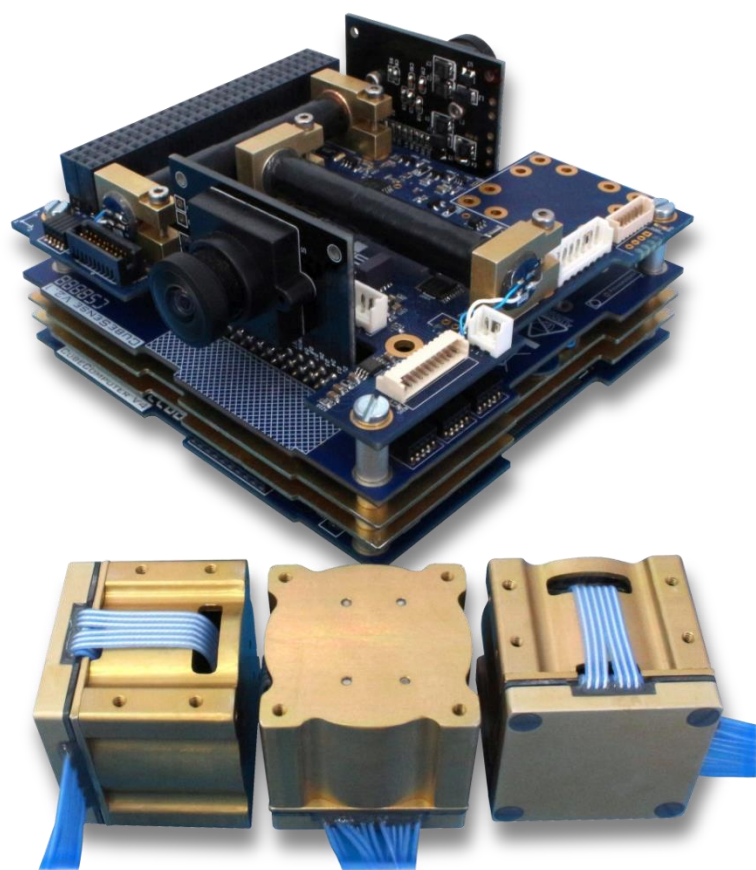




CUBEADCS

THE COMPLETE ADCS SOLUTION



USER MANUAL

ESL
Electronic Systems
Laboratory

Contact Us


Phone : +27 21 808 9499
E-mail : info@cubespace.co.za
Web : www.cubespace.co.za
Facebook : /CubeSpaceADCS
Twitter : @CubeSpace_ADcs

Physical Address

CubeSpace
The LaunchLab
Hammanhand Road
Stellenbosch, 7600
South Africa


UNIVERSITEIT
STELLENBOSCH
UNIVERSITY



| | |
|----------------------|--|
| Document | CubeADCS User Manual |
| Version | 3.07 |
| Domain | Public |
| Date modified | 29 March 2018 |
| Approved by | Name: Mike-Alec Kearney |
| | Signature:  |

Document Version History

| Version | Responsible person(s) | Pages | Date | Description of change |
|---------|-----------------------|--------|------------|---|
| 3.0 | MK | ALL | 20/03/2017 | V3 First draft |
| 3.01 | J | 13, 14 | 23/03/2017 | Adding epoxy note of connectors |
| 3.02 | MK | ALL | 22/06/2017 | Merge Y mom and 3-axis |
| 3.03 | CG | 37-41 | 04/07/2017 | CubeStar Configuration |
| 3.04 | WHS | ALL | 12/07/2017 | Update Control Modes, Magnetometer calibration, Rate sensor range |
| 3.05 | CJG | ALL | 24/07/2017 | Formatting and version number correction. |
| 3.06 | MK | ALL | 27/07/2017 | Accept/Reject changes |
| 3.07 | CCH | All | 29/03/2017 | General language editing |

List of Acronyms/Abbreviations

| | |
|------------------|---|
| ACP | ADCS Control Program |
| ADCS | Attitude Determination and Control System |
| CSS | Coarse Sun Sensor |
| ESD | Electrostatic Discharge |
| I ² C | Inter-Integrated Circuit |
| MCU | Microcontroller Unit |
| MEMS | Microelectromechanical System |
| OBC | Onboard Computer |
| PCB | Printed Circuit Board |
| RTC | Real-Time Clock |
| SBC | Satellite Body Coordinate |
| SPI | Serial Peripheral Interface |
| TC | Telecommand |
| TLM | Telemetry |
| UART | Universal Asynchronous Receiver/Transmitter |

Relevant reference documents

This document is to be used in combination with the following documents:

| Reference | Document name | Document Version |
|--------------|---------------------------------|------------------------------------|
| Ref 1 | CubeADCS - ICD | V3.0 or higher |
| Ref 2 | CubeADCS – Option Sheet | Completed by user at order of unit |
| Ref 3 | CubeADCS – Health Check | V3.0 or higher |
| Ref 4 | CubeADCS – Reference Manual | V3.0 or higher |
| Ref 5 | CubeADCS – CubeSupport Manual | V3.0 or higher |
| Ref 6 | CubeADCS – Configuration Sheet | V3.0 or higher |
| Ref 7 | CubeADCS – Commissioning Manual | V3.0 or higher |
| Ref 8 | CubeADCS – Library Manual | V3.0 or higher |

Table of Contents

| | | |
|----------|--|-----------|
| 1 | Introduction..... | 6 |
| 2 | Handling..... | 7 |
| 3 | Getting Started | 8 |
| 3.1 | Unpacking and setting up the unit | 8 |
| 3.2 | Powering the unit | 9 |
| 3.3 | Performing health check..... | 9 |
| 3.4 | Firmware Uploads..... | 10 |
| 4 | Interfacing with the unit | 11 |
| 4.1 | Connecting with OBC | 11 |
| 4.2 | Mounting in satellite..... | 12 |
| 4.3 | ADCS System description | 16 |
| 4.4 | ADCS operational modes | 19 |
| 5 | Usage | 27 |
| 5.1 | Power Management | 27 |
| 5.2 | Bootloader..... | 27 |
| 5.3 | Firmware Management | 30 |
| 5.4 | Configuration and setup | 30 |
| 5.5 | Status and error flags..... | 45 |
| 5.6 | Logging..... | 45 |
| 5.7 | Magnetometer boom deployment..... | 46 |
| 5.8 | Image Downloads..... | 46 |
| 5.9 | File Downloads | 46 |
| 5.10 | GPS Measurements..... | 52 |
| 6 | Special considerations | 53 |
| 6.1 | FOV of sensors..... | 53 |
| 6.2 | Magnetic loops and solar panels..... | 53 |
| 6.3 | Configuration and mounting | 54 |
| 6.4 | Deployable magnetometer | 56 |
| 6.5 | Series resistance of power supply & inrush currents..... | 56 |
| 6.6 | Y-axis MOI not the largest for Y-Thomson control | 57 |

1 Introduction

The purpose of this document is to provide the user with:

- instructions of completing the initial setup of the CubeADCS unit for the health check
- instructions for connecting the unit to an OBC
- a guide to mounting sensors inside a satellite and calculating mounting transforms.
- a functional description of the control modes that the CubeADCS unit is capable of executing
- general usage instructions for all major functionality
- an overview of the hardware included in this bundle and their performance

The document is to be used in combination with the **Ref 1** and **Ref 4** to: design the software for interfacing with the ADCS unit; connect the unit electrically and mechanically to the rest of the satellite; configure the unit correctly; and plan the ADCS operations once in space.

A CubeADCS can be configured in various ways with the system having more or less sensors and actuators to accommodate different mission ADCS objectives. This document is applicable to all CubeADCS configurations in general. Certain sections are only applicable to a CubeADCS that contains the relevant sensors or actuators. The user's discretion is required to determine whether or not a specific section is relevant to the CubeADCS under investigation.

2 Handling



Anti-static

The bundle contains a variety of static sensitive devices. The appropriate electrostatic discharge (ESD) protection measures must therefore be implemented. The unit must never be handled without proper grounding.



Cleanliness

It is recommended that the CubeADCS unit be handled in a clean environment. Therefore, an appropriate laminar flow workbench or clean room of ISO class 8 or better, should be used.



Moisture

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



Shock

The unit must be handled with care. Dropping or bumping the unit should be avoided completely.



Camera lens cleanliness

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



Camera lens structural integrity

The camera sensors are aligned in parallel with the CubeSense PCB. This is important as misalignment of the cameras influence the performance of the system. External forces on the camera modules should therefore be avoided entirely.



Camera lens covers

The Sun and nadir optics are fitted with dust caps which should be removed before flight.



Momentum wheel

The aluminium housing of the momentum wheel should NOT be tampered with. Tampering with the housing may damage the wheel. No attempt should be made to loosen or remove the fasteners that secure the housing.

3 Getting Started

3.1 Unpacking and setting up the unit

The ADCS unit will be packaged in an anti-static bag inside a Peli Case. Remove the unit inside a clean environment using gloves. Use a sturdy workbench that is covered with a clean anti-static mat. Any person working on the unit must be grounded with an anti-static wristband.



Magnetometer measurements will be distorted if a workbench containing ferrous parts, such as iron; steel; stainless steel; etc, is used. Working on such a surface will have a dramatic effect on sensor measurements and may cause observed results to differ significantly from reference results.

The Peli Case will also contain the following components:

- Deployable magnetometer
- Redundant magnetometer (optional)
- Coarse Sun sensors
- GPS (optional)
- CubeStar (optional)
- CubeConnect and CubeWheels
- UART-to-USB cable
- Flash drive

Remove all parts from their anti-static bags and place them on the bench.



Take care to avoid touching the optics of any of the camera-based sensors.

Next, connect all peripherals to the ADCS stack - refer to **Figure 1**.

1. Connect the GPS to the GPS header on top of the ADCS stack.
2. Connect the redundant magnetometer.
3. Connect the deployable magnetometer using the in-line round connector.
4. Connect the coarse Sun sensor in-line connector to the ADCS stack and connect each photodiode to the in-line connector.

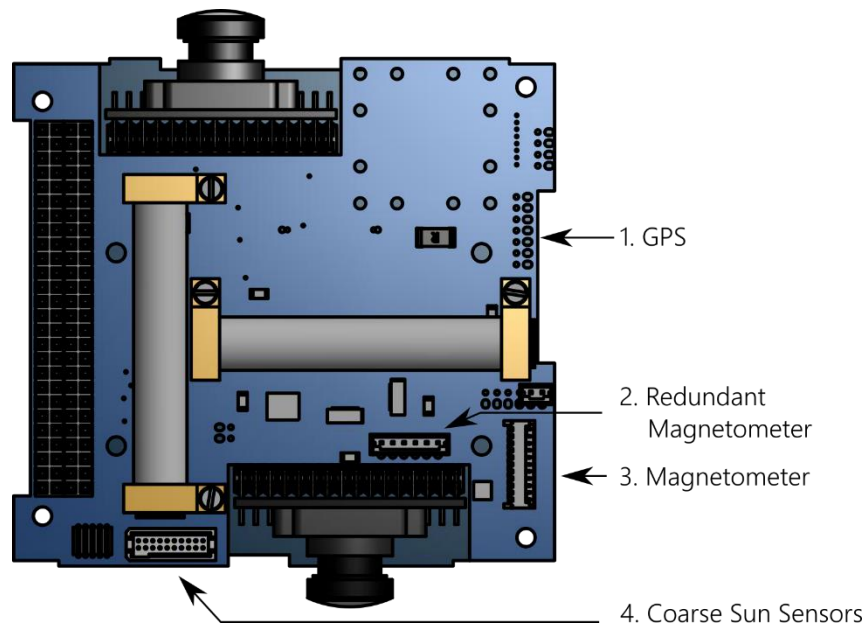


Figure 1 – Peripheral connectors (configuration may differ depending on options).

After all the parts of the ADCS are connected, proceed to powering the unit.

3.2 Powering the unit

A bench power supply is required to power the ADCS stack. Ensure that this supply is set to the battery voltage that the satellite will operate at and that the current limit is set to at least 1 A. This current limit should be high enough to supply the inrush currents that components such as the GPS (optional) or CubeSense draws upon start-up. Try to keep cables between the power supply and the ADCS short and thick, to try to reduce series resistance.



A current limit set too low or a series resistance set too high in the power connection can cause the system to experience voltage dips when powering up. This may lead to unexpected behaviour and/or resets.

Refer to **Ref 1** and **Ref 2** for the relevant PC104 pin locations when connecting power to the stack. It is preferable to use a voltage regulator to supply the 3.3 V and 5 V sources.

3.3 Performing health check

Connect the provided UART-to-USB cable to the CubeComputer as shown in **Figure 2**.



The black wire (ground) should be connected to the pin closest to the corner of the PCB farthest away from the PC104 connector

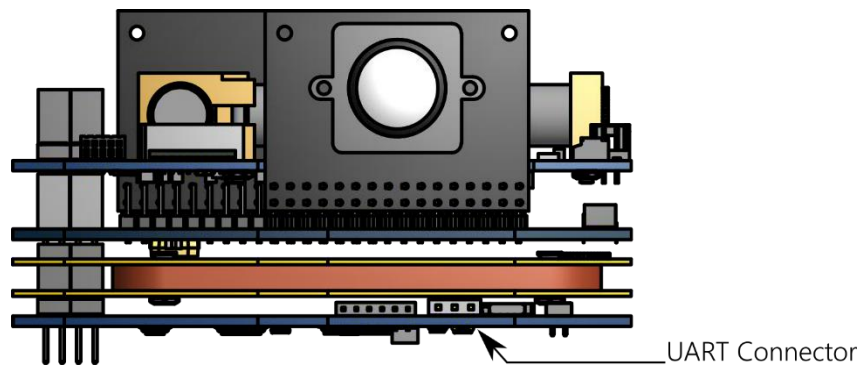


Figure 2 – UART connector (configuration may differ depending on options).

The health check can now be performed as described in **Ref 3**.

3.4 Firmware Uploads

CubeADCS units are delivered with a generic version of the ACP firmware that is not yet optimised for a specific mission. Typically, the completed health check is returned to the CubeSpace team for review. After the health check has been reviewed and all configuration data have been gathered from the client, a new firmware file that must be uploaded to the ADCS unit, is provided.

New firmware can be uploaded through the CubeADCS interfacing program, CubeSupport. The process is described in **Ref 5**.

4 Interfacing with the unit

4.1 Connecting with OBC

CubeADCS units have the option of UART, I2C and CAN as communication bus. Refer to **Ref 1** and **Ref 2** for the relevant PC104 pin locations of the communication bus to the OBC. For each communication bus, the telecommand and telemetry definitions are the same, but the protocol differs slightly.

Table 1 displays the meaning of the first byte of a message sent to the stack. This byte will determine whether the message is a telecommand or telemetry request and will also contain the ID of the telecommand or telemetry request. The first seven bits contain the ID. The most significant bit is the last one, and it determines whether it is a telecommand or telemetry request.

Table 1 – 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 will be in the range of 0 to 127 and telemetry requests in the range of 128 to 255.

Please refer to **Ref 1** for the details of the communications protocol for each bus.

4.2 Mounting in satellite

Placement of the ADCS actuators or sensors in the satellite has a significant influence on the ADCS performance of the system. In some cases, bad placement can even cause the ADCS to completely fail. The following sections will provide guidelines for the layout of the various components of the CubeADCS in the satellite structure.

4.2.1 Magnetometer

The magnetometer is the part of the ADCS that is most sensitive to bad placement. The magnetometer measures the Earth's magnetic field very accurately, and any magnetic disturbances near the magnetometer directly influence the accuracy of the measurements. For this reason, the magnetometer should not be placed in close proximity of any other part of the satellite that:

- **Generates large magnetic fields** - the most common source of such generated magnetic fields is any part that draws large currents.
- **Is ferrous** - ferrous components warp the magnetic field. Even though the ferrous components are not a source of magnetic field itself, they warp the Earth's field and cause an inaccurate measurement on the magnetometer. Further, these parts can be magnetised by the magnetic torquer rods. In this case the ferrous part keeps a magnetic dipole even after the torquer rod is switched off.

Common parts of the satellite that causes significant disturbances on the magnetometer include:

- **Stainless steel** screws or structural parts – use austenitic types of SS if required.
- **Solar panels** and **solar panel harnesses**.
- Any other **harness carrying large currents**.
- **Electrical motors** - motors have permanent magnets that are of considerable strength.
- **Antennas** - significant magnetic fields are generated specifically at the point where the antennas are fed.

As a rule of thumb, the magnetometer should ideally be kept 80+mm from any of the above-mentioned items.

Another important consideration with regards to the **mounting of the deployable magnetometer is to avoid interference with other sensors**. Fine Sun sensors, nadir sensors, or star trackers typically have a relatively large field of view. If they are placed incorrectly, the magnetometer **can obstruct the view of these sensors after being deployed**.

The magnetometer cable **must not** be routed inside the satellite a manner such that the cable **is tight** where it exits from the magnetometer. **The cable must exit perpendicular to the mounting facet for 1 cm, it should have 2 cm slack and must be loose (movable with slight touch).** See Figure 1Figure 3 for the correct illustration. Failure to conform may influence magnetometer deployment significantly.



Figure 3 – Magnetometer cable slack.

4.2.2 Coarse Sun Sensors

The coarse Sun sensors should be mounted on the external surfaces of the satellite panels. It is necessary to ensure that these sensors are not shadowed by other deployable structures. Ten coarse Sun sensors are supplied. The four extra photodiodes can be placed on any faces, bearing in mind that shadowing is to be minimised. The sensors should be glued down using any space grade epoxy.



Ensure that the photodiodes are kept clear of epoxy.

If solar panels already have photodiodes mounted on them, please contact the CubeSpace team to discuss the possibility of pairing these diodes with the CubeADCS.

The connectors on CubeControl need to be epoxied prior to final integration.

4.2.3 Wheels

CubeWheels are covered in MuMetal to shield the rest of the satellite from the strong magnets inside the electrical motors of the wheel. This shielding significantly warps the magnetic field in close proximity, therefore it should not be mounted within 4 cm of the magnetorquers or the magnetometer.

Even though all CubeWheels are balanced, they still have some residual imbalance that will cause vibrations. Typically, reaction wheel vibrations may cause problems in missions where imaging payloads with narrow FOV are used - the star tracker is a good example. In this case it is important not to mount the wheel directly on the same mounting surface that the camera

is mounted. Some structural separation between the wheels and the camera payload would allow for damping of the vibrations through the structure.

4.2.4 Star Tracker

As mentioned before, the star tracker should not be mounted on the same surface as the wheels.

The star tracker should be placed in such a way that it will point away from the Sun and Earth in the nominal flying orientation. Deployable structures should also not impede the view of the camera. It is recommended that CubeSpace is always contacted to discuss the star tracker mounting, and the possibility of a baffle to reduce stray light.

4.2.5 CubeSense

CubeSense is less susceptible to noise from the CubeWheels and typically the CubeSense sensors can be mounted in close proximity to the wheels.

The CubeSense sensors must be mounted in an orientation where the nadir camera is pointing nadir during nominal satellite operation. The Sun camera should typically point zenith, or orbit normal towards the Sun's side of the orbit. The exact placement of the Sun sensor would depend on the orbit the satellite is launched in.

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 Sun sensor is more robust against obstructions and does not have the masking capability. However, it is necessary to ensure that there are no surfaces that will reflect direct sunlight to the Sun sensor, as this may cause false measurements.

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 cameras have 180-degree fisheye lenses, therefore if the lens does not protrude all the way through the side panel, the sensor will detect reflections from the side panels.

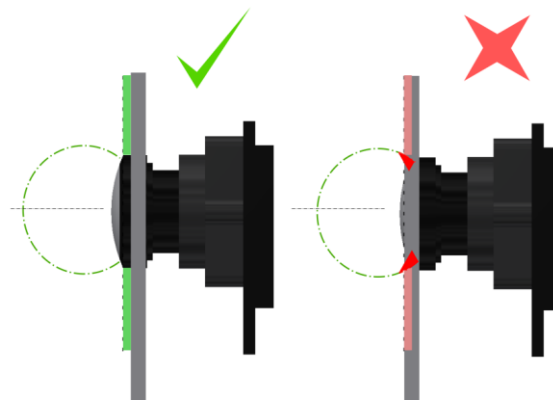


Figure 4 – Correct and incorrect CubeSense camera mounting.

4.2.6 Magnetorquers

Magnetorquers generate a magnetic dipole that interacts with Earth's magnetic field to generate torque. This generated magnetic field can be distorted by any ferrous objects on the satellite. Therefore, it is advised that completely non-ferrous metals are used for all mechanical parts. If any satellite parts contain stainless steel (typically in fasteners), the austenitic types of stainless steel should be used.

Reaction wheels contain relatively large magnets and magnetic shield material which both influence the magnetic field in their proximity. The magnetorquers and reaction wheels should therefore be separated.



Please contact the CubeSpace team when unsure of any mounting or layout of the ADCS components.

4.2.7 Epoxy of connector cables

It is recommended to epoxy all connector cables to peripheral components prior to final integration.

4.3 ADCS System description

4.3.1 System diagram

To estimate the attitude of the satellite, the CubeADCS unit uses, depending on the options, all or a sub-set of the following sensor measurements:

- Magnetometer, Coarse Sun sensors, MEMS rate sensor(s)
- Fine Sun sensor, Fine Earth sensor
- Star tracker

Estimation and control algorithms are run on the ADCS on-board computing unit that is included in the CubeADCS.

It uses magnetorquers and, if applicable, one momentum wheel or three reaction wheels to actuate the satellite's attitude.

The CubeADCS unit consists of up to four integrated PC104-standard PCBs and several peripheral components, which are to be mounted separately. A basic diagram of a complete CubeADCS solution with all options is shown in **Figure 5**.

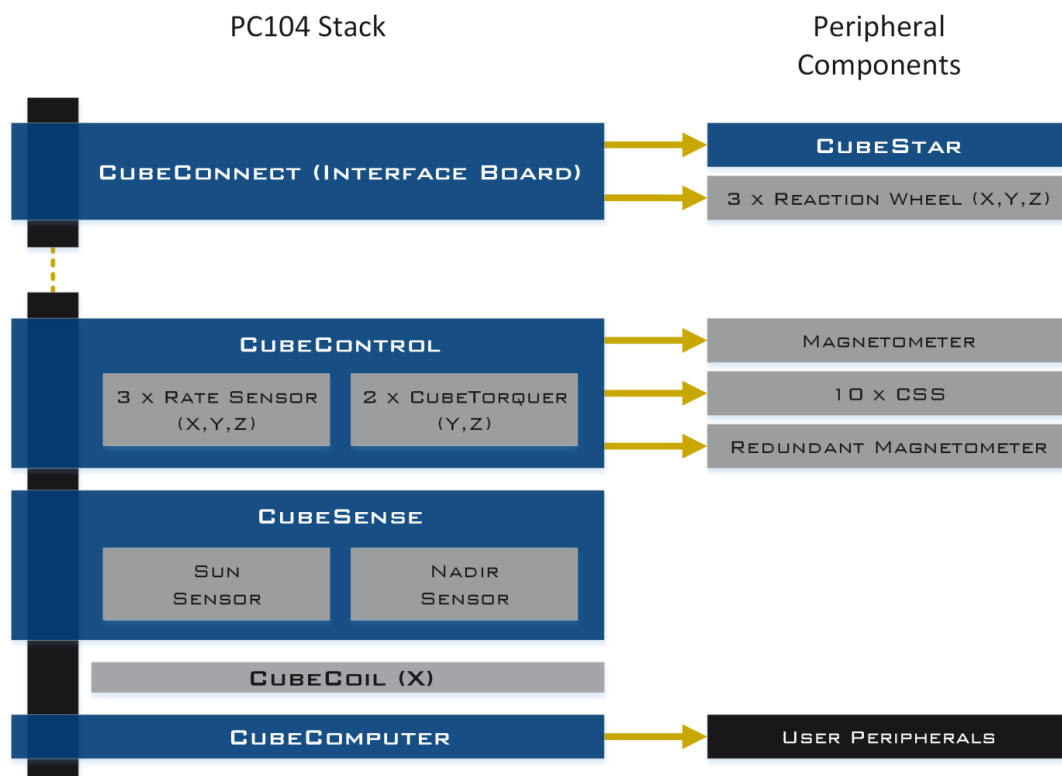
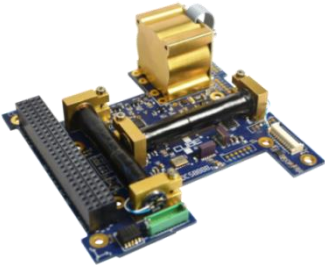
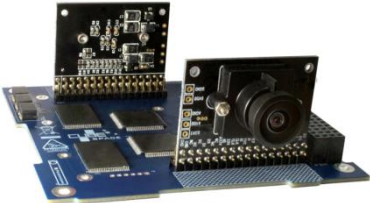

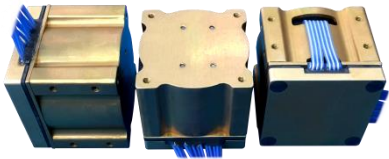



Figure 5 – ADCS stack.

Note that the reaction wheels can be selected as small, medium or large. Small CubeWheels can be mounted on CubeConnect in any axis, whereas medium and large CubeWheels must be mounted separately from the CubeADCS bundle.

A brief description of the core components of the CubeADCS 3-Axis is given in Table 2.

Table 2 – Modules of the CubeADCS solution.

| Module | Description |
|--|--|
| CubeControl  | <p>CubeControl is an actuator and sensor interface module for nanosatellites with advanced attitude control requirements. It can control 3 magnetorquers and a momentum wheel. CubeControl can also interface with 2 magnetometers, 10 coarse Sun sensors, and 3 MEMS rate sensors.</p> <p>NB: The momentum wheel mounted on CubeControl is not included in the CubeADCS 3-Axis bundle.</p> |
| CubeSense  | <p>CubeSense is an integrated Sun and nadir sensor for attitude sensing. It makes use of two CMOS cameras – one dedicated to Sun sensing and another for horizon detection. The Sun sensor has a neutral density filter included in the optics. Both cameras have a field of view (FOV) of at least 160°.</p> |
| CubeComputer  | <p>CubeComputer is a generic CubeSat OBC. It can perform the required ADCS functions, as well as the satellite's main OBC tasks. The module is based on ARM Cortex-M3 architecture and implements error detection and correction (EDAC) techniques.</p> |
| CubeWheel  | <p>CubeWheel is a compact standalone reaction wheel unit for nanosatellites. It provides the satellite with the ability to achieve 3-axis stability and 3-axis control. Each CubeWheel is magnetically shielded and is mountable in 3 axes. Various sizes are available to suit every need.</p> |
| CubeStar  | <p>CubeStar is a compact star tracker for nanosatellites. The module is based on ARM Cortex-M3 architecture and has an exceptionally low power consumption.</p> |

4.3.2 Coordinate Frame

The ADCS body and orbital coordinate system definitions used by the CubeADCS bundle is shown in Figure 6 and Figure 7. The ADCS coordinate system is related to the satellite body coordinate (SBC) system through a transformation matrix. When the ADCS is controlling the attitude to zero roll, pitch, and yaw angles, the SBC system will coincide with the orbit coordinate system (referred to as the nominal orientation).

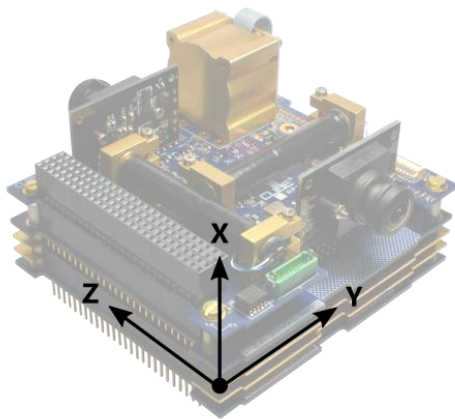


Figure 6 – ADCS coordinate system.

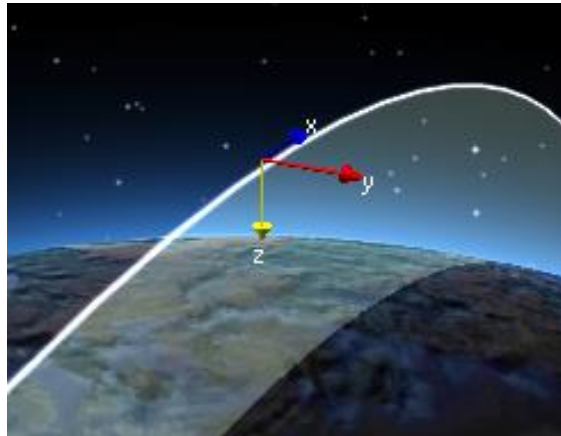


Figure 7 – Orbit coordinate system.

4.3.3 CubeComputer software options

The computer in the CubeADCS unit can either run a standalone ADCS control program (ACP) or it can run an unlocked version of the ACP that allows for the addition of custom user functionality. The two options are illustrated in Figure 8.

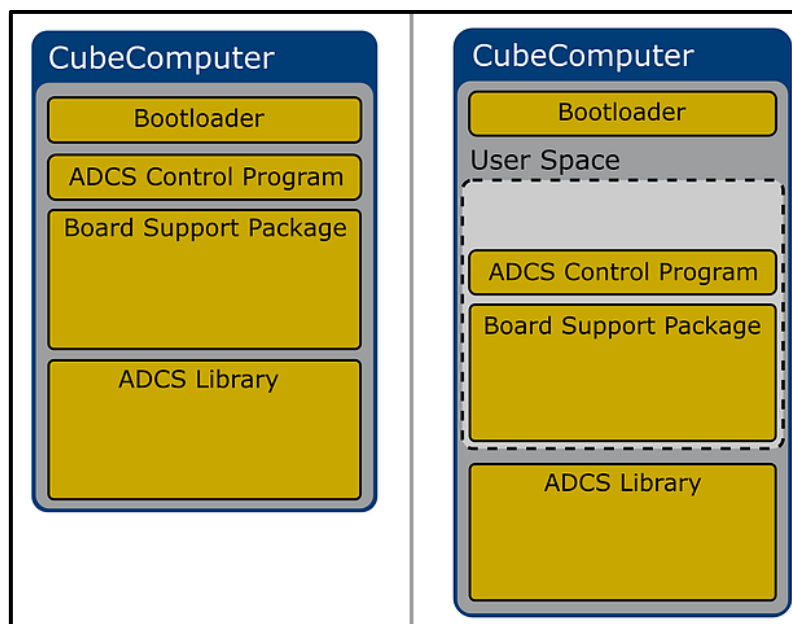


Figure 8 – CubeComputer software options.

In the standalone ACP case, the CubeComputer is commanded by a main OBC. The ADCS is controlled by sending telemetries and telecommands from the main OBC through one of the communication busses (I2C, UART or CAN). The complete listing of TCs and TLMs are shown in **Ref 4**.

In the other case, where the CubeComputer is unlocked for user development, the ADCS is controlled by making function calls to the compiled ADCS library. In this case the computer is typically used as ADCS as well as main OBC in the satellite. For more information on the ADCS library's functions, see **Ref 8**.

4.4 ADCS operational modes

Whether or not the CubeComputer is running the standalone or unlocked ACP, the ADCS has modes that determine what estimators and control algorithms are run, and at which frequency they are executed. This section gives a description of the different mode-settings of the ACP, as well as the options for each of these modes.

4.4.1 ADCS Run Modes

The *ADCS Run Mode (ADCS State)* is an enumeration that specifies how the ACP control loop behaves.

- In the *Off* state, the loop does not execute. The ACP is idle but still responds to commands and telemetry requests.
- In the *Enabled* state the control loop will execute with a 1s period.
- In the *Triggered* state, single iterations of the control loop are executed only when a specific command is received. On receiving a *Trigger ADCS Loop* command (see **Ref 4**), one iteration of the control steps execute and the processor will remain idle until another *Trigger* telecommand. This mode is typically used for HIL setups.

This mode only applies to the ADCS control loop. All other functions of the ACP remain active and it will respond to commands and telemetry requests.

4.4.2 ADCS Estimation Modes

The *Estimation Mode* determines which sensor measurements are used and what information is estimated. The available estimation modes are shown in Table 3, as well as which sensors are used in which mode.

Table 3 – Estimation modes.

| Enumeration Value | Estimation Mode | Sensors used | Estimated information |
|-------------------|---|---|--|
| 0 | No attitude estimation | None | None |
| 1 | MEMS rate sensing | X,Y,Z –axis rate sensors | X,Y,Z angular rates |
| 2 | Magnetometer rate filter | Magnetometer | X,Y,Z angular rates |
| 3 | Magnetometer rate filter with pitch angle estimator | Magnetometer | X,Y,Z angular rates; pitch angle |
| 4 | Magnetometer and fine or coarse Sun TRIAD algorithm | Magnetometer; CSS and/or Sun Sensor | Roll, Pitch & Yaw angles; X,Y,Z angular rates |
| 5 | Full State EKF | Magnetometer; Sun Sensor; Nadir Sensor; Star Tracker | Roll, Pitch & Yaw angles; X,Y,Z angular rates |
| 6 | MEMS Gyro EKF | X,Y,Z–axis rate sensors; Magnetometer; Sun Sensor; Nadir Sensor; Star Tracker | Roll, pitch & Yaw angles; X,Y,Z angular rates; X,Y,Z gyro bias |

Estimation Mode 1 does not perform any estimation algorithms, but rather uses the rate sensor measurement directly. It provides angular rate measurements for all the body axes. The rate sensor bias is not estimated, but simply compensated for using a temperature dependant offset. The constant offset can be changed by the *Set Rate Sensor Configuration telecommand (TC ID 36)* and is also included in the *ADCS configuration block telecommand (TC ID 20)*.

Estimation Mode 2 makes use of successive magnetometer measurements in a robust Kalman filter to estimate the X, Y and Z angular rates. The system noise covariance used in the filter can be changed by the *Set Estimation Parameters 1 telecommand (TC ID 43)*.

Estimation Modes 3 to 6 provide both angular rates and attitude angle estimates, and they make use of modelled vectors for the magnetic field and/or Sun. To calculate these modelled vectors, the orbital position and velocity of the satellite must be known. The latter information is obtained from an SGP4 orbit propagator. ***The mean orbital elements used by the propagator must be updated regularly to ensure that the estimation errors remain low.*** Orbital elements are updated by sending a *Set SGP4 Orbit Parameters* telecommand (TC ID 45) to the ACP. The orbit parameters can be stored in flash memory by the *Save Orbit Parameters telecommand (TC ID 64)*.

Estimation Mode 3 is the same as Mode 2, but also includes simple pitch estimation from magnetometer measurements only.

Estimation Mode 4 combines the magnetometer rate filter of mode 2 with a simple TRIAD algorithm. The latter gives the estimated attitude angles based on matching modelled and measured vectors from the coarse or fine Sun sensor and magnetometer. This mode will not yield attitude angle estimates in eclipse, since the Sun sensor will not give a valid output. The TRIAD estimation mode can be used to initialize the EKF state vector so that a faster EKF convergence time is achieved (this, however, is not a requirement).

Estimation Mode 5 uses an Extended Kalman Filter (EKF) to estimate the full attitude state. The system and measurement noise covariance used in the filter can be changed by *Set Estimation Parameters 1 and 2 telecommands (TC IDs 43 & 44)*. The Kalman filter estimators make use of the satellite moment of inertia tensor, and estimated control torque in order to propagate the attitude state (the moment of inertia and control torque is used in the Euler dynamics equation). ***The satellite moment of inertia tensor should ideally be measured before flight and is part of the configuration block.*** The *Set Moments of Inertia* telecommand (TC ID 41) and *Set Products of Inertia* telecommand (TC ID 42) can be sent to update it. The control torque used in the propagation step is calculated from the actuator commands and model parameters for the actuators. The latter are also configuration parameters modifiable via *Set Magnetorquer Configuration* telecommand (TC ID 20) and *Set Wheel Configuration* telecommand (TC ID 21).

Estimation Mode 6 uses an EKF with the MEMS rate (gyro) measurements to estimate the attitude and rate sensor offsets (gyro bias). As before, the measurement noise covariance can be changed with the *Set Estimation Parameters 1 and 2 telecommands (TC IDs 43 & 44)*. This EKF does not require the satellite moment of inertia tensor, control or disturbance torques as the angular rates are directly measured and the Euler dynamic model is not required. This EKF is therefore more robust against un-modelled external disturbance torques and actuator modelling errors.

The current estimation mode is changed by *Set Attitude Estimation Mode telecommand (TC ID 14)*.

4.4.3 ADCS Control Stages and Control Modes

The ADCS has three distinct control stages namely *Detumbling*, *Y-momentum* and, *XYZ-Wheel* control. In each stage, different control modes and estimation modes are used.

The first stage, *Detumbling*, is used to recover from any initial tumble condition and place the satellite in a slow, stable and known tumbling motion - the so-called Y-Thomson or Z-Thomson spin. In this stage the satellite will end-up spinning only about the body Y/Z-axis and the spin axis will align itself with the orbit normal or Sun vector direction.

The second control stage, *Y-momentum control*, can only be activated once the satellite is in a stable Y-Thomson tumbling state. In *Y-momentum* control the satellite will stop spinning and stabilize to the nominal orientation (zero roll, pitch and yaw angles). During *Y-momentum* stabilization the pitch angle may be controlled to a specific pitch reference value using the *Set commanded Attitude Angles telecommand (TC ID 15)*.

The third control stage, *XYZ-wheel control* uses 3-axis reaction wheels with zero momentum bias. This stage has a control mode to track the nadir vector (zero roll, pitch and yaw angles) or constant attitude reference angles which are set using the *Set commanded Attitude Angles telecommand (TC ID 15)*. It also has control modes to track the Sun vector with the satellite's solar panels for maximum power or to track a reference target at a specific location on earth. The target must first be selected with the *Set Tracking Controller Target Reference telecommand (TC ID 55)*.

4.4.3.1 [Detumbling](#)

The *Detumbling* stage can make use of 4 control modes.

- Control Mode 1 uses a *Bdot Detumbling* magnetic controller to dump the X- and Z-body rates. Only instantaneous magnetometer measurements are required and only the Y-axis magnetorquer is used.
- Control Mode 2 adds *Y-Thomson Spin* to Mode 1 using the X- and Z-axis magnetorquers. It requires an estimate for the Y-angular rate. Thus, either Estimation Mode 1 or 2 should be active. Control Mode 1 and 2 can only detumble the satellite from angular rates below 30 deg/s. For higher initial rates, Control Mode 3 or 4 can be used.
- Control Mode 8 is also a magnetic controller running at 1 Hz using all magnetorquers and instantaneous magnetometer measurements. Mode 8 can detumble initial body rates in all axes as high as 100 deg/s down to zero.

- Control Mode 9 is similar to Mode 8, but is implemented in the CubeControl subsystem at 10 Hz. Mode 9 can detumble initial body rates in all axes as high as 1000 deg/s down to zero. For body rates above 35 deg/s the Magnetometer Rate Filter (Estimation Mode 2) will not be accurate and the MEMS rate sensors (Estimation Mode 1) should be used. The range of the MEMS rate sensors must be configured with the *Set Rate Sensor Configuration* telecommand (TC ID 36) depending on the expected maximum body rate. For example, a *RateSensorMult* value equals 1 for a maximum rate of ± 75 deg/s, the value equals 12 for a maximum rate of ± 900 deg/s, etc.

The *Set Detumbling Control Parameters telecommand (TC ID 38)* can be used to set the Control Mode 1 to 3, 8 and 9 controller gains and reference spin rate ($\omega_{y,ref}$).

4.4.3.2 [Y-Momentum](#)

The *Y-momentum* stage requires pitch angle and body rate estimates, thus either the magnetometer rate filter with pitch estimation (Estimation Mode 3) or the Full State EKF (Estimation Mode 5) or the MEMS Gyro EKF (Estimation Mode 6) must be active. The *Y-momentum* stage contains two separate control modes. In the initial mode (Control Mode 3), the Y-wheel speed will be ramped up to an initial reference momentum. This should absorb most of the body Y-Thomson spin, but keep a slow pitch rotation about the Y-axis. This will continue until the estimated pitch angle is within 25 degrees of the reference pitch angle. At this point the steady-state Y-momentum control mode (Control Mode 4) will be entered, which uses a PD controller to control the pitch angle to the reference pitch angle. This transition will occur automatically on the ACP and reading the *Current ADCS State telemetry (TLM ID 145)* will confirm if the transition has occurred.

In the steady-state Y-momentum mode (Control Mode 4) the magnetorquers are used to maintain the wheel momentum to a reference value, and to damp nutation in the X- and Z-body axes. The magnetorquer and Y-wheel PD controller gains as well as the reference Y-wheel momentum used by this controller can be changed by *Set Y-Wheel Control Parameters telecommand (TC ID 39)*.

4.4.3.3 [XYZ-Wheel](#)

The *XYZ-Wheel* stage implements 3-axis zero bias reaction wheel control and requires full attitude and body rate estimates, thus either the Full State EKF (Estimation Mode 5) or the MEMS Gyro EKF (Estimation Mode 6) must be active. This stage contains 3 separate 3-axis stabilization control modes. In the first mode (Control Mode 5) the nominal attitude will be nadir pointing (for zero roll, pitch and yaw angles), but any constant roll, pitch and yaw reference attitude can be commanded using the *Set Commanded Attitude Angles telecommand (TC ID 15)*. The second mode (Control Mode 6) is used to track the Sun vector normal to the satellite's (typically deployed) solar panels for maximum power. The body facet to track the

Sun will be preconfigured in the code (currently the –Y or –Z facet can be preconfigured). The *XYZ-Reaction Wheel* Q-feedback PD gains can be changed by *Set Reaction Wheel Control Parameters telecommand (TC ID 40)*.

The third mode (Control Mode 7) is used to track a reference ground target at a specific location on earth. The target must first be selected with the *Set Tracking Controller Target Reference* telecommand (TC ID 55). The calculation of the Ground Target Reference parameters is discussed in Section 14.1 of **Ref 7**. The *XYZ-Reaction Wheel* Q-feedback PID gains of Control Mode 7 can be changed by *Set Tracking Controller Gain Parameters telecommand (TC ID 54)*.

The current control mode is changed by *Set Attitude Control Mode* telecommand (TC ID 13).

The command also includes an override flag that is not currently implemented. Please contact CubeSpace if this feature is required

A third parameter allows all the control modes involving the magnetorquers to be activated with a time-out. This is meant to be used during the commissioning phase of the control system, e.g. when the magnetorquer polarities are uncertain. Specifying a time-out period other than 0 or 0xFFFF (65535) seconds will let control proceed until the timer has expired. This is useful to verify that all configuration and mounting is set correctly, and that the control system is behaving nominally. During commissioning it is advised to always start with a timeout on the magnetic control modes.

The various control modes with the actuators used and required estimated states are listed in Table 4 below.

Table 4 – Control mode actuators and estimation requirements.

| Enum value | Control mode | Actuators used | Required estimated state |
|------------|--|---|--|
| 0 | No control | None | None |
| 1 | Detumbling control | Y-Magnetorquer | None |
| 2 | Y-Thomson spin | XYZ-Magnetorquers | Y-body rate |
| 3 | Y-Momentum wheel (Initial pitch acquisition) | Y-Momentum wheel & XYZ-Magnetorquers | X-,Y-,Z- body rates & pitch angle |
| 4 | Y-Momentum wheel (Steady state) | Y-Momentum wheel & XYZ-Magnetorquers | X-,Y-,Z- body rates & pitch angle |
| 5 | XYZ-Wheel control | XYZ-Reaction wheels & XYZ-Magnetorquers | X-,Y-,Z- body rates & Roll, pitch and yaw attitude |
| 6 | R-wheel Sun tracking control | XYZ-Reaction wheels & | X-,Y-,Z- body rates & |

| | | | |
|----|-----------------------------------|---|--|
| | | XYZ-Magnetorquers | Roll, pitch and yaw attitude |
| 7 | R-wheel target tracking control | XYZ-Reaction wheels & XYZ-Magnetorquers | X-,Y-,Z- body rates & Roll, pitch and yaw attitude |
| 8 | Very Fast-spin Detumbling control | XYZ-Magnetorquers | None |
| 9 | Fast-spin Detumbling control | XYZ-Magnetorquers | None |
| 10 | Z-Thomson spin | XYZ-Magnetorquers | Z-body rate |
| 11 | Z-Sun tracking spin | XYZ-Magnetorquers | Z-body rate |
| 12 | Stop XYZ Reaction wheels | XYZ-Reaction wheels | None |

4.4.4 Valid Estimation and Control Transitions

Table 5 displays the combinations of estimation and control modes which are allowed.

Table 5 – Valid estimation and control combinations.

| | | Control Mode | | | | |
|-----------------|---------------------------|--------------|-------------------------|-----------|------------|-----------------------|
| | | None | Detumbling & fast modes | Y-Thomson | Y-Momentum | XYZ-Wheel (all modes) |
| Estimation mode | None | OK | OK | X | X | X |
| | MEMS rate | OK | OK | OK | X | X |
| | Magnetometer rate | OK | OK | OK | X | X |
| | Magnetometer rate + pitch | OK | OK | OK | OK | X |
| | TRIAD | OK | OK | OK | OK | X |
| | Full-state EKF | OK | OK | OK | OK | OK |
| | MEMS Gyro EKF | OK | OK | OK | OK | OK |

An attempt to switch to an estimation mode or control mode (by sending the relevant commands) will fail if it will result in an invalid combination from the table above. This behavior cannot be overridden.

In addition, when switching control mode, the following checks, as found in Table 6, are performed based on the current control mode and state.

Table 6 – Valid control transitions.

| Current Control Mode | New Control Mode | | | | | |
|-------------------------|------------------|-------------------------|-----------|--------------------|-------------------------|-----------------------|
| | None | Detumbling & fast modes | Y-Thomson | Y-Momentum initial | Y-momentum steady-state | XYZ-Wheel (all modes) |
| None | N/A | OK | OK | X | OK | OK |
| Detumbling & fast modes | OK | N/A | OK | X | X | OK |
| Y-Thomson | OK | OK | N/A | OK | X | X |
| Y-momentum initial | OK | OK | OK | N/A | OK | X |
| Y-momentum steady-state | OK | OK | OK | X | N/A | OK |
| XYZ-Wheel (all modes) | OK | OK | OK | X | X | OK |

Control mode transitions will not be permitted if a suitable estimator is not active to generate the required attitude and angular rate estimates. Transitions will also not be permitted if the required ADCS subsystems (e.g. CubeControl Signal, CubeControl Motor and CubeWheel) power switches are not enabled.

5 Usage

5.1 Power Management

The ACP controls power to the nodes (including the sensors and actuators they are connected to) using several enable lines, connected directly to the CubeComputer MCU. Refer to **Ref 1** and **Ref 2** for detailed PC104 pin locations for these enable lines. Nominally, the enabled state of CubeSense, CubeWheel(s), CubeStar and the CubeControl nodes are managed by the ACP, depending on the control and estimation selection and state, but the enabled state can also be changed explicitly by sending the *Power Control* telecommand to the ACP (see **Ref 4**).

The latter telecommand also allows the GPS LNA (low noise amplifier supply to the GPS antenna) to be switched on or off. In this case, the GPS LNA is switched on by the CubeControl Signal MCU only when the ACP sends it a command over I2C.

The *Power Selection* parameter for each component can be either 0 - permanently off; 1 - permanently on; 2 - not currently implemented; or 3 - fake-on for simulated HIL mode.

5.2 Bootloader

5.2.1 [Bootloader](#)

The bootloader is programmed at internal MCU flash address 0, and it will start to execute when the CubeComputer is powered up. The bootloader will run for 5s before it will attempt to boot the currently selected application program.

The currently selected application is determined by the *Boot Index*, and can be any of the following listed in Table 7:

Table 7 – Boot programs list (boot index) enumeration values.

| Numeric Value | Name | Description |
|---------------|--------------------------|--------------------------|
| 0 | Bootloader | Bootloader |
| 1 | Internal Flash Program | Internal Flash Program |
| 2 | EEPROM | EEPROM |
| 3 | External Flash Program 1 | External Flash Program 1 |
| 4 | External Flash Program 2 | External Flash Program 2 |
| 5 | External Flash Program 3 | External Flash Program 3 |
| 6 | External Flash Program 4 | External Flash Program 4 |
| 7 | External Flash Program 5 | External Flash Program 5 |
| 8 | External Flash Program 6 | External Flash Program 6 |
| 9 | External Flash Program 7 | External Flash Program 7 |

If, during the 5s interval, before the bootloader attempts to start the application program, an *Identification* telemetry request is made, it will remain active and not attempt to start an application program (unless explicitly commanded to do so with a telecommand later).

The bootloader will also remain active if the current *Boot Index* is set to 0, or if there have been three or more unsuccessful attempts to start a particular application program.

The Boot Index and Boot Status are variables that are stored in non-volatile memory. The *Boot Status* keeps track of the number of boot attempts. Boot status enumeration values can be found in Table 8.

Table 8 – Boot status enumeration values.

| Numeric Value | Description |
|---------------|------------------------|
| 0 | New Selection |
| 1 | Boot Success |
| 2 | 1 Failed boot attempt |
| 3 | 2 Failed boot attempts |
| 4 | 3 Failed boot attempts |

Prior to starting the application program, the bootloader will change the value of *Boot Status* to 2 (1st failed boot attempt). The application should, upon start-up, overwrite this value with 1 (Boot Success). If it fails to do this, and the bootloader runs again (either as a result of commanded reset; manual power off and on; watchdog or other reset), the *Boot Status* will read back as 2 (1st failed boot attempt). The next time the bootloader attempts to start the application, it will increment the *Boot Status*. This will continue until it reaches 4 (3 failed boot attempts), at which point it will no longer automatically attempt to run the selected program (*Boot Index*), or the application overwrites the *Boot Status* with a Boot Success value. If the *Boot Index* is changed to a different value via telecommand, the *Boot Status* will also be changed to 0 (New selection) and again three boot attempts will be allowed.

A *Reset Counter* is also stored in flash memory and maintained by the bootloader. The *Reset Counter* is incremented every time the bootloader starts.

The *Identification* telemetry frame can be used to identify the program currently running. The *Node Type identifier* is a numeric value that is unique to the specific application (bootloader, CubeACP etc.).

5.2.2 Recovery on reset

If a CPU reset, or power cycle of the ADCS occurs while controlling the attitude of the satellite, four scenarios, found in Table 9, may occur:

Table 9 – Boot reset scenarios.

| Condition at reset/power cycle | Condition at ACP start-up (10 – 20s later) |
|---|---|
| Satellite is tumbling randomly. Detumbling controller is active (or not) | Satellite is still tumbling randomly |
| Satellite is in steady-state Y-Thomson spin – Detumbling controller is active | Satellite will still be in Y-Thomson spin |
| Satellite is in Y-Momentum control mode – stable attitude | Wheel spinning down, satellite spinning about Y-axis (Y Thomson spin) |
| Satellite is in 3-Axis pointing mode | Wheels will have had low bias and satellite will be in slow random spin |

The first scenario should only occur during commissioning of the satellite. In this case, automatic recovery will most likely not be preferred.

In the second scenario, the satellite remains in a Y-Thomson spin after the reset event. The satellite will remain in this state without active control, as long as there are no significant attitude disturbances. For a typical start-up time of 10s, there will be no change in the spinning motion of the satellite.

If a reset occurs while the satellite is in the Y-momentum stabilized mode, the wheel speed will ramp down because of the friction of the wheel bearings. Angular momentum will be transferred back to the satellite and the satellite will transition into a Y-Thomson spin. The wheel will spin from 2000 rpm to 0 rpm in about 30s, so it is likely that the wheel will still be spinning slowly when the ACP comes online.

If the satellite is in 3-Axis reaction wheel control mode, it will revert to a slow random tumble motion, depending on wheel bias at the reset.

5.3 Firmware Management

It is possible to find out which programs are programmed into FLASH memory by first issuing a *Read Program Information* command (ID 102), and subsequent *Program Information* telemetry request (ID 230). The *Busy* field of the *Program Information* telemetry frame will indicate if the CubeComputer is still busy reading the FLASH memory contents. Returned program information includes the CRC and file size of programs in FLASH memory slots.

The program that the bootloader should attempt to run automatically can be set with the *Set Boot Index* command (ID 100). This will automatically reset the *Boot Status* to 0 (*New Selection*).

The program segments in the internal MCU flash cannot be directly written to (see next section on File Transfers for firmware upload details). But it is possible to copy any of the programs in external flash memory to internal flash, using the *Copy Program to Internal Flash* command (ID 103). Note that overwriting the bootloader is not recommended.

After issuing the *Copy to Internal Flash* command, the progress of the operation can be polled using the *Copy to Internal Flash Progress* telemetry request (ID 231).

It is possible to start a program directly, without having to go through the bootloader process. The *Run Selected Program* command (ID 101) will jump to the program pointed to by the current *Boot Index*.

5.4 Configuration and setup

The ADCS unit must be configured to specify the mounting of the sensors and actuators relative to the satellite reference frame and to specify calibration setting unique to each sensor.

The ADCS Configuration Command TC 20, is a single command that contains all the ADCS configuration. There are also various shorter TCs to set the configuration for individual sections of the ADCS. Table 10 displays the different sensors, actuators and attributes that are contained in TC 20, as well as the IDs for shorter TCs to set the individual configurations.

Table 10 – Telecommand 20 and corresponding individual telecommands.

| Part | Dedicated TC ID(s) |
|---------------------------|--------------------|
| Magnetorquer | |
| Axis Configuration (1-3) | ID 21 |
| CubeWheel | |
| Axis Configuration (1-4) | ID 22 |
| MEMS Gyro | |
| Axis Configuration (1-3) | ID 23 |
| Configuration (1-3) | ID 36 |
| CSS | |
| Axis Configuration (1-10) | ID 24 |

| | |
|--------------------------------------|----------|
| Scaling (1-10) | ID 25 |
| CubeSense | |
| Sun mounting & configuration | ID 26 |
| Earth mounting & configuration | ID 27 |
| Earth mask (1-5) | ID 28-32 |
| Magnetometer | |
| Mounting | ID 33 |
| Configuration | ID 34-35 |
| CubeStar | |
| Mounting & configuration | ID 37 |
| ADCS Controllers | |
| Y-Wheel control configuration | ID 39 |
| Reaction wheel control configuration | ID 40 |
| Tracking control configuration | ID 54 |
| Tracking control target reference | ID 55 |
| ADCS Estimators | |
| Estimation parameters | ID 43-44 |
| Satellite General | |
| Moments of inertia | ID 41 |
| Products of inertia | ID 42 |
| SGP4 Orbit parameters | ID 45 |

If these configuration settings have been set using the large TC, or the smaller sub-TCs, the configuration can be saved to flash using TC 63 and 64. If this configuration is not saved, it will be lost after reset of the ADCS OBC.

5.4.1 Mounting transformations

The sensors and actuators in the ADCS bundle are either classed as **single** or **multiple axes**. **Single axis sensors or actuators** are sensors or actuators where the alignment of only one axis is relevant, such as a wheel or single-axis rate sensor. In this case the axis about which rotation or sensing takes place is important, but the mounting angle about the rotation or sensing axis is irrelevant.

Multiple axes sensors or actuators are sensors or actuators for which the alignment of all three principle axes are relevant.

The mounting of the sensors and actuators of the ADCS must be defined by:

- **For single axis sensors or actuators:** specifying the principle SBC axis to which the sensor or actuator is aligned.
- **For multiple axes sensors:** specifying mounting transform angles to rotate the sensor or actuator coordinate system to match the SBC.

Single-axis sensors or actuators include the following:

- Coarse Sun sensors (TC 24)
- MEMS rate sensors (TC 23)
- Torquer rods (TC 21)
- Reaction/Momentum wheels (TC 22)

Multi-axis sensors include the following:

- Deployable or redundant magnetometer (TC 33)
- Fine Sun sensor (CubeSense) (TC 26)
- Fine Earth sensor (CubeSense) (TC 27)
- Star tracker (CubeStar) (TC 37)

A complete guide to identifying these mounting transforms (with examples) is given in Ref 6. Review this document carefully, complete it, and send it back to CubeSpace. These transforms are critical to the correct functioning of the ADCS and require careful review.

5.4.2 Sensor calibration settings

The raw sensor measurements are put through calibration functions before they are used by the ADCS estimation and control algorithms. Each sensor or actuator has its own unique calibration settings, which is explained in the sections below.

5.4.2.1 Magnetometer

The magnetometer pre-calibration (ground calibrated) equation is:

$$\mathbf{B}_{precalib} = \mathbf{S}_{pre}(T)\mathbf{B}_{raw} - \mathbf{d}_{pre}(T)$$

Where:

$\mathbf{B}_{precalib}$ is the pre-calibrated measured magnetic field in the body coordinate frame;

$\mathbf{B}_{raw} = [B_{rawx} \ B_{rawy} \ B_{rawz}]^T$ is the raw measurement returned by CubeControl;

$\mathbf{S}_{pre}(T)$ is the temperature dependent sensitivity matrix; and

$\mathbf{d}_{pre}(T)$ is the temperature dependent offset vector.

The magnetometer post-calibration (in-orbit calibrated) equation is:

$$\mathbf{B}_{calib} = \mathbf{S}_{post}\mathbf{M}_{mounting} \mathbf{B}_{precalib} - \mathbf{d}_{post}$$

Where:

$\mathbf{M}_{mounting}$ is the mounting transformation DCM set as in Section 5.4.1;

\mathbf{S}_{post} is the in-orbit calibrated sensitivity matrix; and

\mathbf{d}_{post} is the in-orbit calibrated offset vector.

The mounting angles will also have to be updated after deploying the magnetometer boom, since the orientation of the magnetometer is now different.

The diagonal components of the \mathbf{S} matrix are scaling factors for each channel, while the off-diagonal components account for non-orthogonal measurements axes.

The temperature dependent pre-calibration sensitivity matrix $\mathbf{S}_{pre}(T)$ and offsets $\mathbf{d}_{pre}(T)$ are **measured during integration for each magnetometer** and will differ for different sets of hardware. The magnetometer is placed in a Helmholtz cage with a high accuracy flux-gate magnetometer. The sensitivity and offsets are determined, and the magnetometer is also temperature calibrated. Measured values are recorded in the build history for each ADCS module.

Ideally the magnetometer would measure the undistorted ideal magnetic field of the Earth once in space. Once the magnetometer is mounted on the satellite, the magnetic field is however distorted by certain parts of the satellite - like ferrous materials and wires carrying large currents. The magnetic field measured by the magnetometer is therefore not ideal and **recalibration of the magnetometer might be necessary in orbit. Please consult Ref 7 for more information on this procedure.**

It is important to note that the ACP performs a check on the calibrated measurement. The post-calibrated magnetic field vector \mathbf{B}_{calib} magnitude must be between 15,000 and 65,000 nT otherwise the measurement is not used by the estimator. In addition, the ACP sets the *Magnetometer Range Error* flag in the *ADCS State* telemetry if the measurement is out of range (see **Ref 4**).

[5.4.2.2 Coarse Sun Sensor](#)

The mounting of the coarse Sun sensors is described in Section 5.4.1.

The CSS photodiodes have bias values even when they are not illuminated. These values are not significant since the CSS only provide a relatively coarse measurement. To reduce the effect of these offsets, and to make the sensors more robust against measurement of reflections off the moon for instance, the CSS measurements are only seen as 'valid' and is only used if the measured value is above a certain threshold, s_{min} . Owing to the varying CSS sensors' sensitivity, the CSS outputs can be scaled. For each of the ten CSSs the configurations settings are then:

- Axis definition
- Scaling
- Minimum threshold

The default scaling and offset configuration can be used for any satellite and the mounting settings must be configured by the user.

5.4.2.3 Sun and nadir sensor

CubeSense has the following calibration settings:

- Mounting – As described in Section 5.4.1
- Detection threshold
- Auto-adjust exposure flag
- Manual exposure time
- Boresight X & Y
- Shift flag

Each of these are discussed below.

Mounting

Each CubeSense module has two cameras that can be mounted on the CubeSense PCB, or mounted externally and connected to the CubeSense PCB through a harness. Each camera's axes are defined as illustrated in Figure 9.

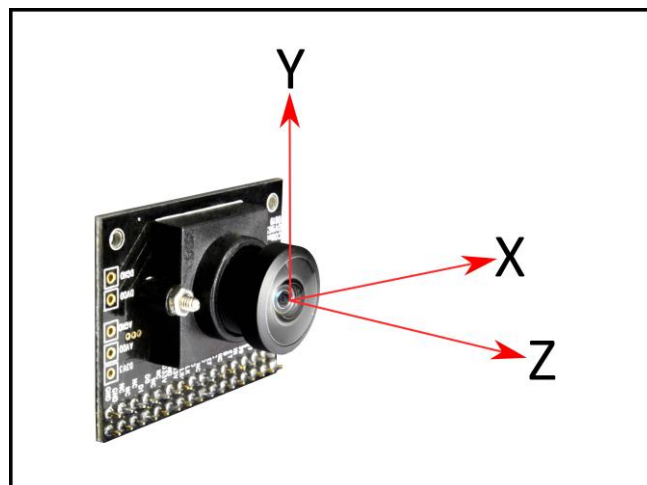


Figure 9 – Camera axes definitions.

The procedure discussed in Section 5.4.1 must be followed to configure the CubeSense mounting angles.

Detection threshold

CubeSense is a visible light image-based sensor. Part of the image processing that is done inside the module is applying a threshold to distinguish between background pixels and pixels

that form part of an object – The Sun, Earth and Moon. This threshold must be chosen correctly for a given exposure value. The default threshold values are correct for the default exposure values. If either is changed, the CubeSpace team should first be consulted.

Auto-adjust exposure flag

Each of the cameras can run an auto-adjust algorithm to automatically adjust the exposure based on the lighting conditions it is experiencing. By default, the Earth sensors run with the auto-adjust on, and the Sun sensors with it off. This should not be changed except if it is for a justified reason that has been approved by CubeSpace.

Manual exposure time

If auto-exposure is switched off the camera exposure is determined by the manual exposure value that is set in the configuration. This value is between 0-1000. Where 0 is the shortest exposure (darkest photo) and 1000 is the longest exposure (lightest photo). These values are correct by default and should only be changed after consulting CubeSpace.

Boresight X & Y

Each CubeSense camera provides two angles, α and β , as output. Angle α relates to the angle θ which is the smallest angle between the Sun vector and the sensor boresight (which is the Z-axis). Angle β relates to the rotation φ angle around the boresight. **The α and β angles are called the *Sun Centroid X and Y angles in the Raw Measurements telemetry frame*.** The direction vector can be calculated using the following procedure:

First calculate θ and φ using the following formula:

$$\theta = \sqrt{\left(\frac{\alpha}{100}\right)^2 + \left(\frac{\beta}{100}\right)^2}$$

$$\varphi = \text{atan2}(\beta, \alpha)$$

Where atan2 is the 4-quadrant arctangent function. Then, the direction vector is:

$$\mathbf{s}_{\text{sensor}} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \sin(\theta) \cos(\varphi) \\ \sin(\theta) \sin(\varphi) \\ \cos(\theta) \end{bmatrix}$$

In finding the α and β detection values, the sensor makes use of a calibrated α and β “bore-sight”-location on the imaging sensor. This pixel location is the intersection of the optical axis of the lens with the imaging sensor. This value is unique for every sensor and must be configured.



This boresight value is unique to each camera. Each CubeSense board is paired with a specific set of camera modules. Boresights **MUST be updated whenever these camera modules are swapped for others.**

Shift

If the Sun or nadir sensor measured **current levels to its SRAM that exceed the defined limit, the ACP will set the *Sun Sensor Overcurrent* or *Nadir Sensor Overcurrent* error flags** in the ADCS State telemetry, and it will reject the measurement. This should not occur under nominal conditions. If an SRAM overcurrent is detected, the operator of the satellite can choose to clear the error flags and start using the sensor again. Alternatively, if the error persists, the SRAM might have been damaged by a radiation event. In this case, CubeSense can be forced to do only Sun or nadir detection to the remaining working SRAM. This is done by setting the shift flag for the given problematic sensor. If the Sun sensor SRAM gives an overcurrent condition for instance, the Sun shift flag can be set, and the Sun sensor will operate using the Earth sensor's SRAM. It is important to note that in this case, the nadir sensor will no longer be used.

Sampling and errors

The ACP will reject a Sun or nadir measurement if the sensor reports that it is still busy performing the detection or if the detection result is anything other than successful. If the **sensor is busy** at the time the Sun or nadir telemetry is requested, the ACP will set the ***Sun Sensor Busy Error* or *Nadir Sensor Busy Error* flags in the ADCS State telemetry frame.** This should not occur under normal operating conditions.

If the sensor reports a detection failure, the *Sun Detection Error* or *Nadir Detection Error* flags in the ADCS State telemetry frame will be set. These flags will also be set if the measured Sun azimuth and elevation angles are larger than 90 deg (or smaller than -90 deg), and the measurement will not be used for estimation. For the nadir sensor, the valid azimuth and elevation range is limited to -60 to +60 deg. These flags will regularly be set during nominal operations since the Sun will often not be in range, or the Earth will only be partially visible. **The user does not have to manage these flags, however. The ACP will automatically use the measurements only when they are valid. When a valid measurement is received, these flags are automatically cleared by the ACP.**

In some situations, such as commissioning, it will be required to sample raw Sun or nadir measurements without using the calibrated measurements in the EKF estimator. The Sun and nadir sensor sampling can be controlled through the *Sun and Nadir Sampling Period* setting in the *Set Estimation Parameters 2* TC ID 44. This parameter is an 8-bit value of which the lower

4 bits is the Sun sampling period and upper 4 bits is nadir sampling period. Setting either to zero will prevent sampling of that particular sensor, even if the CubeSense is powered on.

The *Mask Sun Sensor* and *Mask Nadir Sensor* flags in the Estimation Parameters will further determine if the calibrated measurement vectors will be used in the EKF. The sampling of the CubeSense measurements will also be skipped if the ADCS computes that it is currently in eclipse, based on current TLEs and unix time.

5.4.2.4 [Rate sensor](#)

Axis definition of the rate sensors is done as described in Section 5.4.1. The first part of the rate sensor calibration in the ACP is a multiplier that scales the raw rate sensor measurements. This multiplier can either be $\times 1$ (± 75 deg/s), $\times 2$ (± 150 deg/s), $\times 4$ (± 300 deg/s) or $\times 12$ (± 900 deg/s). The reason for this multiplication is to make the rate sensor range adjustable. The rate sensor TLM measurements are formatted as a 16-bit integer. If a satellite is spinning at very large rates, the $\times 12$ multiplier is implemented giving the rate sensor a ± 327.68 deg/s ($\pm 32k$ centi-deg/s) measurement range and a centi-deg/s TLM resolution, given the 16-bit variable size. When the satellite is spinning at low rates, it is desirable to have milli-deg/s resolution for more accurate measurement. In this case the $\times 1$ default multiplier is implemented which decreases the range to ± 32.768 deg/s ($\pm 32k$ milli-deg/s) that can fit in the 16-bits, but providing a milli-deg/s TLM resolution.

The multiplier is set to $\times N$ ($N = 1, 2, 4, 12$), when the *RateSensorMult* parameter in *Set Rate Sensor Offsets* TC ID 36, is set to N . By default it sets to $\times 1$ ($N = 1$). In extreme cases where a satellite is tumbling at rates higher than ± 32.768 deg/s, the multiplier can be set to $N = 2, 4$, or 12.

Further, the rate sensors are calibrated by subtracting a factor to compensate for temperature variation, and subtracting a permanent offset value. The temperature calibration is hardcoded into the ADCS and the only configuration parameter that should be set for the rate sensors are the permanent offsets set by the *Set Rate Sensor Offsets* TC ID 36.

5.4.2.5 [CubeStar](#)

The CubeStar configuration parameters are listed in Table 11. All CubeStar units are pre-calibrated and the default configuration parameters are set to optimal values. Most of these parameters can be altered using TCMD 20 or TCMD 37. However, under nominal circumstances, altering should not be necessary. Please contact CubeSpace before altering any of these values. The configuration parameters can be read using TLM 206.

Table 11 – CubeStar configuration parameters.

| Parameter | Default Value |
|---------------|---------------|
| Exposure Time | 1704 |

| | |
|------------------------------|-------------------|
| Detection Threshold | 22 |
| Star Threshold | 5 |
| Maximum Stars Matched | 15 |
| Maximum Star Noise | 200 |
| Maximum Star Pixels | 60 |
| Minimum Star Pixels | 4 |
| Error Margin | 85 |
| Centroid X | Roughly ± 640 |
| Centroid Y | Roughly ± 519 |
| Focal Length | Roughly ± 6.2 |
| Sync Delay | 45 |

5.4.2.5.1 Exposure

The exposure parameter is a 16-bit value. This value translates to an exposure time which can be calculated using the formula:

$$Exposure_{ms} = Exposure_{16\text{ bit paramter}} \times 0.1795 + 14.565ms$$

The default exposure value of 2704 translates to an exposure value of 499.93 milliseconds. Increasing the exposure will cause the overall processing time and noise of an image to increase. An increase in exposure time may also prevent CubeStar from providing new measurements every second. This value should not be changed without consulting CubeSpace.

5.4.2.5.2 Detection and Star Threshold

The star detection algorithm searches though the image by evaluating every second pixel against the *Detection Threshold* value. If the pixel value is greater than the threshold value it indicates that this pixel may be part of a star. Once such a pixel is found, the algorithm searches the neighbouring pixels to find all the pixels with values more than or equal to the *Star Threshold*.

Increasing the *Detection Threshold* will cause dimmer stars not to be detected.

Decreasing the *Detection Threshold* value will cause more noise to be found and will increase the time the detection algorithm takes to execute. It may also cause CubeStar to find dimmer stars not detected previously. There are limits on the number of stars and how much noise can be found in an image, therefore decreasing the detection threshold will cause the algorithm to reach the noise and stars detection limits earlier in its search, which will affect performance.

Increasing the *Star Threshold* will result in fewer pixels being detected as part of a star. This will decrease the execution time of the detection algorithm, but may cause dimmer stars to be seen as noise.

Decreasing the *Star Threshold* will result in more pixels being detected as part of a star. This will increase the execution time of the detection algorithm. It may also cause more than the

maximum number of pixels to be detected for a star. This will cause larger stars to be seen as invalid stars.

Once these threshold values have been set, it takes immediate effect. The *star pixel threshold* must always be less than the *detection threshold*.

5.4.2.5.3 Max and Min Pixels per Star

Once all the pixels above or equal to the *Star Threshold* have been found, it is necessary to determine whether these pixels form a star. This is done by evaluating how many pixels were found, against the minimum and maximum pixels a star can have. If the number of pixels is greater or equal to the minimum number of pixels and less or equal to the maximum amount of pixels they form a valid star. CubeStar counts the number of valid stars that is found as it searches through the image.

If the number of pixels found is less than the minimum number, it is considered as noise detected. CubeStar counts how many of these invalid detections are found as it searches through the image. The pixels or small grouping of pixels that fall below the minimum number of pixels for a valid star, is referred to as noise in CubeStar.

If the number of pixels per star is found to be more than the maximum number of pixels allowed, the pixels are considered to be part of a bright object in the field of view, e.g. The Moon. CubeStar counts how many of these invalid objects are detected as it searches through the image. These clusters of pixels above the threshold values that exceed the maximum number of pixels for a star are referred to as invalid stars.

The largest value for the maximum number of pixels that can be set is 99. If an attempt is made to set the value higher than 99, an error flag will be set.

Decreasing the minimum pixels to three or less pixels will cause noise to be considered as stars. Increasing the minimum pixels will cause small stars to be rejected as noise.

Decreasing the maximum pixels might cause larger or brighter stars to be seen as invalid stars.

The minimum and maximum pixel values also depend on the detection threshold value. A high detection threshold will cause less pixels to be detected, while a lower threshold value will cause more pixels to be detected.

5.4.2.5.4 Max stars

As stated earlier CubeStar keeps count of how much noise, invalid and valid stars are detected, while it searches through the image. The detection algorithm needs to ensure that the execution time is not too long, and that the image being processed is valid. The default limits are shown in Table 12:

Table 12 – Detection limits.

| Parameter | Default limit |
|---------------|---------------|
| Noise | 200 |
| Invalid stars | 5 |
| Valid stars | 15 |

If any of these limits are reached during the star search, the star search is ended and an error flag is set. This does not always mean that the image could not be processed. For example, in the case where the maximum number of stars is detected, the search is stopped and the max stars detected error flag is set.

Once the detection is complete, the identification algorithm will execute, as long as three or more stars were detected.

5.4.2.5.5 Error Margin

CubeStar uses an error margin when calculating the distance between stars. This error margin is a percentage of the distance. The default value is 0.85 % or as a gain value: 0.0085. This telecommand allows this margin to be changed and accepts an 8-bit parameter.

This 8-bit parameter is the gain value multiplied by 10 000.

$$parameter = gain \times 10\,000$$

For the default gain of 0.0085 the parameter will have the value of 85.

When CubeStar receives the parameter it is divided by 10 000 to obtain the correct gain to be used in the calculations.

Owing to the conversion and the byte size of the parameter, the gain value can only be between 0% and 2.25% (gain value: 0 to 0.0225).

5.4.2.5.6 Centroid

The *Centroid* is the position on the image plane in line with the point on the lens with no distortion. This is close to the center of the lens and is determined during calibration.

5.4.2.5.7 Focal Length

The *Focal Length* is a value in millimetres. This value is determined during calibration and should be close to 6.2 mm. An error flag will be set if the focal length is not set between 5 and 7 mm.

5.4.2.5.8 Sync Delay

This delay is used along with the trigger command to schedule the image capture and establish a one second loop. The one second update rate is achieved by overlapping the capture of a new image with the processing of the previous image. After the trigger command is received

the delay timer is started. When the timer reaches the timeout value it triggers a new image capture.

This command accepts one byte as a parameter. This 8-bit value will be divided by 100 in CubeStar to calculate the timeout of the timer in seconds. Given a certain timeout period, the time parameter can be calculated using the following equation:

$$Time_{8-bit\ parameter} = \text{int}(Timeout_s \times 100)$$

This formula allows for a resolution of 10 ms. The default timeout is 450 ms. This translates to a parameter value of $0.45 \times 100 = 45$. The value 45 is the parameter value that will be received if you request TLM 173 if the default timeout value of 450 ms is set. If the default timeout value of 450 ms is set, the value 45 is the parameter value that will be received if the user requests TLM 173 (see **Ref 4**).

The timeout in milliseconds must be a value between 0 to 0.99 s. This implies that the parameter value will range between 0 and 99.

5.4.3 Actuator configuration settings

5.4.3.1 [Magnetorquers](#)

The only configuration for the magnetorquers is the mounting, as described in Section 5.4.1. The *Axis Selection* enumeration (see **Ref 4**), for each magnetorquer rod, is used to map the control command to torquer output.

5.4.3.2 [Reaction wheels](#)

The axes of each of the reaction wheels are set using the ADCS Configuration telemetry (see **REF 4**). Figure 10 displays the polarity of the reaction wheel.

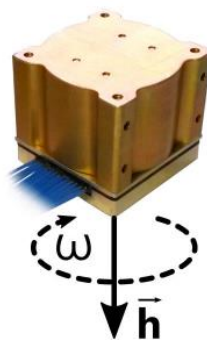


Figure 10 – Reaction wheel polarity.

5.4.4 ADCS Controller configuration

The ADCS controller configuration consists of settings for detumbling, Y-wheel, and reaction wheel controllers.

5.4.4.1 [*Detumbling controller settings*](#)

The detumbling controller setting contains three gain values, K_s , K_d and K_{df} . These values are specified by CubeSpace at delivery and should not be changed unless consulting CubeSpace.

Further, the reference spin rate for the detumbling controller can be specified. The default value for the spin rate is -2 deg/s. This can be adjusted in specific cases, however CubeSpace should be consulted before changing this value.

5.4.4.2 [*Y-Wheel controller settings*](#)

The Y-Wheel controller settings consist of four gain values K_h , K_n , K_{p1} , and K_{d1} . These gains are set at delivery of each ADCS unit and should only be changed after consultation with CubeSpace.

Further, the reference wheel momentum for the Y-Wheel controller can be specified. The default value for the wheel momentum depends on the satellite's Y-axis MOI and will be specified by CubeSpace at delivery. This can be adjusted for specific cases, however CubeSpace should be consulted before changing this value.

5.4.4.3 [*Reaction wheel controller settings*](#)

The reaction wheel controller setting consists of two gains, K_{p2} and K_{d2} , having values which are specified by CubeSpace at delivery. The values should not be changed, unless CubeSpace is consulted.

5.4.5 ADCS Estimator configuration

The estimator setting consists of three parts: sensor EKF measurement noise; sensor masking; and Sun or nadir sampling period.

5.4.5.1 [*Measurement noise*](#)

The estimator algorithms make use of filters that combine sensor measured vectors and modelled vectors. The measurement noise parameters in the ADCS configurations settings determine how heavy or light, in weight, the measured vectors are, relative to the modelled vectors. These values are correctly set up at delivery, and CubeSpace should be consulted if they are to be changed.

[5.4.5.2 Sensor masking](#)

The estimator algorithm will use whatever measurements are available on the satellite. All sensors that are switched on and enabled will be sampled by the ACP and will be available for use in the estimators. **In some cases, especially during commissioning, it is desirable to sample sensors but not to use it in the filter. In this case, the sensor can be masked. The sensor is included in the filter, if the masking value is set to 1 and excluded, if it is set to 0.**

[5.4.5.3 Sun/nadir sampling period](#)

The Sun and Earth sensors (CubeSense) are sampled at 1Hz by default.

[5.4.6 General satellite configuration](#)

[5.4.6.1 MOI and POI](#)

The moments of inertia and the products of inertia for a specific satellite must be correctly set for the ADCS to function nominally. **If any deployable structure on the satellite is activated, the MOI and POI of the satellite will typically change. It is critically important that the MOI and POI in the ADCS configuration is updated when the deployables are activated.**

[5.4.6.2 Orbit parameter configuration](#)

The orbit SGP4 parameters can be set using a single command TC ID 45, or individual parts of the SGP4 parameters can be set separately using TC ID 46-53. The SGP4 parameters must be regularly updated to account for a changing orbit. This is especially important at LEO altitudes.

It is recommended that the TLEs be updated daily during commissioning, and at least once a week, thereafter. See Figure 11 below.

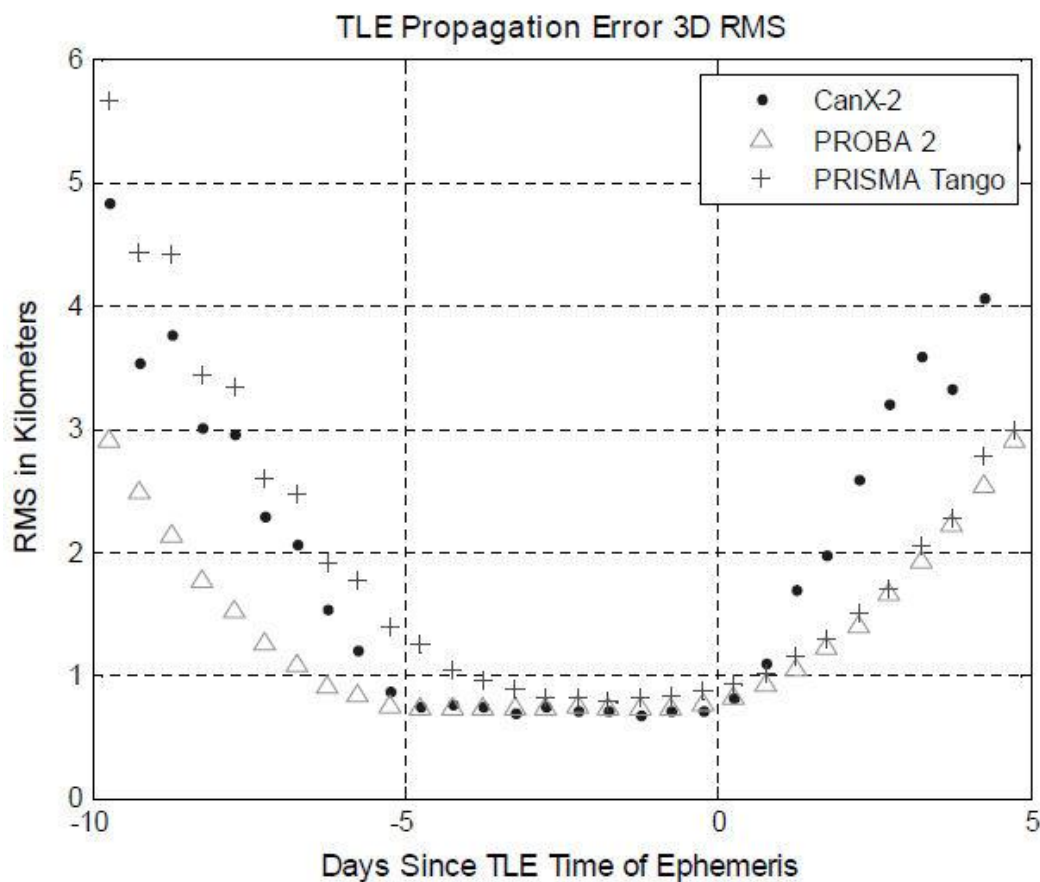


Figure 11 – TLE Time since Epoch errors.

5.4.7 General considerations with configuration

Both the configuration block and SGP4 orbit parameters are stored in flash memory on the CubeComputer and persists between power cycles.

On start-up, the configuration in flash memory will be read and transferred to RAM. The configuration block in RAM is what will be used by the ACP for controller and estimation parameters, and sensor calibration. If the configuration block in flash memory is corrupt or has not been programmed yet (a CRC check is performed when doing the initial read) a flag in the *ADCS State* telemetry frame (see **Ref 4**) is set, and a default RAM configuration block is populated from code.

The current RAM configuration can be changed by sending the *Set Configuration* telecommand (see **Ref 4**) or smaller individual targeted telecommands as described in the previous sections. The current RAM configuration can be read via the *Configuration* telemetry request (see **Ref 4**).

The current RAM configuration is written to flash memory upon receiving a *Save Configuration* command (see **Ref 4**).

The process for the current SGP4 orbit parameters is the same. For command and telemetry formats, refer to the *Set SGP4 Orbit Parameters* telecommand and the *SGP4 Orbit Parameters* telemetry request.

5.5 Status and error flags

The *ADCS State* telemetry (see **Ref 4**) contains the status and error flags. Error flags are latched whenever the particular error occurs, and must be cleared by telecommand.

5.6 Logging

It is possible to configure logging on the CubeComputer application (the CubeComputer bootloader does not support logging, but the CubeACP and programs built using the BSP framework do). Logging can be enabled for a variable selection of telemetry frames, with selectable period, up to 1s resolution.

It is possible to log telemetry to the on-board SD card or as “unsolicited” telemetry on the UART communications link. For logging to SD card, two independent selections can be configured. This allows for logging of different telemetry frames at different periods. This can be used, for instance, when most telemetry frames are only required at a slow interval (house-keeping telemetry) but certain telemetry such as magnetometer measurements, are needed at 1s interval.

The logging selections are configuration items – they can be set via command and read back using telemetry request. See **Ref 4** for SD Log1 Configuration, SD Log2 Configuration and UART Log Configuration message format.

If the on-board telemetry logging functionality is not being used, and telemetry logging is to be performed by an external master, it is important that the logging ensures synchronization of telemetry data.

Since the ACP loop executes at 1s periods, telemetry frames containing estimated attitude state and sensor measurements will be updated with a 1s period. For logging purposes, these frames should be requested from the ACP with integer second periods and preferably synchronized to the ADCS loop. The ideal sample time would be to request the frames after one control loop has executed, before the next iteration.

This can be achieved by using the runtime milliseconds counter in the *Identification* telemetry request (See **Ref 4**, TLM ID 128) to synchronize telemetry sampling. The ADCS control loop should execute in the first 300ms of 1s period, leaving about 700ms in which the telemetry may be sampled. It is suggested that telemetry sampling is synchronized to start when the runtime milliseconds count reaches 500ms.

5.7 Magnetometer boom deployment

Both CubeControl MCUs should be on when sending the *Deploy Magnetometer Boom* command (see **Ref 4**). In addition, the *ADCS Run Mode* should be set to *Enabled*. A deployment timeout of 2s is suggested. This timeout can be increased in case of non-deployment. Please **Ref 7** for the suggested boom deployment procedure.

5.8 Image Downloads

The ACP can download images from the CubeSense and CubeStar cameras. For this functionality to be available, an SD card must be present in the CubeComputer SD card slot. Image downloads are initiated by the *Save Image* telecommand (see **Ref 4**). A parameter in the command selects which camera to download from (Sun, nadir or star) and the size of the image to download to the SD card.

The time that the download process takes to complete is dependent on the size of the image that is selected. The status of the download can be polled by reading the *Status of Image Capture and Save Operation* telemetry frame (see **Ref 4**). While the download is taking place, the CubeSense will not perform any detection. If the ADCS loop is running while an image is being downloaded, both the Sun and nadir sensor will automatically be masked.

Once the download is complete, the file can be downloaded from the ACP using the protocol described in section 5.9.

5.9 File Downloads

Files that are stored on the SD card of CubeComputer can be downloaded and managed using the protocol described here.

5.9.1 Download File List

The current download-able list of files on the SD card can be found in two ways. The first method involves a sequence of *File Information* telemetry requests. But first, the file list read pointer must be reset with a *Reset File List Read Pointer* command (see **Ref 4**).

Following the *Reset File List Read Pointer* command, the next available file in the download list can be queried with the *File Information* telemetry request. The *File Information* telemetry frame should be polled until the *Busy Updating* flag is cleared. At this point the populated file information will be valid. If the populated file information comes back as all zeroes, the end of the file list has been reached. If not, the file list read pointer can be advanced to the next file with the *Advance File List Read Pointer* command.

The complete process is described by the flow diagram in Figure 12.

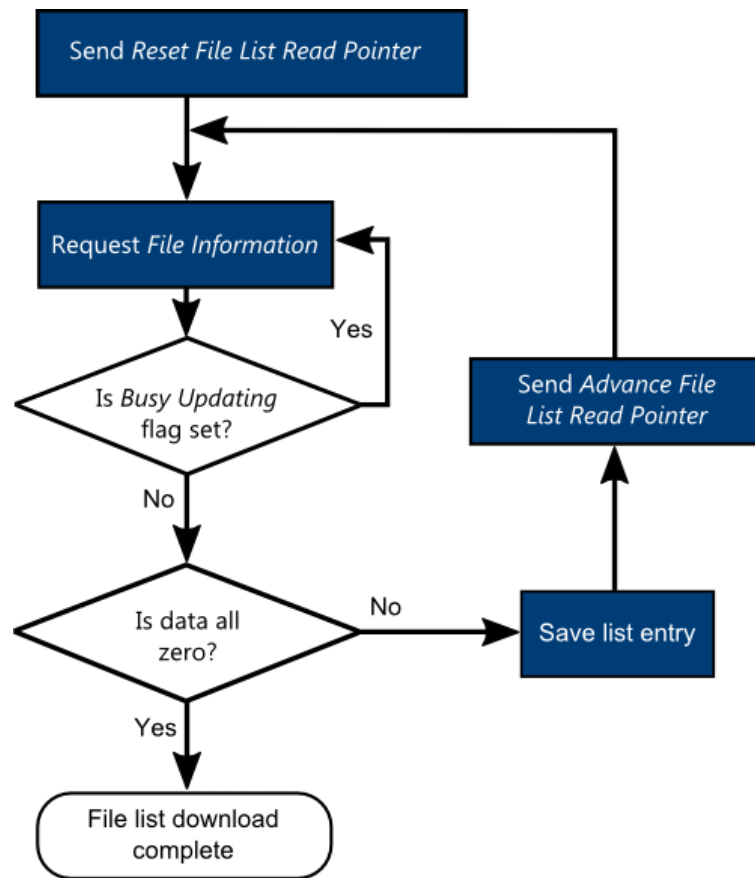


Figure 12 – File list download process.

The other method of retrieving the download-able file list is to perform an actual file transfer. In this case the file to download is the Index File (see **Ref 4**).

5.9.2 File Download

Files are downloaded from CubeComputer in blocks. A block is typically downloaded as a burst of packets, of which some packets may get lost. A "Hole Map" is used to keep track which packets in a block have been successfully received. The Hole Map is then communicated to the CubeComputer, and a burst retry is performed for all the missed packets.

The download of a file block is initiated by sending a *Load File Download Block* command (see **Ref 4**). This will read a block of data from the indicated file and buffer it in SRAM on the CubeComputer.

The file is uniquely identified by the *File Type* and *Counter* parameters. The *Offset* and *Block Length* parameters indicate which part of the file to buffer in memory. The maximum *Block Length* is 20 kB.

The *Download Block Ready* telemetry frame can be polled to see if the block is done loading. Once the block is ready, the burst can be initiated by sending an *Initiate Download Burst* command, with the *Ignore Hole Map* parameter set to true.

If the Initiate Download Burst command was sent on the UART, the CubeComputer application will respond by sending up to 1024 packets (the number depends on the *Block Length* specified with the *Load File Download Block* command), each with a payload length of 20 bytes in rapid succession. Each download packet will have a header ID that matches that of the *Initiate File Download Burst* command. The following two bytes will contain the counter of the packet in the burst. The counter makes it possible to keep track of which packets have been received on the remote side.

Table 13- File packet telemetry definition.

| Msg Type | File packet | | Parameters Length (bytes) | 22 | |
|-------------|---|---------------|---------------------------|-----------|---|
| Description | Fill download buffer with file contents | | | | |
| Parameters | Offset (bits) | Length (bits) | Name | Data Type | Description |
| | 0 | 16 | Packet counter | UINT | Packet counter of this file download packet |
| | 16 | 160 | File bytes | ARRAY | File bytes |

If the Initiate File Download Burst command is issued on the I2C communications link, it is expected to successively perform up to 1024 read transactions from the remote side, each with 22 bytes length, again with the same format as the *Download File Packet* above.

The *Hole Map* is a bit-map, where each bit represents one 20-byte packet out of the complete download block. Since there are at most 1024 packets in one file download block, the *Hole Map* is also 1024 bits long, or 128 bytes.

To make the CubeComputer resend only the packets that have been missed by the remote side, the remote component must upload the Hole Map to the CubeComputer by performing up to 8 *Hole Map* commands (each *Hole Map* command contains 16 bytes of the Hole Map).

After uploading the Hole Map, another Initiate Download Burst command can be issued, but this time with the Ignore Hole Map parameter set to false. The CubeComputer will then transmit only the packets that have a corresponding '0' in the Hole Map.

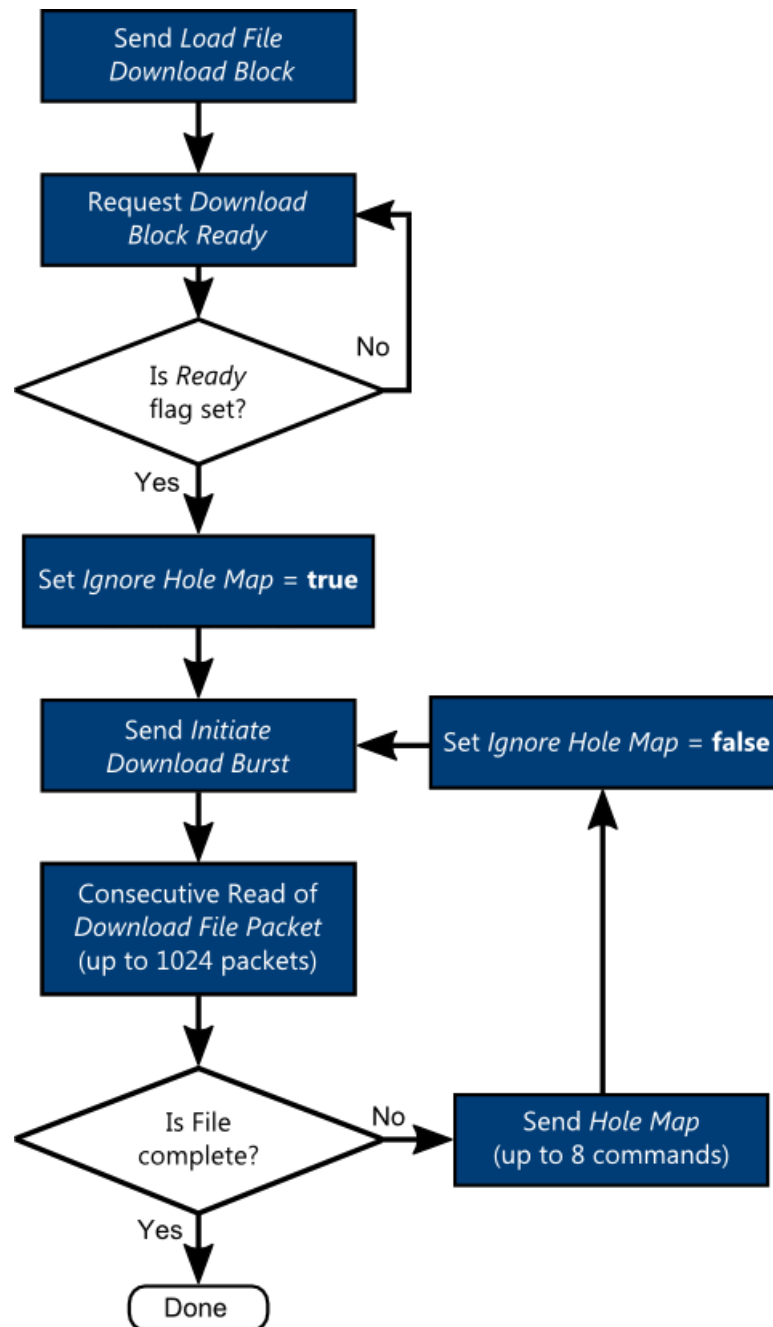


Figure 13 – File download process.

5.9.3 Index File

The Index File is a virtual file on the CubeComputer. It contains the list of downloadable files on the SD card with the file meta-data included. The downloaded file index will contain a 37-byte entry for every file on the SD card. See **REF 4** for the Index file contents.

5.9.4 File Erase

To delete files, use the *Erase File* command (see **Ref 4**).

5.9.5 File Uploads

File uploads make use of the same approach as file downloads – i.e. a burst of packets without acknowledges. The Hole Map is also used, but this time it is updated by the CubeComputer as it receives packets.

Uploaded file packets are first buffered in RAM on the CubeComputer before they are committed to flash memory or stored to SD card. The RAM buffer and packet sizes are the same as for file downloads.

As the ultimate destination of uploaded file packets is either flash memory or one of the SD card upload slots, it is necessary to first clear the firmware program area or SD card file with an *Initiate File Upload* command.

The status of the flash or SD card erase operation can be polled with the *Initialise Upload Complete* telemetry request (see **Ref 4**).

After the flash or SD upload slot has been erased, the file can be uploaded block-for-block. Prior to uploading a new block of file packets, it is necessary to clear the Hole Map on the CubeComputer with a *Reset Upload Block* command.

The file data can now be send in rapid succession of *File Upload Packet* commands (they are not acknowledged by the CubeComputer).

To check if the entire block was received successfully by the CubeComputer application, the remote component should request the 8 Hole Map telemetry frames. If the Hole Map indicates some packets were not received, they should be uploaded again (only the packets indicated by the Hole Map as not received) using the *File Upload Packet* command.

If the entire block has been uploaded, it can be committed to non-volatile storage by sending a *Finalize Upload Block* command. Here the *Offset* and *Block Length* will tell the CubeComputer application where in flash memory to write the data (the *Offset* is ignored for SD card uploads, since the block is simply appended to the file).

The status of the write-to-flash or write-to-SD card operation can be polled using the *Upload Block Complete* telemetry request.

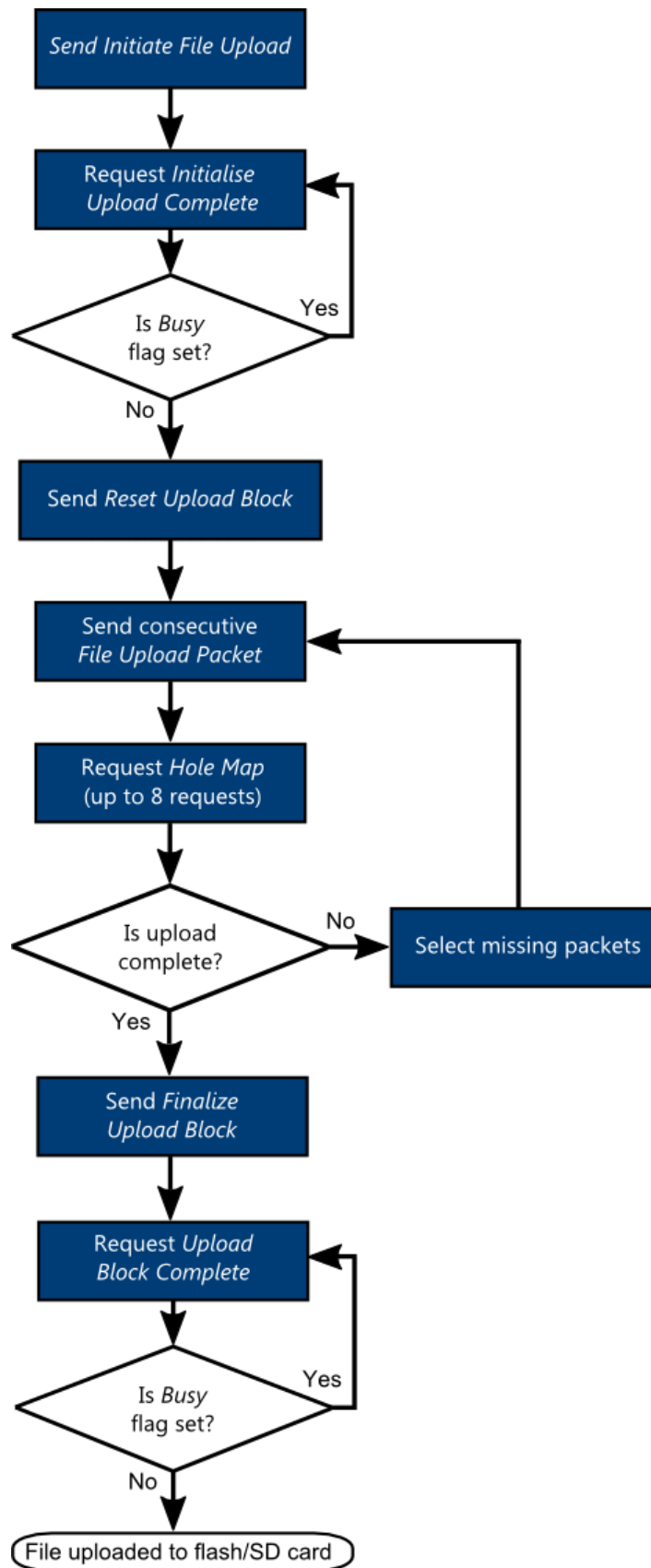


Figure 14 – File upload process.

5.9.6 CRC Checksum

File transfers make use of a CRC16 checksum. The following code can be used to compute the checksum of a buffer.

```
uint16_t CRC_Calc(uint8_t *start, uint32_t len)
{
    uint16_t crc = 0;
    uint8_t *data;
    uint8_t *end = start + len;

    for (data = start; data < end; data++)
    {
        crc = (crc >> 8) | (crc << 8);
        crc ^= *data;
        crc ^= (crc & 0xff) >> 4;
        crc ^= crc << 12;
        crc ^= (crc & 0xff) << 5;
    }
    return crc;
}
```

5.10 GPS Measurements

The CubeControl board has the capability to interface to a Novatel OEM615 GPS receiver module with Special Space Firmware (6.210SII Special Space) and with the SkyFoxLabs piNAV-NG GPS/FM. When enabled, it is possible to read the GPS provided position, velocity and time measurements via TLM request to the ADCS. The GPS receiver measurements are not used or required by the ADCS.

The use of the GPS receiver requires a harness and specific components on the CubeControl board of the ADCS module to be populated. The harness and components are not populated on standard ADCS units. The mechanical standoffs for the GPS receiver are also not attached by default. Users of the ADCS should explicitly request that these components are populated.

Even when the GPS interfacing components are fitted to the ADCS, the GPS receiver itself is never supplied with the ADCS module due to export restrictions.

6 Special considerations

6.1 FOV of sensors

Obstacles in the FOV of the optical sensors like CubeSense and CubeStar, may degrade the quality of the sensor significantly; may prevent the sensor from outputting measurements at all; or in the worst case, may output incorrect measurements that lead to control issues.

6.2 Magnetic loops and solar panels

Current flowing in solar panels can form a loop and cause a magnetic moment vector with Sun dependent direction - see Figure 15. This can cause a disturbance torque that leads to spin-up of the satellite body.

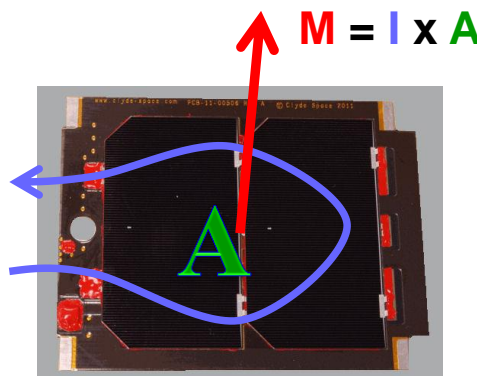


Figure 15 – Solar panel magnetic moment disturbance.

It is therefore advised to make use of back-wiring to cancel the disturbance magnetic moment, see the figure below.

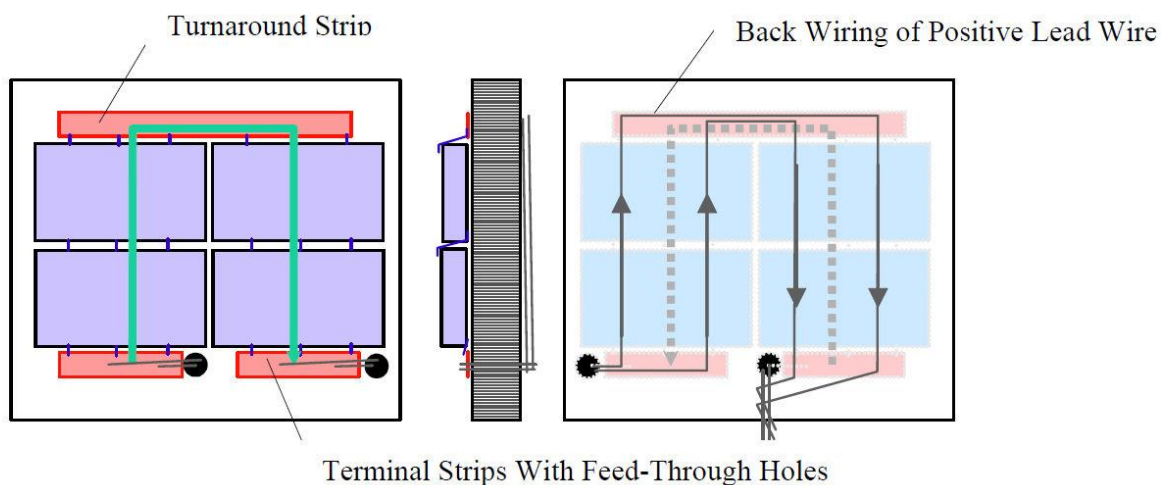


Figure 16 – Back-wiring of solar panels to cancel magnetic moments.

6.3 Configuration and mounting

If controllers are activated with mounting configurations (for sensors or actuators) that do not represent the actual physical mounting of the sensors or actuators, relative to the satellite coordinate frame, the ADCS can have the incorrect effect on the orientation of the satellite. This may, in extreme cases, cause the satellite to spin up to a very high rate, which may be fatal to the satellite mission.

For this reason, it is important to verify that sensors and actuators are configured correctly on the ground before launch. Sensors should be stimulated, and the calibrated vector outputs should be compared with the test setup to confirm that the vectors are correct.

The CubeSense camera and ADCS bundle axes are defined as shown in Figure 17.

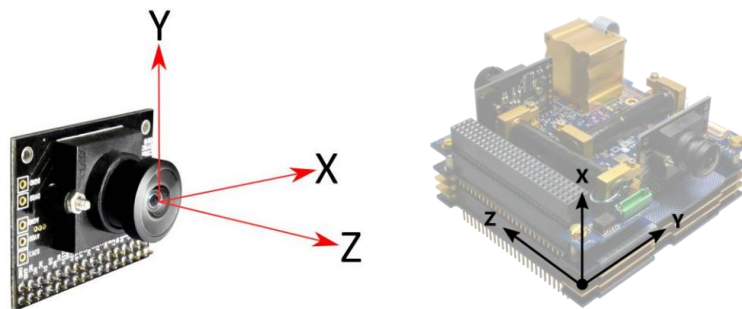


Figure 17 – Camera and ADCS bundle coordinate frame.

The mounting transformations for the Sun and nadir cameras are set in the CubeSupport ADCS *Config* tab. If the transformation is set to $[0, 0, 0]$, the abovementioned camera coordinate frame is used. The mounting transformation must be set to transform the camera coordinate frame to the satellite coordinate frame as used by the ADCS. For example, if the satellite coordinate frame matches the ADCS bundle axes, the transformation for the Sun camera is shown below in Figure 18 together with the Sun sensor stimulated in the bundle.

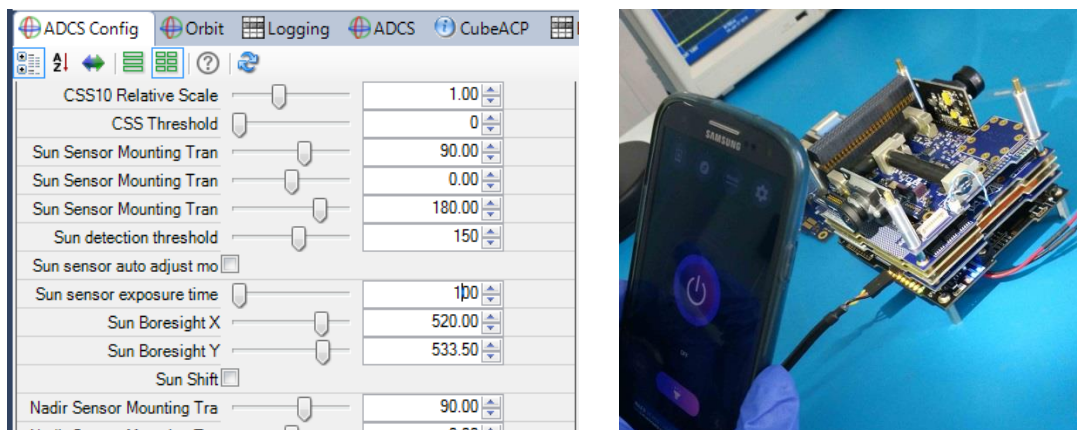


Figure 18 – Camera mounting transformation.

If the Sun sensor is stimulated with a vector positive in all three axes as shown above (45 degrees to the right and upwards), the vector direction of all axes can be verified in the CubeSupport ADCS tab. Figure 19 displays the Sun sensor vector for the above example, with the light in the ADCS bundle positive X, positive Y and negative Z direction.

| Fine Sun Vector | | |
|-----------------|---------|--|
| Sun X | 0.6044 | |
| Sun Y | 0.6350 | |
| Sun Z | -0.4811 | |

Figure 19 – CubeSupport fine Sun vector.

Similarly, actuators' axes must be defined correctly and should be checked. In the CubeSupport ADCS *Config* tab, the ADCS bundle defined axes (1, 2, 3) must be set to the corresponding satellite axes (PosX, PosY and PosZ). Figure 20 is an example of a possible setting.

| Adcs 3-Axis - Configuration | | |
|-----------------------------|------|--|
| ADCS Configuration | | |
| Magnetorquer 1 Configurati | PosY | |
| Magnetorquer 2 Configurati | PosX | |
| Magnetorquer 3 Configurati | NegZ | |

Figure 20 – Magnetorquer mounting transform.

Using the above setting, if a positive X Magnetorquer command is given, the ADCS bundle axis 2 magnetorquer will be activated and can be measured with a compass or external magnetometer as shown in Figure 21 (be aware that the magnetometer pulses on and off).

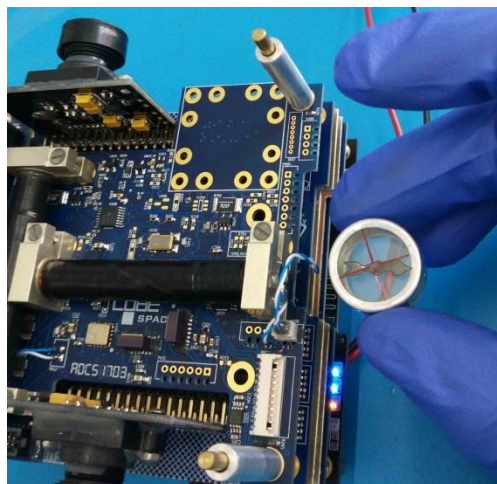


Figure 21 – Magnetorquer axes transformation verification.

6.4 Deployable magnetometer

Placement of the magnetometer is EXTREMELY critical for effective ADCS performance. Placement of the magnetometer next to magnetic parts of the satellite will severely affect the magnetometer, and may compromise the ADCS performance, even to the extent where the ADCS is unusable. Items to avoid having in close proximity to the magnetometer include:

- Electrical motors (reaction wheels, deployment wheels, etc.)
- Batteries
- Steel (or any ferrous) screws
- Torquer rods

6.5 Series resistance of power supply & inrush currents

The CubeADCS unit requires a good power supply to operate reliably. Most satellite EPS systems are designed to handle the power requirements of typical CubeSat subsystems and connects through the PC104 connection.

When the CubeADCS unit is tested on ground though, a normal bench power supply is often used. These power supplies are typically rated for supplying large currents, and do have the capability to power the CubeADCS unit. The supply is, however, usually connected to the ADCS unit through jumper wires, which very often have large resistances or unreliable connections. When the ADCS draws large currents, especially in-rush currents, the resistance of the wire or connection causes a voltage drop. This voltage drop, in many cases, is large enough to cause failures in the CubeADCS electronics. It is therefore recommended to use thick wires to connect bench power supplies to the CubeADCS, and to avoid using leads that have crimp connections.

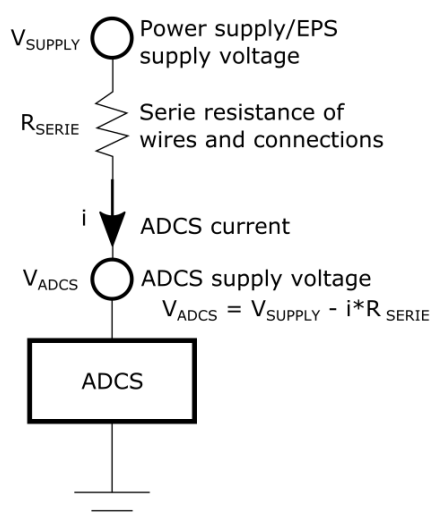


Figure 22 – Power supply to ADCS.

6.6 Y-axis MOI not the largest for Y-Thomson control

In some satellite configurations the Y-axis MOI is not the largest or at best only similar in size to another axis. In these cases, the magnetic Y-Thomson controller (Control Mode 3) will have difficulty to sustain a Y-spin. By commanding a constant Y-wheel momentum (*Set Wheel Speed* TC ID 17), the magnetic Y-Thomson controller will be able to work as expected. A negative Y-wheel speed of 5% to 20% of the maximum speed will be sufficient. The speed required depends on the Y-MOI/max-MOI ratio, e.g. 1 = 5%, 0.8 = 20%.