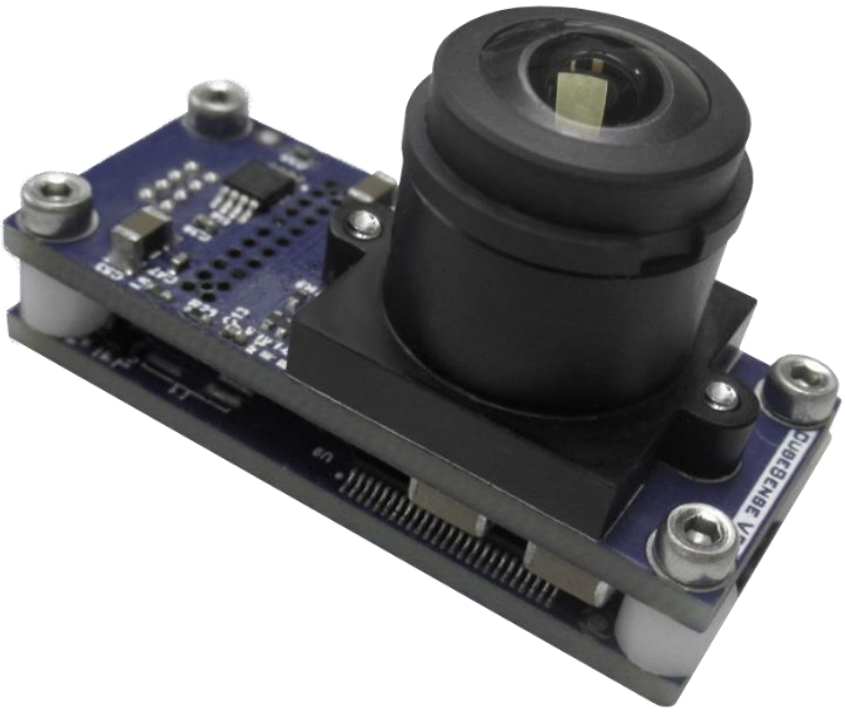
CubeSense V3

An integrated sun and nadir sensor module



Interface Control Document

Contact Us

Phone (0027) 79 945 9957
E-mail info@cubespace.co.za
Web www.cubespace.co.za
Facebook /CubeSpaceADCS
Twitter @CubeSpace_ADCS

Physical Address

CubeSpace
Hammanshand Road
Stellenbosch
7600
South Africa

Table of Contents

Document Approved By

Document Approved By	
Document	CubeSense V3 ICD
Version	1.4
Date Modified	04 November 2022
Approved By:	Douw Steyn
Signature	

List of Acronyms/Abbreviations

CMOS	Complementary metal-oxide semiconductor
ADCS	Attitude and Determination Control System
FPGA	Field Programmable Gate Array
I^2C	Inter- Integrated Circuit
MCU	Microcontroller Unit
OBC	Onboard Computer
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver/Transmitter
COTS	Commercially Off-the-shelf
PCB	Printed circuit board

1. Introduction

The CubeSense module is available as either a sun or nadir sensor and is used for CubeSat attitude sensing. If configured as a sun sensor, a neutral density filter is added to the optics to ensure that only the sun will be visible in the image. The camera has a wide field-of-view optics (200 degrees) for increased operating range.

The primary output of the sensor is the measured sun/nadir vector in the sensor's coordinate frame. If the CubeSense is configured as a nadir sensor then it can also be used as a camera to capture and download 1024x1024 pixel greyscale images.

This document is only applicable to CubeSense V3.

The unit contains a variety of static sensitive devices. The appropriate electrostatic protection measures must thus be implemented. **The unit must never be handled without proper grounding.**

It is recommended that the unit be handled in a clean environment. A clean room of ISO class 8 or higher or an appropriate laminar flow workbench is recommended.

The unit should be **kept free of moisture or liquids.** Liquids and moisture could have corrosive effects on the electronics and electronic joints which may lead to degradation and loss of reliability of the circuits.

The unit must be handled with care and **dropping or bumping the unit should be completely avoided.**

The camera **lens should be kept clean** and free of any dirt that may obstruct the images captured by the camera. Dust should be removed with a cloth. If required, the lens may be cleaned using ethanol and appropriate lens cleaning equipment, but unnecessary cleaning of the lens should be avoided.

The optic is fitted with a **dust cap which should be removed before flight**.

The position of the lens relative to the image sensor is of extreme importance for accurate detection. **Any external force on the lens or lens holder should be completely avoided.**

2. Specifications

Table 1 – Performance specifications

Physical	
Size	43 x 17.7 x 27.5 mm
Mass	19.7 g
Power	
Maximum power use	200 mW
Nominal mode	< 100 mW
Performance	
Maximum Update Rate	2 Hz
Accuracy (2σ)	
Nadir option	< 0.22° (Full earth in FOV)
Sun option	< 0.22° (Full range)
Detection Range	
Nadir option	130° vertical/horizontal and 160 degrees diagonal
Sun option	170° vertical/horizontal and 180 degrees diagonal

Table 2 – Electrical characteristics

Symbol	Parameter	Min	Nom	Max	Unit
V_{CC}	3V3 Supply voltage	3.2	3.3	3.4	V
T_A	Operating Temperature	-20	-	70	°C
I_{CC}	Idle Single camera capturing @ 1Hz	16 16	20 25	24 57	mA
V_{I2C}	I^2C voltage levels		3.3		V
R_{I2C}	I^2C bitrate		100		kHz
V_{UART}	UART voltage levels		3.3		V
R_{UART}	UART baudrate		57600		bps

3. Electrical Interface

The electrical interface to the CubeSense is achieved through an 8-pin connector with the location shown in Figure 1. This connector is either a *Samtec TFM-104-01-L-D* (straight) or a *Samtec TFC-104-01-F-D-RA* (right-angle). The harness used to connect to this connector is called the *Samtec SFSDT-04-28-G-XX.XX-S*.

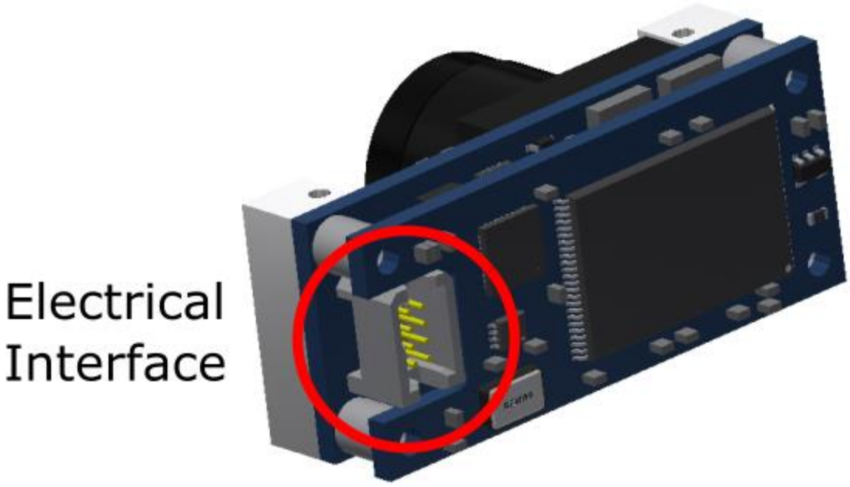


Figure 1 – Location of CubeSense's connector

The pinout of this connector is shown in Table 3 with the pinout direction shown in Figure 2.

Table 3 – Pinout of CubeSense Connector

Pin	Name	Description
1	3.3V	Input voltage of 3.3V
2	SDA[1]	SDA line for I2C communication
3	SCL[1]	SCL line for I2C communication
4	GND	Ground pin (both ground pins must be used)
5	RX	RX line for UART communication
6	TX	TX line for UART communication
7	EN	Enable line that turns the CubeSense on or off (active high)
8	GND	Ground pin (both ground pins must be used)

[1] There are no pull-up resistors on the I2C bus. Any pull-up resistor should be implemented by the master.

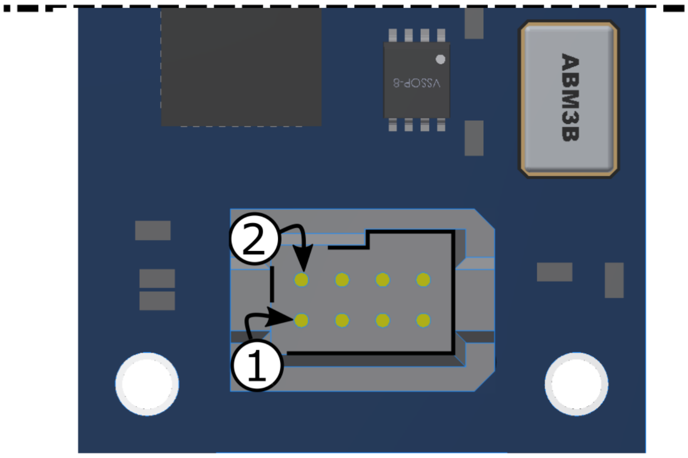


Figure 2 – Location of pin 1 on connector on the CubeSense's bottom

4. Mechanical Interface

The CubeSense must be mounted on the satellite's side-panel. The orientation for a nadir sensor is ideally pointing nadir during nominal satellite operation, and the sun sensor should ideally point towards the Sun's side of the orbit which can be either zenith or orbit normal. The exact placement of a sun sensor would depend on the orbit the satellite is launched in. The mounting hole layout is shown in Figure 3. The module comes with M2 treaded mounting brackets attached to the front with a hole depth of 3 mm.

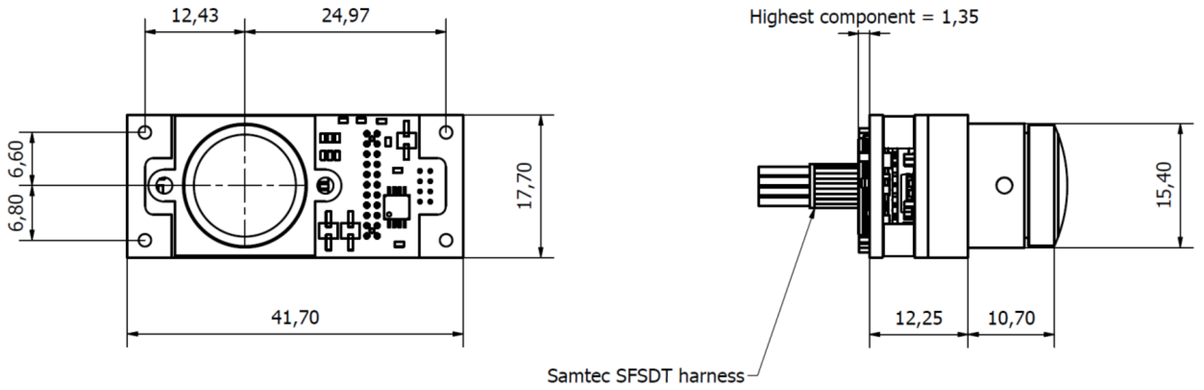


Figure 3 – CubeSense dimensions

An important mounting consideration is ensuring that the camera lens protrudes completely through the side panels of the satellite, as demonstrated in Figure 4. The CubeSense has a 190-degree fisheye lens, therefore if the lens does not protrude all the way through the side panel, the sensor will detect reflections from the side panels. Best practice is to have the lens protrude as far as possible from the side panel.

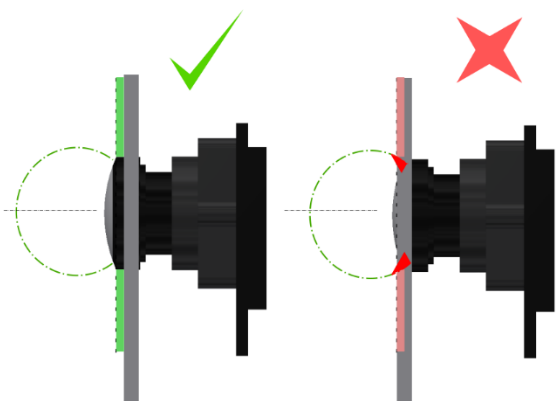


Figure 4 – Correct and incorrect protruding distance

Additionally:

- The CubeSense is less susceptible to noise from the reaction/momentum-wheels and typically the CubeSense sensors can be mounted in close proximity to them.
- CubeSense can mask interfering deployable structures in the nadir camera's field of view, but it is highly recommended to avoid having any obstructions as this will greatly decrease its detection capability.
- The connector needs to be epoxied in place after final integration of the satellite to ensure that it does not shake loose during launch.

5. Software Interface

CubeSense can communicate using either a UART channel or an I2C bus. In both cases the telecommand and telemetry definitions are the same, but the protocol differs.

CubeSense acts as a slave in both cases – it will only respond to telecommands or telemetry requests from a master.

The first byte of a message sent to the CubeSense will determine whether the message is a telecommand or telemetry request and also contain the ID of the telecommand or telemetry request. The most significant bit determines whether it is a telecommand or telemetry request, and the lower 7 bits contain the ID.

Table 4 – Content of first byte of message

Bit(s)	Meaning
7	0 = telecommand, 1 = telemetry request
0:6	Telecommand or telemetry frame ID

When considering the full byte identifier, telecommands' first byte is in the range 0-127 and telemetry requests in the range 128-255.

5.1 I2C Protocol

The CubeSense module acts as an I2C slave node with 7-bit addressing. The 8-bit read and write addresses of the node are:

Table 5 – I2C node address

	8-bit Hex byte	7-bit Hex address	Binary
I2C write	0x20	0x10	0b0010 000**0**
I2C read	0x21	0x10	0b0010 000**1**

5.1.1 Requesting telemetry

Telemetry is requested from CubeSense over the I2C bus by performing a combined write-read operation.

- The first write following the start condition is the write-address of the node (0x20).

- This is followed by the telemetry frame identifier which is the ID of the TLM that should be read.
- A repeated start condition is then given
- The read address is then written by the master (0x21)
- The master then issues several read cycles depending on the length of the telemetry frame.

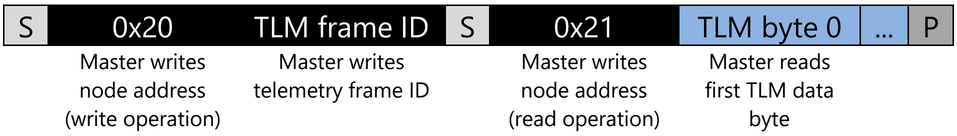


Figure 5 – I2C Telemetry request

The length and content of the telemetry frames are summarized in Table 10. For the complete listing of TLMs and TCs, see Appendices A and B.

Because the master determines the number of bytes that are read, it is possible to read past the end of a telemetry frame or to read an incomplete telemetry frame. CubeSense will flag an error if an incorrect number of bytes are read for a given frame identifier. This flag is stored in the *Communications Status* telemetry frame (ID = 2), and can be read using a telemetry request. The flag will remain set until the communication status telemetry frame is read.

5.1.2 Sending telecommands

Telecommands are given to CubeSense by performing a master write to the module.

- The first write following the start condition is the write-address of the node (0x20).
- The first data byte (after the address byte) is the telecommand identifier
- This is followed by the telecommand parameters.

Figure 6 – I2C Telecommand

The number and format of these parameters vary for each telecommand and are summarized in Table 11.

The telecommand acknowledge status can be polled via a telemetry request (ID = 3) to ensure that CubeSense successfully registered the telecommand that was given to it.

Table 6 – *Telecommand Acknowledge* telemetry frame

Telemetry frame ID	3				
Name	Telecommand Acknowledge				
Frame length (bytes)	3				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	1	Last TC id	Unsigned 8-bit	ID of last received telecommand
	1	1	Processed flag	Unsigned 8-bit	0 = TC has not been processed. Sending another TC while this flag is 0 will corrupt the TC buffer
	2	1	TC error flag	Unsigned 8-bit	0 = no error, 1 = invalid TC ID, 2 = parameters invalid

It is not a requirement that the telecommand acknowledge status has to be read following a telecommand, but an error will occur if another telecommand is sent before the *Telecommand Processed* flag (contained in the *Telecommand Acknowledge* telemetry frame) has been set. In this case the telecommand buffer will be overwritten, while the first telecommand is being processed, leading to corrupt telecommand data.

The *Processed* flag is not an indication of the telecommand execution status. Some telecommands may take a while to execute (such as imaging and detection functions) and other telemetry channels are available to monitor their execution status. The *Processed* flag is only an indication that the module is ready for another telecommand to be sent.

The *Telecommand Acknowledge* telemetry frame also contains a *TC Error* flag. This flag will be set if an invalid telecommand ID was received for the last telecommand or if the number of data bytes were incorrect or contained invalid data.

The following sequence illustrates the actions that the master must take to ensure proper telecommand execution:

1. Send telecommand.
2. Poll *Telecommand Acknowledge* telemetry until the *Processed*-flag equals 1.
3. Confirm telecommand validity by checking the *TC Error* flag of the last read *Telecommand Acknowledge* telemetry.
4. Back to step 1 (if another telecommand is to be sent).

5.2 UART protocol

The UART operates at a baudrate of 57600 bps, with 8 data bits, 1 stop bit and no parity.

The UART protocol makes use of start-of-message (SOM) and end-of-message (EOM) identifiers to mark the beginning and end of a transmission. An escape character precedes the SOM and EOM identifiers.

Table 7 – UART protocol message identifiers

Escape character	0x1F
SOM identifier	0x7F
EOM identifier	0xFF

A message will therefore begin with the sequence 0x1F, 0x7F and end with the sequence 0x1F, 0xFF. Whenever data occurs in the message, where the data byte matches the escape character, this will be replaced with the sequence 0x1F, 0x1F. When decoding a CubeSense UART message, on reception of the escape character, the byte following the escape character has the following implications:

Table 8 – UART message decoding

Byte received after escape character	Meaning
0x7F	Start of message
0xFF	End of message
0x1F	Data byte: 0x1F
other	Should not occur – error

When formatting a message to be sent to the CubeSense UART, the same protocol applies. CubeSense will set internal error flags to indicate that a protocol error occurred or if an incomplete message was received (if a SOM identifier occurred without a preceding EOM identifier). These flags can be read via the *Communication Status* telemetry request. Once set, they will remain set until the *Communication Status* telemetry is requested.

5.2.1 Requesting telemetry

A telemetry request to the CubeSense (via the UART) will have the following form:

0x1F	0x7F	TLM frame ID	0x1F	0xFF
Start-of-message identifier			End-of-message identifier	

Figure 7 – UART telemetry request

The reply from the CubeSense will then have the following form:

0x1F	0x7F	TLM byte 0	...	0x1F	0xFF
Start-of-message identifier				End-of-message identifier	

Figure 8 – UART telemetry reply

5.2.2 Sending telecommands

A telecommand to the CubeSense (via the UART) will have the following form:

0x1F	0x7F	TC ID	TC data byte 0	...	0x1F	0xFF
Start-of-message identifier					End-of-message identifier	

Figure 9 – UART telecommand

The CubeSense will reply to the telecommand with an acknowledge message.

0x1F	0x7F	TC Error flag	0x1F	0xFF
Start-of-message identifier		0 = no error 1 = invalid TC ID 2 = invalid parameters	End-of-message identifier	

Figure 10 – UART telecommand acknowledge

The reply will contain a single data byte with the *TC Error* flag. This is the same flag that can be read via the *Telecommand Acknowledge* telemetry request. The receipt of the telecommand-acknowledge will indicate that the CubeSense is ready to receive another telecommand. Sending another telecommand before the acknowledge will corrupt the telecommand buffer.

5.3 Telecommand and telemetry IDs

The first byte of a message sent to the CubeSense will determine whether the message is a telecommand or telemetry request, and also contain the ID of the telecommand or telemetry request. The most significant bit determines whether it is a telecommand or telemetry request, and the lower 7 bits contain the ID.

Table 9 – Telecommand and telemetry IDs

Bit(s)	Data
7	0 = telecommand, 1 = telemetry request
0:6	Telecommand or telemetry frame ID

When considering the full byte identifier, telecommands will have ID's in the range 0-127 and telemetry requests will have ID's in the range 128-255. Table 10 and Table 11 show lists of telemetry and telecommand frames that CubeSense accepts. For the complete listing of frames and their content, see the Node Definition.

Table 10 – List of telemetry frames**

ID (Decimal)	Full ID-Byte (HEX)	Telemetry frame	Length
0	0x80	Status	8
1	0x81	Serial Number	2
2	0x82	Communication Status	8
3	0x83	Telecommand Acknowledge	3
14	0x8E	Nadir Bad-Fit Threshold	2
15	0x8F	Nadir Angular Radius Threshold	2
16	0x90	Nadir Measured Angular Radius	1
19	0x93	Operation Status	1
20	0x94	Sensor result	6
22*	0x96	Sensor result & start new detection	6
26	0x9A	Power	6
40	0xA8	Configuration	7
64	0xC0	Image frame	128
65	0xC1	Image frame info	3
66	0xC2	Full image – SRAM Location 1 (UART only)	1048576
67	0xC3	Full image – SRAM Location 2 (UART only)	1048576
72	0xC8	Read Sensor Mask	40

* Telemetry frame 22 will also trigger a detection operation.

** This definition is for CubeSense Nodedef V3. If the 'Interface version' field of TLM 0 of your CubeSense is not V3, please contact our team for the updated list of TLMs.

Table 11 – List of telecommands*

ID (Decimal)	Full ID-Byte (HEX)	Telecommand	Length
0	0x00	Reset	1
11	0x0B	Clear SRAM overcurrent flags	0
14	0x0E	Nadir Bad-Fit Threshold	2
15	0x0F	Nadir Angular Radius Threshold	2
20	0x14	Capture & detect	0
21	0x15	Capture Image	1
40	0x28	Set sensor detection threshold	1
42	0x2A	Set sensor auto-adjust	1
43	0x2B	Set sensor settings	5
50	0x32	Set sensor boresight pixel location	4
52	0x34	Set sensor mask	9
54	0x36	Set sensor distortion correction coefficients	15
64	0x40	Initialize image download	2
65	0x41	Advance image download	2

* This definition is for CubeSense Interface V3. If the 'Interface version' field of TLM 0 of your CubeSense is not V3, please contact our team for the updated list of TCs.

6. Telecommand and Telemetry Frames

6.1 Telemetry Frames

Telemetry frame ID	0				
Name	Status				
Frame length (bytes)	8				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	1	Node type	Unsigned 8-bit	Identification of type of CubeComponent Node
	1	1	Interface version	Unsigned 8-bit	Interface definition version
	2	1	Firmware version (major)	Unsigned 8-bit	
	3	1	Firmware version (minor)	Unsigned 8-bit	
	4	2	Runtime (seconds)	Unsigned 16-bit	Number of seconds since processor start-up
	6	2	Runtime (milliseconds)	Unsigned 16-bit	Number of milliseconds (after the integer second) since processor start-up

Telemetry frame ID	1				
Name	Serial number				
Frame length (bytes)	2				
Channels	**Byte No**	**Length (bytes)**	**Channel**	**Data type**	**Detail**
	0	2	Serial number	Unsigned 16-bit	Number that defines the serial of CubeSense

Telemetry frame ID	2				
Name	Communication Status				
Frame length (bytes)	8				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	2	TC counter	Unsigned 16-bit	No. of telecommands received
	2	2	TLM counter	Unsigned 16-bit	No. of telemetry request received
	4	1	TC buffer overrun flag	Unsigned 8-bit	TC buffer was overrun while receiving a telecommand
	5	1	I2C TLM read error flag	Unsigned 8-bit	While reading a TLM buffer in an I2C transaction, either the read carried on past the end of the buffer, or the read stopped before all bytes were read
	6	1	UART protocol error flag	Unsigned 8-bit	UART protocol error occurred
	7	1	UART incomplete message flag	Unsigned 8-bit	UART start-of-message identifier was received without a preceding end-of-message

Telemetry frame ID	3				
Name	Telecommand Acknowledge				
Frame length (bytes)	3				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	1	Last TC id	Unsigned 8-bit	ID of last received telecommand
	1	1	Processed flag	Unsigned 8-bit	0 = TC has not been processed. Sending another TC while this flag is 0 will corrupt the TC buffer
	2	1	TC error flag	Unsigned 8-bit	0 = no error, 1 = invalid TC ID, 2 = parameters invalid

Telemetry frame ID	14				
Name	Nadir Bad-Fit Threshold				
Frame length (bytes)	2				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	1	Max Deviation Percentage	Unsigned 8-bit	Percentage = raw / 10 (e.g. 5 = 0.5%)
	1	1	Max Bad Edges	Unsigned 8-bit	Maximum number of edges

Telemetry frame ID	15				
Name	Nadir Angular Radius Threshold				
Frame length (bytes)	2				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	1	Max Radius	Unsigned 8-bit	Angle in degrees
	1	1	Min Radius	Unsigned 8-bit	Angle in degrees

Telemetry frame ID	16				
Name	Nadir Measured Angular Radius				
Frame length (bytes)	1				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	1	Angular Radius	Unsigned 8-bit	Angle in degrees

Telemetry frame ID	20 and 22				
Name	Detection result & Trigger				
Frame length (bytes)	6				
Channels	**Byte No**	**Length (bytes)**	**Channel**	**Data type**	**Detail**
	0	2	α	Signed 16-bit	α angle in centi-degrees (range = -100 to 100 degrees)
	2	2	β	Signed 16-bit	β angle in centi-degrees (range = -100 to 100 degrees)
	4	1	Capture Result	Unsigned 8-bit	0 = start-up 1 = capture pending 2 = successfully captured 4 = camera timeout 5 = SRAM overcurrent
	5	1	Detection result	Unsigned 8-bit	0 = start-up 1 = no detection scheduled 2 = detection pending 3 = Nadir error – too many detected edges 4 = Nadir error – not enough detected edges 5 = Nadir error – Bad fit 6 = Sun error – Sun not found 7 = Successful detection

Telemetry frame ID	26				
Name	Power				
Frame length (bytes)	10				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	2	3.3V current	Unsigned 16-bit	To obtain current from sample value: I= 0.208 * TLM_3V3CURRENT (returns current in mA)
	2	2	SRAM current	Unsigned 16-bit	I=0.208 * TLM_SRAMCURRENT (returns current in mA)
	4	1	3V3 over-current	Unsigned 8-bit	0 = no overcurrent 1 = 3V3 overcurrent detected
	5	1	SRAM over-current	Unsigned 8-bit	0 = no overcurrent 1 = SRAM overcurrent detected

Telemetry frame ID	40				
Name	Configuration				
Frame length (bytes)	7				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	1	Camera detection threshold	Unsigned 8-bit	
	1	1	Camera auto adjust mode	Unsigned 8-bit	0 = disabled, 1 = enabled
	2	1	Camera exposure	Unsigned 16-bit	
	4	1	Camera AGC	Unsigned 8-bit	
	5	1	Camera Blue Gain	Unsigned 8-bit	
	6	1	Camera Red Gain	Unsigned 8-bit	

Telemetry frame ID	64				
Name	Image frame				
Frame length (bytes)	128				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	128	Image bytes	Array of 8-bit unsigned	Image bytes

Telemetry frame ID	65				
Name	Image frame info				
Frame length (bytes)	3				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	2	Image frame number	Unsigned 16-bit	Number of current frame loaded into download buffer
	2	1	checksum	Unsigned 8-bit	XOR checksum of frame loaded into download buffer

Telemetry frame ID	66*				
Name	Full image – SRAM Location 1				
Frame length (bytes)	1048576				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	1048576	Image bytes	Array of 8-bit unsigned	Image bytes

Telemetry frame ID	67*				
Name	Full image – SRAM Location 2				
Frame length (bytes)	1048576				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	1048576	Image bytes	Array of 8-bit unsigned	Image bytes

* Telemetry 66 to 69 can only be acquired through UART.

Telemetry frame ID	72				
Name	Read Sensor Mask				
Frame length (bytes)	40				
Channels	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	2	Minimum X of area 1	Unsigned 16-bit	Specifies the lower X-boundary of masked area 1
	2	2	Maximum X of area 1	Unsigned 16-bit	Specifies the upper X-boundary of masked area 1
	4	2	Minimum Y of area 1	Unsigned 16-bit	Specifies the lower Y-boundary of masked area 1
	6	2	Maximum Y of area 1	Unsigned 16-bit	Specifies the upper Y-boundary of masked area 1
	8	2	Minimum X of area 2	Unsigned 16-bit	Specifies the lower X-boundary of masked area 2
	10	2	Maximum X of area 2	Unsigned 16-bit	Specifies the upper X-boundary of masked area 2
	12	2	Minimum Y of area 2	Unsigned 16-bit	Specifies the lower Y-boundary of masked area 2
	14	2	Maximum Y of area 2	Unsigned 16-bit	Specifies the upper Y-boundary of masked area 2
	16	2	Minimum X of area 3	Unsigned 16-bit	Specifies the lower X-boundary of masked area 3
	18	2	Maximum X of area 3	Unsigned 16-bit	Specifies the upper X-boundary of masked area 3
	20	2	Minimum Y of area 3	Unsigned 16-bit	Specifies the lower Y-boundary of masked area 3
	22	2	Maximum Y of area 3	Unsigned 16-bit	Specifies the upper Y-boundary of masked area 3
	24	2	Minimum X of area 4	Unsigned 16-bit	Specifies the lower X-boundary of masked area 4
	26	2	Maximum X of area 4	Unsigned 16-bit	Specifies the upper X-boundary of masked area 4

[redacted]	28	2	Minimum Y of area 4	Unsigned 16-bit	Specifies the lower Y-boundary of masked area 4
	30	2	Maximum Y of area 4	Unsigned 16-bit	Specifies the upper Y-boundary of masked area 4
	32	2	Minimum X of area 5	Unsigned 16-bit	Specifies the lower X-boundary of masked area 5
	34	2	Maximum X of area 5	Unsigned 16-bit	Specifies the upper X-boundary of masked area 5
	36	2	Minimum Y of area 5	Unsigned 16-bit	Specifies the lower Y-boundary of masked area 5
	38	2	Maximum Y of area 5	Unsigned 16-bit	Specifies the upper Y-boundary of masked area 5

6.2 Telecommand Frames

Telecommand ID	0				
Name	Reset				
Parameters length (bytes)	1				
Fields	**Offset**	**Length**	**Field**	**Data type**	**Detail**
	0	1	Reset type	Unsigned 8-bit	1 = reset communication interfaces (I2C & UART) including message counts 2 = reset camera 3 = reset MCU

Telecommand ID	11
Name	Clear SRAM overcurrent flag
Parameters length (bytes)	0

Telecommand ID	14				
Name	Nadir Bad-Fit Threshold				
Parameters length (bytes)	2				
Fields	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	1	Max Deviation Percentage	Unsigned 8-bit	Percentage = raw / 10 (e.g. 5 = 0.5%)
	1	1	Max Bad Edges	Unsigned 8-bit	Maximum number of edges

Telecommand ID	15				
Name	Nadir Angular Radius Threshold				
Parameters length (bytes)	2				
Fields	**Offset**	**Length**	**Channel**	**Data type**	**Detail**
	0	1	Max Radius	Unsigned 8-bit	Angle in degrees
	1	1	Min Radius	Unsigned 8-bit	Angle in degrees

Telecommand ID	20
Name	Image capture & detection
Parameters length (bytes)	0

Telecommand ID	21				
Name	Image capture				
Parameters length (bytes)	1				
Fields	**Offset**	**Length**	**Field**	**Data type**	**Detail**
	0	1	SRAM Location selection	Unsigned 8-bit	0 = Top Half 1 = Bottom Half

Telecommand ID	40				
Name	Set sensor detection threshold				
Parameters length (bytes)	1				
Fields	**Offset**	**Length**	**Field**	**Data type**	**Detail**
	0	1	Detection threshold	Unsigned 8-bit	Detection threshold

Telecommand ID	42				
Name	Set sensor exposure auto-adjust				
Parameters length (bytes)	1				
Fields	**Offset**	**Length**	**Field**	**Data type**	**Detail**
	0	1	Auto-adjust enabled	Unsigned 8-bit	0 = disabled, 1 = enabled

Telecommand ID	43				
Name	Set sensor settings				
Parameters length (bytes)	5				
Fields	**Offset**	**Length**	**Field**	**Data type**	**Detail**
	0	1	Exposure time	Unsigned 16-bit	Exposure register value
	2	1	AGC	Unsigned 8-bit	Gain control register
	3	1	Blue gain	Unsigned 8-bit	Blue gain control register
	4	1	Red gain	Unsigned 8-bit	Red gain control register

Telecommand ID	50				
Name	Set sensor boresight pixel location				
Parameters length (bytes)	4				
Fields	**Offset**	**Length**	**Field**	**Data type**	**Detail**
	0	2	X-Pixel	Unsigned 16-bit	100*(X Pixel location of CAM1 boresight)
	2	2	Y-Pixel	Unsigned 16-bit	100*(Y Pixel location of CAM1 boresight)

Telecommand ID	52				
Name	Set sensor mask				
Parameters length (bytes)	9				
Fields	**Offset**	**Length**	**Field**	**Data type**	**Detail**
	0	1	Mask Number	Unsigned 8-bit	Number indicating which of 5 (0-4) areas will be specified
	1	2	X Minimum	Unsigned 16-bit	Lower X limit of specified masked area
	3	2	X Maximum	Unsigned 16-bit	Upper X limit of specified masked area
	5	2	Y Minimum	Unsigned 16-bit	Lower Y limit of specified masked area
	7	2	Y Maximum	Unsigned 16-bit	Upper Y limit of specified masked area

Telecommand ID	54				
Name	Set sensor distortion correction coefficients				
Parameters length (bytes)	15				
Fields	**Offset**	**Length**	**Field**	**Data type**	**Detail**
	0	2	Mantissa1	Unsigned 16-bit	Mantissa of coefficient 1
	2	1	Exponent1	Unsigned 8-bit	Exponent of coefficient 1
	3	2	Mantissa2	Unsigned 16-bit	Mantissa of coefficient 2
	5	1	Exponent2	Unsigned 8-bit	Exponent of coefficient 2
	6	2	Mantissa3	Unsigned 16-bit	Mantissa of coefficient 3
	8	1	Exponent3	Unsigned 8-bit	Exponent of coefficient 3
	9	2	Mantissa4	Unsigned 16-bit	Mantissa of coefficient 4
	11	1	Exponent4	Unsigned 8-bit	Exponent of coefficient 4
	12	2	Mantissa5	Unsigned 16-bit	Mantissa of coefficient 5
	14	1	Exponent5	Unsigned 8-bit	Exponent of coefficient 5

Telecommand ID	64				
Name	Initialize image download				
Parameters length (bytes)	2				
Fields	**Offset**	**Length**	**Field**	**Data type**	**Detail**
	0	1	SRAM location	Unsigned 8-bit	0 = Top 1 = Bot
	1	1	Size selection	Unsigned 8-bit	0 = 1024x1024 (8192 frames) 1 = 512x512 (2048 frames) 2 = 256x256 (512 frames) 3 = 128 x 128 (128 frames) 4 = 64 x 64 (32 frames)

Telecommand ID	65				
Name	Advance image download				
Parameters length (bytes)	2				
Fields	**Offset**	**Length**	**Field**	**Data type**	**Detail**
	0	2	Next frame number	Unsigned 16-bit	Number of next frame to be loaded

7. Document Version History

Version	Person	Pages	Date	Description of Change
0.1	DS	ALL	07/02/2017	First draft
1.0	DS	ALL	10/08/2017	Major updates
1.1	DS	ALL	11/03/2019	Removed references to 5V
1.2	HW, DS	ALL	03/09/2019	Update for CubeSense V3
1.3	DS	ALL	06/01/2020	Updated TLMs and TCMDs
1.4	DS	4	04/11/2022	Corrected specifications