning the bottom surface to minimize the fluid that enters the vent hole.
3. Do not poke the vent hole with sharp objects.
4. Store the star tracker in a clean environment that is temperature and humidity controlled.

3.7. Fluorine Outgassing

Star trackers that use Rev 4 electronics (typically 4 V input voltage units) contain internal Tefzel type wiring that will slowly offgas minute quantities of fluorine. These units must be stored in a ventilated space so that fluorine does not build up and cause corrosion.

Star trackers that use Rev 5 or later electronics replace the internal wiring with a flex circuit. They do not produce any fluorine, and can be safely stored either in a ventilated clean space or in a sealed container with desiccant.

3.8. Lens Cleaning

When the baffle is not installed the lens is easily accessible, and can be cleaned using conventional camera-cleaning products. The front surface is glass, and so care should be taken to avoid scratching. Do not under any circumstances attempt to unscrew the lens from the star tracker body as the precision focus will be lost.

Once a baffle is installed it is extremely difficult to clean the lens. It is suggested that the star tracker be kept in a clean environment, or capped, at all times when a baffle is used.

3.9. Dust Covers

The ST-16 star tracker is supplied with a black plastic cover on the lens. This must be removed before flight, and before the installation of any baffle. The ST-16RT and ST-16RT2 are fitted with a machined aluminum lens cover which is removed with a screwdriver.

The connector is recessed into the star tracker body such that a standard dust cap will not fit. If specified, the unit will ship with an attached connector saver.

3.10. Connector

The connector built into the star tracker is a Glenair part MWDM2L-9S-0J7-18B (rev 4) or MWDM2L-9S-5C3-.125B (rev 5 and later). It is built on the same line and to the same drawings as MIL-DTL-83513 parts, but is provided without the certifications to reduce the price. The built-in connector has socket contacts and is fitted with jackposts.

Star trackers may be supplied with a mating connector. This is Glenair part MWDM2L-9P-4J7-72M. It has pin contacts, and is fitted with jackscrews. There are nine flying leads, each 72” long. The leads are made from 24 AWG high-strength copper wire with ETFE insulation per M22759/33. Leads are identified by colour. Fluorine outgassing precautions must be observed with these materials.

3.11. Interface Configurations

The star tracker is available from the factory with a number of electrical interface options. The following three options can be built on the 4 V (Rev 4b) PCB by selective component population.

- RS485 (compatible with RS422)
- ASYNC
- CAN

The 28 V (Rev 5) PCB can be configured for Dual RS485 (compatible with RS422) or RS485 + CAN.

3.12. Pinout

Table 3: Connector Pinout, 4 V Option

Pin Number	Wire Colour	RS485 Option	ASYNC Option	CAN Option
1	Black	Power Ground		
2	Brown	Telemetry A	Command	CAN_L
3	Red	Command A	Command	CAN_L
4	Orange	Address Ground		
5	Yellow	/Reset		
6	Green	Power In		
7	Blue	Telemetry B	Telemetry	CAN_H
8	Violet	Command B	Telemetry	CAN_H
9	Grey	Address In		

Table 4: Connector Pinout, 28 V Option

Pin Number	Wire Colour	Dual RS485 Option	RS485 + CAN Option
1	Black	Chassis Ground	Chassis Ground
2	Brown	Power In 1	Power In 1
3	Red	RS485-1 A	CAN_H
4	Orange	RS485-0 A	RS485-0 A
5	Yellow	Power Ground	Power Ground
6	Green	Power Ground	Power Ground
7	Blue	RS485-1 B	CAN_L
8	Violet	RS485-0 B	RS485-0 B
9	Grey	Power In 2	Power In 2

Table 3 shows the connector pinout for the various 4 V build options. Table 4 shows the pinout for star trackers built with 28 V interface. The wire colour is provided for convenience, for those instances where Sinclair Interplanetary provides a mating connector with the star tracker.

3.13. Thermal

3.13.1. Thermal Environment

Operating Temperature	-40 ºC to +50 ºC
Survival Temperature	-40 ºC to +95 ºC

The temperature is defined as the Detector Temperature field in the Hardware Telemetry structure. At the high limit of the operating temperature, dark noise in the detector increases to the point where stars can no longer be detected. There is no danger of hardware damage* at this point, but the star tracker will not reliably produce quaternions.

* Caution: Star trackers sold with short rigid baffles are provided with a yellow dust cover installed on the end of the baffle. This cover is NOT compatible with the hot survival temperature of the star tracker and should be removed prior to any thermal testing that will or could approach the Survival extremes. If the cover is removed, care must be taken to ensure alternate dust protection is provided.

3.13.2. Thermal Interface

The thermal interface to the spacecraft is through conduction. There are 3 mounting feet. For the ST-16, the total area is 168 mm^2. For the ST-16RT2, the total area is 187 mm^2. The contact material is aluminum 6061-T6, with yellow chemical film coating.

The black areas on the body of the star tracker are aluminum 6061-T6, treated with a thick sulfuric acid anodize. If a baffle is fitted, the baffle is also made from aluminum 6061-T6. Yellow areas on the baffle are chemical film coating. Black areas on the baffle are the proprietary Deep Space Black coating from N-Science. The baffle is conductively connected to the body of the star tracker.

In normal operation, the maximum heat load generated by the star tracker is 1.0 W. Since the star tracker may be exposed to the outside of the spacecraft, it will also be radiatively coupled to space, sun and Earth. This can result in large positive or negative heat flows through the mounting feet.

3.13.3. Alternate Baffle Coatings

Sinclair Interplanetary does not provide alternate baffle coatings. Some customers have chosen to apply their own coatings to baffles after delivery. These have included:

- White paint on the outside of the baffle
- Multi-layer insulation (MLI) jacket to cover the outside of the baffle and the star tracker body

Many users mount the star tracker and baffle behind a structural panel, looking out through a circular cutout. This panel provides radiative isolation from space.

4. Protocol Layer 1 (Physical Layer)

4.1. Power Ground

All signals are referred to Power Ground. The 28 V electronics has multiple Power Ground lines, connected together inside the star tracker. Power Ground is connected to the chassis by three capacitors and one 1 MΩ resistor. In the 4 V option the capacitors are 10 nF, 50 V each. In the 28 V option the capacitors are 33 nF, 100 V each.

4.2. Chassis Ground (28 V option only)

Table 5: Chassis Ground Parameters

Absolute Maximum	-50 V to +50 V WRT Power Ground

The Chassis Ground signal is connected to the star tracker housing. It can be used to terminate harness shields if required.

4.3. Power In (4 V option only)

Table 6: Power In Parameters

Operating Range (CAN option)	+4.5 V to +5.5 V
Operating Range (Other options)	+3.3 V to +6 V
Absolute Maximum (CAN option)	-20 V to +6 V
Absolute Maximum (Other options)	-20 V to +20 V

The unit is powered from the Power In signal, which should be connected to a positive voltage with respect to Power Ground. Several built-in linear and switch-mode power supplies generate the internal voltages from this signal. When built with the CAN configuration the CAN transceiver is powered directly from the Power In voltage. This constrains the operating and absolute maximum ranges.

A MOSFET protection switch disconnects the internal supplies when the Power In signal is out of range, protecting the unit from damage in the event of inverted power polarity or unexpectedly high voltages. This switch has a finite turn-off time. Complete protection is not guaranteed in a scenario where the Power In signal suddenly jumps up to a high voltage from a voltage within the Operating Range.

Table 7: Typical Power Consumption, RS485 Configuration

Bootloader mode	6 mA @ 5 V
Idle mode	18 mA @ 5V
Processing mode	250 mA @ 5V peak 150 mA @ 5V typical
Maintenance mode	60 mA @ 5V

Table 7 shows some typical measured power consumption values. In general the supervisor circuits, which are responsible for the bootloader and idle mode, run from linear

regulators. Thus, their current consumption is constant regardless of input voltage. The functional circuits, which are responsible for the bulk of the processing and maintenance mode power, run from DC/DC converters. Thus, at higher voltages they will consume less current.

4.4. Power In [1..2] (28 V option only)

Table 8: Power In[1..2] Parameters

Operating Range	+9 V to +34 V
Absolute Maximum WRT Power Ground	-50 V to +50 V
Absolute Maximum Differential between Power In 1 and Power In 2	-50 V to +50 V

The unit can be powered from either the Power In 1 or Power In 2 signal. Internal diodes select the signal with the highest voltage. These diodes also prevent damage in the case of a shorted or reversed power bus.

The lower end of the operating range is determined by the built-in under-voltage lockout circuit. Contact the factory for units with wider operating ranges, down to as low as 5 V.

Table 9: Typical Power Consumption, RS485 Configuration

Bootloader mode	5.5 mA @ 28 V
Idle mode	7.6 mA @ 28 V
Processing mode	55 mA @ 28 V peak 26 mA @ 28 V typical
Maintenance mode	14.0 mA @ 28 V

Table 9 shows some typical measured power consumption values.

4.5. Address (4 V option only)

Table 10: Address In Parameters

Absolute Maximum	-15 V to + 20 V
Logic High	> 2.3 V
Logic Low	< 1.0 V
Protection	2 kΩ series resistor, 6.8 V TVS diode
Termination	~100 kΩ to +3.3 V

The star tracker determines its network address from the cable harness. If the Address In signal is shorted to Address Ground, then the unit is designated Star Tracker A. If the Address In signal is disconnected then the unit is designated Star Tracker B.

The Address Ground signal is directly connected to Power Ground inside the unit. While Address In is technically a digital input, it is strongly suggested that it be used only for addressing purposes by hard-wiring to Address Ground or isolating. If shorting the two

signals together, do so close to the connector. If isolating, cut the wire short near the connector. This will help to reduce radiated susceptibility EMC problems.

4.6. /Reset (4 V option only)

Table 11: /Reset Parameters

Absolute Maximum	-15 V to + 20 V
Logic High	> 2.3 V
Logic Low	< 1.0 V
Protection	2 kΩ series resistor, 6.8 V TVS diode
Termination	2 kΩ to +3.3 V

The /Reset signal is used in the factory during the bootloader programming operation. In flight it should be left unconnected, with the wire snipped as close to the connector as possible. The internal pull-up resistor will keep the star tracker running normally.

4.7. Command [A|B] (RS485)

Table 12: Command Parameters (RS485)

Absolute Maximum	-11 V to +15 V, each signal, WRT Power Ground
ESD rating	±15 kV (Human-body model)
Polarity	B > A in Mark (Idle, "1") state A > B in Space (On, "0") state
Input Resistance	> 96 kΩ each signal to Power Ground
Input Differential Threshold	± 0.2 V max

The RS-485 pair carrying commands from the spacecraft to the star tracker is called Command. Pay very careful attention to the polarity designation. The formal standard of EIA/TIA-485 is used. Some devices/chips use incorrect A/B designations.

The Command signals are inputs to the star tracker. No line termination is used.

Star trackers will only interpret commands that are addressed to them. Thus, two star trackers with different NSP addresses can share a common Command link.

4.8. Telemetry [A|B] (RS485)

Table 13: Telemetry Parameters (RS485)

Absolute Maximum	-11 V to +15 V, each signal, WRT Power Ground
ESD rating	±15 kV (Human-body model)
Polarity	B > A in Mark (Idle) state A > B in Space (ON) state
Differential Output Voltage	> 2 V into 50 Ω termination. > 1.5 V into 27 Ω termination.
Short-circuit Output Current	300 mA max
Three-State Output Current	± 10μA max
Output Impedance	100 Ω @ 100 MHz, common-mode

The RS-485 pair carrying telemetry from the star tracker to the spacecraft is called Telemetry. Pay very careful attention to the polarity designation. The formal standard of EIA/TIA-485 is used. Some devices/chips use incorrect A/B designations.

The Telemetry signals are outputs from the star tracker. When there is no data to transmit the star tracker will three-state its outputs to allow another unit to drive the bus. Thus two star trackers can share a common Telemetry link.
The Telemetry line driver IC is equipped with an overtemperature shutdown feature which will turn it off if it overheats: for example, if the bus is short-circuited for an extended period.

4.9. RS-485-[0|1] [A|B] (28 V option only)

Table 14: RS-485 Parameters

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

The 28 V option electronics feature one or two bidirectional RS485 pairs. Pay very careful attention to the polarity designation. The formal standard of EIA/TIA-485 is used. Some devices/chips use incorrect A/B designations.

4.10. Command (ASYNC)

Table 15: Command Parameters (ASYNC)

Absolute Maximum	-15 V to +20 V, WRT Power Ground
Input Voltage WRT Power Ground	> 2.2 V in Mark (Idle, “1”) state < 0.7 V in Space (On, “0”) state
Input Resistance	~100 kΩ to +3.3 V

The low-voltage CMOS signal carrying commands from the spacecraft to the star tracker is called Command. The Command signal is an input to the star tracker. No line termination is used, but a small current source biases the signal high if it is left floating.

Star trackers will only interpret commands that are addressed to them. Thus, two star trackers with different NSP addresses can share a common Command link. The Command signal is available on two pins to facilitate daisy-chain wire harnesses.

4.11. Telemetry (ASYNC)

Table 16: Telemetry Parameters (ASYNC)

Absolute Maximum	-15 V to +20 V, WRT Power Ground
Output Voltage	+3.3 V in Mark (Idle, “1”) state, 4.7 kΩ series

	V in Space (On, “0”) state, 4.7 kΩ series +3.3 V in high-Z state, ~100 kΩ series
Short-circuit Output Current	300 mA max
Three-State Output Current	± 10μA max
Output Impedance	100 Ω @ 100 MHz, common-mode

The low-voltage CMOS signal carrying telemetry from the star tracker to the spacecraft is called Telemetry. The Telemetry signal is an output from the star tracker. When there is no data to transmit the star tracker will three-state its outputs to allow another unit to drive the bus. Thus two star trackers can share a common Telemetry link.

The telemetry output has a series resistor to limit the current. It also has a small current source to bias the signal high in high-Z mode. Care must be taken to ensure that the pull-up is sufficient, but also that the signal has enough drive to overcome external pull-up devices in the Space state.

4.12. CAN[_L|_H] (CAN)

Table 17: CAN Parameters

Absolute Maximum	-80 V to +80 V, each signal, WRT Power Ground (4 V option) -36 V to +36 V, each signal, WRT Power Ground (28 V option) (But do not exceed 0.25 W continuous into termination resistor if installed)
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	0.15 V typ (4 V) 0.1 V typ (28 V)
Dominant Output Voltage	VCANH – VCANL > 1.5 V into 45 Ω load (4V) VCANH – VCANL > 2.3 V into 60 Ω load (28V)
Slew Rate	Configurable – Contact factory

The CAN_L and CAN_H signals comprise a differential pair used for both input and output. In the 4 V option the signals are available on two pins each to facilitate daisy-chain wire harnesses.

4.13. Full- and Half-Duplex Operation

4.13.1. 4 V Option

The CAN bus is inherently half-duplex, and the CAN pinout is compatible with daisy-chain wiring by default. The 4 V ASYNC and RS485 options can be used in either full-duplex or half-duplex modes.

Full-duplex is the nominal case. The star tracker has separate Command and Telemetry lines. If there are multiple units on one bus, all of the Command lines are connected together, and all of the Telemetry lines are connected together. The ASYNC pinout permits

daisy-chaining in full-duplex mode. The RS485 pinout does not provide for internal loopback, so the wire connections to put multiple units on one bus must be made elsewhere within the spacecraft.

Half-duplex is achieved by connecting the Command and Telemetry lines together. For ASYNC builds, the Command and Telemetry wire would be connected. For RS485, Command A would be connected to Telemetry A and Command B would be connected to Telemetry B. Half-duplex operation uses fewer wires and does not impact the data throughput. However, the flight computer can no longer shout over a misbehaving subsystem, so take care when attaching critical devices (such as power switch units) to shared half-duplex busses.

Half-duplex operation can be implemented by connecting wires together outside the star tracker. For ASYNC units, one of the command wires can be connected to one of the telemetry wires right at the mating connector. RS485 units can also be operated in half-duplex mode with external splices. Alternatively, jumpers can be installed inside the unit by special order.

4.13.2. 28 V Option

The 28 V option star tracker can have its communications configured in a number of different ways. If the dual RS485 option is selected, the two pairs can be used as follows:

- Two pairs can be used together to make a 4-wire full-duplex RS485 connection (compatible with RS422) handing NSP packets.
- Each pair can be used as a half-duplex RS485 connection handling NSP packets. Thus there are two independent redundant NSP links.
- One pair (RS485-0) can be used as a half-duplex RS485 connection handling NSP packets. The second pair (RS485-1) can be used as a synchronization pulse input or output.

If the CAN option is selected then the CAN bus is inherently a half-duplex communications link. The RS485 pair can then be used for:

- An additional half-duplex RS485 connection handling NSP packets.
- A synchronization pulse input or output.
- A redundant CAN bus selector. The RS485 pair can drive an external DPDT latching relay (some diodes may also be needed) to connect the CAN pair to one of two external busses.

Other uses are also possible. An unused communications pair could drive external bakeout heater, Peltier cooler or baffle deployment actuators, for example.

5. Protocol Layer 2 (Data Link Layer)

5.1. Asynchronous Serial

Units built with the ASYNC and RS485 options 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 18: Default Asynchronous Serial Parameters

Nominal Baud Rate	115.2 kbps
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 star tracker master oscillator. Revision 4 and 5a star trackers use a trimmed CMOS oscillator with ±0.5% tolerance. Revision 5b star trackers use a MEMS silicon oscillator with quartz-like accuracy.

Table 19: Actual Asynchronous Serial Baud Rates

Actual Telemetry Baud Rate	114.8 kbps to 116.0 kbps
Permissible Command Baud Rate	111.9 kbps to 118.8 kbps

5.2. CAN

Units built with the CAN option use the ISO 11898 CAN protocol. The baud rate is programmed into the unit bootloader at the factory, and should be negotiated at the time of purchase. Rates of up to 2 Mbps can be achieved. The star tracker's CAN controller is fed by a 24 MHz clock with ±0.5% tolerance. This information can be used to calculate the actual and permissible CAN baud rates.

6. 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 star tracker uses NSP messages for all communication. This includes communication between the host spacecraft and the supervisor processor, as well as communication between the internal supervisor and functional processors. NSP messages sent between the host and the star tracker must be encapsulated in a manner compatible with the data link layer.

6.1. 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 20: 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.

6.2. CAN NSP Encapsulation

The mechanism for encapsulating NSP messages into one or more CAN messages is TBD.

7. Protocol Layer 4 (Transport Layer)

7.1. Command and Reply

The star tracker 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 star tracker 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.

Notwithstanding the above, the star tracker 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.

7.2. NSP Message Format

Table 21: 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 supervisor bootloader supports a maximum data length of 516 bytes, giving a total message length of 521 bytes. The supervisor application program and the functional processor software both support 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.

7.3. NSP Address

Each star tracker contains two distinct processors: the supervisor processor and the functional processor. Each has its own NSP address.

The supervisor processor is responsive whenever the star tracker is turned on. Direct communication between the functional processor and the outside is possibly only when the supervisor processor is configured to forward packets, typically during troubleshooting or reconfiguration operations.

The user is free to pick one or more NSP addresses for flight computers and other units that may talk to the star tracker. 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.

Whenever the star tracker generates a reply message, its destination address is equal to the source address of the corresponding command message. Other than this, the star tracker pays no attention to the NSP addresses used by the host spacecraft.

7.3.1. 4 V NSP Addresses

Table 22: 4 V NSP Addresses

	Star Tracker A	Star Tracker B
Supervisor Processor	0x0C	0x0Eh
Functional Processor	0x0D	0x0Fh

Table 22 shows the NSP addresses of both processors, as a function of the Star Tracker A/B designation. The A/B designation is controlled by the Address In pin.

7.3.2. 28 V NSP Addresses

The 28 V (Revision 5) star trackers have a different addressing scheme. In bootloader mode, the supervisor processor has the following address options:

Table 23: 28 V Supervisor NSP Addresses

Supervisor NSP Address	Command Port	Telemetry Port
0x0A	RS485-0	RS485-0
0x0C	RS485-0	RS485-1
0x0E	RS485-1	RS485-0
0x08	RS485-1	RS485-1

The supervisor bootloader will accept any of the above addresses, provided the command comes in on the appropriate command port. The associated reply will be sent on the appropriate telemetry port. Thus, addresses 0x0A and 0x08 are half-duplex, while 0x0C and 0x0E are full duplex.

When the supervisor is commanded to transition from bootloader to idle mode, the addressing used in that INIT command is latched. From that point on, the supervisor will only respond to that one address, received on that one command port. Similarly, telemetry will only be sent to the appropriate selected port.

The functional processor address is determined from the supervisor address, as shown.

Table 24: 28 V Functional Processor Address

Supervisor Processor Address	Functional Processor Address
0x08	0x09
0x0A	0x0B
0x0C	0x0D
0x0E	0x0F

7.3.3. Multicast Address

The supervisor processor will respond to an additional NSP address, called the multicast address. Multicast commands will never generate replies. Multicast functionality is not available in bootloader mode. The functional processor has no multicast address.

Table 25: Multicast Addresses

Supervisor Multicast Address	0x07

7.4. Message Control Field

Table 26: 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 star tracker. On telemetry reply messages sent by the star tracker 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 star tracker 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 star tracker 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'.

7.5. 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.

7.6. 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;
      }
}
```

7.7. Error Conditions

The star tracker will ignore NSP command messages where the destination address does not correspond to the NSP address of either the supervisor or functional processor. NSP messages addressed to the functional processor will be ignored if the unit is not in maintenance mode. 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’.

7.8. Command Timing

For the star tracker attitude data to be useful, it is necessary to communicate quaternion epoch information with the host spacecraft. In those configurations where a discrete synchronization pulse connection is not available, timing information is carried over the command and telemetry serial link.

When a command is sent by the host to the supervisor processor, the time that the packet is received is recorded. This is defined as the time of reception of the final FEND, including the stop bit. The host spacecraft may elect to transmit most of a command, and

then delay the final FEND until the exact desired moment. Since there are no timeouts on the NSP link this delay can be arbitrarily long.

When the star tracker returns epoch information in telemetry, the time is measured relative to the FEND character that finishes the GO or COMBINATION command that has caused the imaging cycle.

8. Protocol Layer 5 (Session Layer)

8.1. Operating Modes

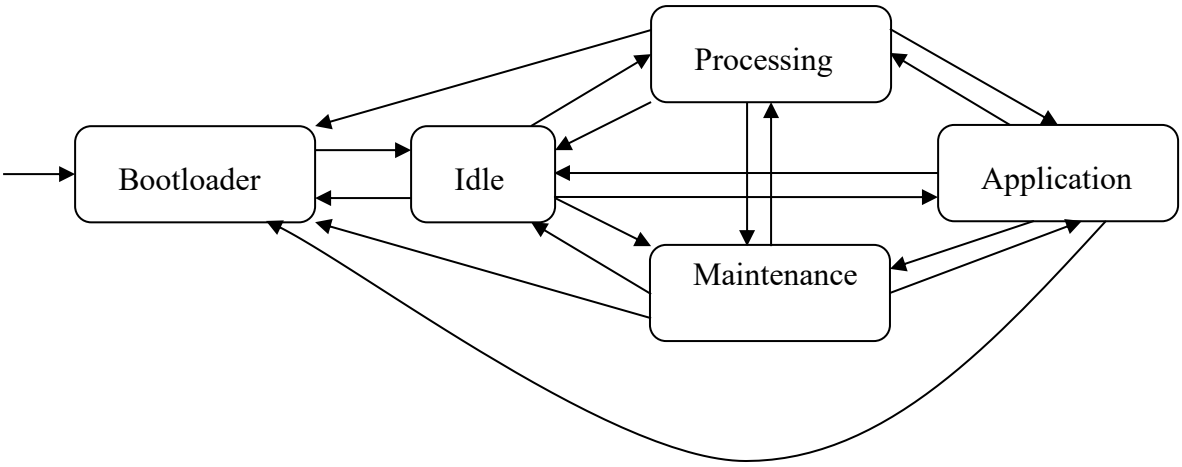


Figure 1: Mode Transition Diagram

Power-on starts the unit in bootloader mode. The remaining mode transitions have the following triggers. In each case where a mode transition is caused by a command, the command is assumed to have been addressed to the supervisor processor.

Table 27: Mode Transition Mechanisms

		From				
		Bootloader	Idle	Processing	Maintenance	Application
To	Bootloader		Supervisor INIT with no data	Supervisor INIT with no data	Supervisor INIT with no data	Supervisor INIT with no data
	Idle	INIT with data 0x00002000		Emergency notification GO requesting functional processor turn-off Timeout, if GO command does not permit remaining on Processing complete, if GO command does not permit remaining on	Emergency notification GO requesting functional processor turn-off Timeout, if GO command does not permit remaining on	Emergency notification Go requesting functional processor turn-off Timeout, if GO command does not permit remaining on
	Processing		GO requesting turn-on, boot from NAND, and control structure sent		GO requesting turn-on, boot from NAND, and control structure sent	GO requesting turn-on, boot from NAND, and control structure sent
	Maintenance		GO requesting turn-on, boot from supervisor flash	GO requesting turn-on, boot from supervisor flash		GO requesting turn-on, boot from supervisor flash Functional INIT with no data
	Application		GO requesting turn-on, boot from NAND, no control structure sent	GO requesting turn-on, boot from NAND, no control structure sent	GO requesting turn-on, boot from NAND, no control structure sent	

				Processing complete, if GO command permits remaining on	Functional INIT with data 0x00008000	

Table 28: Mode Description

<table>
<tr><th>Mode</th><th>Supervisor Processor</th><th>Functional Processor</th><th>Detector</th></tr>
<tr><td>Bootloader</td><td>Running bootloader software.
Vdd regulator at 2.5 V
Clock at 6 MHz
Polling communication</td><td rowspan="2">Powered Off</td><td rowspan="2">Powered Off</td></tr>
<tr><td>Idle</td><td rowspan="4">Running application software
Vdd regulator at 2.1 V
Clock at 48 MHz
Interrupt driven</td></tr>
<tr><td>Processing</td><td>Powered On
Clock at 480 MHz
Busy processing star data</td><td>Powered On
Pixel clock at 85 MHz</td></tr>
<tr><td>Application</td><td>Powered On
Clock at 96 MHz
Polling for NSP messages
Running code from DRAM, copied from NAND image</td><td rowspan="2">Powered On
Held in reset, unclocked</td></tr>
<tr><td>Maintenance</td><td>Powered On
Clock at 96 MHz
Polling for NSP messages
Running code from SRAM, copied from boot image or sent from supervisor</td></tr>
</table>

The table above shows the configuration of the major hardware elements in the different modes.

8.2. Self-Test Sequence

The star tracker can be commanded to run a self test sequence. It will run through the following steps:

1. Turn on the main power switch. Log analog telemetry. Wait 1 second.
2. Turn on the +1.8 V Vdd IO rail. Log analog telemetry. Wait 1 second.
3. Turn on the +2.8 V detector rail. Log analog telemetry. Wait 1 second.
4. Turn on the Vdd core rail. Log analog telemetry. Wait 1 second.
5. Turn on the Vdd MPU rail. Log analog telemetry. Wait 1 second.
6. Release the functional processor reset signal. Log analog telemetry.
7. Boot the functional processor. Switch to full speed. Put the power supplies into default state. Log analog telemetry. Wait 1 second.
8. Command the Vdd MPU rail up to its maximum voltage. Log analog telemetry. Wait 2 seconds.
9. Command the Vdd MPU rail to its default. Command the Vdd core rail to its maximum voltage. Log analog telemetry. Wait 2 seconds.
10. Command the Vdd core rail to its default. Command the Vdd IO rail to its maximum voltage. Log analog telemetry. Wait 2 seconds.
11. Command Vdd IO rail to its default. Engage Smart Reflex dynamic voltage adjustment. Configure the detector for a diagonal test pattern and read one image. Log analog telemetry.
12. Read second image. Wait 1 second. Compare the images to the predicted test pattern, and to each other. Log analog telemetry.
13. Configure the detector with its operational settings and read two images. Return hardware health telemetry. Log analog telemetry. Wait 1 second.
14. Compute image statistics on dark columns. Log analog telemetry.
15. Drop functional processor to idle speed. Wait 1 second. Log analog telemetry.

The timeout period is overridden and set to 30 seconds for the duration of the self-test. This gives the test time to complete before returning to idle mode.

If the self-test command is sent while the functional processor is running the first 5 steps are skipped. The analog telemetry that would have normally been recorded for those steps is replaced with NaN, indicating that the value is unknown.

8.3. Byte Order

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

8.4. Command Codes

Table 29: Command Codes

Command Code	Command	Mode			
		Bootloader	Idle	Processing	Maintenance / Application
0x00	PING	Supervisor	Supervisor	Supervisor	Supervisor + Functional
0x01	INIT	Supervisor	Supervisor	Supervisor	Supervisor + Functional
0x02	PEEK	Supervisor	Supervisor	Supervisor	Supervisor + Functional
0x03	POKE	Supervisor	Supervisor	Supervisor	Supervisor + Functional
0x04	DIAGNOSTIC	Supervisor	Supervisor	Supervisor	Supervisor
0x05	STORE		Supervisor		
0x06	FLASH/CRC	Supervisor	Supervisor	Supervisor	Supervisor + Functional
0x07	READ FILE		Supervisor	Supervisor	Supervisor
0x08	WRITE FILE		Supervisor	Supervisor	Supervisor
0x09	READ EDAC		Supervisor	Supervisor	Supervisor
0x0A	WRITE EDAC		Supervisor	Supervisor	Supervisor
0x0B	GO		Supervisor	Supervisor	Supervisor
0x0C	GATHER RESULT		Supervisor	Supervisor	Supervisor
0x0D	READ RESULT		Supervisor	Supervisor	Supervisor
0x0E – 0x0F	Reserved				
0x10	IMAGE				Functional (Application Only)
0x11	Reserved				
0x12	COMBINATION		Supervisor	Supervisor	Supervisor
0x13	READ TIME		Supervisor	Supervisor	Supervisor
0x14	WRITE TIME				
0x15	WRITE KEPS		Supervisor	Supervisor	Supervisor
0x16 – 0x1F	Reserved				

The table above shows the command codes that can be used by the host spacecraft to communicate with the star tracker. It shows which of the star tracker processors will accept each type of command in each mode. The functional processor will only accept host commands in maintenance and application modes, and the supervisor processor supports only a limited command set in bootloader mode.

In addition to the codes shown above, there are a number of private command codes used by the star tracker's two processors to communicate with each other. These are not documented here.

8.5. PING (0x00)

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

- The type of device and the manufacturer
- The name, and compile time and date of the software that is currently running on the target processor.

8.5.1. Command Format

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

8.5.2. Reply Format

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

8.6. INIT (0x01)

The INIT command is used to change the operating mode of a processor. 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 supervisor 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 supervisor processor primary application program stored at address 0x00002000. Thus, a command of INIT 0x00002000 will start the default behaviour.

If the functional processor is in application mode, it will respond to an INIT with no data by exiting the Application Module and returning to its previous program. This is probably its bootloader (maintenance mode). If one application module is called from a second, then the INIT will return to the first application module.

The functional processor will respond to an INIT with 4 bytes of data by attempting to load an Application Module from the corresponding 32-bit page address in NAND memory. If the CRC of the module fails, NAK will be returned. Otherwise the module will be executed. By convention, devices will ship from the factory with the functional processor primary application program stored at address 0x0008000. Thus, a command of INIT 0x00008000 will start the default behaviour.

When using the INIT command to the functional processor in application mode, be sure that the second application module is compatible with the first. If both use the same DRAM

resources, trouble will occur. Do not attempt to INIT to an Application Module that is already running.

8.6.1. Command Format

Reboot command:

No payload bytes

Application start command:

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

8.6.2. Reply Format

Reboot reply:

No payload bytes

Application start reply:

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

8.7. PEEK (0x02)

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

The supervisor processor has no restriction on the alignment or length of a peek. The functional processor can perform only the following peeks:

- 1 byte length, to any address
- 2 byte length, to any even address
- 4N length, to any 32-bit aligned address

8.7.1. Short Command Format

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

8.7.2. Long Command Format

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

8.7.3. Reply Format

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

8.8. POKE (0x03)

The POKE command is used to write the device memory. The supervisor processor will not permit a POKE into flash memory when any Application Module is running. Each 512 byte block of supervisor processor 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 supervisor processor has no restriction on the alignment or length of a poke. The functional processor can perform only the following pokes:

- 1 byte length, to any address
- 2 byte length, to any even address
- 4N length, to any 32-bit aligned address

8.8.1. Command Format

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

8.8.2. Reply Format

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

8.9. DIAGNOSTIC Command (0x04)

The DIAGNOSTIC command gathers error count data from the supervisor.

8.9.1. Command Format

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

8.9.2. 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

8.10. STORE Command (0x05)

The STORE command saves the supervisor processor parameter file to non-volatile memory. It will only function in idle mode.

8.10.1. Command Format

Byte 0	0 to reset the stored parameter file to defaults 1 to store the current parameter file

8.10.2. Reply Format

Byte 0	0 if stored parameter file reset to defaults 1 if current parameter file stored

8.11. CRC Command (0x06) [Supervisor Processor Only]

Command 0x06 is interpreted by the supervisor processor as a CRC request. The 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 supervisor processor's entire 128 kB flash memory. This can take a number of seconds, especially in bootloader mode where the system clock is much slower.

8.11.1.1. 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

8.11.1.2. 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

8.12. FLASH Command (0x06) [Functional Processor Only]

Command 0x06 is interpreted by the functional processor as a FLASH request. The first byte of the data field is consulted to determine which subcommand is required.

Table 30: FLASH Subcommands

Subcommand Index	Function
0	Read page buffer
1	Write page buffer
2	Erase NAND block
3	Write NAND page
4	Read NAND page
5	Count NAND errors
6	Find NAND bad blocks
7	CRC buffer
8	CRC NAND
9	CRC RAM
10	Make boot block
11	Make bit error
12	Copy NAND

13	NAND read disturbance test
14	Rewrite NAND
15	Read NAND page raw
16	Read NAND ID
17	Read NAND ONFI Parameters
18	Get NAND Features
19	Set NAND Features
20	Upgrade NAND ECC
21	Downgrade NAND ECC
22	Inspect NAND ECC

The NAND flash memory is divided into pages (2 kB each), and blocks (128 kB each). A 2 kB page buffer is maintained in RAM, and allows multiple NSP messages to interact with individual pages. The NAND flash is protected by Error Correcting Codes (ECC) which are generally invisible to the user. The codes can correct 1 bit error in a 512 byte section.

NAND flash can be additionally protected using the backup feature, where a block is assigned a second backup block to be used in case of failure.

8.12.1. Read Page Buffer

The Read Buffer subcommand reads the current contents of the page buffer. Note that this command is of no use until another command has been used to put data into the page buffer. The command is available in long and short variants.

8.12.1.1. Short Command Format

Byte 0	Value 0, indicating Read Page Buffer subcommand
Bytes 1 – 2	16-bit offset within buffer to start reading
Byte 3	Number of bytes to read. 0 indicates 256 bytes should be read.

8.12.1.2. Long Command Format

Byte 0	Value 0, indicating Read Page Buffer subcommand
Bytes 1 – 2	16-bit offset within buffer to start reading
Byte 3 - 4	Number of bytes to read.

8.12.1.3. Reply Format

Byte 0	Value 0, indicating Read Page Buffer subcommand
Bytes 1 – 2	16-bit offset within buffer where reading started
Byte 3 - N	Bytes read

8.12.2. Write Page Buffer

The Write Buffer subcommand writes into the page buffer. Note that the page buffer must then be used by another command for this to be useful.

8.12.2.1. Command Format

Byte 0	Value 1, indicating Write Page Buffer subcommand

Bytes 1 – 2	16-bit offset within buffer to start writing
Byte 3 - N	Bytes to write

8.12.2.2. Reply Format

Byte 0	Value 1, indicating Write Page Buffer subcommand
Bytes 1 – 2	16-bit offset within buffer where writing started
Byte 3 - N	Bytes written

8.12.3. Erase NAND Block

This subcommand erases a block of NAND memory. The entire block is erased so that all bytes read 0xFF.

8.12.3.1. Command Format

Byte 0	Value 2, indicating Erase NAND Block subcommand
Bytes 1 – 4	32-bit number of a page within the target block

8.12.3.2. Reply Format

Byte 0	Value 2, indicating Erase NAND Block subcommand
Bytes 1 – 4	32-bit number of a page within the target block

8.12.4. Write NAND Page

This subcommand copies the contents of the page buffer into the target NAND page. For correct operation the page should have been previously erased.

8.12.4.1. Command Format

Byte 0	Value 3, indicating Write NAND Page subcommand
Bytes 1 – 4	32-bit number of target page to write to

8.12.4.2. Reply Format

Byte 0	Value 3, indicating Write NAND Page subcommand
Bytes 1 – 4	32-bit number of target page that was written.

8.12.5. Read NAND Page

This subcommand copies the contents of the target NAND page into the page buffer. It will return NAK if there are uncorrectable ECC errors or if the page is in the erased condition.

Even if NAK is returned the page buffer will have been updated. This allows some data to be recovered from the page, even if there are errors.

8.12.5.1. Command Format

Byte 0	Value 4, indicating Read NAND Page subcommand
Bytes 1 – 4	32-bit number of target page to read from

8.12.5.2. Reply Format

Byte 0	Value 4, indicating Read NAND Page subcommand
Bytes 1 – 4	32-bit number of target page that was read.

8.12.6. Count NAND Errors

This subcommand examines a number of contiguous NAND pages for ECC errors. It counts the number of pages that contain fixable errors, the number of pages that contain uncorrectable ECC errors, the number of pages that require access to the backup, and the number of erased pages.

8.12.6.1. Command Format

Byte 0	Value 5, indicating Count NAND Errors subcommand
Bytes 1 – 4	32-bit number of the first page to examine
Bytes 5 - 8	32-bit number of the last page to examine

8.12.6.2. Reply Format

Byte 0	Value 5, indicating Count NAND Errors subcommand
Bytes 1 – 4	32-bit number of the first page that was examined
Bytes 5 – 8	32-bit number of the last page that was examined
Bytes 9 – 12	32-bit number counting the number of pages with fixable ECC errors
Bytes 13 – 16	32-bit number counting the number of pages with uncorrectable ECC errors that cannot be corrected from the backup (either there is no assigned backup, or the backup also has uncorrectable ECC errors). Blocks tagged as bad are included here.
Bytes 17 – 20	32-bit number counting the number of pages where the primary has uncorrectable ECC errors but the backup is readable
Bytes 21 – 24	32-bit number counting the number of pages that are in a completely erased state

Each page is considered to belong to at most one of these categories.

8.12.7. Find NAND Bad Blocks

This subcommand examines a number of contiguous NAND pages for the factory bad block flag.

8.12.7.1. Command Format

Byte 0	Value 6, indicating Count NAND Errors subcommand
Bytes 1 – 4	32-bit number of the first page to examine
Bytes 5 - 8	32-bit number of the last page to examine

8.12.7.2. Reply Format

Byte 0	Value 6, indicating Count NAND Errors subcommand
Bytes 1 – 4	32-bit number of the first page that was examined
Bytes 5 – 8	32-bit number of the last page that was examined

Bytes 9 – 12	32-bit number counting the number of pages with bad block flags
Bytes 13 – 16	32-bit number showing the page number of the first page with a bad block flag. Zero is returned if no bad block flags were found.

8.12.8. CRC Buffer

This subcommand computes a 16-bit CRC of the page buffer.

8.12.8.1. Command Format

Byte 0	Value 7, indicating CRC Buffer subcommand

8.12.8.2. Reply Format

Byte 0	Value 7, indicating CRC Buffer subcommand
Bytes 1 – 2	16-bit CRC of page buffer.

8.12.9. CRC NAND

This subcommand computes a 16-bit CRC of a number of contiguous NAND pages. It will return NAK if any of the pages contains uncorrectable ECC errors or if the page is in the erased condition.

8.12.9.1. Command Format

Byte 0	Value 8, indicating CRC Buffer subcommand
Bytes 1 – 4	32-bit number of the first page to CRC
Bytes 5 – 8	32-bit number of the last page to CRC

8.12.9.2. Reply Format – Success

Byte 0	Value 8, indicating CRC Buffer subcommand
Bytes 1 – 4	32-bit number of the first page that was commanded CRCed
Bytes 5 - 8	32-bit number of the last page that was commanded CRCed
Bytes 9 - 10	16-bit CRC of NAND region

If successful, a 16-bit CRC result is returned with an ACK code.

8.12.9.3. Reply Format – Failure due to ECC error

Byte 0	Value 8, indicating CRC Buffer subcommand
Bytes 1 – 4	32-bit number of the first page that was commanded CRCed
Bytes 5 - 8	32-bit number of the last page that was commanded CRCed
Bytes 9 - 12	32-bit number of the page where the error was located

If there is an uncorrectable ECC error, no CRC is returned. Instead, the page where the error is located is returned. The CRC attempt stops on the first uncorrectable ECC error, so it is possible that there are also additional errors in subsequent pages.

8.12.10. CRC RAM

This subcommand computes a 16-bit CRC of a region of RAM. [It is grouped with FLASH commands for convenience only.] Be careful to specify correct addresses, as a data abort emergency notification will be generated if unimplemented memory is read.

8.12.10.1. Command Format

Byte 0	Value 9, indicating CRC RAM subcommand
Bytes 1 – 4	32-bit address of the first byte to CRC
Bytes 5 - 8	32-bit address of the last byte to CRC

8.12.10.2. Reply Format

Byte 0	Value 9, indicating CRC RAM subcommand
Bytes 1 – 4	32-bit address of the first byte that was CRCed
Bytes 5 - 8	32-bit address of the last byte that was CRCed
Bytes 9 - 10	16-bit CRC of RAM region

8.12.11. Make Boot Block

This subcommand performs the following actions:

- Erases the NAND block containing the target page
- Copies 32 kB from SRAM to NAND memory, starting with the target page, prepending the required Configuration Header data.

If the target page is 0, this writes the default boot block allowing the functional processor to load its software from NAND in the future. Target pages of 64, 128 and 196 can be used to write the backup boot bocks which are searched, in order, if the primary boot block suffers an irrecoverable ECC error.

The command takes approximately 20 msec to execute, with the reply being sent once execution is complete.

8.12.11.1. Command Format

Byte 0	Value 10, indicating Make Boot Block subcommand
Bytes 1 – 4	32-bit number of target page to begin writing.

8.12.11.2. Reply Format

Byte 0	Value 10, indicating Make Boot Block subcommand
Bytes 1 – 4	32-bit number of first page that was written.

8.12.12. Make Bit Error

This subcommand reads the entire block of NAND memory into an internal RAM buffer, inverts one of the bits, erases the block, and reprograms it from the buffer. It has the effect of flipping a single bit. The intent of this command is to test the ECC codes. Note that the

address used here is a physical address. The block relocation and backup tables have no effect on this command.

The target bit number is encoded as follows:

Bits 0 – 3 (4 least significant bits)	Number of target bit within a target 16-bit word. 0 corresponds to the LSB, and 15 to the MSB.
Bits 4 – 14 (11 bits)	Number of target word within the target page. 0 – 1023 span the data range of the 2 kB page, while 1024+ targets the extra ECC area.
Bits 15 – 31 (17 bits)	Number of target page within the flash memory.

8.12.12.1. Command Format

Byte 0	Value 11, indicating Make Bit Error subcommand
Bytes 1 – 4	32-bit number of target bit to flip

8.12.12.2. Reply Format

Byte 0	Value 11, indicating Make Bit Error subcommand
Bytes 1 – 4	32-bit number of target bit to flip.

8.12.13. Copy NAND

This subcommand copies a number of contiguous pages of NAND memory from one location to another. The target location is not automatically erased. However, the subcommand will ensure that there is no good data (i.e. no pages with good ECC) in the target area before beginning to write.

8.12.13.1. Command Format

Byte 0	Value 12, indicating Copy NAND subcommand
Bytes 1 – 4	32-bit destination page number
Bytes 5 – 8	32-bit source page number
Bytes 9 – 12	32-bit number of pages to copy

8.12.13.2. Reply Format

Byte 0	Value 12, indicating Copy NAND subcommand
Bytes 1 – 4	32-bit destination page number
Bytes 5 – 8	32-bit source page number
Bytes 9 – 12	32-bit number of pages to copy

8.12.14. NAND Read Disturbance Test

This subcommand repeatedly reads a number of contiguous pages of NAND memory and checks the ECC. If no uncorrectable errors are found after the required number of cycles, an ACK and a successful reply packet is returned. If an uncorrectable ECC error is found the subcommand will stop and return an ECC failed reply packet with a NACK. A syntax failed reply format packet will be immediately returned with a NACK if the command parameters are out of range – for example, if the final page is before the start page. NACK

will also be returned if a large enough memory buffer could not be allocated. The largest area that can be disturbance tested at one time is approximately 16 MB.

This command can take quite some time (roughly 1 second per 9000 pages read) to execute if the read cycle time is large. It will emit a functional processor message (which will be stored in the supervisor EDAC) after every 1000 read cycles. This can be polled to determine the command progress.

To abort a read disturbance test, command the supervisor to reset the functional processor. The functional processor itself is non-responsive while performing this test.

8.12.14.1. Command Format

Byte 0	Value 13, indicating NAND Read Disturbance Test subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number
Bytes 9 – 12	32-bit maximum read cycle count

8.12.14.2. Successful Reply Format

Byte 0	Value 13, indicating NAND Read Disturbance Test subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number
Bytes 9 – 12	32-bit number of read cycles completed

8.12.14.3. Syntax Failed Reply Format

Byte 0	Value 13, indicating NAND Read Disturbance Test subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number
Bytes 9 – 12	32-bit maximum read cycle count

8.12.14.4. ECC Failed Reply Format

Byte 0	Value 13, indicating NAND Read Disturbance Test subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number
Bytes 9 – 12	32-bit number of read cycles completed before failure
Bytes 13 – 16	32-bit number of page that failed ECC

8.12.15. Rewrite NAND

This subcommand reads blocks of NAND into RAM, erases the NAND, and reprograms them from the RAM image. This will fix any correctable ECC errors and remove the effects of cumulative read disturbance.

If a block has a backup then one of the two blocks will be fully erased and rewritten before the other is erased. This should leave the NAND workable even if there is an unexpected reset or power loss during the operation. The block that has the most errors is erased and rewritten first.

The subcommand takes a start and end page number. The page numbers will be extended on either end so that a whole number of blocks is accessed.

Pages that started in an erased state will not be reprogrammed. Blocks that start in an erased state will not be re-erased.

The subcommand will fail if a block cannot be successfully read. If there is a backup, this means that neither the primary nor secondary pass ECC. In this case it will abort before erasing the block.

8.12.15.1. Command Format

Byte 0	Value 14, indicating Rewrite NAND subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number

8.12.15.2. Successful Reply Format

Byte 0	Value 14, indicating Rewrite NAND subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number

8.12.15.3. Syntax Failed Reply Format

Byte 0	Value 14, indicating Rewrite NAND subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number

8.12.15.4. Operation Failed Reply Format

Byte 0	Value 14, indicating NAND Read Disturbance Test subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number
Bytes 9 – 12	32-bit page number of fault

8.12.16. Read NAND Page Raw

This subcommand reads a page from NAND into the page buffer. It is intended as a low-level debugging operation, and differs from the regular Read NAND Page in the following ways:

- The block relocation table is ignored, and the addressed physical page is read
- No attempt is made to use ECC to correct errors. Any bit errors in the page are presented.
- Additional ECC information is appended to the end of the page buffer
- The command has no failure condition, and will always ACK

The page buffer format is:

Bytes 0 – 2047	2048 Data bytes from the NAND page
Bytes 2048 – 2111	64 Data bytes from the NAND page spare area
Bytes 2112 – 2127	16 bytes of the expected ECC registers, based on the spare area
Bytes 2128 – 2144	16 bytes of the actual ECC registers following the read

8.12.16.1. Command Format

Byte 0	Value 15, indicating Read NAND Page Raw subcommand
Bytes 1 – 4	32-bit number of target page to read from

8.12.16.2. Reply Format

Byte 0	Value 15, indicating Read NAND Page Raw subcommand
Bytes 1 – 4	32-bit number of target page that was read.

8.12.17. Read NAND ID

This subcommand reads the NAND device ID. This is a 40-bit code which describes some of the basic parameters of the NAND IC.

8.12.17.1. Command Format

Byte 0	Value 16, indicating Read NAND ID subcommand

8.12.17.2. Reply Format

Byte 0	Value 16, indicating Read NAND ID subcommand
Bytes 1 – 5	40-bit ID code

8.12.18. Read NAND ONFI Parameters

This subcommand reads the NAND ONFI (Open NAND Flash Interface) parameters table into the page buffer.

The page buffer format is:

Bytes 0 – 255	256 byte ONFI parameter table copy 1
Bytes 256 – 511	256 byte ONFI parameter table copy 2
Bytes 512 – 767	256 byte ONFI parameter table copy 3

8.12.18.1. Command Format

Byte 0	Value 17, indicating Read NAND ONFI Parameters subcommand

8.12.18.2. Reply Format

Byte 0	Value 17, indicating Read NAND ONFI Parameters subcommand

8.12.19. Get NAND Feature

This subcommand reads one of the 32-bit feature registers from the NAND IC.

Known feature registers are:

Address	Feature Register
01h	Timing Mode
80h	Programmable I/O Drive Strength
81h	Programmable R/B# Pull-Down Strength
90h	Array Operation Mode

8.12.19.1. Command Format

Byte 0	Value 18, indicating Get NAND Feature subcommand
Byte 1	8-bit address of the desired feature

8.12.19.1. Reply Format

Byte 0	Value 18, indicating Get NAND Feature subcommand
Byte 1	8-bit address of the desired feature
Bytes 2 - 6	32-bit contents of feature register

8.12.20. Set NAND Feature

This subcommand writes one of the 32-bit feature registers in the NAND IC. Attempts to write to feature register 0x90 will result in NAK. This is to prevent accidental interference with the ECC and OTP functionality.

8.12.20.1. Command Format

Byte 0	Value 19, indicating Set NAND Feature subcommand
Byte 1	8-bit address of the desired feature
Bytes 2 - 6	32-bit contents of feature register

8.12.20.2. Reply Format

Byte 0	Value 19, indicating Set NAND Feature subcommand
Byte 1	8-bit address of the desired feature
Bytes 2 - 6	32-bit contents of feature register

8.12.21. Upgrade NAND ECC

This subcommand is used to change the error correcting codes in NAND blocks to 4-bit from 1-bit. Not all hardware supports 4-bit ECC, and even on supported hardware the boot blocks cannot be upgraded. Upgrades are performed on a block-by-block basis, and the commanded page addresses will be expanded to include a whole number of blocks.

This subcommand addresses physical blocks, and is unaffected by reload or backup tables.

8.12.21.1. Command Format

Byte 0	Value 20, indicating Upgrade NAND ECC subcommand
Bytes 1 – 4	32-bit start page number

Bytes 5 – 8	32-bit final page number

8.12.21.2. Reply Format

Byte 0	Value 20, indicating Upgrade NAND ECC subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number
Bytes 9 – 12	32-bit number of blocks upgraded
Bytes 13 – 16	32-bit number of blocks untouched because they are already 4-bit ECC
Bytes 17 – 20	32-bit number of blocks untouched because they are fully erased
Bytes 21 – 24	32-bit number of blocks untouched either because they cannot be read (irrecoverable ECC errors) or they cannot be upgraded (ECC hardware not present, or boot blocks)

8.12.22. Downgrade NAND ECC

This subcommand is used to change the error correcting codes in NAND blocks to 1-bit from 4-bit. Not all hardware supports 4-bit ECC. Downgrades are performed on a block-by-block basis, and the commanded page addresses will be expanded to include a whole number of blocks.

This subcommand addresses physical blocks, and is unaffected by reload or backup tables.

8.12.22.1. Command Format

Byte 0	Value 21, indicating Downgrade NAND ECC subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number

8.12.22.2. Reply Format

Byte 0	Value 21, indicating Downgrade NAND ECC subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number
Bytes 9 – 12	32-bit number of blocks downgraded
Bytes 13 – 16	32-bit number of blocks untouched because they are already 1-bit ECC
Bytes 17 – 20	32-bit number of blocks untouched because they are fully erased
Bytes 21 – 24	32-bit number of blocks untouched because they cannot be read (irrecoverable ECC errors)

8.12.23. Inspect NAND ECC

This subcommand is used to inspect a range of NAND memory, reporting on the type of ECC used. It addresses physical blocks, and is unaffected by reload or backup tables.

8.12.23.1. Command Format

Byte 0	Value 22, indicating Inspect NAND ECC subcommand

Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number

8.12.23.2. Reply Format

Byte 0	Value 22, indicating Inspect NAND ECC subcommand
Bytes 1 – 4	32-bit start page number
Bytes 5 – 8	32-bit final page number
Bytes 9 – 12	32-bit number of blocks that read properly only with 1-bit ECC
Bytes 13 – 16	32-bit number of blocks that read properly only with 4-bit ECC
Bytes 17 – 20	32-bit number of blocks that read properly with both 1-bit and 4-bit ECC (This should never happen)
Bytes 21 – 24	32-bit number of blocks that are entirely erased
Bytes 25 – 28	32-bit number of blocks that cannot be read with either ECC

8.13. READ FILE (0x07)

The Read File command returns one or more "files", which are four consecutive bytes of EDAC protected memory from the supervisor processor. The read process is atomic, so that consistent data is always returned. This command is intended for back-compatibility with other NSP devices with 32-bit floating-point file systems. There are three command formats available: short, long and list. These return corresponding short, long and list replies. These can be distinguished by their lengths.

8.13.1. Short Command Format

Byte 0	EDAC address divided by 4 (0 – 255)

8.13.2. Long Command Format

Bytes 0 - 1	EDAC address divided by 4 (0 – 65535)

8.13.3. List Command Format

Bytes 0 - 1	First EDAC address divided by 4 (0 – 65535)
Bytes 2 - 3	Second EDAC address divided by 4 (0 – 65535)
Bytes 4 - N	Addresses for additional files, as desired

8.13.4. Short Reply Format

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

8.13.5. Long Reply Format

Bytes 0 - 1	EDAC address divided by 4 (0 – 65535)
Bytes 2 - 5	EDAC data bytes read from memory

8.13.6. List Reply Format

Bytes 0 - 1	First EDAC address divided by 4 (0 – 65535)
Bytes 2 - 5	First EDAC data bytes read from memory
Bytes 6 - 7	Second EDAC address divided by 4 (0 – 65535)
Bytes 8 - 11	Second EDAC data bytes read from memory
Bytes 9 - N	Addresses and data for additional files requested

8.14. WRITE FILE (0x08)

The Write File command writes one or more files, which are four consecutive bytes, to EDAC memory. The write process is atomic, so that consistent data is always stored. This command is intended for back-compatibility with other NSP devices with 32-bit floating-point file systems. There are three command formats available: short, long and list. These return corresponding short, long and list replies. These can be distinguished by their lengths.

8.14.1. Short Command Format

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

8.14.2. Long Command Format

Bytes 0 – 1	EDAC address divided by 4 (0 – 65535)
Bytes 2 – 5	Data bytes to write to EDAC memory

8.14.3. List Command Format

Bytes 0 - 1	First EDAC address divided by 4 (0 – 65535)
Bytes 2 - 5	First EDAC data bytes to write to EDAC memory
Bytes 6 - 7	Second EDAC address divided by 4 (0 – 65535)
Bytes 8 - 11	Second EDAC data bytes to write to EDAC memory
Bytes 9 - N	Addresses and data for additional files to write

8.14.4. Short Reply Format

Bytes 0	EDAC address divided by 4 (0 – 255)
Bytes 1 - 4	Data bytes that were written to EDAC memory

8.14.5. Long Reply Format

Bytes 0 – 1	EDAC address divided by 4 (0 – 65535)
Bytes 2 – 5	Data bytes that were written to EDAC memory

8.14.6. List Reply Format

Bytes 0 - 1	First EDAC address divided by 4 (0 – 65535)
Bytes 2 - 5	First EDAC data bytes written to memory
Bytes 6 - 7	Second EDAC address divided by 4 (0 – 65535)
Bytes 8 - 11	Second EDAC data bytes written to memory
Bytes 9 - N	Addresses and data for additional files written

8.15. READ EDAC (0x09)

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

8.15.1. 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.

8.15.2. Long Command Format

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

8.15.3. Reply Format

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

8.16. WRITE EDAC (0x0A)

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

8.16.1. Command Format

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

8.16.2. Reply Format

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

8.17. GO (0x0B)

The Go Code command starts (or terminates) a star tracker sequence. The code is a bitfield, with the following contents:

Bit 0 (LSB)	0: Power off functional processor and detector immediately 1: Power on functional processor and detector immediately
Bit 1	0: Load functional processor software from supervisor flash 1: Load functional processor software from functional NAND flash
Bit 2	0: Power off functional processor and detector when functional processor indicates done, or after timeout 1: Do not turn off functional processor and detector when done
Bit 3	0: Do not send a command to the functional processor 1: Load the compact control structure into the functional processor
Bit 4	0: Do not execute a built-in-test 1: Execute a built-in-test.
Bit 5	0: Reboot the functional processor and then load software into it

	1: Keep the functional processor running with whatever software is currently loaded. The timeout timer is reset.
Bits 6 – 7 (MSB)	Reserved for future use. Write as zero.

If bit 1 is set the functional processor will load its bootloader from NAND flash (blocks 0, 1, 2 or 3). It will then immediately attempt to load and execute an Application Module image from block 0x8000, exactly as if it had received an INIT 0x00008000 command.

If bit 1 is clear the functional processor will load its bootloader from the supervisor flash. This will take several seconds, and when complete the star tracker will be in maintenance mode. Do not clear bit 1 when bit 3 or 4 is set.

Bit 5 is only valid when the sequence state is "running" or "completed". If this is not the case, the bit will be considered cleared regardless of its commanded state.

If bit 2 is clear, the functional processor will be powered down after a timeout, completion, or an emergency terminate, whichever comes first.

If bit 2 is set but bit 5 is clear, the functional processor will be powered down by an emergency terminate only.

If bit 2 and bit 5 are set, the functional processor will be powered down if timeout occurs before completion. It will also be powered down by an emergency terminate. If it reaches completion before timeout it will remained powered.

8.17.1. Command Format

Bytes 0	Go code to execute

8.17.2. Reply Format

Bytes 0	Go code that is being executed

8.18. GATHER RESULT (0x0C)

The Gather Result command is used to read a number of separate sections of the result structure from the supervisor memory. All of the data requested must fit into a single reply message.

8.18.1. Command Format

Bytes 0 – N	List of gather command structures

8.18.1.1. Gather Command Structure

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

8.18.2. Result Format

Bytes 0 – N	List of gather result structures

8.18.2.1. Gather Result Structure

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

8.19. READ RESULT (0x0D)

The Read Result command is used to read the result structure from the supervisor memory. Short and long commands are available. If a single reply message would not be sufficient to contain the data requested then several messages will be sent back-to-back. The PF bit of the final message will be '1', and all other messages have a PF bit of '0'.

8.19.1. Short Command Format

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

8.19.2. Long Command Format

Bytes 0 - 1	Result address to start reading
Byte 2 - 3	Number of bytes to read.

8.19.3. Reply Format

Bytes 0 - 1	Result address where reading started
Bytes 2 - N	The data bytes read from result structure

8.20. IMAGE (0x10)

Command 0x10 is interpreted by the functional processor as an IMAGE request. This is allowed only in application mode.

There are a number of different ways in which the image buffers may be manipulated. The first byte of the data field is consulted to determine which subcommand is required.

Table 31: IMAGE Subcommands

Subcommand Index	Function
0	Draw Rectangle
1	Rectangular Poke
2	Draw Ellipse
3	Blank Image Buffer
4	Sequenced Poke
5+	Reserved

8.20.1. Draw Rectangle Subcommand

This subcommand sets a rectangular window in a RAM image buffer to a constant brightness state. The reply is simply an echo of the command. NACK is returned if any of the parameters is out of range.

8.20.1.1. Command Format

Byte 0	Value 0, indicating Draw Rectangle subcommand
Byte 1	Index of image buffer to write to
Bytes 2 – 3	Index of first row of the rectangle
Bytes 4 - 5	Index of first column of the rectangle
Bytes 6 – 7	Number of rows in the rectangle
Bytes 8 – 9	Number of columns in the rectangle
Bytes 10 – 11	Colour value 0
Bytes 12 – 13	Colour value 1
Bytes 14 – 15	Colour value 2
Bytes 16 – 17	Colour value 3
Bytes 18 – 19	Colour value 4
Bytes 20 – 21	Colour value 5
Bytes 22 – 23	Colour value 6
Bytes 24 – 25	Colour value 7

8.20.1.2. Reply Format

Byte 0	Value 0, indicating Draw Rectangle subcommand
Byte 1	Index of image buffer to write to
Bytes 2 – 3	Index of first row of the rectangle
Bytes 4 - 5	Index of first column of the rectangle
Bytes 6 – 7	Number of rows in the rectangle
Bytes 8 – 9	Number of columns in the rectangle
Bytes 10 – 11	Colour value 0
Bytes 12 – 13	Colour value 1
Bytes 14 – 15	Colour value 2
Bytes 16 – 17	Colour value 3
Bytes 18 – 19	Colour value 4
Bytes 20 – 21	Colour value 5
Bytes 22 – 23	Colour value 6
Bytes 24 – 25	Colour value 7

8.20.2. Rectangular Poke Subcommand

This subcommand allows a rectangular window in a RAM image buffer to be filled in with commanded pixel values. The number of pixels written must be a multiple of the number of pixels in a window row. The number of columns in the window is automatically determined from the number of pixels given. The pixel value written is summed with the value already in the image buffer, and is saturated at 4095 as needed.

8.20.2.1. Command Format

Byte 0	Value 1, indicating Rectangular Poke subcommand
Byte 1	Index of image buffer to write to
Bytes 2 – 3	Index of first row of the poke window
Bytes 4 - 5	Index of first column of the poke window
Bytes 6 – 7	Number of pixels in each row
Bytes 8 – 9	First pixel to write
Bytes 10 – 11	Second pixel to write
Bytes 11+	Subsequent pixel data

8.20.2.2. Reply Format

Byte 0	Value 1, indicating Rectangular Poke subcommand
Byte 1	Index of image buffer to write to
Bytes 2 – 3	Index of first row of the poke window
Bytes 4 - 5	Index of first column of the poke window
Bytes 6 – 7	Number of pixels in each row
Bytes 8 – 9	First pixel to write
Bytes 10 – 11	Second pixel to write
Bytes 11+	Subsequent pixel data

8.20.3. Draw Ellipse Subcommand

This subcommand sets an elliptical window in a RAM image buffer to a constant brightness state. The reply is simply an echo of the command. NACK is returned if any of the parameters is out of range.

The ellipse is specified using a rectangular window. It is permitted to be partially clipped by the edges of the imager.

8.20.3.1. Command Format

Byte 0	Value 2, indicating Draw Ellipse subcommand
Byte 1	Index of image buffer to write to
Bytes 2 – 3	Index of first row of the rectangle
Bytes 4 - 5	Index of first column of the rectangle
Bytes 6 – 7	Number of rows in the rectangle
Bytes 8 – 9	Number of columns in the rectangle
Bytes 10 – 11	Colour value 0
Bytes 12 – 13	Colour value 1
Bytes 14 – 15	Colour value 2
Bytes 16 – 17	Colour value 3
Bytes 18 – 19	Colour value 4
Bytes 20 – 21	Colour value 5
Bytes 22 – 23	Colour value 6
Bytes 24 – 25	Colour value 7

8.20.3.2. Reply Format

Byte 0	Value 2, indicating Draw Ellipse subcommand
Byte 1	Index of image buffer to write to
Bytes 2 – 3	Index of first row of the rectangle
Bytes 4 - 5	Index of first column of the rectangle
Bytes 6 – 7	Number of rows in the rectangle
Bytes 8 – 9	Number of columns in the rectangle
Bytes 10 – 11	Colour value 0
Bytes 12 – 13	Colour value 1
Bytes 14 – 15	Colour value 2
Bytes 16 – 17	Colour value 3
Bytes 18 – 19	Colour value 4
Bytes 20 – 21	Colour value 5
Bytes 22 – 23	Colour value 6
Bytes 24 – 25	Colour value 7

8.20.4. Blank Image Buffer Subcommand

This subcommand writes zero over every byte of an image buffer. The entire 16 MB region is blanked, including the unused memory between lines.

8.20.4.1. Command Format

Byte 0	Value 3, indicating Blank Image Buffer subcommand
Byte 1	Index of image buffer to write to

8.20.4.2. Result Format

Byte 0	Value 3, indicating Blank Image Buffer subcommand
Byte 1	Index of image buffer written to

8.20.5. Sequenced Poke Subcommand

This subcommand is equivalent to the Rectangular Poke subcommand. However, it introduces a sequence number. A sequence number is stored in the functional processor. At turn-on it is set to zero. It is also zeroed by reception of any IMAGE command that is not Sequenced Poke, even if that command is later NAKed due to out-of-range parameters.

The command should contain a new sequence number which is one greater than the stored sequence number (looping back to 0 after 255). If this is the case, then the command will be executed. Since the stored sequence number starts at zero, the first command issued should have a sequence number of one.

If a command is received with a sequence number equal to the current sequence number then it will be considered a duplicate of the previously received command. It will not be executed, but will receive an ACK.

A command with any other sequence number will be NAKed.

A special short command can be issued as a “Sequence number enquiry”. The “Duplicate reply” will be issued in response.

8.20.5.1. Normal Command Format

Byte 0	Value 4, indicating Sequenced Poke subcommand
Byte 1	New sequence number
Byte 2	Index of image buffer to write to
Bytes 3 – 4	Index of first row of the poke window
Bytes 5 - 6	Index of first column of the poke window
Bytes 7 – 8	Number of pixels in each row
Bytes 9 – 10	First pixel to write
Bytes 11 – 12	Second pixel to write
Bytes 12+	Subsequent pixel data

8.20.5.2. Sequence Number Enquiry Command Format

Byte 0	Value 4, indicating Sequenced Poke subcommand

8.20.5.3. Executed Reply Format

Byte 0	Value 4, indicating Sequenced Poke subcommand
Byte 1	Current stored sequence number
Byte 2	Index of image buffer to write to
Bytes 3 – 4	Index of first row of the poke window
Bytes 5 - 6	Index of first column of the poke window
Bytes 7 – 8	Number of pixels in each row
Bytes 9 – 10	First pixel to write
Bytes 11 – 12	Second pixel to write
Bytes 12+	Subsequent pixel data

8.20.5.1. Duplicate Reply Format

Byte 0	Value 4, indicating Sequenced Poke subcommand
Byte 1	Current stored sequence number

8.20.5.1. Failed Reply Format

Byte 0	Value 4, indicating Sequenced Poke subcommand
Byte 1	Current stored sequence number
Bytes 2 – N	Remainder of command, echoed back

8.21. COMBINATION (0x12)

The combination command may be used by resource constrained host spacecraft to start the star tracker (equivalent to sending a GO command) and then receive result data (equivalent to polling with READ RESULT commands).

Up to half a second may elapse between the command and its replies in normal mode. If a self-test is commanded, tens of seconds may elapse. Be extremely careful using this command if the star tracker shares the NSP bus with other devices.

Table 32: Combination Bitfield

Bit	Result Offset	Data Length	Name
0	0x0000	0x0004	Sequence Number
1	0x0004	0x0004	Return Code
2	0x0008	0x0020	Attitude Quaternion
3	0x0028	0x0018	Angular Velocity
4	0x0040	0x0008	Epoch time
5	0x0048	0x0038	Hardware Telemetry
6	0x0080	0x00B0	Statistics Telemetry
7	0x0130	0x0310	Image Telemetry
8	0x0440	0x0068	ERS Telemetry
9	0x04A8	0x0340	Centroid Telemetry
10	0x07E8	0x0160	Match Telemetry
11	0x0000	0x02CC	Built-in Test Result

Each bit in the bitfield causes a particular piece of data to be returned. Table 32 shows the offset and length into the result structure referenced by each bit.

To run a built-in test, send a Go code consistent with a test, and a bitmap of 0x000800. For normal operation, use a bitmap of 0x0007FF or less.

8.21.1. Command Format

Bytes 0	Go code to execute
Byte 1 - 3	Bitmap indicating desired telemetry return

8.21.2. Successful Reply Format

Bytes 0 - 1	Number of bytes already transmitted
Bytes 2 - N	Result telemetry data

For a success, the ACK bit will be set to ‘1’. The number of reply messages generated depends on the amount of data requested. The final message will have the PF bit set to ‘1’. All other reply messages will have the PF bit set to ‘0’.

8.21.3. Failure Reply Format

Bytes 0	Sequence state
Bytes 1 - N	Error message

If the functional processor stops before generating the full set of data required by the COMBINATION command, a failure reply will be generated. The ACK bit will be set to ‘0’, to distinguish this from a successful reply.

The first byte of the return is the sequence state, as described in Table 37. The subsequent bytes contain the error message. This may be the ASIC ID message, or the result of the most recent warning or terminate message. If the error message would not fit into a single

NSP packet it is truncated. Only one failure reply message will be generated, and the PF bit is set to '1'.

8.22. READ TIME (0x13)

The READ TIME command returns the supervisor's realtime clock, latched at the moment the final FEND character of the command is received. The data is returned as a 56-bit unsigned integer count of microseconds since J2000. The least significant bit of the count will always be zero, so the effective precision is 2 microseconds. The 56-bit count will roll over in the year 4283, which is longer than the feasible lifetime of this software.

By sending several READ TIME commands, separated by minutes or hours, an accurate comparison can be made between the star tracker clock and the spacecraft computer clock. Rev 4 star tracker clocks are not particularly accurate, and may benefit from calibration.

At turn-on the clock is set to zero. It remains steady at zero until initialized by a WRITE TIME command, at which point it begins to tick. Reading a zero result shows that the clock has not yet been initialized.

8.22.1. Command Format

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

8.22.2. Reply Format

Bytes 0 – 6	56-bit unsigned integer, counting microseconds since J2000

8.23. WRITE TIME (0x14)

The WRITE TIME command sets the supervisor's realtime clock. The data is latched at the moment the final FEND character of the command is received, allowing the star tracker clock to be accurately synchronized with the spacecraft clock.

Writing a time of exactly zero indicates that the star tracker should consider its realtime clock invalid. In this state the clock will hold at exactly zero until a subsequent WRITE TIME command is received.

8.23.1. Command Format

Bytes 0 – 6	56-bit unsigned integer, counting microseconds since J2000

8.23.2. Reply Format

Bytes 0 – 6	56-bit unsigned integer, counting microseconds since J2000

8.24. WRITE KEPS (0x15)

The WRITE KEPS command allows the user to upload a set of osculating Keplerian elements that describe the host spacecraft's orbit about the Earth.

There are various ways to describe Keplerian elements. The set here is chosen for reasons of computation efficiency, and because they can be easily obtained from TLEs.

The time of perigee passage must be in the past (i.e. less than the current time) and must occur after J2000. It should be recent (within the past few days) or numerical approximations will result in poor accuracy.

8.24.1. Command Format

Bytes 0 – 3	32-bit IEEE 754 float, Orbit eccentricity. [0.0 <= e < 1.0]
Bytes 4 – 7	32-bit IEEE 754 float, Orbit inclination, rads
Bytes 8 – 11	32-bit IEEE 754 float, Right ascension of the ascending node, rads
Bytes 12 – 15	32-bit IEEE 754 float, Argument of perigee, rads
Bytes 16 – 19	32-bit IEEE 754 float, Mean motion, rads/sec
Bytes 20 – 26	56-bit unsigned integer, Time of perigee passage, microseconds since J2000

8.24.2. Reply Format

Bytes 0 – 3	32-bit IEEE 754 float, Orbit eccentricity. [0.0 <= e < 1.0]
Bytes 4 – 7	32-bit IEEE 754 float, Orbit inclination, rads
Bytes 8 – 11	32-bit IEEE 754 float, Right ascension of the ascending node, rads
Bytes 12 – 15	32-bit IEEE 754 float, Argument of perigee, rads
Bytes 16 – 19	32-bit IEEE 754 float, Mean motion, rads/sec
Bytes 20 – 26	56-bit unsigned integer, Time of perigee passage, microseconds since J2000

9. Protocol Layer 6 (Presentation Layer)

9.1. Supervisor Mapping

9.1.1. Memory Map

Table 33: Supervisor Memory Map

Address Range	Function
0x00000000 – 0x000015FF	Bootloader program memory
0x00001600 – 0x00002FFF	Supervisor program memory (flash)
0x00003000 – 0x00003047	Default control structure (flash)
0x00003048 – 0x00007DFF	Supervisor program memory (flash)
0x00007E00 – 0x00007FFF	Stored parameters (flash)
0x00008000 – 0x0000FFFF	Unused (flash)
0x00010000 – 0x0001F9FF	Functional processor bootloader image (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
0x04000900 – 0x0400090F	Power supply SMBus mapped registers
0x04004A00 – 0x04004A03	Detector temperature sensor SMBus mapped registers
0x04005D00 – 0x04005DFF	Active pixel detector SMBus mapped registers

The supervisor 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 5.5 kB of program memory contain the bootloader. These are protected against POKEs so that the bootloader cannot be accidentally changed. The next 58.5 kB contains the supervisor 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.

Starting at address 0x00010000 is a 56 kB bootloader image for the functional processor. The first four bytes indicate the length of the program, and the first actual program byte lives at address 0x00001004. When a GO command is received to boot the functional processor from supervisor flash, the byte from 0x00001004 on the supervisor is loaded into byte 0x40200000 of the functional processor. Successive bytes from the supervisor are loaded into successive locations in the functional processor. When all of the bytes (indicated by the length field at the start) have been loaded the functional processor begins execution at address 0x40200000.

The supervisor 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.

The SMBus mapped registers are similarly only to be used by those who know what they are doing. There is no interlock preventing the supervisor from trying to access the SMBus at the same time as the functional processor, potentially leading to corruption for both units.

9.1.2. Diagnostics

Table 34: Diagnostic Channels

Diagnostic Channel	Function
0x00	Reset Reason
0x01	Reset Count
0x02	Internal Framing Error Count
0x03	Internal Runt Packet Count
0x04	Internal Oversize Packet Count
0x05	Internal Bad CRC Count
0x06	Internal FIFO Overflow Count
0x07	External Framing Error Count
0x08	External Runt Packet Count
0x09	External Oversize Packet Count

0x0A	External Bad CRC Count
0x0B	External FIFO Overflow Count

Each diagnostic channel is presented as a 32-bit unsigned integer. The internal storage for many of these is only 16 bits, so overflows may occur after 64k counts.

Internal errors represent bad NSP events on the communications link between the supervisor and functional processors. External errors represent bad NSP events on the communication link between the supervisor and the host spacecraft. Reset will clear all of the error counters.

9.1.2.1. Reset Reason

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

Table 35: Reset Reason Codes

Reset Reason Code	Meaning
0	Power cycle. The star tracker has either been freshly turned on, or the input voltage has dropped below approximately 2 V.
1	Flash error. An illegal attempt has been made to read or write flash memory.
2	Overcurrent. The star tracker input current has momentarily exceeded the 750 mA maximum threshold, and the supervisor processor has been reset in an attempt to clear the fault.
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 internal oscillator has failed momentarily. Obviously if you are reading this code then the oscillator must have restarted.
5	Pin reset. The external /Reset signal has been pulled low.
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 supervisor software encounters an irrecoverable fault, such as a spurious interrupt.

9.1.2.2. Reset Count

The reset count contains the number of supervisor 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.

9.1.2.3. Framing Error Count

A framing error is declared if an NSP message is incorrectly encapsulated on the communications link. For ASYNC and 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 TBD.

9.1.2.4. Runt Packet Count

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 star tracker's address, 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 ASYNC or RS485 link two FEND characters back-to-back is a valid bus condition and not a runt.

9.1.2.5. Oversize Packet Count

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 7.2 for the length constraints.

9.1.2.6. Bad CRC Count

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 star tracker.

9.1.2.7. FIFO Overflow Count

The bootloader does not use FIFO buffers, and will never increment this counter.

The application program uses a mix of hardware and software FIFO buffers on its serial inputs. If a FIFO overflows and loses data then this counter will increment. Due to the constraints of the hardware it is not guaranteed that all overflow events will be noticed.

9.1.3. EDAC Memory

The supervisor processor supports 512 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 STORE command will save EDAC memory into non-volatile flash memory.

Table 36: Supervisor EDAC Memory Map

EDAC Address	File Address	Function	Format
0x00 – 0x01	N/A	Flash table CRC	16-bit unsigned integer
0x02	N/A	EDAC load source	8-bit unsigned integer
0x03	N/A	Reserved	
0x04 – 0x07	0x01	Asynchronous current sense telemetry	32-bit float, Amps
0x08 – 0x0B	0x02	Asynchronous bus voltage telemetry	32-bit float, volts
0x0C – 0x0F	0x03	Asynchronous Vdd Core telemetry	32-bit float, volts

0x10 – 0x13	0x04	Asynchronous Vdd MPU telemetry	32-bit float, volts
0x14 – 0x17	0x05	Asynchronous Vdd IO telemetry	32-bit float, volts
0x18 – 0x1B	0x06	Asynchronous supervisor temperature telemetry	32-bit float, °C
0x1C – 0x1F	0x07	Asynchronous Vdd supervisor telemetry	32-bit float, volts
0x20 – 0x23	0x08	Asynchronous ADC ground telemetry (Rev4) Asynchronous Vdd detector telemetry (Rev5)	32-bit float, volts
0x24 – 0x27	0x09	Synchronous current sense telemetry	32-bit float, Amps
0x28 – 0x2B	0x0A	Synchronous bus voltage telemetry	32-bit float, volts
0x2C – 0x2F	0x0B	Synchronous Vdd Core telemetry	32-bit float, volts
0x30 – 0x33	0x0C	Synchronous Vdd MPU telemetry	32-bit float, volts
0x34 – 0x37	0x0D	Synchronous Vdd IO telemetry	32-bit float, volts
0x38 – 0x3B	0x0E	Synchronous supervisor temperature telemetry	32-bit float, °C
0x3C – 0x3F	0x0F	Synchronous Vdd supervisor telemetry	32-bit float, volts
0x40 – 0x43	0x10	Synchronous ADC ground telemetry (Rev4) Synchronous Vdd detector telemetry (Rev5)	32-bit float, volts
0x44 – 0x47	0x11	SEU count	32-bit unsigned integer
0x48 – 0x4B	0x12	SEU scrub index	32-bit unsigned integer
0x4C – 0x4F	0x13	Result structure length	32-bit signed integer
0x50 – 0x53	0x14	Control structure length	32-bit unsigned integer
0x54 – 0x57	0x15	Timeout period	32-bit float, seconds
0x58 – 0x5B	0x16	Sample point	32-bit float, seconds
0x5C	N/A	Sequence state	8-bit enum
0x5D	N/A	Functional processor message length	8-bit unsigned integer
0x5E – 0x97	N/A	Functional processor message	Sequence of 8-bit characters. May be human-readable.
0x98 – 0x18F	0x26-0x63	Control structure	
0x190 – 0x193	0x64	Uptime (low precision, for convenience)	32-bit float, Julian days since supervisor boot
0x194	N/A	Unused	
0x195 – 0x19B	N/A	Time offset	56-bit unsigned integer, recording offset between realtime clock and uptime clock in microseconds
0x19C	N/A	Vdd IO tune	8-bit unsigned integer
0x19D	N/A	Vdd detector tune	8-bit unsigned integer
0x19E	N/A	Detector config	8-bit bitfield
0x19F	N/A	Unused	
0x1A0 – 0x1A3	0x68	Thermistor temperature (in star trackers built after Jan 2021, thermistor is not populated. Value will always read 0xffffffff).	32-bit float, °C

0x1A4 – 0x1AA	N/A	Time of last GO command	56-bit unsigned integer, counting in units of microseconds of uptime
0x1AB	N/A	Previous epoch override	8-bit Boolean. Non-zero values prevent automatic write of control structure epoch of previous return.
0x1AC – 0x1DB	0x6B – 0x76	Reserved area written by WRITE KEPS	Reserved
0x1DC – 0x1E7	0x77 – 0x79	Satellite velocity about the Earth	Array of three 32-bit floats, meters/second
0x1E8 – 0x1F3	0x7A- 0x7C	Earth velocity about the sun	Array of three 32-bit floats, meters/second
0x1F4 – 0x1F7	0x7D	Time (low precision, for convenience)	32-bit float, Julian days since J2000.0
0x1FA	N/A	Ephemeris control	8-bit bitmap
0x1FB - 1FF	N/A	Realtime clock	40-bit unsigned integer, counting in units of 0.065536 sec, since J2000.0

9.1.3.1. EDAC Load Source

Upon entry to the supervisor application program, the supervisor flash memory is checked for a valid EDAC table. If such a table is found it will be loaded. The EDAC Load Source byte will be set to '1'.

Otherwise, the EDAC Load Source byte will be set to '0', and factory default values will be loaded into the EDAC table.

The STORE command can be used to create or delete the flash memory table.

9.1.3.2. Asynchronous Analog Telemetry

Asynchronous telemetry is continually recorded using the ADC in the supervisor processor. Reads to these files show the most recently collected values. The current and voltage telemetry points are quite accurate. The supervisor temperature telemetry point is based on a junction integrated with the silicon die and is not well calibrated. It can be used for a general indication of hot or cold, but should not be relied on as an absolute measure of temperature.

9.1.3.3. Synchronous Analog Telemetry

Synchronous telemetry is recorded in a single snapshot at a time after the functional processor startup as defined by the Sample point file. If the star tracker is in regular cyclic

operation, this can be used as an apples-to-apples comparison to look at trends in voltages, currents and temperatures.

9.1.3.4. SEU Count

Each byte of the EDAC memory is stored in three SRAM cells, and triple-voting is used whenever one is retrieved. As a background task the EDAC memory is scrubbed for errors. Every time an error is found and fixed the SEU Count is incremented.

9.1.3.5. SEU Scrub Index

The SEU Scrub Index contains the pointer to the current location to be scrubbed. This counter is frequently incremented until it reaches the end of the EDAC space and loops back to zero. It is of little interest to the user.

9.1.3.6. Result Structure Length

The functional processor sends telemetry results to the supervisor processor. The result structure return is not atomic, so the result structure length file will slowly grow as individual result packets are received by the supervisor processor. The result structure length file will be zeroed immediately after a GO command, deleting the results from the previous cycle and making room for a fresh result structure.

The result structure length file will be set to -1 if out-of-order result packets are received from the functional processor.

9.1.3.7. Control Structure Length

A control structure is held in EDAC memory, and is sent to the functional processor on each GO command. The Control structure length file determines how many bytes are sent to the functional processor. It is critically important that this number match the number of bytes that the functional processor is expecting. The functional processor will not operate if there is a mismatch.

9.1.3.8. Timeout Period

The timeout period determines how long the functional processor should be allowed to run before it is turned off. This is ignored if the GO command stated that the functional processor should be allowed to run indefinitely. If the built-in-test is run then the timeout period file is not used, and a value of 30 seconds is specified instead.

9.1.3.9. Sample Point

The sample point determines when the synchronous telemetry snapshot should be taken. It is measured in seconds after the moment that the /Reset signal on the functional processor is de-asserted. Note that if the sample point value is too long (i.e. longer than the timeout period) the synchronous telemetry may not be sampled.

9.1.3.10. Sequence State

The sequence state shows the power state of the functional processor.

Table 37: Sequence States

Sequence State	Meaning
0x00	The main power switch has been turned ON
0x01	The DC/DC converter running the 1.8 V I/O rail has been started
0x02	The LDO running the 2.8 V detector power supply has been started
0x03	The DC/DC converter running the core Vdd rail has been started
0x04	The DC/DC converter running the MPU Vdd rail has been started
0x05	The functional processor's /Reset line has been released
0x06	The functional processor's ASIC ID message is being received
0x07	The supervisor is commanding the functional processor's boot mode
0x08	The supervisor is loading the functional processor with a program from its flash memory
0x09	The functional processor is booting
0x0A	The functional processor has sent a message indicating that its software is running
0x0B	The functional processor has been turned off by GO command. This is also the default state upon starting the supervisor processor.
0x0C	The functional processor has been turned off after it has reported successful completion.
0x0D	The functional processor has been turned off following a timeout waiting for the ASIC ID message.
0x0E	The functional processor has been turned off following a timeout while commanding the boot mode.
0x0F	The functional processor has been turned off following a timeout while sending the program.
0x10	The functional processor has been turned off following a timeout while waiting for it to indicate that its software is running.
0x11	The functional processor has been turned off following a timeout while running normally.
0x12	The functional processor has been turned off because the input voltage to the star tracker exceeded the safe limit (Rev 4). OR The functional processor has been turned off because the supervisor failed in an attempt to write on the SMBus (Rev 5).
0x13	The functional processor has been turned off because it emitted an emergency terminate message.
0x14	The supervisor processor Vdd regulator has entered low-voltage dropout (at about 2.3 V). The sequencer has been reset so that excessive currents are not drawn at low supply voltages.
0x15	The supervisor processor has detected a parity error in the boot message received from the functional processor. The functional processor has been turned off.

0x16	The supervisor processor UART which connects to the functional processor has experienced a FIFO overflow during the boot process. The functional processor has been turned off. Note that overflows subsequent to the boot process are handled by the diagnostic counters instead.
0x17	The supervisor processor has detected serial data from the functional processor when it was not expected in the boot process. The functional processor has been turned off.
0x18	The supervisor processor has detected a serial transmit interrupt when it did not think it was trying to transmit. The functional processor has been turned off in the ensuing confusion.

9.1.3.11. Functional Processor Message

The functional processor can send notification messages to be stored in the EDAC memory. Only one message can be stored at a time, and a newer message replaces an older one. The length of the currently stored message is stored in EDAC, followed by the message itself.

The Rev 4 star tracker captures the functional processor ASIC ID sequence, which is a 58 byte structure identifying the chip. This will be used as the first functional processor message. The Rev 5 hardware lacks the ability to read the ASIC ID.

At certain points in the functional program it may send debugging notifications. These are human-readable ASCII strings. They are not NULL terminated, since the message length is stored separately. If the functional processor encounters a serious error it will send an emergency terminate message. This message will be stored in EDAC, and receipt of it will also cause the supervisor to immediately power down the functional processor. Emergency terminate messages are human-readable ASCII.

9.1.3.12. Vdd IO Tune

By setting this byte to a non-zero value, the voltage on the Vdd IO rail can be reduced from its +1.8 V nominal value. Each step reduces it by 3%. Values of 0 and 1 keep all components within specified ranges. Anecdotal testing suggests that the star tracker may keep working with values of up to 7. 15 is the largest acceptable value.

Increasing this value reduces the power consumption and heat generation of the star tracker. It does not appear to have a significant effect on detector noise levels.

This functionality is only available on the Rev 5 hardware.

9.1.3.13. Vdd Detector Tune

By setting this byte to a non-zero value, the voltage on the Vdd detector rail can be reduced from its +2.8 V nominal value. Each step reduces it by 3%. Values of 0 to 2 keep all components within specified ranges. Anecdotal testing suggests that the star tracker may keep working with values of up to 15. 15 is the largest acceptable value.

Increasing this value slightly reduces the power consumption and heat generation of the star tracker. It does not appear to have a significant effect on detector noise levels.

This functionality is only available on the Rev 5 hardware.

Analog offsets should be recomputed when this setting is changed.

9.1.3.14. Detector Config

By setting this byte to a non-zero value, the detector can be configured in non-default ways. This functionality is only available on the Rev 5 hardware.

Bit	Meaning
0 (LSB)	Pixel clock slew rate. 0 = Fast (default) 1 = Slow
1	Pixel data slew rate. 0 = Fast (default) 1 = Slow
2	Pixel clock rate 0 = 86 MHz (default) 1 = 84 MHz (do not use!)
3	ADC gain 0 = x15.75 (default) 1 = x8 (Doubles dynamic range, slight increase in noise)

9.1.3.15. Thermistor Temperature

If the star tracker is fitted with a chassis thermistor, this telemetry gives a calibrated measurement in °C. The measurement is asynchronous to the GO cycle.

9.1.3.16. Time of Last GO Command

This file stores the time of the last GO command, assuming the clock has been correctly set. Only relatively low precision data is available in EDAC memory. An internal higher precision structure is used for epoch feedback.

9.1.3.17. Previous Epoch Override

When this byte is zero the previous epoch field in the control structure will be automatically updated to tell the functional processor the time of the previous solution relative to its own clock. If it is non-zero then this update will not occur, and the user can force the previous epoch field to any value. This is useful in HITL testing. It should be zero for flight operations.

9.1.3.18. Satellite Velocity about the Earth

These three files contain the velocity of the satellite with respect to the Earth, in the ECI frame. Depending on the ephemeris control byte they may be set by the user or automatic. These files can be read to test the function of the ephemeris.

9.1.3.19. Earth Velocity about the Sun

These three files contain the velocity of the Earth with respect to the sun, in the ECI frame. Depending on the ephemeris control byte they may be set by the user or automatic. These files can be read to test the function of the ephemeris.

9.1.3.20. Time

This file contains the time as a 32-bit floating point value. Its precision is not very good – by 2014 it has a granularity approaching 30 seconds. It is provided for convenience only.

9.1.3.21. Ephemeris Control

The star tracker has the ability to correct star positions for stellar aberration. It does this through the spacecraft velocity field of the control structure. The data in the velocity field is controlled by the Ephemeris control in EDAC memory.

Table 38: Ephemeris Control

Bits	Function
0 – 1 (LSB)	0: Earth ephemeris disabled. Earth velocity files can be set by user. 1: Earth ephemeris runs. Earth velocity files automatically updated. If time is zero, Earth velocity files are held at zero. 2+: Earth ephemeris disabled. Earth velocity files are held at zero.
2 – 3	0: Satellite ephemeris disabled. Satellite velocity files can be set by user. 1: Satellite ephemeris runs. Satellite velocity files automatically updated. If time is zero, satellite velocity files are held at zero. 2+: Satellite ephemeris disabled. Satellite velocity files are held at zero.
4 – 6	0: Control spacecraft velocity fields can be set by user. 1: Control spacecraft velocity fields are set by Earth velocity files. 2: Control spacecraft velocity fields are set by Satellite velocity files. 3: Control spacecraft velocity fields are set by sum of Earth velocity files and Satellite velocity files. 4+: Control spacecraft velocity fields are held at zero
7 (MSB)	Reserved

9.1.4. Result Structure

Received result data from the functional processor is stored in the result structure. It can be read using the READ RESULT command. Before reading it is a good idea to read the result structure length from EDAC memory. READ RESULT will return NACK if an attempt is made to read beyond the valid areas of the result structure.

The format of the result depends on whether the star tracker is being used operationally, or in self-test. In both cases, the structure is made up of sub-structures

9.1.4.1. Operational Result

Table 39: Operational Result Structure

Result address	Type	Notes

0x0000	Unsigned 32-bit integer	Sequence number
0x0004	Unsigned 32-bit integer	Return code
0x0008	Array of 4 64-bit IEEE floating-point values	Inertial attitude quaternion. Scalar component first.
0x0028	Array of 3 64-bit IEEE floating-point values	Angular velocity of sensor wrt ECI in sensor frame
0x0040	64-bit IEEE floating-point value	Epoch time
0x0048	Hardware Telemetry	
0x0080	Statistics Telemetry	
0x012C	Unsigned 32-bit integer	Reserved
0x0130	Array of 2 Image Telemetry structures	
0x0440	ERS Telemetry	
0x04A8	Array of 2 Centroid Telemetry structures	
0x07E8	Array of 2 Matching Telemetry structures	
0x0948	Reserved	

The entire operational result structure is 0x0A38 bytes long.

9.1.4.2. Self-Test Result

Table 40: Self-Test Result Structure

Offset	Type	Notes
0x0000	Analog Frame	From step 1 in 8.2
0x0020	Analog Frame	From step 2 in 8.2
0x0040	Analog Frame	From step 3 in 8.2
0x0060	Analog Frame	From step 4 in 8.2
0x0080	Analog Frame	From step 5 in 8.2
0x00A0	Analog Frame	From step 6 in 8.2
0x00C0	Analog Frame	From step 7 in 8.2
0x00E0	Analog Frame	From step 8 in 8.2
0x0100	Analog Frame	From step 9 in 8.2
0x0120	Analog Frame	From step 10 in 8.2
0x0140	Analog Frame	From step 11 in 8.2
0x0160	Signed 32-bit int	0 if first test pattern image matches expected value, negative otherwise
0x0164	Signed 32-bit int	0 if second test pattern image matches first image, negative otherwise
0x0168	Analog Frame	From step 12 in 8.2
0x0188	Hardware Telemetry	
0x01C0	Analog Frame	From step 13 in 8.2
0x01E0	Statistics Telemetry	
0x028C	Analog Frame	From step 14 in 8.2
0x02AC	Analog Frame	From step 15 in 8.2

The complete self-test structure is 0x02CC bytes long. It is possible that the self-test will fail somewhere along the sequence. For example, if the detector is not working the sequence may stall waiting for an image. In this case the structure will be shorter, and the location of the stall may be determined by the missing data. The Combination command will not deal gracefully with a test failure.

9.1.4.3. Return Code

The return code is a bitfield that allows the success of the operation to be determined at a glance. New in this revision is a revised set of return codes that clarify interpretation of the sensor status for the end user. The original codes are still set, but are labeled 'Legacy' codes.

Table 41: Return Code

Bit	Meaning	Status
Bit 0	Image 1 output quality	Legacy
Bit 1	Image 2 output quality	Legacy
Bit 2	Image 1 processing success	Legacy
Bit 3	Image 2 processing success	Legacy
Bit 4	Full processing	Legacy
Bit 5	Detector image	Legacy
Bit 6	Consistent image solutions	Legacy
Bit 7	Reserved	
Bit 8	Master Return	Master Return
Bits 9-10	Image 1 Status	Advanced
Bits 11-12	Image 2 Status	Advanced
Bit 13	Rate Source	Advanced
Bit 14	Solution 1 consistent with previous	Advanced
Bit 15	Solution 2 consistent with previous	Advanced

Master Return

The Master Return bit is an overall indicator of the sensor's confidence in its result. Generally, when this bit is set, the sensor is confident in its return. If this bit is not set, the quaternion and omega fields in the primary telemetry will be zeroed and should not be used.

Advanced Status Return

When the Master Return bit is not set, users may find useful attitude information by consulting the extended telemetry. However, care must be taken to assess the reasonableness of any such estimates. Interpreting these results relies on understanding the sensor's evaluation of each image.

Each image's result is assigned a quality label of BAD (00), MARGINAL (01), or GOOD (10). A good result has at least four matched stars, or a three matched star solution that is consistent between images. For the Master Return bit to be set, the sensor requires at least one GOOD image, as well as error-free execution.

During ground-testing we have observed a rare condition where atmospheric scintillation causes a failure of the rolling shutter processing. Thus, one of both images may be marked as good, yet the Master Return is not set. This condition should be very rare on-orbit.

In the case of a marginal processing result, the image-specific quaternions found in the extended telemetry may be useful if external attitude estimates are available for cross-checking. When even degree-accurate estimates are available, it is usually fairly easy to assess the quality of marginal returns. When in doubt, we recommend that marginal returns be treated as 'bad'.

Legacy Return

This section documents the original set of status messages. In general a '1' result is good, while a '0' result should be cause for caution before using the attitude and rate data.

- Bits 4 and 5 are mainly diagnostic. These indicate that images were taken by the detector, and the processing completed without aborting.
- Bits 2 and 3 indicate that a match was found for at least three stars in the corresponding image. Attitude estimates are available.
- Bits 0 and 1 indicate greater confidence in the returned attitude solution. When the success bit is '1', but this bit is '0', the algorithm was unable to match several stars detected in the image. This bit will never be '1' if the corresponding success bit is '0'. As of Rev. 1.13, all returns require a minimum of four matched stars for the quality bits to be set.
- Bit-6 indicates that the attitude solutions for the two images agree to within 0.6 degrees (this limit based on maximum supported satellite rates). This allows greater confidence in the return.

Rate Source is set if the output rate is based on the difference between the quaternion this frame, and last frame's quaternion. Rate source is clear if the output rate is based on the difference between the two images in this frame.

9.1.4.4. Hardware Telemetry Structure

Table 42: Hardware Telemetry Structure

Offset	Type	Notes
0x0000	Unsigned 16-bit integer	Number of fixable flash errors
0x0002	Unsigned 16-bit integer	Number of unfixable flash errors
0x0004	Reserved	
0x000C	Signed 16-bit integer	Detector Temperature
0x000E	Unsigned 16-bit integer	Functional Processor Temperature
0x0010	Unsigned 32-bit integer	Status bitfield
0x0014	Unsigned 8-bit integer	Vdd Core set point
0x0015	Unsigned 8-bit integer	Vdd MPU set point

0x0016	Reserved	
0x0018	Array of 16 unsigned 16-bit integers	Dark offsets

The flash error counts show the number of fixable and unfixable page errors seen since the start of the functional application program. Note that this does not include any errors seen while loading the application program from flash.

The detector temperature is a 16-bit signed quantity, with the four least significant bits always reading zero. It represents the temperature of the detector, in °C, multiplied by 16. For example, a return of 0x5000 indicates a detector temperature of 80 °C. A return of 0xE700 indicates a detector temperature of -25 °C. The detector temperature sensor is the most accurate of the various sensors in the device.

The functional processor temperature sensor indicates the processor's die temperature. A temperature of 25 °C will give a return of 0x0032, and the slope is approximately 1.5 C° / ADU. This sensor is not particularly accurate.

The status bitfield contains several flags related to the health of the unit.

Table 43: Status Bitfield

Bit	Meaning
Bit 19	POP memory overtemperature, set at approximately 85 °C
Bit 31	Thermal shutdown, set at approximately 160 °C
All other bits	Reserved

The bit 31 thermal shutdown has not been tested. It is quite possible that the other thermal shutdown mechanisms in the unit will trip first and render the star tracker uncommunicative.

The Vdd set points are automatically adjusted by the functional processor's Smart Reflex peripheral to provide the needed voltages as a function of device temperature and age. For revision 4 hardware, the two most significant bits will always be set and the six least-significant bits can be decoded as 6-bit integers commanding the respective DC/DC converters. For revision 5 hardware the values are in the range 0-15.

The first eight dark offsets correspond to the first image buffer, and the second eight to the second image buffer. These are the average brightness, across the eight colour channels, of the dark columns.

9.1.4.5. Statistics Telemetry Structure

Table 44: Complete Statistics Telemetry Structure

Offset	Type	Notes
0x0000	Unsigned 16-bit integer	Good rows in image 1
0x0002	Unsigned 16-bit integer	Good rows in image 2
0x0004	Array of 8 32-bit IEEE floating-point	Mean value in image 1
0x0024	Array of 8 32-bit IEEE floating-point	Mean value in image 2
0x0044	Array of 8 32-bit IEEE floating-point	Spatial variation in image 1
0x0064	Array of 8 32-bit IEEE floating-point	Spatial variation in image 2

0x0084	Array of 8 32-bit IEEE floating-point	Temporal variation between images
0x00A4	Unsigned 32-bit integer	Reserved
0x00A8	Unsigned 32-bit integer	Reserved

The complete statistics telemetry is only valid if the functional processor has been specifically commanded to gather statistics.

Table 45: Partial Statistics Telemetry Structure

Offset	Type	Notes
0x0000	32-bit zero	Reserved
0x0004	32-bit IEEE floating-point	Integrated brightness of all pixels in ROI of image 1
0x0008	32-bit IEEE floating-point	Integrated brightness of all pixels in ROI of image 2

The partial statistics telemetry is returned if the functional processor has not been specifically commanded to gather statistics. It returns the total integrated brightness in both images, which can be used as a quick check to determine if the sensor is pointed close to the sun or Earth.

9.1.4.6. Image Telemetry Structure

The image telemetry structure contains information on the location of bright spots on the image. There are two image structures, one for each of the two images.

Table 46: Image Telemetry Structure

Offset	Type	Notes
0x0000	Unsigned 32-bit integer	Initialization flag
0x0004	Signed 32-bit integer	Image return code
0x0008	Signed 32-bit integer	Number of lit pixels
0x000C	Unsigned 32-bit integer	Number of peaks
0x0010	64-bit IEEE floating-point	Exposure error
0x0018	64-bit IEEE floating-point	Image capture time relative to functional processor start
0x0020	Array of 30 peak-entry structures	List of peaks

Each image structure contains 30 peak-entry structures. Their format is shown below.

Table 47: Peak Entry Structure

Offset	Type	Notes
0x0000	Array of 2 signed 32-bit integers	Peak position (row, column)
0x0008	Signed 32-bit integer	Peak intensity

9.1.4.7. ERS Telemetry Structure

The ERS telemetry structure contains information on the angular velocity estimation and electronic-rolling-shutter distortion compensation.

Table 48: ERS Telemetry Structure

Offset	Type	Notes
0x0000	Unsigned 32-bit integer	Initialization flag
0x0004	Signed 32-bit integer	ERS return code
0x0008	Array of 10 32-bit signed integers	Star mapping
0x0030	64-bit IEEE floating-point	Fit residual
0x0038	Array of 6 64-bit IEEE floating-points	Upper triangle of angular velocity covariance

9.1.4.8. Centroid Telemetry Structure

The centroid telemetry structure contains information on the centroids of stars in the image. There are two centroid structures, one for each of the two images.

Table 49: Centroid Telemetry Structure

Offset	Type	Notes
0x0000	Unsigned 32-bit integer	Initialization flag
0x0004	Signed 32-bit integer	Centroid return code
0x0008	Signed 32-bit integer	Number of good stars
0x000C	Unsigned 32-bit integer	Reserved
0x0010	Array of 10x2 64-bit IEEE floating-points	Best centroids (row, column)
0x00B0	Array of 10x3 64-bit IEEE floating-points	Best star vectors

9.1.4.9. Matching Telemetry Structure

The matching telemetry structure contains information on the mapping between stars in the image and stars in the catalog. There are two matching structures, one for each of the two images.

Table 50: Matching Telemetry Structure

Offset	Type	Notes
0x0000	Unsigned 32-bit integer	Initialization flag
0x0004	Signed 32-bit integer	Matching return code
0x0008	Unsigned 32-bit integer	Number of triangle tests
0x000C	Unsigned 32-bit integer	Reserved
0x0010	64-bit IEEE floating-point	Matching error
0x0018	64-bit IEEE floating-point	Confidence index
0x0020	Signed 32-bit integer	Total star tests
0x0024	Signed 32-bit integer	Total set tests
0x0028	Signed 32-bit integer	Conset size
0x002C	Unsigned 32-bit integer	Reserved
0x0030	64-bit IEEE floating point	Conset RMS error
0x0038	Array of 10 Signed 32-bit integers	Matched stars
0x0060	Array of 4 64-bit IEEE floating-point values	Inertial attitude quaternion. Scalar component first.

0x0080	Array of 6 64-bit IEEE floating-point values	Upper triangle of attitude covariance

9.1.4.10. Analog Frame Structure

Table 51: Analog Frame Structure

Offset	Type	Notes
0x0000	32-bit IEEE floating-point, Amps	Synchronous current sense telemetry
0x0004	32-bit IEEE floating-point, Volts	Synchronous bus voltage telemetry
0x0008	32-bit IEEE floating-point, Volts	Synchronous Vdd Core telemetry
0x000C	32-bit IEEE floating-point, Volts	Synchronous Vdd MPU telemetry
0x0010	32-bit IEEE floating-point, Volts	Synchronous Vdd IO telemetry
0x0014	32-bit IEEE floating-point, °C	Synchronous supervisor temperature telemetry
0x0018	32-bit IEEE floating-point, Volts	Synchronous Vdd supervisor telemetry
0x001C	32-bit IEEE floating-point, Volts	Synchronous ADC ground telemetry (Rev4) Synchronous Vdd detector telemetry (Rev5)

The analog frame structure is used by the self-test mode to capture snapshots of the analog telemetry under various conditions.

9.2. Functional Processor Mapping

9.2.1. Memory Map

Table 52: Functional Processor Memory Map

Address Range	Function
0x40200000 – 0x4020FFFF	SRAM, containing bootloader
0x80000000 – 0x87FFFFFF	SDRAM, general purpose
0x88000000 – 0x88FFFFFF	SDRAM, Image buffer 0
0x89000000 – 0x89FFFFFF	SDRAM, Image buffer 1
0x8A000000 – 0x8AFFFFFF	SDRAM, Image buffer 2
0x8B000000 – 0x8BFFFFFF	SDRAM, Image buffer 3
0x8C000000 – 0x8CFFFFFF	SDRAM, Image buffer 4
0x8D000000 – 0x8DFFFFFF	SDRAM, Image buffer 5
0x8E000000 – 0x8EFFFFFF	SDRAM, Image buffer 6
0x8F000000 – 0x8FFFFFFF	SDRAM, Image buffer 7

The functional processor has memory mapped RAM as shown above that can be accessed using PEEK and POKE commands. In addition, there are a number of memory-mapped peripheral devices. Attempts to access unimplemented memory regions will not result in a NACK, but will result in a “Data abort” emergency terminate message.

9.2.2. NAND Flash

Table 53: Functional Processor NAND Map

NAND Page	Function
0x00000 – 0x0001F	Boot image 0
0x00040 – 0x0005F	Boot image 1

0x00080 – 0x0009F	Boot image 2
0x000C0 – 0x000DF	Boot image 3
0x00100 – 0x013F	Calibration structure
0x00140 – 0x001BF	Block Relocation Table
0x001C0 – 0x0023F	Block Relocation Table Backup
0x00240 – 0x002BF	Block Backup Table
0x002C0 – 0x0033F	Block Backup Table Backup
0x00340 – 0x00340	ECC Signal Page
0x00400 – 0x00BFF	Hash table catalog
0x00C00 – 0x00FFF	Star table catalog
0x01000 – 0x07FFF	Triangle table catalog
0x08000 – 0x08FFF	Application image
0x10000 – 0x11FFF	NAND, Image buffer 0
0x12000 – 0x13FFF	NAND, Image buffer 1
0x14000 – 0x15FFF	NAND, Image buffer 2
0x16000 – 0x17FFF	NAND, Image buffer 3
0x18000 – 0x19FFF	NAND, Image buffer 4
0x1A000 – 0x1BFFF	NAND, Image buffer 5
0x1C000 – 0x1DFFF	NAND, Image buffer 6
0x1E000 – 0x1FFFF	NAND, Image buffer 7

The functional processor NAND memory is divided into pages and blocks. Each page contains 2 kB of ECC corrected data (the ECC check-bits are not easily visible to the user). A page is the smallest programmable unit. 64 consecutive pages make up a block, which is the smallest erasable unit.

If the functional processor is commanded to boot from NAND flash, it will first attempt boot image 0. If there are errors uncorrectable by ECC in this image, it will move on to boot image 1, and so on for boot images 2 and 3.

9.2.2.1. Calibration Structure

Table 54: Calibration Structure

Offset	Type	Notes
0x0000	32-bit unsigned integer	Sensor serial number
0x0004	32-bit unsigned integer	Revision
0x0008	Array of 2 x 64-bit IEEE-754 floating point	Pixel pitch in two directions (m)
0x0018	64-bit IEEE-754 floating point	Focal length (m)
0x0020	64-bit IEEE-754 floating point	Inverse focal length (1/m)
0x0028	Array of 2 x 64-bit IEEE-754 floating point	Sin(eta) x 2
0x0038	Array of 2 x 64-bit IEEE-754 floating point	Cos(eta) x 2
0x0048	Array of 2 x 64-bit IEEE-754 floating point	Beta_r x 2
0x0058	Array of 2 x 64-bit IEEE-754 floating point	Beta_s x 2
0x0068	Array of 2 x 64-bit IEEE-754 floating point	Beta_t x 2
0x0078	Array of 2 x 64-bit IEEE-754 floating point	Optical center pixel location

0x0088	Array of 3 x 64-bit IEEE-754 floating point	Optical axis vector, encoding focal length (m)
0x00A0	Array of 3 x 3 x 64-bit IEEE-754 floating point	C_MD rotation matrix
0x00E8	Array of 2 x 32-bit signed integer	Active array dimensions x 2
0x00F0	Array of 2 x 32-bit signed integer	Raw array dimensions x 2
0x00F8	Array of 2 x 32-bit signed integer	First Pixel dimensions x 2
0x0100	Array of 2 x 8-bit unsigned integer	Flip axis x 2
0x0102	16-bits of padding	
0x0104	Array of 4 x 16-bit signed integer	Analog offsets – no longer recommended for use
0x010C	32-bit unsigned integer	Use ROI
0x0110	ROI definition	

The Use ROI field may be one of the following:

Table 55: Use ROI Field

Use ROI Field	Meaning
0	No ROI bound
1	Row ROI bound
2	Ellipse ROI bound
3	Compound ROI bound

If No ROI bound is selected then the entire imager is considered to be the region of interest. Otherwise, the ROI definition field in the calibration structure specifies the ROI. The form of the definition depends on the Use ROI field.

Table 56: Row ROI Bound ROI Definition

Offset	Type	Notes
0x0000	Array of 389 x 16-bit unsigned integers	Row start
0x030A	Array of 389 x 16-bit unsigned integers	Row end

Each entry in the table controls the ROI bound for five rows in the image. If the row start is greater than the row end, then the row is considered to have no valid pixels in it. Using the Row ROI bound a rectangular window can be easily created. Custom window shapes can also be made, though they will be somewhat jagged due to the five row bundling.

Table 57: Ellipse ROI Bound ROI Definition

Offset	Type	Notes
0x0000	16-bit signed integer	Top row of bounding ellipse
0x0002	16-bit signed integer	Left column of bounding ellipse
0x0004	16-bit signed integer	Bottom row of bounding ellipse
0x0006	16-bit signed integer	Right column of bounding ellipse
0x0008	16-bit signed integer	In boundary width
0x000A	16-bit signed integer	Out boundary width

This structure defines an elliptical ROI. The ellipse is specified by the elliptical bounds – it fits within a rectangular bounding box which is not rotated with respect to the detector axes.

The boundary width parameters are relevant if the boundary weighting is activated in the control structure. This algorithm penalizes stars that are detected near the ROI boundary. Part of the required definition is the in_bnd (where penalties begin) and out_bnd (where penalties are maximum). The in_bnd field specifies the number pixels inside the boundary where penalties should begin; out_bnd specifies the number of pixels outside the boundary where they hit their maximum. Centroids within this boundary range are assigned a linearly-interpolated weight from 1 to 0.01 (max penalty). This weight multiplies any intensity weighting, if applicable. If the ROI and boundary regions are not defined appropriately, no weights are assigned.

Table 58: Compound ROI Bound ROI Definition

Offset	Type	Notes
0x0000	Array of 389 x 16-bit unsigned integers	Row start
0x030A	Array of 389 x 16-bit unsigned integers	Row end
0x0614	Array of 4 x 16-bit signed integers	Elliptical bounds
0x061C	16-bit signed integer	In boundary width
0x0626	16-bit signed integer	Out boundary width

The compound ROI bound combines both the row bound and the elliptical bound. The row bound is used to produces the ROI shape, while the elliptical bound is used for boundary weighting only.

9.2.2.2. Block Relocation Table

After booting, the functional processor will read the block relocation table from NAND flash. This table allows the mapping of logical NAND blocks to physical NAND blocks to be changed. Its nominal use is to bypass bad NAND blocks.

Each of the 128 pages within the table may have zero or more block relocation entries.

Table 59: Block Relocation Entry

Offset	Type	Notes
0	32-bit integer	Address of page in logical block
4	32-bit integer	Address of page in physical block

Block relocation entries are read from a page until an out of range marker (such as 0xFFFFFFFF) is found. If the marker is found in the first entry in the page, the table is considered complete. Otherwise the next page will be read. If a page cannot be read due to ECC failure (bad, or erased) the table is considered complete.

Later entries will replace earlier entries. Blocks that are not explicitly relocated in the table default to a physical address equal to the logical address.

The mapping need not be one-to-one. While each logical block maps to one physical block, multiple logical blocks may map to one physical block.

The block relocation table, and the four boot images, may not be relocated.

9.2.2.3. Block Backup Table

After booting, the functional processor will read the block backup table from NAND flash. This table allows one logical block (secondary) to serve as a backup for another logical block (primary).

A write or erasure to a primary block will have an identical effect on the secondary. If a read to a primary block fails due to ECC mismatch, a second read will be attempted from the backup.

Each of the 128 pages within the table may have zero or more block backup entries.

Table 60: Block Backup Entry

Offset	Type	Notes
0	32-bit integer	Logical address of page in primary block
4	32-bit integer	Logical address of page in secondary block

Block backup entries are read from a page until an out of range marker (such as 0xFFFFFFFF) is found. If the marker is found in the first entry in the page, the table is considered complete. Otherwise the next page will be read. If a page cannot be read due to ECC failure (bad, or erased) the table is considered complete. Later entries will replace earlier entries.

Each block may have only one backup. It is valid, but not necessary, to have a bidirectional backup structure. I.e. block A may backup block B, and block B may backup block A.

The block relocation table and the block backup table have preset backup locations, as shown in Table 53, which cannot be changed. It is legal to enter backup blocks for any of the four boot images, but the boot process itself will only consider the first four physical blocks.

9.2.2.4. ECC Signal Page

Immediately after booting (prior to reading the backup or relocation tables), the functional processor will attempt to read the ECC Signal Page using 4-bit ECC. If this read is successful, and returns a non-erased result, then 4-bit ECC will be used as the default for all subsequent NAND reads and writes. Otherwise, 1-bit ECC will be used.

If the functional processor ECC default does not match the ECC mode used in the NAND flash then reads (including the backup and relocation tables) will not be successful.

9.2.3. Image Buffers

Image buffers may be stored in SDRAM or in NAND flash. In either event, they have the same format.

Table 61: Image Buffer Format

Offset	Type	Note
0x00000000	Reserved	Dark header
0x0000C000	Array of 2608 x 16-bit unsigned integers	First row of active pixels
0x0000D460	Array of 36 x 16-bit unsigned integers	First row of leading barrier pixels
0x0000D4A8	Array of 84 x 16-bit unsigned integers	First row of dark pixels

0x0000D550	Array of 14 x 16-bit unsigned integers	First row of trailing barrier pixels
0x0000D56C	Padding	Uninitialized garbage data
0x0000D800	Array of 2608 x 16-bit unsigned integers	Second row of active pixels
0x0000EC60	Array of 36 x 16-bit unsigned integers	Second row of leading barrier pixels
0x0000ECA8	Array of 84 x 16-bit unsigned integers	Second row of dark pixels
0x0000ED50	Array of 14 x 16-bit unsigned integers	Second row of trailing barrier pixels
…	…	…
0x00B6E800	Array of 2608 x 16-bit unsigned integers	1944th row of active pixels
…		

Each pixel is represented as a 16-bit unsigned integer. The detector uses a 12-bit ADC, and valid values are between 0 and 4095. Dark pixels give a measure of the signal chain offset. Barrier pixels have undetermined values and should not be used.

9.2.4. Control Structure

The control structure is sent by the supervisor to the functional processor when a suitable GO command is received, and causes the functional processor to begin a cycle.

Table 62: Control Structure

Offset	Type	Notes
0x0000	32-bit IEEE-754 floating point	Exposure length
0x0004	32-bit IEEE-754 floating point	Time between first and second exposure starts
0x0008	32-bit bitfield	Functional GO bitfield
0x000C	8-bit signed integer	Thumbnail downsample
0x000D	8-bit bitfield	Heater select
0x000E	8-bit bitfield	Weighting algorithm control
0x000F	8-bit signed integer	Heater target (deg C x 2)
0x0010	Array of 4 16-bit signed integers	Analog offsets for 4 colour channels
0x0018	16-bit unsigned integer	Thresholding raw threshold
0x001A	8-bit bitfield	Thresholding algorithm
0x001B	8-bit unsigned integer	Centroid window radius
0x001C	16-bit unsigned integer	Thresholding bin threshold
0x001E	16-bit unsigned integer	Thresholding maximum average
0x0020	16-bit unsigned integer	Thresholding minimum lit-pixels/star
0x0022	16-bit unsigned integer	Thresholding maximum lit-pixels/star
0x0024	Array of 3 32-bit IEEE-754 floating-point	Local gravity vector (for atmospheric correction)
0x0030	16-bit signed integer	ERS matching threshold
0x0032	16-bit signed integer	Lit pixel limit
0x0034	32-bit IEEE-754 floating point	Matching max set multiplier
0x0038	32-bit IEEE-754 floating point	Matching set stopping threshold multiplier
0x003C	32-bit IEEE-754 floating point	Matching gamma
0x0040	32-bit IEEE-754 floating point	Matching max conset error
0x0044	32-bit IEEE-754 floating point	Matching target arc length

0x0048	32-bit IEEE-754 floating point	Time offset between functional processor t=0 and supervisor processor t=0
0x004C	32-bit unsigned integer	Sequence counter, echoed back by functional processor in telemetry
0x0050	Array of 3 32-bit IEEE-754 floating-point	Spacecraft velocity, relative to the sun in ECI frame
0x005C	32-bit unsigned integer	Previous return bitfield
0x0060	Array of 4 32-bit IEEE-754 floating-point	Previous quaternion
0x0070	Array of 3 32-bit IEEE-754 floating-point	Previous rate
0x007C	32-bit IEEE-754 floating-point	Epoch of previous return relative to supervisor processor t=0 (sec)
0x0080	32-bit IEEE-754 floating-point	Maximum cross-axis rate change between frames for ERS compensation (rad/sec).
0x0084	32-bit IEEE-754 floating-point	Maximum about-axis rate change between frames for ERS compensation (rad/sec).
0x0088	32-bit IEEE-754 floating-point	Maximum age of previous solution for ERS compensation. (sec)
0x008C	32-bit IEEE-754 floating-point	Max angle change from previous solution for poor quality match. (rad)
0x0090	32-bit unsigned integer	Realtime clock (1.048576 sec ticks since J2000)

9.2.4.1. Exposure Length

This field allows the shutter length of the detector to be controlled. Longer shutters collect more photons and potentially detect more stars, but give worse streaking when the spacecraft has a high angular rate. Longer exposures also increase the total time-to-solution for the star tracker.

9.2.4.2. Time Between First and Second Exposure Starts

This field determines the time between the starts of the two exposures. If it is set to a length shorter than the exposure length, the exposure length will be used instead. Usually the minimum value would be used to give better time-to-solution, though a longer timebase will help with rate estimation if previous solution feedback is not being used.

When set to a negative value (typically -1.0) the second exposure is inhibited. Only one exposure is taken. This makes the star tracker faster, though external rate information is now required to allow correct ERS operation.

9.2.4.3. Functional GO bitfield

The functional GO bitfield allows the various processing stages to be enabled or disabled.

Table 63: Functional GO Bitfield

Bit	Function
0 (LSB)	Activate the detector to take two frames

1	Compute image noise statistics
2	Process the images
3	Synchronize DC/DC converters to reduce noise (No noticeable effect. Not recommended to use)
4	Produce a thumbnail image
5	Threshold and centroid the images
6	Compute star vectors
7	Perform ERS correction
8	Perform star matching
9	Perform time correction
10	Perform Two-Pass Star Selection (for matching)
11	Perform self-test
12 – 15	Copy first image out
16 – 19	Copy first image in
20 – 23	Copy second image out
24 – 27	Copy second image in
28	Cache triangle table
29	Perform benchmark
30	Use hardware BLC to set analog offsets
31	Perform atmospheric refraction correction

The supervisor processor will automatically set bit 11 when commanded to self-test. It should typically be cleared otherwise.

Table 64: Copy Image Out Code

Copy Image Out Code	Function
0x0	Do nothing
0x1	Copy image into RAM buffer 1
0x2	Copy image into RAM buffer 2
0x3	Copy image into RAM buffer 3
0x4	Copy image into RAM buffer 4
0x5	Copy image into RAM buffer 5
0x6	Copy image into RAM buffer 6
0x7	Copy image into RAM buffer 7
0x8	Save image into Flash buffer 0
0x9	Save image into Flash buffer 1
0xA	Save image into Flash buffer 2
0xB	Save image into Flash buffer 3
0xC	Save image into Flash buffer 4
0xD	Save image into Flash buffer 5
0xE	Save image into Flash buffer 6
0xF	Save image into Flash buffer 7

The Copy Image Out Codes, located at offsets 12 and 20, may be used to copy the images that have just been taken by the detector into additional RAM or flash buffers.

Table 65: Copy Image In Code

Copy Image In Code	Function
0x0	Do nothing
0x1	Copy image from RAM buffer 1
0x2	Copy image from RAM buffer 2
0x3	Copy image from RAM buffer 3
0x4	Copy image from RAM buffer 4
0x5	Copy image from RAM buffer 5
0x6	Copy image from RAM buffer 6
0x7	Copy image from RAM buffer 7
0x8	Load image from Flash buffer 0
0x9	Load image from Flash buffer 1
0xA	Load image from Flash buffer 2
0xB	Load image from Flash buffer 3
0xC	Load image from Flash buffer 4
0xD	Load image from Flash buffer 5
0xE	Load image from Flash buffer 6
0xF	Load image from Flash buffer 7

The Copy Image In Codes, located at offsets 16 and 24, may be used to replace the images that have just been taken by the detector with images from RAM or flash buffers. These images will then be processed as if they were live.

9.2.4.4. Thumbnail Downsample

This field, together with the thumbnail bit in the GO bitfield, can be used to produce a compressed image. Further details of this feature are TBD.

9.2.4.5. Weighting Algorithm Control

Table 66: Weighting Algorithm Control

Bit	Function
0 (LSB)	Use intensity weighting
1	Use boundary penalties
2 – 7 (MSB)	Reserved

When these bits are zero, all detected star vectors are weighted equally when the final quaternion is calculated. Intensity weighting allows for brighter stars, which are presumed to have smaller centroid error, to be weighted above dimmer stars. Boundary penalties reduce the weighting of stars near the edge of the ROI, which may have erroneous centroids due to clipping effects.

9.2.4.6. Heater Control

The star tracker can be configured with thermostatic controlled heaters to keep temperatures constant across an orbit. This is most effective in the ST-16, where there is a considerable thermal resistance between the circuit board and its chassis. The ST-16RT's

and ST-16RT2's detector is well coupled to the chassis, and so a modest amount of heater power has little effect on its temperature.

Table 67: Heater Select

Bit	Function
0 (LSB)	Heat the detector by keeping the electronics at full speed even when not needed.
1	Heat the functional processor by keeping its clock at full speed even when not needed.
2 – 7 (MSB)	Reserved

The Heater Select parameter allows various onboard heaters to be enabled. By setting this parameter to zero, all heater functions are disabled.

The Heater Target parameter adjusts the thermostat's setpoint. It is an 8-bit signed quantity, and the units are half degrees. Thus the setpoint can lie in the range of -64 C to +63.5 C. This setpoint is compared to the detector temperature to determine when to apply heat.

9.2.4.7. Analog Offsets

The analog offsets for the four colour channels drive DACs in the detector signal chains to keep the amplifiers in their linear region. The offsets are calibrated for best performance at the factory, and the factory values should be used unless there is good reason to change them.

If the Use Hardware BLC bit in the GO code is set then these values are not used, and instead the detector will attempt to determine the analog offsets in realtime. That may give better compensation for temperature effects, but risks bad frames from radiation hits.

9.2.4.8. Thresholding raw threshold

This determines the required brightness for a single pixel to be considered 'lit'. Smaller values will detect more stars, but risk false stars and excessive processing time.

9.2.4.9. Thresholding Algorithm

Table 68: Thresholding Algorithm

Bit	Function
0 (LSB)	Use adaptive thresholds
1	Use two-pass centroiding
2 – 7 (MSB)	Reserved

If adaptive thresholds are selected then the average brightness of a 128 pixel wide row window is subtracted from a pixel's brightness before the raw threshold comparison is made. This compensates for elevated background due to stray light. If adaptive thresholds are not selected then the dark offset alone is subtracted from the brightness before the comparison.

If two-pass centroiding is selected then centroids will only be calculated for the best candidate stars based on their total brightness. Otherwise all stars will be centroided.

Setting this bit can give a speed improvement for a scene with many lit pixels. There is no known disadvantage.

9.2.4.10. Centroid Window Radius

If this value is set to zero then the centroid of a star is based on the lit pixels that make it up.

If this value is non-zero, then the centroid is based on a circular window, centered on a first-pass centroid made from the lit pixels. The central pixel is not included in the radius. Thus, a value of 7 gives a window diameter of 15 pixels.

Circular windows give lower noise performance at low rate, but are unable to deal very well with motion streaks.

9.2.4.11. Thresholding Bin Threshold

For a star to be considered the integrated brightness of all of its lit pixels must reach this value. Setting this value higher will result in fewer star detections, but also fewer spurious peaks.

9.2.4.12. Thresholding Maximum Average

This parameter is only relevant if adaptive thresholds have been selected in the thresholding algorithm field. If the average of a 128 pixel wide row window exceeds this value then the pixel will not be considered as lit. Set this value to zero to disable the function.

The intent of this feature is to reject the moon and other bright extended objects.

9.2.4.13. Thresholding Minimum Lit-Pixels/Star

If a peak has fewer adjacent lit pixels (determined by horizontal and vertical, but not diagonal, connectivity) than this value, it is not considered a candidate for a star.

This is the first line of defense against thermal and proton noise. If it is set too high it may eliminate real stars.

9.2.4.14. Thresholding Maximum Lit-Pixels/Star

If a peak has greater adjacent lit pixels than this value, it is not considered a candidate for a star.

This is intended to reject the moon, Earth limb, and bright baffle edges. It can also be used to intentionally eliminate bright planets and stars. Bright stars may saturate the detector and centroid poorly, while planets are not currently in the star tracker catalog.

9.2.4.15. Local Gravity Vector

If atmospheric refraction is turned on, the star tracker uses this field to determine the nadir direction. It is intended for sea-level ground-based testing only.

9.2.4.16. ERS Matching Threshold

TBD

9.2.4.17. Lit Pixel Limit

The total number of lit pixels permitted in an image is:

$$\frac{1{,}000{,}000}{Lit\ Pixel\ Limit + 1}$$

The star tracker will abort processing of an image if the number of lit pixels approaches this value. If this value is set high then more scenes will abort, but the maximum time-to-reply is reduced. If this value is set low then fewer scenes will abort, but the star tracker may take a long time to process bright scenes.

9.2.4.18. Matching Max Set Multiplier

TBD

9.2.4.19. Matching Set Stopping Threshold Multiplier

TBD

9.2.4.20. Matching Gamma

TBD

9.2.4.21. Matching Max Conset Error

TBD

9.2.4.22. Matching Target Arc Length

TBD

9.2.4.23. Time Offset Between Functional Processor t=0 and Supervisor Processor t=0

This field is updated by the supervisor after a GO command is received, after the functional processor clock is reset and just before the control structure is sent to the functional processor. It can be read as a measure of the boot-up time.

9.2.4.24. Sequence Counter

The supervisor increments the sequence counter one for each GO command. The user may write to this value as desired.

9.2.4.25. Spacecraft Velocity

This vector contains three scalar elements, encoding the spacecraft velocity with respect to the sun in ECI frame. Depending on the ephemeris control the supervisor may automatically update the vector, or it may be set by the user. The velocity is used by the functional processor for stellar aberration correction (not currently implemented).

9.2.4.26. Previous Return Bitfield

This field contains the bitfield from the previous star tracker result.

9.2.4.27. Previous Quaternion

This field contains the quaternion from the previous star tracker result. While the result structure uses 64-bit floating point values, here only 32-bit floating point values are used.

9.2.4.28. Previous Rate

This field contains the rate from the previous star tracker result. While the result structure uses 64-bit floating point values, here only 32-bit floating point values are used.

9.2.4.29. Epoch of Previous Return

This field is used as an input to the functional processor, telling it the epoch of the previous return relative to the functional processor t=0 clock. If the Previous Epoch Override value is non-zero then the supervisor will not automatically update this field. Otherwise it is set by the supervisor after the functional processor clock is reset but before the control structure is sent.

Since the previous return happened prior to the current moment, this value should always be negative.

9.2.4.30. Maximum Cross-axis/About-axis Rate Change

These fields specify the maximum difference between the internally calculated rate (from comparing image 1 and image 2) and the previous rate in the control structure. If the limit is exceeded then the star tracker will assume that the satellite is under angular acceleration at that the more recent (internally calculated) rate is valid. Otherwise the star tracker will use the more accurate previous rate information.

There are two fields, one for cross-axis rate and the other for about-axis rate. The about-axis rate knowledge is typically worse, so this field can be set higher. Both comparisons must pass in order to use the previous rate information.

Setting either value to zero forces the comparison to fail, and the internally calculated rate to be used. Setting a value to a negative number forces that one comparison to pass.

The previous rate will never be used if the previous return bitfield does not have the master return bit set.

9.2.4.31. Maximum Age of Previous Solution

This field specifies the maximum age of the previous solution, relative to the start of the functional processor clock. If the previous solution is too old then the previous rate is never used and the internally generated rate is chosen.

Setting this value to zero forces the comparison to fail, and the internally calculated rate to be used. Setting the value to a negative number forces the comparison to pass.

9.2.4.32. Maximum Angle Change

This field specifies the maximum quaternion rotation between the previous quaternion and the current quaternion, if the current solution is a poor quality match. The assumption is that if a poor quality match is consistent with a previous solution then it is likely good,

whereas if it is very different from the previous solution it is likely erroneous. If the difference is too great, the master return bit will be cleared. This has no effect if the current solution is high quality.

Setting this value to zero specifies that poor quality solutions never be used, while setting the value to a negative number disables the check.

10. Operating Concept

The star tracker is a flexible device which can be used in many ways. Nominal operation for ACS purposes will follow this sequence:

1. Power on
2. Send INIT 0x00002000 to start Idle mode
3. Send GO 0x0B to transition to Processing mode, booting the functional processor from NAND flash, and enabling the timeout
4. Poll with READ EDAC commands the "Result structure length" channel until enough bytes have arrived to encompass all of the desired star tracker data
5. Send one or more READ RESULT commands to read the star tracker data. Feed this into the satellite ACS algorithm.
6. Optionally, send one or more READ EDAC commands to read status data from the supervisor processor. Feed this into the satellite housekeeping telemetry system.
7. Go to step 3

If polling is not desired, follow this simpler sequence:

1. Power on
2. Send INIT 0x00002000 to start Idle mode
3. Send COMBINATION 0x00001E0B to transition to Processing mode, booting the functional processor from NAND flash, and enabling the timeout.
4. Wait for the COMBINATION reply packet, which will contain the return code, quaternion, angular rate, and epoch time.
5. Optionally, send one or more READ EDAC commands to read status data from the supervisor processor. Feed this into the satellite housekeeping telemetry system.
6. Go to step 3

To get the very fastest operation, turn on the functional processor with a GO 0x07 command. Then send a sequence of COMBINATION 0x00001E2F commands. These will keep the functional processor running at all times and not reboot between cycles. This will save ~38 msec from each cycle time (for Rev 4. Less for Rev 5), but may risk radiation induced crashes.

11. Frames of Reference and Data Definitions

The primary data return from the star tracker is a four-element quaternion vector (scalar component as the first element) representing the rotation from the inertial reference frame into the frame of the sensor. This section presents a short summary of these reference frame definitions and some helpful mathematical relationships to help make best use of the sensor output.

The inertial frame used by the sensor is the Earth-Centred Inertial J2000 system. Star positions are not corrected for parallax, sacrificing some precision for simpler clock-free operation.

11.1. ST-16

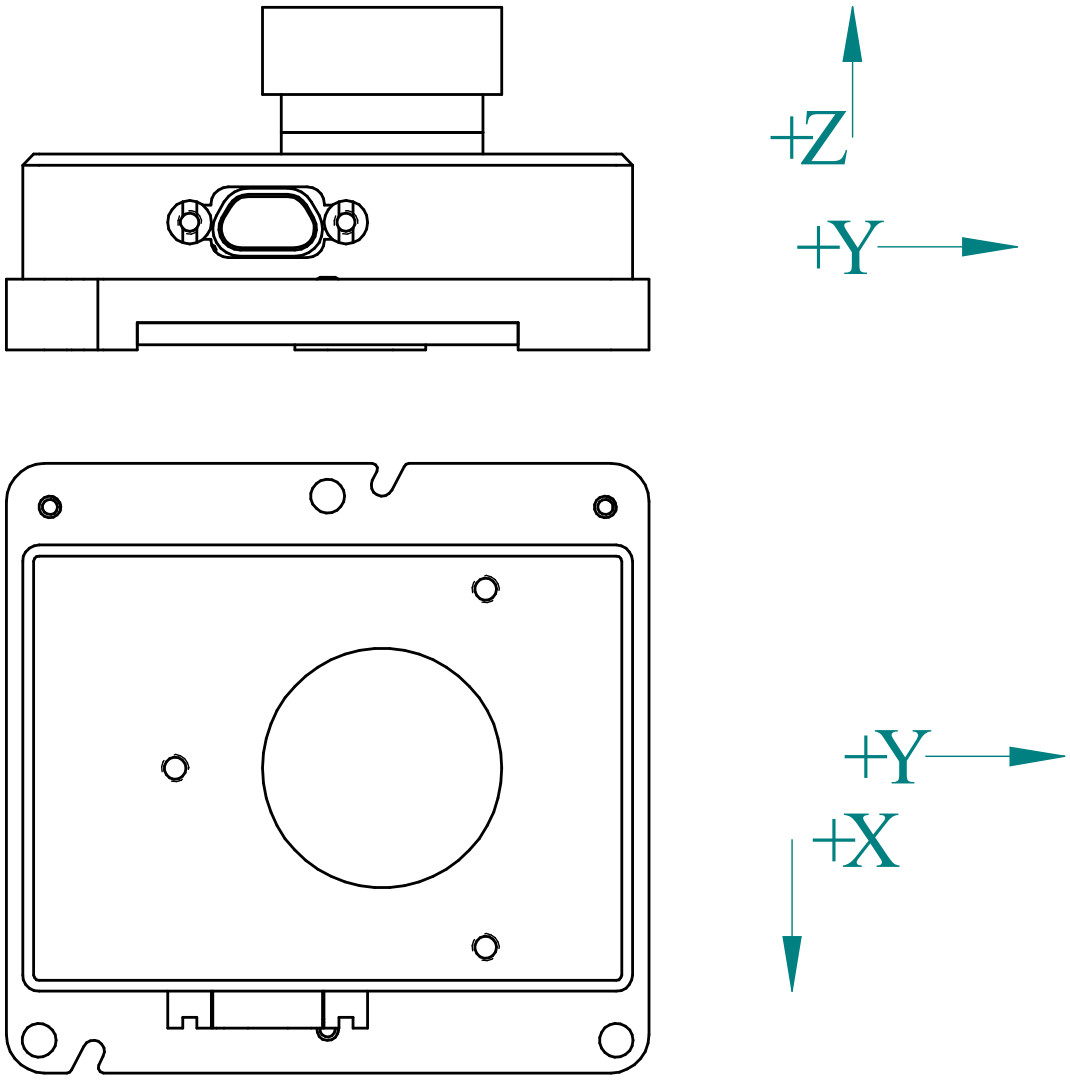


Figure 2: Informal ST-16 Coordinate Directions

The sensor reference frame is defined by mechanical features on the housing. Two primary vectors define the sensor reference frame. We have included both informal and formal definitions of these vectors.

- The +Z axis is informally the outward boresight of the sensor (i.e., direction the lens is facing). Formally, this direction is normal to the mounting bosses on the back of the sensor, pointing away from host spacecraft.
- The +X axis is informally in the direction of the power/data connector. More formally, the line of this vector is oriented 26.997° counterclockwise (facing the lens), from the datum formed by the two alignment pins.
- The +Y is chosen to make the coordinate system right-handed.

11.2. ST-16RT and ST-16RT2 – Chassis with Prism

The sensor reference frame is defined by the optical prism. The informal vector definitions are the same as for the ST-16. The formal definitions are:

- The mirror face pointed away from the power/data connector (Surface 1) defines the –X axis.
- The cross-product between the two mirror faces defines the Z axis.
- The +Y axis is chosen to make the coordinate system right-handed.

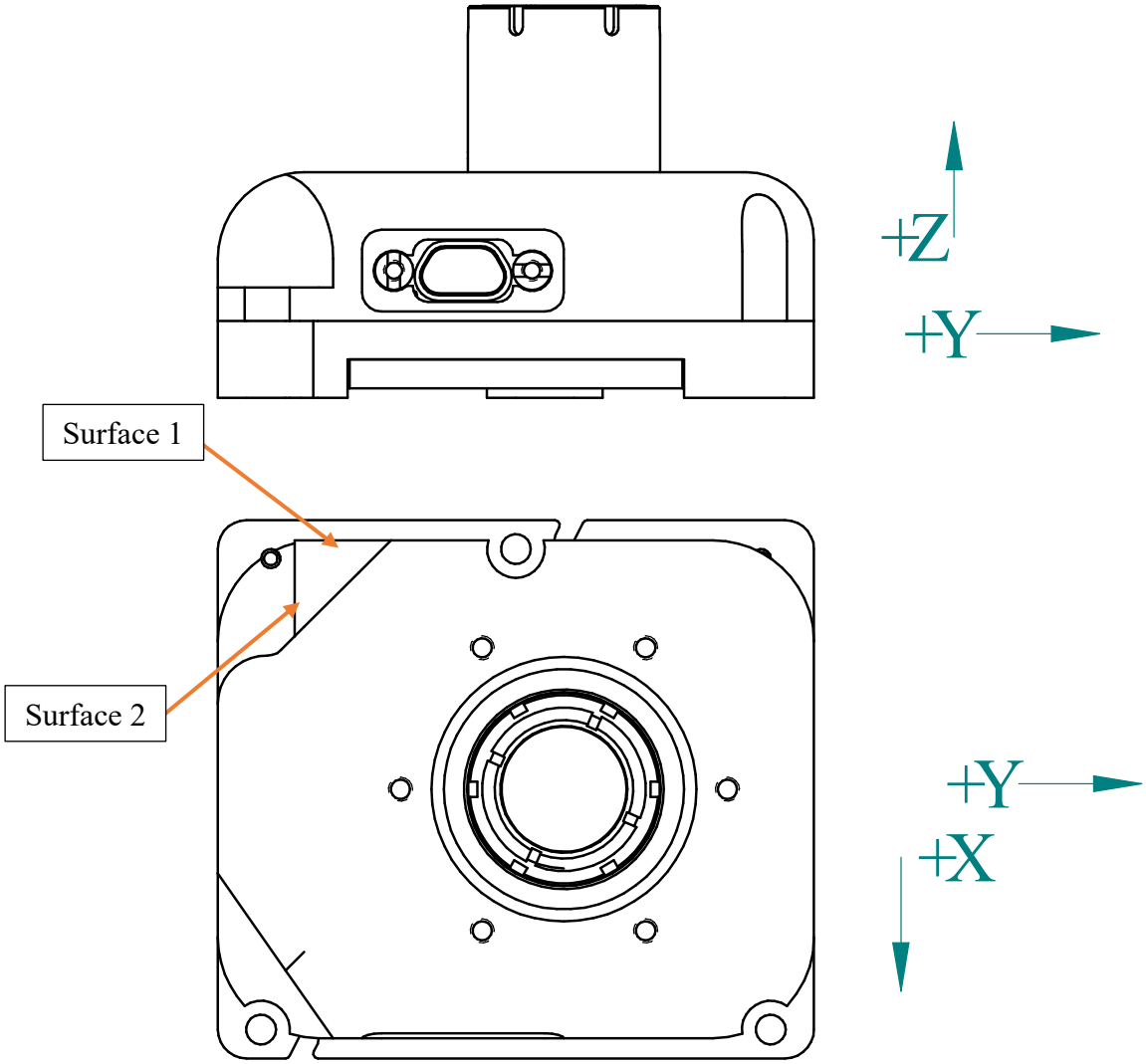


Figure 3: Informal Coordinate Definitions for ST-16RT and ST-16RT2 with Prisms

Note that while the Y axis is very close to the second mirror face normal, they may differ slightly if the prism angle is not exactly 90°. The ST-16RT and ST-16RT2 retain slots for alignment pins to make its mounting backwards-compatible with the ST-16, but the pins are not surveyed for calibration.

11.3. ST-16RT2 – Chassis with Polished Corners

The sensor reference frame is defined by measurements of the polished corners. First, an informal frame definition is needed to establish idealized surface normal vectors of the two polished corners. This informal frame is first defined with:

- The +X axis in the outward direction of the power/data connector;
- The +Z axis in the outward boresight of the sensor (i.e., direction the lens is facing);
- And the +Y axis is chosen to make the coordinate system right-handed.

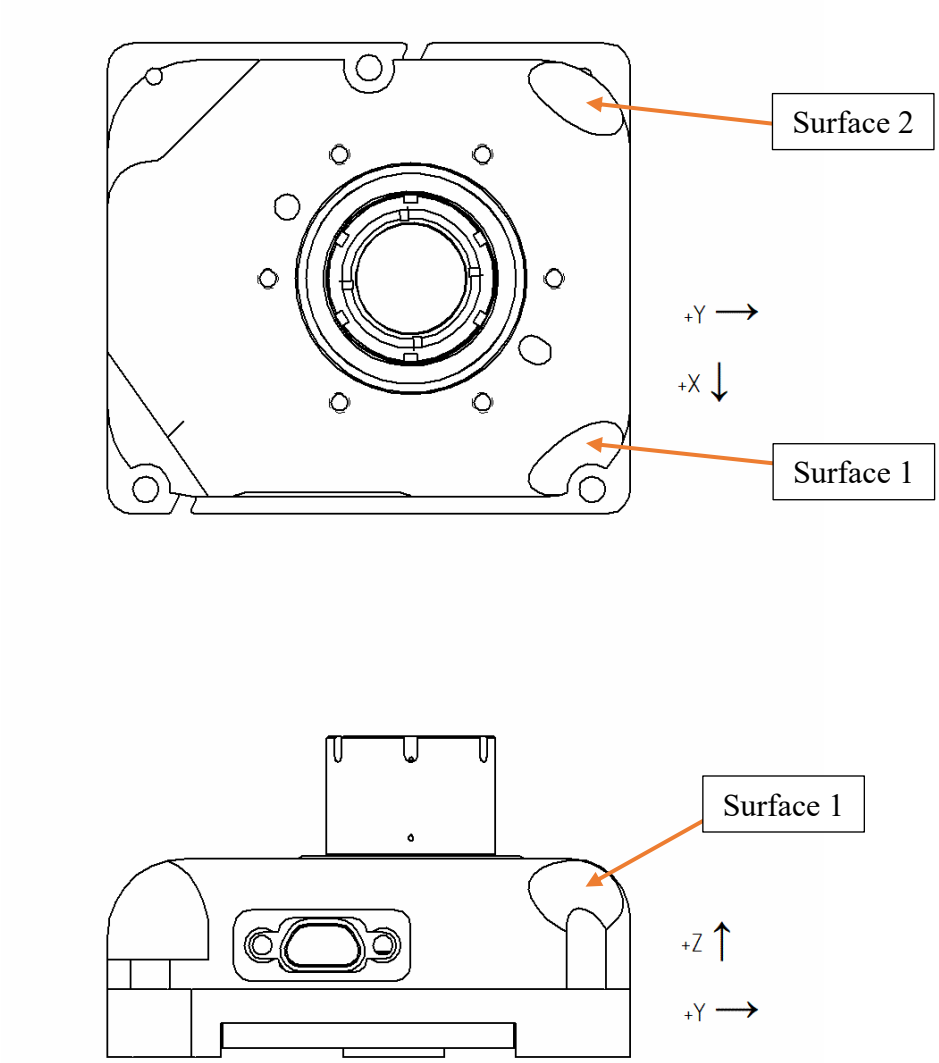


Figure 4: Informal ST-16RT2 with Polished Corners Coordinate Definitions

Using this informal reference frame, an ideal surface normal of each polished corner can be defined. Surface 1 is selected as the polished surface that is visible from the connector side of the star tracker.

The ST-16RT2 with polished corners retains slots for alignment pins to make its mounting backwards-compatible with the ST-16, but the pins are not surveyed during calibration.

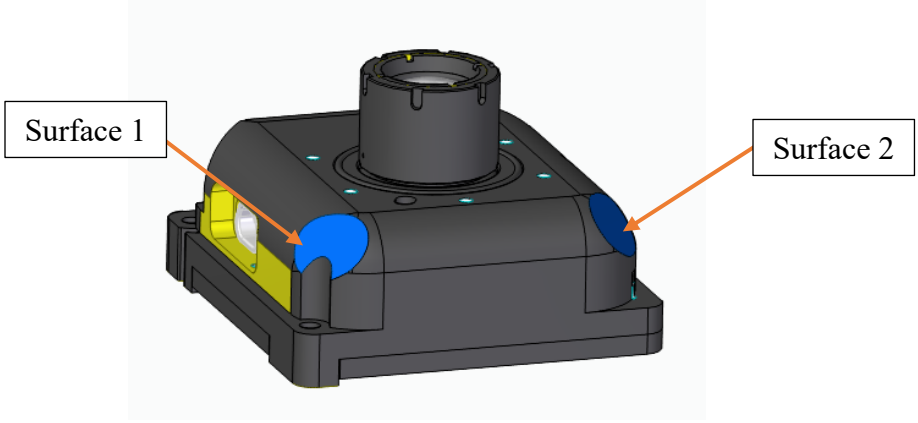


Figure 5: An alternate view of the Polished reference corners (blue)

11.4. Sensor Frame Measurement

The transformation between an external reference frame (i.e. the spacecraft body frame) and the sensor frame is determined with the following procedure.

Let $\hat{\boldsymbol{n}}_i$ be the unit vector that represents a measured surface normal in the external reference frame. For both the chassis with prism and the chassis with polished corners, the normal vector of the reflective surfaces are labelled as $\hat{\boldsymbol{n}}_1$ and $\hat{\boldsymbol{n}}_2$ for Surface 1 and Surface 2 respectively.

For each chassis type, the normal of Surface 1 defines a vector in the sensor frame. If $\hat{\boldsymbol{n}}_i$ is a measured surface normal in the external frame, $\hat{\boldsymbol{s}}_i$ represents the same vector in the sensor reference frame. The surface normal vector of Surface 1 is defined in the table below.

Table 69: Surface 1 definitions in the sensor frame

Chassis with Prism	Chassis with Polished Corners
$\hat{\boldsymbol{s}}_1 = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$	$\hat{\boldsymbol{s}}_1 = \begin{bmatrix} \cos(45°) \\ \cos(\alpha) \\ \cos(55°) \end{bmatrix}$

The angle α is determined as

$$\cos \alpha^2 = 1 - \cos(55°)^2 - \cos(45°)^2$$

$$\alpha \cong 65.573°$$

Measurements of Surface 2 are not used to directly define a vector in the sensor frame. Instead, the cross product of Surface 1 and Surface 2 is used to define an orthogonal vector.

$$\hat{\boldsymbol{n}}_3 = [\hat{\boldsymbol{n}}_1]_\times \hat{\boldsymbol{n}}_2$$

Note that the cross product operator, $[\quad]_\times$, is defined for a vector $\vec{\boldsymbol{a}} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$ as

$$[\vec{\boldsymbol{a}}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

The cross product in the sensor frame is similarly determined and the definitions are tabulated below.

$$\hat{\boldsymbol{s}}_3 = (\hat{\boldsymbol{s}}_1)_\times \hat{\boldsymbol{s}}_2$$

Table 70: Vector cross product definitions in the sensor frame

Chassis with Prism	Chassis with Polished Corners
$\hat{\boldsymbol{s}}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\hat{\boldsymbol{s}}_3 = \begin{bmatrix} 0 \\ -2\cos(45°)\cos(55°) \\ 2\cos(45°)\cos(\alpha) \end{bmatrix}$

With two surface normal measurements and the vector definitions above, the transformation between the sensor frame and the external measurement frame can be computed. A direction cosine transformation matrix from the sensor frame to the external measurement frame, $\boldsymbol{C}_{MS}$, can be calculated as

$$\boldsymbol{C}_{MS} = [\hat{\boldsymbol{n}}_1 \quad (\hat{\boldsymbol{n}}_3)_\times \hat{\boldsymbol{n}}_1 \quad \hat{\boldsymbol{n}}_3][\hat{\boldsymbol{s}}_1 \quad (\hat{\boldsymbol{s}}_3)_\times \hat{\boldsymbol{s}}_1 \quad \hat{\boldsymbol{s}}_3]^T$$

Use of this result to transform an arbitrary vector in the sensor frame, $\vec{\boldsymbol{s}}$, into the external measurement frame, $\vec{\boldsymbol{m}}$, is accomplished simply as

$$\vec{\boldsymbol{m}} = \boldsymbol{C}_{MS}\vec{\boldsymbol{s}}$$

11.5. Quaternion Form

The quaternion returned by the sensor is a vector of four double-precision numbers. The scalar part of the quaternion is the first element; the vector part, the last three.

$$\mathbf{q} = \begin{bmatrix} q_s \\ \mathbf{q}_v \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

The rotation matrix from the inertial frame (I) to the sensor body frame (B) can be found from:

$$\mathbf{C}_{BI} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & 1-2(q_3^2+q_1^2) & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix}$$

The attitude error covariance returned from the sensor follows the classic definition by Shuster, calculated from the body-frame star vectors.

$$\mathbf{P}_{\theta\theta} = \left(\sum_i \left(\mathbf{I} - \mathbf{b}_i \mathbf{b}_i^T \right) \right)^{-1}$$

Note: This covariance is expressed in the sensor frame.

11.6. Epoch Time

The epoch time measurement is expressed in seconds from the reception of the command that initiated the cycle (see section 7.8). If time correction is turned off, then this represents the moment in time at which the returned quaternion is valid. If time correction is turned on, the star tracker uses its attitude and rate knowledge to extrapolate the quaternion to the moment the command was received. In this case the quaternion telemetry can be used in the satellite ACS without requiring epoch telemetry.

12. Flash Memory Issues

12.1. Introduction

There are two flash memories in the star tracker, as shown in Table 71.

Table 71: Flash Memory Parameters

	Supervisor Processor	Functional Processor
Flash Technology	NOR SLC Flash	NAND SLC Flash
Program/Erase Endurance	20,000 minimum 150,000 typical	100,000
Read Endurance	Unlimited	Variable

It is well known that flash memories have a write endurance limit. After a certain number of program/erase cycles, a page or block may become damaged and unusable. It is the user's responsibility to ensure that these limits are not approached. In the event that a small region of flash wears out due to accidental program/erase loops it should be possible to modify the software to avoid this region and recover use of the device.

The supervisor NOR flash has no read endurance limit – the memory contents can be read an infinite number of times without disturbing them. This is not true for the functional NAND flash. Each read cycle disturbs the contents of the memory, and after a large number (literature suggests 1,000,000) bits may become spuriously programmed.

In normal operations, the functional processor reads from flash in the following ways:

- At bootup, the boot block is read
- The ECC signal page, flash relocation and backup tables are read
- The application image is read
- The ECC signal page, flash relocation and backup tables are read again
- The calibration table is read

- The entire star table and hash table are read
- A small subset of the triangle table is read, depending on which stars are identified in the images

If the star tracker is cycling at its maximum rate (~2 Hz) it may accumulate 172,800 cycles in a day. Since each flash block contains 64 pages, this gives 11,059,200 read accesses to a given block. This is well within the literature values for read disturbance.

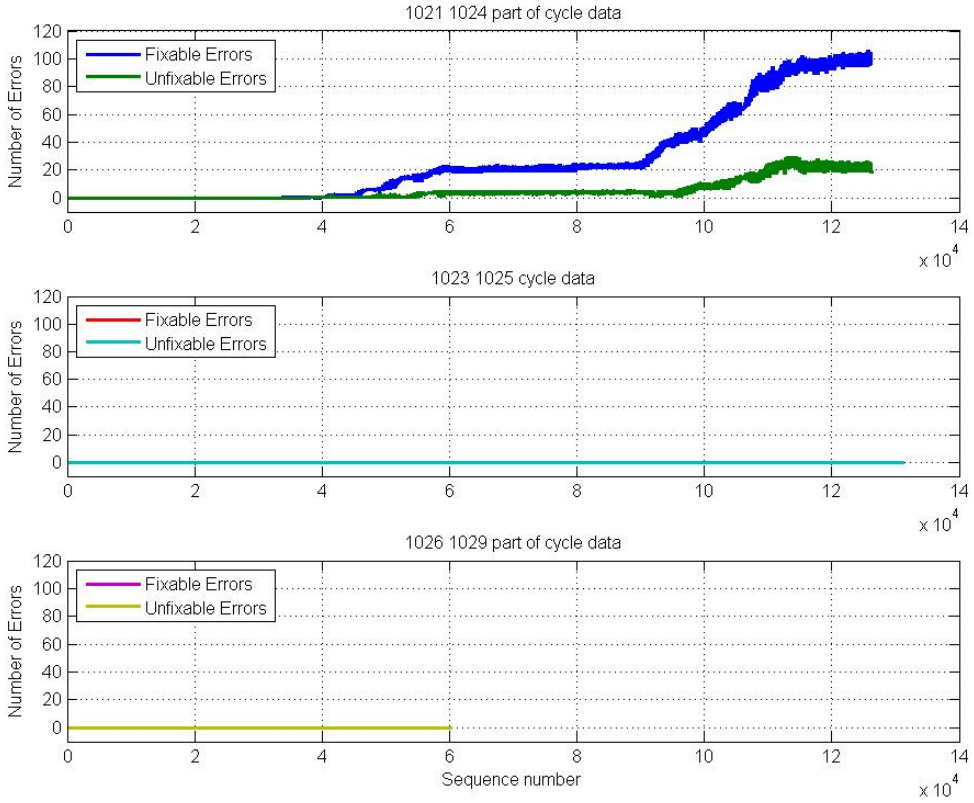


Figure 6: Flash Read Disturbance vs Sequence Number for 6 Star Trackers

Laboratory tests have been very confusing. Some units have shown flash degradation, while others have stubbornly refused to. Figure 6 shows data collected during the thermal acceptance testing of six units. Units 1021 and 1024 show errors, starting at about 40,000 cycles. The other four units show no errors.

In this particular case, units 1021 and 1024 are rev 5 parts, while the others are rev 4. This would seem to be pretty clear correlation, except that some (but not all!) customers with rev 4 hardware also report similar read disturbance errors.

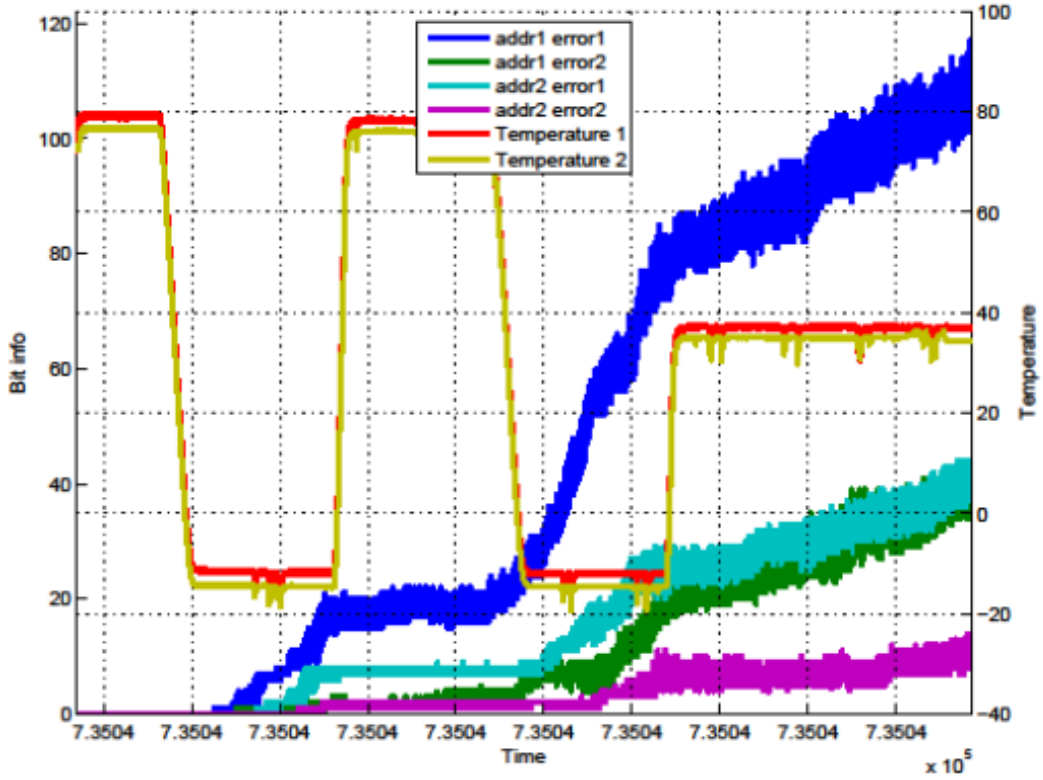


Figure 7: Flash Read Disturbance and Temperature vs Time for 2 Star Trackers

For the rev 5 parts, there appears to be a temperature sensitivity to the read disturbance effect. Figure 7 shows how the rate of error accumulation slows at high temperature, and accelerates at low temperature.

12.2. Error Correcting Codes

The NAND flash uses error correcting codes (ECC) to mitigate against read disturbance errors as well as bad bits that may be present from manufacturing. Two ECC strategies may be available: 1-bit ECC and 4-bit ECC.

12.2.1. 1-bit ECC

1-bit ECC is available to all star trackers, and historically it has been the mode used at delivery. If an EIDP makes no mention of the ECC type, then 1-bit ECC can be assumed.

1-bit ECC is so called because errors of a single bit within a 256 byte area of NAND flash can be corrected. 2-bit errors can be detected. 3 or more bit errors may not be detectable – we have seen more than one 3-bit error that has fooled the 1-bit ECC into believing that it is recoverable.

12.2.2. 4-bit ECC

4-bit ECC is available on most recent star trackers. Some older rev 4 star trackers may not support 4-bit ECC, and will return errors when 4-bit operation is attempted.

4-bit ECC is so called because errors of up to 4 bits within a 512 byte area of NAND flash can be corrected. 5-bit errors can be detected.

4-bit ECC is slightly slower than 1-bit ECC. Experiments show that the time taken for the functional processor to boot and load the star and hash tables increases by about 30 msec with 4-bit ECC as compared to 1-bit ECC.

12.2.3. Migrating Between ECC Types

You may wish to move a star tracker from one type of ECC to another. An older 1-bit ECC star tracker can be changed to 4-bit to gain better error robustness. A 4-bit ECC star tracker may be changed to 1-bit if the very fastest operation is desired.

To move to 4-bit ECC, first make sure that the ECC Signal Page is not empty. If it is erased, write (anything) to it. Next, issue an Upgrade NAND ECC command covering the entire NAND IC. This will take several minutes. Finally, power the functional processor off and back on. The NAND will now be protected by 4-bit ECC, and the functional processor will default to 4-bits when reading and writing.

To move to 1-bit ECC, issue a Downgrade NAND ECC command covering the entire NAND IC. This will take several minutes. Finally, power the functional processor off and back on. The NAND will now be protected by 1-bit ECC, and the functional processor will default to 1-bits when reading and writing.

12.3. Mitigation

Until this issue is fully understood, it is safest to assume that all star trackers are potentially susceptible. There are two mitigation methods that can be used. Both have their advantages and disadvantages, and the choice is left up to the customer.

12.3.1. Periodic Flash Rebuild

To employ this mitigation method, the user must periodically place the star tracker into maintenance mode and send Rewrite NAND commands. These commands should rewrite those areas of flash that have been subject to stress. They include:

- The boot block. Instead of the Rewrite NAND command, the Make Bootable command should be used.
- The flash relocation and backup tables
- The application image
- The calibration table
- The entire star table and hash table
- The triangle table

Of these, the triangle table is probably stressed less than the others. It may be possible to rewrite it less often than the other areas.

The rewrite frequency must be carefully chosen. The program/erase endurance of the NAND flash is only 100,000 cycles. For a 10 year mission, with no margin, the maximum

rewrite frequency would be once an hour. Rewriting only once every few hours will give margin.

The current health of the NAND flash can be estimated by reading the flash error counts at the beginning of the hardware telemetry. By observing these trends, and appropriate rewrite frequency can be determined.

Where possible, perform the rewrite operation in a low radiation portion of the orbit. If in LEO, avoid the South Atlantic Anomaly (SAA). Flash memory is more vulnerable to latchup while its charge pumps are active, performing program and erase functions.

12.3.2. Run From RAM

To employ this mitigation method, set the "Cache triangle table" bit in the functional GO bitfield in the control structure. Then, when sending GO or COMBINATION commands, ensure that bits 2 and 5 are set.

When the first GO or COMBINATION command is received, the star tracker will take slightly longer than normal to execute. In addition to the star and hash tables, the entire triangle table will be copied into RAM. The functional processor will remain turned on, and all accesses will be to RAM. There will be no repeated flash accesses, and so the flash will not accumulate read disturbance errors.

12.3.3. Comparison

Neither mitigation strategy is ideal. The periodic flash rebuild requires that the star tracker be taken out of operation for a number of seconds, several times a day. This may interfere with missions that demand very high ACS availability. It also consumes program/erase cycles – a finite resource. There should be sufficient cycles for any normal mission, but nevertheless it is disquieting.

Running from RAM defeats one of the prime design features of the star tracker – that the functional processor powers down and reboots between cycles. Expect to see periodic SEU events where the functional processor becomes corrupted and requires a full reset (GO code of 0). The spacecraft flight computer will need to detect these events autonomously to guarantee high ACS availability. SEL events might occur as well, and by keeping the functional processor powered continually the chance of a destructive event may increase.

12.3.4. Recommendation

All else being equal, Sinclair Interplanetary recommends the periodic flash rebuild mitigation method. Keep an eye on the flash error counts. Maybe your star tracker will not experience read disturbance degradation in your application, and no mitigation will be required.

The use of 4-bit ECC is recommended unless there is a compelling reason to do otherwise.

12.4. Catalog Rebuild

Some star trackers ship from the factory with the catalog rebuilding program installed in the functional processor NAND flash. If a star tracker was not originally supplied with this, the program can be loaded by the customer at a later date.

To run the program, follow these steps:

- Put the star tracker into maintenance mode
- Send the functional processor an INIT 0x00008080 command
- The star tracker will enter processing mode, and the catalog rebuild process will begin.
- After approximately three minutes the star tracker will transition to idle mode. The catalog rebuilding process will be complete.
- Check the functional processor message field for a statement indicating success or failure.

The catalog rebuilding program uses the star catalog, from its nominal location, as an input. It generates fresh hash tables and triangle tables from this. They are stored in their nominal locations, overwriting whatever may have been stored there before.

The catalog rebuilding program is intended to provide emergency catalog replacement in the event of catalog damage on-orbit. In the future its use may be expanded to allow for proper motion compensation.

13. Stellar Aberration Correction

Stellar aberration correction is now fully functional in the most recent software revisions. There are a number of different operations concepts that can be used.

13.1. No correction

To disable stellar aberration correction, set Ephemeris control to 0x40. The control structure spacecraft velocity fields are held at zero, implicitly nulling the stellar aberration correction.

13.2. Lowest effort

The bulk of the velocity comes from the Earth's orbit about the sun. To correct this term only, first use the WRITE TIME command to set the star tracker's realtime clock to the current epoch. Then set Ephemeris control to 0x11. The star tracker will propagate the Earth's motion around the sun and use that to feed the control structure velocity. No additional configuration or attention is required.

13.3. Osculating Elements

Additional accuracy can be derived from compensating for the satellite's motion about the Earth. To do this, first use the WRITE TIME command to set the star tracker's realtime clock to the current epoch. Then use WRITE KEPS to set the osculating elements of the satellite's current orbit. Finally, set the Ephemeris control to 0x35. The star tracker will propagate the Earth's motion around the sun, and the satellite's motion around the Earth. The sum of these two will be fed into the control structure velocity.

The star tracker's satellite orbit propagator is an idealized two-body model. It does not account for J2 gravity harmonics, lunar interaction, drag, or any other disturbance. For

these reasons the Keplerian elements will quickly grow stale. The WRITE KEPS command should be used periodically (between hours to weeks, depending on mission requirements) to keep the osculating elements current.

13.4. Heliocentric Override

The host spacecraft can take over the duties of computing the spacecraft velocity. This can be done if the spacecraft is on an interplanetary trajectory, or if its orbit evolves such that Keplerian elements are not appropriate. To do this, set the Ephemeris control to 0x00. Periodically write the most recent velocity to the velocity vector in the control structure. This can be updated on a minute-to-minute basis if needed.

14. Rate Estimation, Feedback and Feedforward

14.1. Rate Output

The star tracker produces a vehicle angular velocity estimate in its result structure. This rate may be generated in one of two ways.

a) By comparison of the two images taken in this frame, usually separated by 0.1 sec. The most likely rotation that maps stars seen in one image to stars seen in the other image is chosen.
b) By comparing the quaternion solution from this frame with the "previous quaternion" field of the control structure, computing the time between these quaternions based on the previous epoch data.

The "rate source" bit in the return code can be inspected to determine which of the two methods was used. If "0", the method used in (a) was used. If "1", the method used in (b) was used.

Method (a) is used, unless all of the following conditions hold true:

- The time between the previous quaternion and the quaternion from this frame is less than the "maximum age" field from the control structure (or the maximum age field is negative).
- The previous return bitfield from the control structure has the "master return" bit set.

14.2. Rate Input

The star tracker detector uses an electronic rolling shutter. This produces image distortion when an exposure is taken at a non-zero angular velocity. The star tracker must have an estimate of angular velocity in order to undistort its images and successfully match stars. This rate may be determined in one of two ways:

a) By comparison of the two images taken in this frame, usually separated by 0.1 sec. The most likely rotation that maps stars seen in one image to stars seen in the other image is chosen.
b) By using the "previous rate" field in the control structure.

The “ERS return code” telemetry can be used to determine which method is used. A value of “11” indicates method (b). A positive value of 10 or less indicates method (a) with a successful match of that number of stars.

If the time between the first and second exposure starts is negative then the second image is suppressed and method (b) is always used. Otherwise, method (a) is used unless all of the following conditions hold true:

- The previous epoch from the control structure is less than the maximum age, measured at the frame t=0 time (or the maximum age field is negative)
- The previous return bitfield from the control structure has both the “master return” and “rate source” bits set.
- The cross-axis difference between the “previous rate” and the rate estimate from method (a) is less than the “maximum cross-axis rate change” (or the maximum cross-axis rate change is negative).
- The about-axis difference between the “previous rate” and the rate estimate from method (a) is less than the “maximum about-axis rate change” (or the maximum about-axis rate change is negative).

14.3. Rate Feed-Forward

The star tracker rate estimate is based on a short time-base comparison of quaternions. A longer window filter would reduce noise, and thus give better quaternion solutions, but the latency would result in unacceptable error during high-acceleration spacecraft slews. The nominal behaviour is intended to maintain availability in slews, at the cost of some inertial performance.

The spacecraft attitude control computer may have a better estimate of rate. It can combine star tracker quaternions (possibly from multiple star trackers) with known body dynamics and knowledge of present actuator states. It is possible to feed this rate estimate into the star tracker to produce more accurate quaternions.

To do this, follow this sequence each frame:

1. Send a GO or COMBINATION command to start the frame.
2. Poll for completion, if a GO command was used.
3. Read the result telemetry.
4. Determine the best angular velocity estimate, using all of the information available to the spacecraft.
5. Write this estimate into the “previous rate” field of the control structure.
6. Write into the “previous return bitfield” field of the control structure, setting the “master return” and “rate source” bits.
7. Goto step 1.

For simplicity, configure the star tracker with the maximum cross-axis rate, about-axis rate and age parameters to negative numbers to disable those checks.