CUBESPACE

CubeWheel Generation 2

User Manual

DOCUMENT NUMBER	CS-DEV.UM.CW-01
VERSION	1.00
DATE	17/07/2022
PREPARED BY	C. LEIBBRANDT
REVIEWED BY	JM, LV
APPROVED BY	W. Morgan
DISTRIBUTION LIST	External

Revision History

VERSION	AUTHORS	DATE	DESCRIPTION
A	C. Leibbrandt	03/07/2023	Initial draft release
B	C. Leibbrandt	17/03/2023	Update based on review feedback
1.00	C. Leibbrandt	17/03/2023	Update to publication status

Reference Documents

The following documents are referenced in this document.

[1]	CS-DEV.PD.CW-01	CubeWheel Product Description Ver.1.00 or later
[2]	CS-DEV.ICD.CW-01	CubeWheel Interface Control Document Ver.1.00 or later
[3]	CS-DEV.AMNL.BL-01	Bootloader Application Manual, Ver 3.00 or later
[4]	CS-DEV.FRM.CA-01	CP Firmware Reference Manual, Ver 7.00 or later
[5]	N/A	cube-wheel-2/control-program-3.xml (distributed with software package)
[6]	N/A	cube-computer-5/control-program-8.xml (distributed with software package)
[7]	CS-DEV.UM.COL-01	CubeObcLib User Manual Ver 1.00 or later, and CubeObcLib interfacing library source code (distributed with software package)
[8]	CS-DEV.UM.CU-01	CubeSupport HMI User Manual, Ver.1.00 or later

List of Acronyms/Abbreviations

ACP	ADCS Control Program
ADCS	Attitude Determination and Control System
CAN	Controller Area Network
COTS	Commercial Off The Shelf
CSS	Coarse Sun Sensor
CVCM	Collected Volatile Condensable Materials
DUT	Device Under Test
EDAC	Error Detection and Correction
EHS	Earth Horizon Sensor
EM	Engineering Model
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
ESD	Electrostatic Discharge
FDIR	Fault Detection, Isolation, and Recovery
FM	Flight Model
FSS	Fine Sun Sensor
GID	Global Identification
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GYR	Gyroscope
I2C	Inter-Integrated Circuit
ID	Identification
LTDN	Local Time of Descending Node
LEO	Low Earth Orbit
MCU	Microcontroller Unit
MEMS	Microelectromechanical System
MTM	Magnetometer
MTQ	Magnetorquer
NDA	Non-Disclosure Agreement
OBC	On-board Computer
PCB	Printed Circuit Board
RTC	Real-Time Clock

RWA	Reaction Wheel Assembly
RW	Reaction Wheel
SBC	Satellite Body Coordinate
SOFIA	Software Framework for Integrated ADCS
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SSP	Sub-Satellite Point
STR	Star Tracker
TC	Telecommand
TCTLM	Telecommand and Telemetry (protocol)
TID	Total Ionizing Dose
TLM	Telemetry
TML	Total Mass Loss
UART	Universal Asynchronous Receiver/Transmitter

Table of Contents

Table of Tables

Table of Figures

1 Introduction

This document serves as a general introduction and manual to operating the CubeWheel.

Momentum and reaction wheels are used to exchange angular momentum with a satellite body. This momentum exchange induces a control torque, which can be used to change the satellite's attitude or to absorb disturbances from the space environment (e.g. aerodynamic disturbance). Momentum/reaction wheels are commonly used in satellites that have moderate to high pointing accuracy requirements.

1.1 Document applicability

This version of the document applies to the CubeWheel hardware-, software- and firmware versions as stated in Table 1 below.

Table 1: Document applicability

ELEMENT	VERSION	NOTES
Hardware	M2.0E2.3	Minimum versions applicable.
Firmware	Control-program-cube-wheel ver. 3.1.25 Flash-bootloader-52 ver. 1.7.4	

1.2 Scope

This document assumes the user has already read [1] and [2], typically supplied to a client / user before the User Manual (this document) is provided.

This document is to be used in combination with the referenced documents to: (a) unpack, assemble, integrate and (b) test (conduct a CubeWheel health-check) after receipt.

This document should be studied in full before the (new) user starts working with the CubeWheel. Ensure that all instructions and intended use cases are understood before any action is taken.

As an overview, this User Manual addresses handling precautions applicable to the unit in chapter 2.

A "Getting started" chapter is provided in chapter 3 which guides the user to unpack the CubeWheel while subsection 3.3 details interfacing to it, addressing the necessary power supply and communication harnesses and establishing initial communication. Once the user has completed the "Getting started" chapter, the CubeWheel is correctly set-up for the Health Check test.

Chapter 4 describes a select subset of functions and features offered by CubeWheel focussed on guiding the user to be able to successfully assemble and integrate the CubeWheel for a complete standalone Health Check before fitment into the supersystem / satellite.

CubeSpace is continually innovating on designs and device capabilities. Since the firmware for the CubeWheel is always improving, CubeSpace will release updated software from time to time. Chapter 4.6 explains how to reprogram such an update to the CubeWheel.

Chapter 5 provides a health check sequence that establishes that the CubeWheel is "alive and well" after being delivered to the user. The steps to perform this test are explained and tables are provided with expected results and space for the user to capture the actual results obtained.

The user is requested to complete the Appendix: Health Check/Acceptance test **and to return a digital copy of it to CubeSpace for record keeping purposes**

2 Handling

Anti-static: The unit contains several static sensitive components. Therefore, the appropriate electrostatic discharge (ESD) protection measures must be implemented. The unit must never be handled without proper grounding.

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

Moisture: The unit should be kept free of moisture and liquids. These could have corrosive effects on the electronics and electronic joints, leading 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 altogether.

Housings: The aluminium housing of the different components should NOT be tampered with. Tampering with the housings may damage the components inside. No attempt should be made to loosen or remove the fasteners that secure the housings.

3 Getting started

This section describes what is required from the user's Lab and what support equipment is provided by CubeSpace to be able to start using the CubeWheel.

The following items are required before the user can get started with unpacking and setup of the CubeWheel:

- Multi-meter or oscilloscope
- Computer with an open USB port (running Windows 7 or later)
- Power Supply which can supply a voltage range of 10 V to 17.6 V at a current limit set to 1.5A

The CubeWheel hardware is packaged in anti-static bags inside a protective shipment case. Remove the unit in 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.

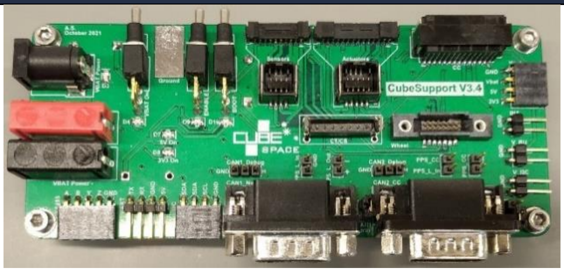
3.1 Ground Support Equipment (GSE)

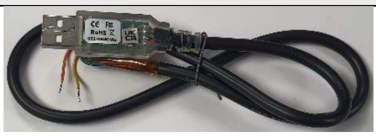
In addition to the CubeWheel itself, the package will contain some or all the following GSE:

- CubeSupport PCB
- Interface harness (CubeProduct dependant)
- DC Power Cable
- PCAN-USB (optional)
- UART-to-USB cable (optional)
- I2C-to-USB cable (optional)
- RS485-to-USB cable (optional)

These GSE items are shown in Table 2. Remove all items from their anti-static bags and place them on the bench.

Table 2: Typical GSE items in CubeADCS package

GSE ITEM	IMAGE
CubeSupport PCB	
Interface Harness (CubeProduct Dependant)	

GSE ITEM	IMAGE
DC Power Cable	
PCAN-USB: CAN Interface for USB (optional)	
FTDI UART to Serial USB cable (Optional)	
FTDI I2C to USB cable (Optional)	
FTDI RS485 to USB cable (Optional)	

3.2 CubeSupport PCB introduction

The purpose of the CubeSupport PCB is to serve as the main interface for a user with CubeSpace products, which may include standalone sensors and (some) actuators as well as a CubeADCS bundle. However, in this CubeWheel user manual, the CubeSupport PCB serves as the main interface between the client's computer and the CubeWheel.

The CubeSupport PCB receives power from a bench power supply via the provided DC power cable. The communication interfaces available on the CubeSupport PCB are CAN, UART, I2C and RS485/RS422. Depending on the CubeWheel communication configuration, connected to the CubeSupport PCB, the corresponding communication connection must be made to an interfacing computer.

The available headers and ports on CubeSupport PCB are shown in Figure 1 below.

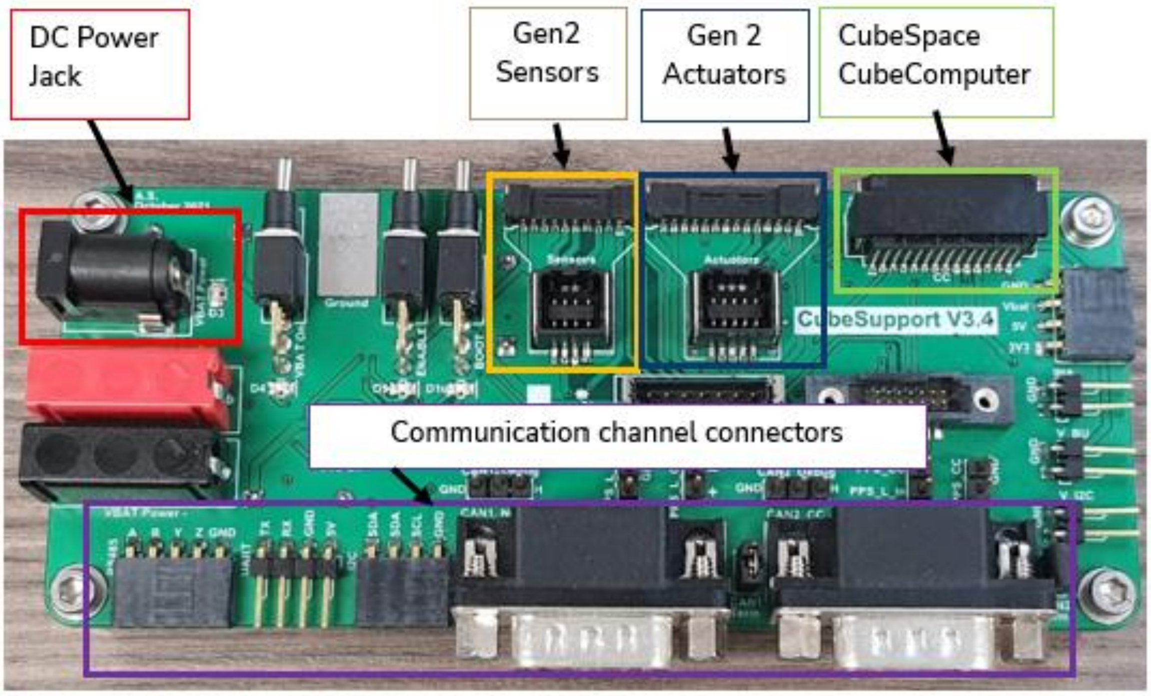


Figure 1: CubeSupport V3.4 PCB

3.3 CubeSupport application introduction

The CubeSupport application, a Windows-based software application is also supplied with the CubeWheel. The CubeSupport application can be opened by running the executable file located in the software release bundle under "\tools\cube-support\cubesupport.exe".

For further details on using the CubeSupport application please refer to the dedicated document referenced as [8].

3.4 Connecting to the CubeWheel

3.4.1 Power setup

Before connecting the bench power supply to the CubeSupport PCB, set the power supply voltage to 10.0V (**the recommended voltage to set for the health check is 10.0 V**) and set the current limit to at least 1.5A. Keep cables between the power supply and the CubeSupport PCB short and thick to reduce series resistance.

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

Before connecting the power source, ensure that all toggle switches on the CubeSupport PCB are in the OFF (down) position. (The toggle switches control the power to CubeWheel along with the Enable and Boot signals).

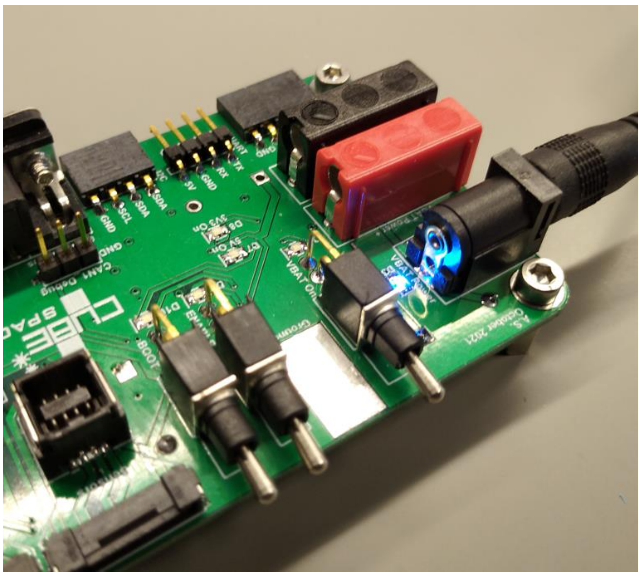


Figure 2: "VBAT On", "BOOT" and "Enable" switches in the OFF position (down)

Note that the **Enable signal** might not be used for CubeWheel, depending on the order configuration.

The DC power cable has one black wire and another wire with a white dashed line on it. The black wire is ground, and the wire with the white dashed line is the supply voltage. Connect these two wires to the bench power supply, the other side of the cable must be plugged into the "VBAT Power" mating connector on the CubeSupport PCB.

With the DC cable connected to the CubeSupport PCB and the power supply switched on, the "VBAT Power" LED will turn on to indicate that power is present. The "VBAT on" switch can now be pushed to the ON position (up). The "VBAT On", "5V On" and "3V On" LEDs on the CubeSupport PCB will turn on, as shown in Figure 3, indicating that power is present and that 3.3V is regulated on the PCB. If at this stage all four LEDs are not turned on, please contact CubeSpace.

Turn off the "VBAT on" switch by pushing it to the OFF (down) position and observe that the LED's turn off (except the "VBAT Power" LED).

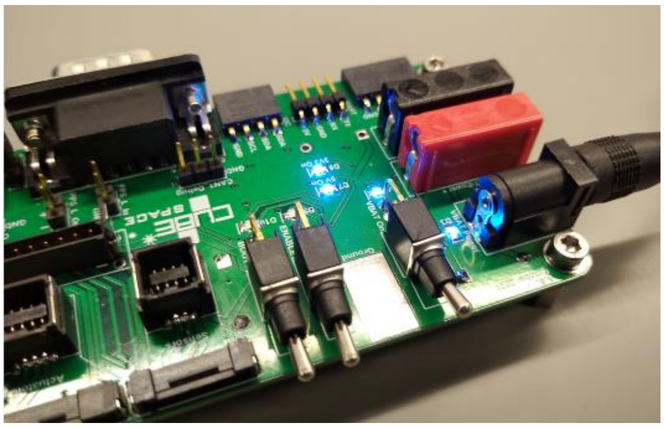


Figure 3: CubeSupport PCB with the DC cable connected, the "VBAT On" switch in the ON (up) position, and the LEDs turned on to indicate power-on condition.

The “VBAT On” switch can now be used along with the enable switch (if the enable line is connected to the CubeWheel) to disable and enable the CubeWheel.

The CubeSupport board has a diode protection circuit which induces a voltage drop on the “VBAT” line. The input voltage is not the same as the voltage that is applied to the actuator headers. Pin 2 (VBat) on the Vout header can be used to measure the voltage supplied to the actuator.

3.4.2 Communication setup

3.4.2.1 CAN and UART

CubeSupport PCB breaks out the communication interfaces to user-friendly headers. The CubeWheel has two main communication interfaces available, CAN and UART. These channels are made available on CubeSupport PCB as the “CAN1 Nodes” and “UART” headers respectively, as indicated in Figure 4. A PC can interface with the CubeWheel through either of these channels.

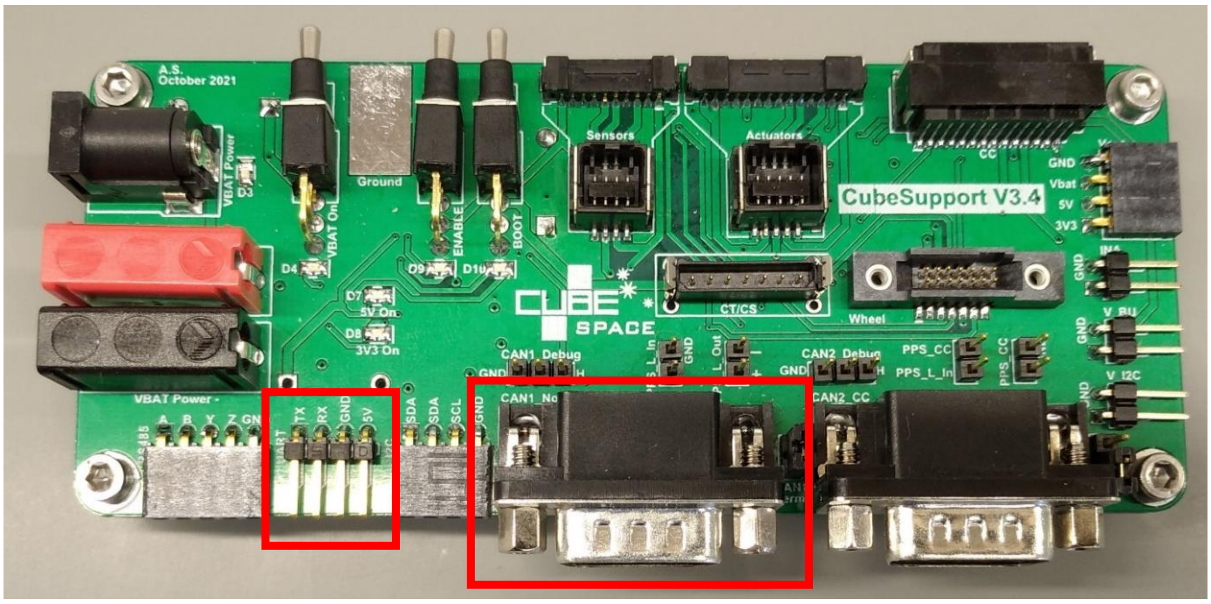


Figure 4: CubeSupport PCB “UART” and “CAN1 Nodes” headers indicated.

Note: **The CAN interface is the default communications channel**, the UART interface is normally not used and usually remains unconnected. It can be used for reprogramming the CubeWheel software, or as a secondary communications interface.

The PCAN-USB cable to connect to the “CAN1 Nodes” header and the UART to Serial FTDI USB cable to connect to the “UART” header are supplied with the CubeWheel. These two cables allow the user’s PC to interface via the CAN- and/or UART communication channels.

Note that the UART to Serial FTDI USB cable is terminated in a three-pin female header containing the GND-, Tx- and Rx connections while the UART header on the CubeSupport PCB is a four-pin male header that contains an additional 5V pin that is not used. Ensure that this cable is correctly connected to the CubeSupport PCB, as indicated in Figure 5, to avoid any damage.

Connect the applicable communication interface/s as show in Figure 5.

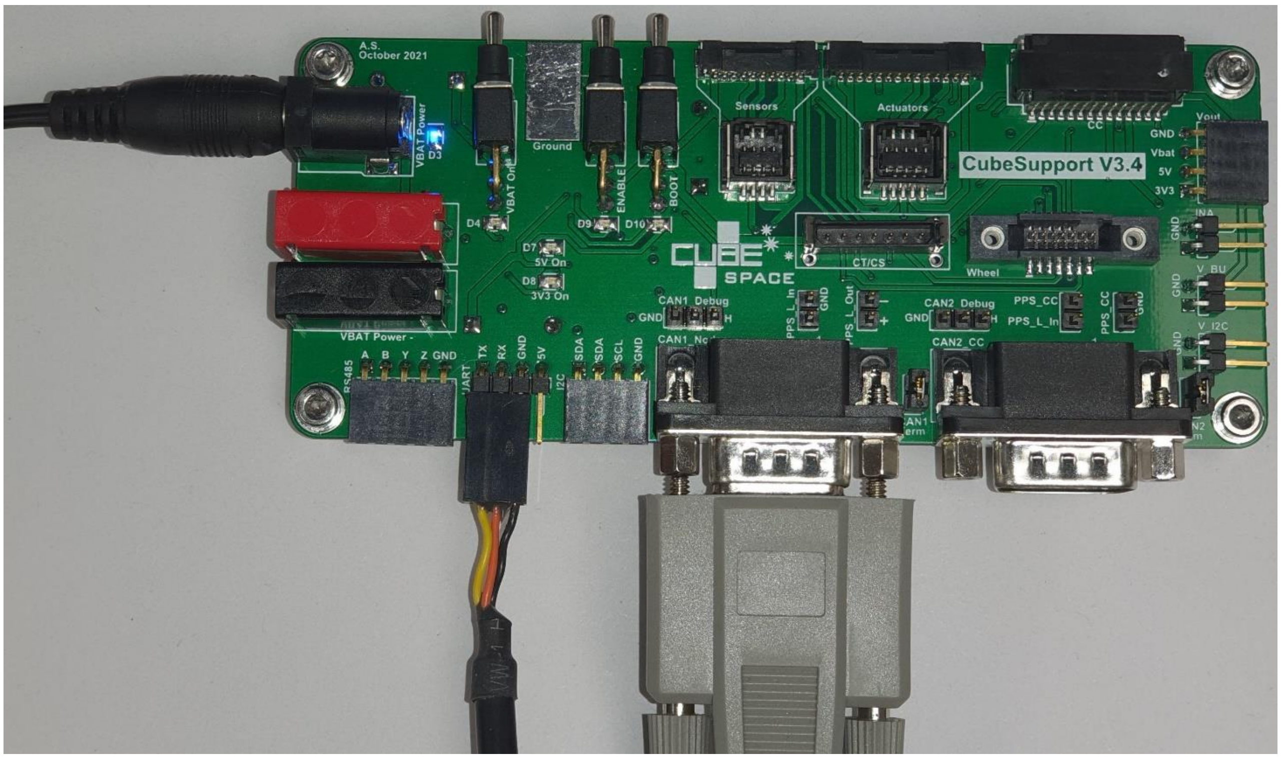


Figure 5: CubeSupport PCB with UART and PCAN-USB connections shown.

To connect the CubeWheel to the CubeSupport PCB, the supplied CubeWheel interface harness can be used.

Generally, the harness has two ends that should be plugged into the CubeWheel itself and into either the "Sensors" or the "Actuators" header on the CubeSupport PCB, as shown in Figure 6. For the CubeWheel, connect to the Gen2 "Actuators" header.

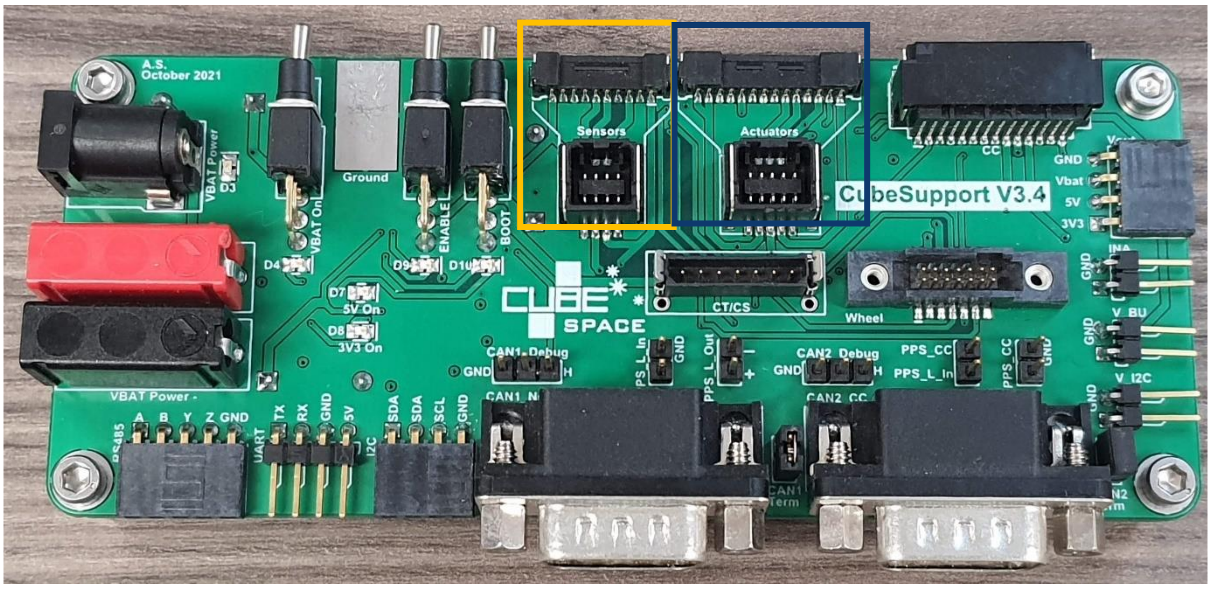


Figure 6: CubeSupport PCB "Sensors" and "Actuators" header

3.4.2.2 I2C

Communication via I2C is not included as part of the standard CubeWheel offering but is available as an option. If there is a need to use a I2C connection between the CubeWheel and the client ADCS / OBC (i.e.

and thus also via the CubeSupport PCB and the CubeSupport application for initial health testing), customers can contact CubeSpace for the necessary details.

Both the CubeWheel and the CubeSupport PCB hardware support I2C communication if so specified at the time of CubeWheel order placement.

3.4.2.3 RS485

Communication via RS485 is not included as part of the standard CubeWheel offering but is available as an option replacing the UART functionality. If there is a need to use a RS485 connection between the CubeWheel and the client ADCS / OBC (i.e. via CubeSupport PCB and the CubeSupport application for initial health testing), customers can contact CubeSpace for the necessary details.

Both the CubeWheel and the CubeSupport PCB hardware support RS485 communication if so specified at the time of CubeWheel order placement.

3.4.3 Communicating with the CubeWheel using CubeSupport PCB and -Application

Following the steps described in sections 3.4.1 and 3.4.2, the CubeWheel should now be setup in such a way to allow the user to be able to power it.

The user is now referred to [8] for a description of how to communicate with the CubeWheel from a PC via the CubeSupport PCB using the *CubeSupport* application.

Once the (new) user has physically worked through this "Getting started" section, the user is requested to proceed with the system health check as defined in chapter 5 Appendix: Health Check/Acceptance test**.**

4 Functions and Features (Some needed on client ADCS / OBC)

This chapter is written explaining how the CubeADCS, and in particular the CubeComputer, would be used to interact and communicate with the CubeWheel. **In the case of the stand-alone CubeWheel, the client ADCS / OBC should implement similar functionality to ensure optimal usage of the CubeWheel within the client satellite / ADCS.**

This chapter describes a select subset of functions and features offered by the CubeWheel focussed on guiding the user to be able to successfully prepare the CubeWheel for integration into the client satellite / ADCS.

The functionality needed on the client ADCS / OBC as described in this chapter is not yet needed to be able to conduct the complete standalone Health Check (see chapter 5 “Appendix: Health Check/Acceptance test”) but is needed before fitment into the supersystem / satellite.

For a summary list of all functions and features offered by the CubeWheel, the user is referred to the CubeWheel Product Description document (see [1]).

For a full list and description of all functions and features offered by the CubeWheel and for commissioning purposes, the user is referred to [4] and [7]

> Note that in some cases, specific Telecommands (TC) and Telemetry Requests (TLM) are included in the text following below. Such TC or TLM messages are highlighted for the user by the following style “`Telecommand Name`”. Telecommand and telemetry names are identified by their “Code Names” in the API documentation, with an optional friendly name in brackets, for instance: `Version` (Firmware Version). The code name identifies the particular telecommand or telemetry message in the API definitions [5] and [6]. The same information is also available in the generated HTML documentation tables, distributed as part of the software package.
>
> Note that a software library is provided with all CubeProducts which implements low-level formatting of communications messages to the CubeADCS and individual CubeProducts. The library serves both as example, and as source code that may be included in the supersystem program code to facilitate integration with the CubeWheel. For details, please see [7].

4.1 Identification

Each CubeSpace CubeProduct, including the CubeWheel, is programmed with a serial number, and details about its firmware.

The serial number of a CubeProduct is important as each CubeProduct may have specific calibration values associated with it, i.e. for a specific serial number. Further, knowledge of which CubeProducts and their associated serial numbers are utilised in a specific client ADCS bundle, allows CubeSpace to accurately maintain configuration records for the CubeWheel, in turn allowing configuration specific support to the client when needed.

Similarly, knowledge of the firmware (versions) programmed onto a CubeProduct is important to allow correlation with the versions of firmware available in the software release packages, and to confirm firmware updates when done.

To check these details about the CubeWheel, the `Identification`, `SerialNumber` and `Version` TLM can be requested. The content of these TLM frames can be referenced in [5].

The client ADCS / OBC should implement a similar CubeWheel identification procedure.

4.2 Configuration

The CubeADCS makes use of configuration files to define the behaviour and setup for the firmware programs utilised by itself and the CubeProducts making up the system. This includes settings that define the hardware specific options, enabling specific communications interfaces, changing protocol, baud rates, which the CubeProducts expects as part of the Auto-Discovery feature etc.

Configuration files are supplied as part of CubeWheel software releases. The configuration files are binary coded versions of configuration settings and are not meant to be viewed or modified by users. For more information on uploading and changing configuration files, please consult reference [3].

The ADCS configuration also includes default values for a subset of settings that may be changed via telecommand. These commands also have matching "get" telemetry requests, to make it easier to modify single values in the larger command message (i.e., read config telemetry, modify a specific parameter only, and send command with updated config).

The CubeWheel configuration messages are listed in Table 3.

Table 3: Configuration messages (matching TCs and TLMs) for CubeWheel configuration settings.

COMMAND/TELEMETRY NAME	CONTENT
MainGain (Main Gain)	Main speed controller gain values
BackupGain (Backup Gain)	Backup speed controller gain values
PWMGain (PWM Gain)	General PWM gain

Changes that are made to the configuration via telecommand are only temporary – until the CubeWheel is reset or power cycled.

The client may request a configuration file update should they want to permanently update configuration parameters such as UART baud rates and communication addresses.

4.3 Power management

The CubeWheel has its own power switch embedded, and it is controlled using an externally controlled enable signal. The enable signal should be connected to the client ADCS / OBC computer, and the client ADCS / OBC must implement the necessary software, so that the power to the CubeWheel can be enabled or disabled as appropriate. Even when automatic power switching is implemented on the client ADCS / OBC based on operating mode, it is highly recommended that low-level functionality to toggle to the enable signal state is available, using direct telecommands to the OBC.

4.4 Bootloader

CubeSpace provides its own Bootloader application for each CubeProduct which executes before the control program. This Bootloader provides an interface consistent with the control program and allows for easy update of the configuration and control program for the entire CubeADCS and the CubeWheel in particular. This interface is also supported by the CubeSpace's CubeSupport application (see [8]).

The `Identification` telemetry frame can be used to identify whether the bootloader or the control program is currently running. The `ProgramType` is a numeric value that is unique to the specific application. Alternatively, the BootStatus telemetry frame can be used to discern if the bootloader is running. The `BootState` will provide more detail of the boot process than the `ProgramType` but can be used for the same purpose. The CubeADCS firmware and command and telemetry interface is described in detail in [4], the accompanying interface definition files [5] and [6], as well as the HTML documentation tables that are generated from the latter references, distributed as part of the software bundle.

For information about the bootloader application(s), refer to [3].

The client ADCS / OBC should implement a similar CubeWheel bootloader procedure.

4.5 Auto-Discovery

The integrated CubeADCS includes an "Auto-Discovery" feature which is implemented as part of the CubeComputer bootloader.

"Auto-Discovery" refers to the CubeADCS bootloader's ability to automatically determine which CubeProducts are present, as well as which port each CubeProduct is connected to, without user intervention.

During auto-discovery, a CubeProduct will identify itself using the normal identification telemetry mentioned in 4.1.

This feature allows the user to connect any CubeProduct to the CubeConnect port as part of the CubeComputer, that best suits their mechanical design, without the need for any changes to the bootloader firmware configuration. The CubeProducts that are discovered in the system are compared to the CubeProducts that are expected in the ADCS system.

An Auto-Discovery is successful if the discovered CubeProducts match the expected CubeProducts. The expected CubeProducts are listed as a bootloader configuration item per CubeADCS system. This configuration is managed by CubeSpace.

For a detailed discussion on the Auto-Discovery feature, refer to [3].

The client ADCS / OBC should consider implementing a similar CubeWheel auto-discovery procedure if this feature could add value to the client's (unknown to CubeSpace) architecture.

4.6 CubeWheel functions and features

4.6.1 Configuration

The CubeADCS makes use of configuration files to define the behaviour and setup for the firmware programs. This includes settings that define the hardware specific options, enabling specific communications interfaces, changing protocol, baud rates. Each CubeWheel also makes use of configuration files to define the behaviour of the respective CubeWheel.

Configuration files are supplied as part of software releases. The configuration files are binary coded versions of configuration settings and are not meant to be viewed or modified by users. For more information on uploading and changing configuration files, please consult reference [3].

The CubeWheel configuration also includes default values for a subset of settings that may be changed via telecommand. These commands also have matching "get" telemetry requests, to make it easier to modify single values in the larger command message (i.e., read config telemetry, modify a specific parameter only, and send command with updated config).

The CubeWheel configuration messages are listed in Table 4.

Table 4: Controller Gains

Command/Telemetry Name	Content
`BackupGain`	The gains of the backup speed controller
`MainGain`	The gains of the main speed controller (i.e. when backup mode IS NOT enabled)
`PWMGain`	The gain of the PWM register that commands the motor driver

4.6.2 Control Modes

The CubeWheel has different control modes in which the wheel operates, this control mode may be accessed by using the `ControlMode` TLM. The content of this TLM frame can be referenced in [5]. Table 5 lists the control modes and functions.

Table 5: Control Modes

ControlMode	Function
Idle	The initial start-up state, during which the unit has the lowest power consumption. The power switches to the motor and other non-critical components are switched off
No Control	Power to the motor is switched off, but sensor feedback (i.e. speed measurements) is still maintained. Note that during this state, the motor current measurements are invalid. The backup speed measurements are also invalid because the motor is switched off
Duty Cycle Control	A duty cycle command will activate the Duty Cycle state. This state will maintain a reference control signal to the motor.
Speed Control	The speed of the wheel can actively be controlled (used either the primary or the secondary speed measurement as feedback) if the Speed Controller state is enabled. The speed reference can be set via telecommand.*

*Speed control mode can be used in backup mode where only the motor hall sensors are used or in not backup mode in which two forms of speed measurements are used.

The backup mode can be set and retrieved using the TC/TLM `BackupMode` referenced in [5].

4.6.3 Backup Mode

Redundancy is incorporated through the availability of two speed measurements from the encoder, and from the Hall sensors. The speed control algorithms are therefore not only reliant on the functioning of the encoder, but it can also use the Hall sensors to follow the given reference speed.

The user has the ability to enable or disable the so-called backup mode, which will switch off the encoder when activated via telecommand `BackupMode`. Backup mode is disabled by default.

One drawback of the backup mode is the inability to determine the direction of rotation autonomously. The user should therefore refrain from changing the reference direction of rotation too often when in backup mode. The CubeWheel unit will detect a change in direction and will automatically command a zero-duty cycle for 30 seconds before controlling the wheel speed to the new reference speed.

4.6.4 Error Codes

The wheel returns a general error flag in the status telemetry that is set when any error has occurred. To further investigate what caused the error, the user should request the Error Flags telemetry. This telemetry will return a byte where every bit represents a unique error that could be set.

The **Invalid Telemetry** error is set when any of the telemetry requests that the wheel has received is not part of its defined ID set.

The **Invalid Telecommand** error is set when any of the telecommands received is not recognised or of incorrect length.

The **Encoder Error** is set when the magnetic wheel encoder fails to return a valid measurement. The wheel will automatically be set to Idle state and be allowed to free wheel. The user has the choice to switch over to Backup control mode to still resume controlling the wheel speed using the Hall sensors built into the motor.

The **UART Error** is set when the UART state machine enters a fault condition. This may happen when a stop condition is received before a start condition.

The **I2C Error** is set when the I2C state machine enters a fault condition. This may happen when a stop condition is received before a start condition.

The **CAN Error** is set when the CAN state machine enters a fault condition. This may happen when a stop condition is received before a start condition.

The **Configuration Error** is set when there is a mismatch between the control program configuration and the bootloader configuration.

The **Speed Error** is set when the referenced speed is not reached within a timeout or there is a measurement mismatch between the two sensors.

The listed errors can be retrieved using the `getErrorFlags` TLM with the data of the frame referenced in [5].

4.6.5 EMI Shielding

The EMI of the motor is shielded by the flywheel of the reaction wheel as well as a Mu-Metal lining placed on the inside of the motor casing.

4.6.6 Input Protection

The battery voltage going to the motor driver and the 3v3 voltage to the MCU and peripherals have input protection. These power subsystems will be shut down if the voltage is outside of the tolerable ranges as defined in the ICD [2].

4.6.7 Qualification Features

The CubeWheel is qualified for vibration, shock, thermal, thermal vacuum and radiation. The CubeWheel is designed to have a low imbalance limiting jitter with a 5-year design life.

4.6.8 Limitations

The CubeWheel is limited to a maximum speed of 10000rpm. Due to the size of the wheel, the momentum storage of the wheel is limited. The maximum momentum storage for each wheel can be found in the ICD [2]. The CubeWheel CW0162 and smaller are speed controlled products and do not have torque control.

4.6.9 CubeWheel Typical use case

In an integrated ADCS typical control loop that makes use of speed control for the reaction wheels, the control of the wheel will follow the procedure as in Table 6. Prior to this loop, the CubeWheel must be powered on as described in section 4.3, and it must be verified that the CubeWheel control program has started through the identification process described in section 4.1.

Table 6: CubeWheel Typical Usage during Speed Control loop in integrated ADCS

Action performed by ADCS computer	Command / telemetry request to CubeWheel	Action performed by CubeWheel
Read the current wheel speed by requesting the `WheelSpeed` telemetry frame	Request `WheelSpeed` telemetry	Respond with most recent measured wheel speed telemetry
(Estimate current attitude using sensor inputs)		
Calculate a torque demand from the wheel(s) based on the current estimated attitude and reference attitude		
Calculate updated wheel reference speed, from torque demand and current wheel speed		
Command new wheel speed	`WheelSpeedRef` telecommand	Accelerate or decelerate wheel to meet the new reference speed
Delay until the start of the next control iteration		

Note that the CubeWheel also accepts low level telecommands to enable the motor driver. In the above scenario it is not required to send this command, because the `WheelSpeedRef` telecommand will automatically enable the motor driver if it is not already on.

4.7 Mounting the CubeWheel in the satellite

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.

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

4.8 Programming and Updating of the CubeWheel

The user does not have access to the low level (SWO/JTAG) programming pins of each MCU in the CubeADCS nor to the programming pins on the CubeWheel. Instead, all CubeProducts are pre-loaded with a bootloader that allows reprogramming of its control program firmware.

The Bootloader application is programmed in the start position of Flash for each CubeProduct. This Bootloader provides an interface consistent with the control program and allows for easy update of the configuration and control program for the entire CubeADCS including the CubeWheel. This interface is also supported by the CubeSpace's CubeSupport application (see [8]). This interface and update method works for all communication interfaces and no additional- or special hardware connections are required. This makes this method to update the CubeADCS and the CubeWheel in particular, the preferred method in orbit since it can be easily implemented by the client ADCS / OBC.

4.8.1 When to Update

The user will need to reprogram or update the CubeWheel for the following scenarios:

a) A next version of a control program for the CubeWheel is available (Update)
b) A next version of configuration for the CubeWheel is available (Update)
c) A combination of the above

4.8.2 Bootloader updates

The CubeSpace bootloader itself is also upgradeable, should a new version become available with features specific to the bootloader. All CubeProducts have a manufacturer provided ROM bootloader. This provides an alternative method for updating firmware; however, this interface is only accessible on the ground. Details on reprogramming the CubeSpace bootloader through the ROM bootloader is supplied in [3].

4.8.3 Software release package

CubeSpace supplies a software release bundle together with the initial hardware delivery of the CubeWheel that contains the latest firmware-, bootloader- and configuration file versions of the CubeWheel that should correspond to the firmware versions on the CubeWheel. Subsequent software release bundles are also made available to offer new functions and features, address earlier software issues and so on.

Each software release bundle contains a number of files:

- The firmware-, configuration- and bootloader files can be found under `<…\sw-bundle-vxxx\firmware>`.
- The CubeProducts are available as subdirectories below it, for example `<…\sw-bundle-vxxx\firmware\cube-computer-5>`
- A subdirectory containing the control program (FW) and configuration file(s) for the CubeProduct is available, for example `<…\sw-bundle-vxxx\firmware\cube-computer-5 \control-program>`
- Another subdirectory containing the bootloader application FW for the CubeProduct is available, for example `<…\sw-bundle-vxxx\firmware\cube-computer-5\flash-bootloader>`.

Each software release package also contains documentation to explain what is contained in the release.

4.8.4 Steps to update control application and configuration through CubeSpace Bootloader

To update the configuration or the application software of the CubeADCS and the CubeWheel in particular, it must first be reset, or power cycled.

Upon power-on or reset, the bootloader application will run first. The bootloader uses a backoff timer and will listen for incoming commands for **5 seconds** before jumping to (starts executing) the CubeWheel application control program. The user must send a `Halt` (Halt bootloader) command within 5 seconds of power-on or reset (this happens when opening the connection, as discussed in section 3.4.3).

The boot process is then halted and the CubeWheel will remain in the bootloader application until a `JumpToApp` (Jump To Application) or `Reset` command is received.

The CubeWheel is now ready to be programmed. The Bootloader Application Manual (see [3]) provides detailed information for interfacing with the CubeSpace bootloader to update the configuration and control application firmware.

4.8.5 CubeObcLib software interfacing library

The client ADCS / OBC should implement a similar CubeWheel reprogramming and update procedure. The CubeObcLib library of software functions (see [7]) provides an implementation of the firmware upgrade procedure and may be incorporated in client ADCS / OBC source code for convenience and to assist with client ADCS / OBC software development.

5 Appendix: Health Check/Acceptance test

This Appendix first requests user contact information and then provides a sequence to perform a "health-check" / Acceptance test on the CubeWheel to confirm that it is working as it should post physical receipt at the client premises.

Please complete this Appendix Health Check / Acceptance test and return a digital copy of the whole appendix to CubeSpace along with copies of the captured image(s) and test pattern(s) as relevant to the CubeWheel received.

Please keep in mind that all required safety measures must be taken when handling the CubeWheel. ESD protection must always be in place. Please use gloves when handling the CubeWheel and ensure that the user and the surface worked on is properly grounded. Flight Models (FM) must always be handled in a dust-free environment such as a cleanroom.

It is essential to perform the basic CubeWheel health check procedures upon receipt of either hardware or software to ensure the unit is in complete working condition before further testing/integration occurs.

5.1 User and CubeWheel details

Please complete Table 7. This table will allow CubeSpace to identify the user using the CubeWheel.

Table 7: User and CubeWheel Details

Information needed	Information supplied
Company/university/institution:	
Contact person:	
Email of contact person	
Satellite name:	
CubeWheel Serial number	
Software package version:	
Person(s) responsible for Health Check testing	
Date Health Check was performed	

5.2 Connection to CubeWheel with CubeSupport software

Following the steps described in section 3.3, the CubeWheel should be setup in such a way to allow the user to power it on via the CubeSupport PCB and communicate with it from a PC using the *CubeSupport* application, a Windows-based software application also supplied with the CubeWheel and described in detail in referenced document [8].

The *CubeSupport* application can be opened by running the executable file located in the software release bundle under "\tools\cube-support\cubesupport.exe".

<u>The UART connection is not required at this stage, only CAN.</u>

In the steps below, commands will be sent to the CubeWheel from the *CubeSupport* application via the CubeSupport PCB.

5.3 Current measurements and expected results interpretation.

For the sections that follow where the expected current consumption values are given, the current was measured from a bench power supply set to **12.9V**. Unless stated otherwise, the value given as the expected current is the current increase that should be shown by the power supply. For example:

> With the CubeWheel plugged in and the "VBAT On" switch in the "Off" position, the current consumption is ~1 mA due to the "VBAT On" LED on the CubeSupport PCB.
>
> When switching the "VBAT On" switch to the "On" position to power on the CubeWheel, the current consumption increases by ~108 mA, for a total of ~109 mA, if the CubeWheel was not optioned with an Enable line present. If the CubeWheel has the Enable line present, the "Enable" switch must be switched to the "On" position for the CubeWheel to be powered on and draw the mentioned current. Note that ~19mA of the drawn current is due to LEDs on the CubeSupport PCB, and the remaining ~90 mA is due to current drawn by the CubeWheel unit through the 3.3V linear regulator on the CubeSupport.

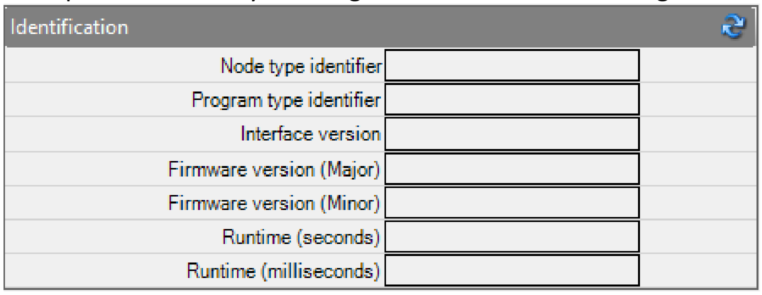
5.4 CubeWheel health check

Refer to [8], and in particular to the CubeWheel User Interface section there-in to open the main CubeWheel User interface page.

If more than one CubeWheel was received, please repeat the test steps for each and indicate actual results per CubeWheel in the tables below.

Follow the steps below to perform the CubeWheel health check.

1. Find and note the **firmware version** numbers that should be loaded on the CubeWheel by navigating to "**firmware/cube-wheel-2/control-program**" in the software release bundle.
The file of interest is named "**app-control-program-CubeWheel-X.Y.Z-freertos-release**", where "X.Y.Z" denotes the firmware version. "X" corresponds to the Major firmware version, "Y" corresponds to the minor firmware version and "Z" denotes the patch version.
2. Similarly, the CubeWheel firmware's **interface version** must also be verified.
This version number can be found by navigating to "**libcubeobc\api\CubeWheel\**" in the software release bundle. This location contains a folder named "**control-program-X**", where "X" denotes the CubeWheel interface version. Note this version.
Complete Table 8 by locating the Identification message:


Identification

Node type identifier	
Program type identifier	
Interface version	
Firmware version (Major)	
Firmware version (Minor)	
Runtime (seconds)	
Runtime (milliseconds)	

and refreshing the relevant data under the CubeWheel tab of the CubeSupport application. The refresh button can be found here .

Table 8: CubeWheel firmware versions: Expected Results

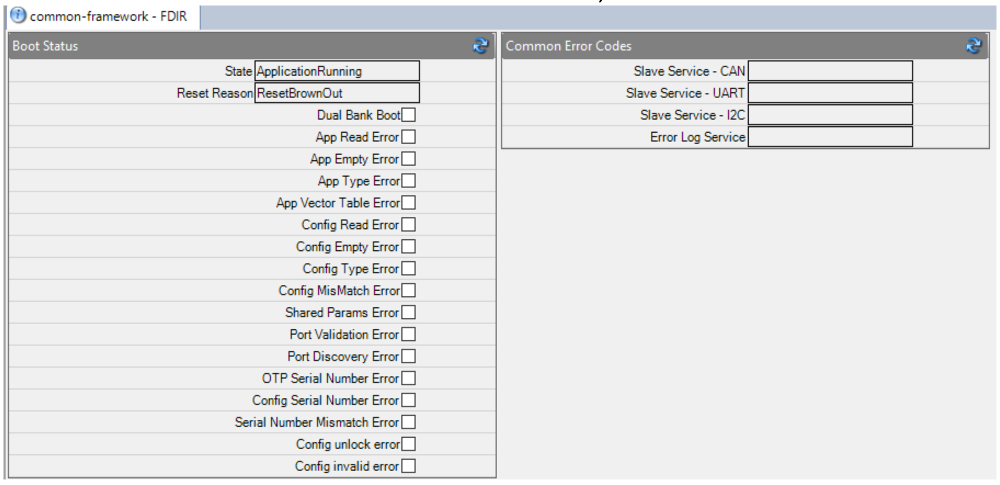
Field	Expected Result	Actual result	Comment	P / F
Node type identifier	NodeTypeCubeWheel			
Program type identifier	ProgramTypeControl			

Field	Expected Result	Actual result	Comment	P / F
Interface Version	3 {Or as verified in the software bundle in step 2}			
Firmware version (Major)	3 {Or as verified in the software bundle in step 1}			
Firmware version (Minor[1])	1 {Or as verified in the software bundle in step 1}			
Runtime (seconds)	Incrementing			
Runtime (milliseconds)	0 – 1000			

If the Interface version, Firmware version (Major) or Firmware version (Minor) differ from the expected results (i.e. do not reflect the latest software versions as per the software bundle), please stop and contact CubeSpace.

CubeSpace ensures that all the latest firmware as per the software release package are in fact programmed on the various CubeADCS units and CubeSpace then conducts the same health check as per this section before shipping to the user. If the user finds discrepancies, the shipped baseline should be considered as not correct.

3. Verify that there are no errors in the firmware configuration and hardware faults by refreshing the Boot Status under and under common framework,


common-framework - FDIR

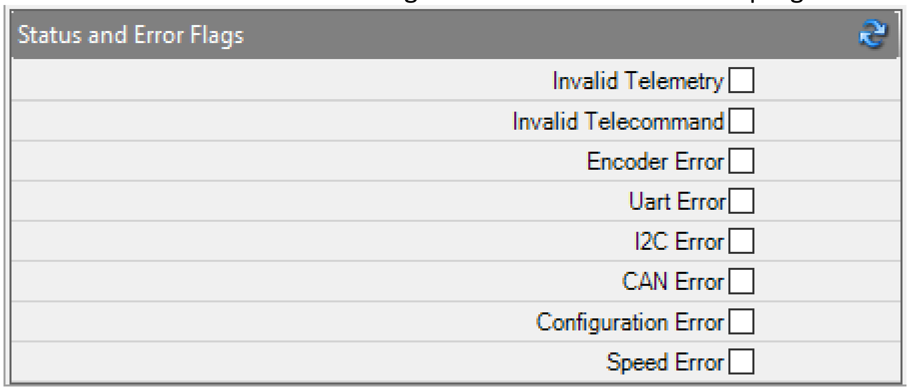
Boot Status

State	ApplicationRunning
Reset Reason	ResetBrownOut
Dual Bank Boot	☐
App Read Error	☐
App Empty Error	☐
App Type Error	☐
App Vector Table Error	☐
Config Read Error	☐
Config Empty Error	☐
Config Type Error	☐
Config MisMatch Error	☐
Shared Params Error	☐
Port Validation Error	☐
Port Discovery Error	☐
OTP Serial Number Error	☐
Config Serial Number Error	☐
Serial Number Mismatch Error	☐
Config unlock error	☐
Config invalid error	☐

Common Error Codes

Slave Service - CAN	
Slave Service - UART	
Slave Service - I2C	
Error Log Service	

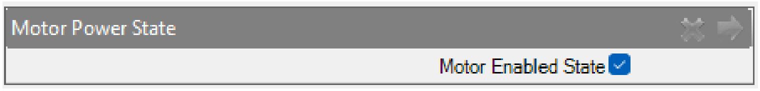
as well as the Status and Error Flags in the CubeWheel control program tab.


Status and Error Flags

Invalid Telemetry	☐
Invalid Telecommand	☐
Encoder Error	☐
Uart Error	☐
I2C Error	☐
CAN Error	☐
Configuration Error	☐
Speed Error	☐

[1] The minor version will depend on the software revision, and it does not impact the Control Program interface. The minor version showed here reflects the latest at the time of writing.

4. Power on the motor driver and, by locating the

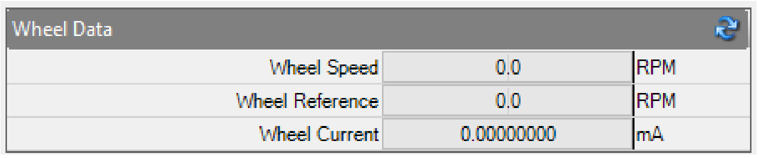

Motor Power State
Motor Enabled State ✓

telecommand and checking the "Motor Enabled State" check box and send the telecommand by pressing the button.

Table 9: CubeWheel Health Check: Power On

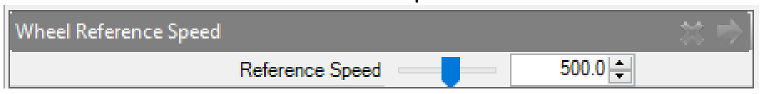
FIELD	EXPECTED RESULT	ACTUAL RESULT	COMMENTS	P / F
CubeWheel power state	PowerOn		Set, then Get to confirm receipt	

5. After the CubeWheel has been powered on, verify and note the current increase caused by powering on the CubeWheel by locating the Wheel Data message:


Wheel Data
Wheel Speed 0.0 RPM
Wheel Reference 0.0 RPM
Wheel Current 0.00000000 mA

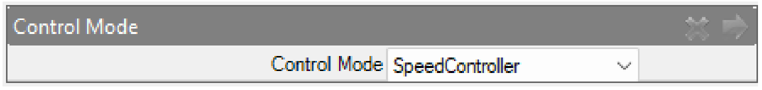
refreshing the message and note the "**Wheel current**" while "**Wheel Speed**" and "**Wheel Reference**" are 0 RPM in Table 10 (Idle CubeWheel current field).

6. Set the "Wheel Reference Speed" to the values indicated in Table 10 by locating

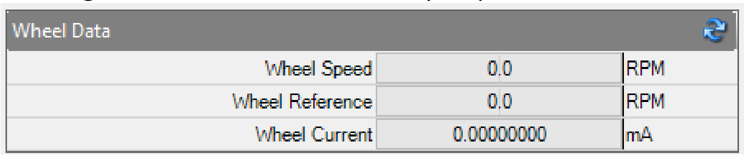

Wheel Reference Speed
Reference Speed 500.0

and, using either the slider or the text field, entering the indicated values in the "**Reference Speed**" field and sending the telecommand by pressing the button.

7. Note the Control Mode should display **"SpeedController"**

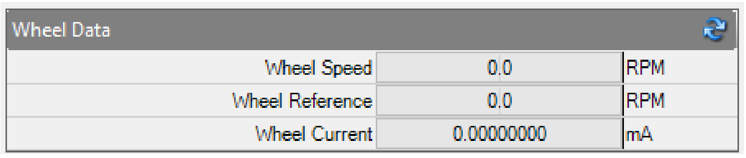

Control Mode
Control Mode SpeedController

8. Note the corresponding active wheel current for the commanded Wheel Speed commanded by locating the "Wheel Data" telemetry request


Wheel Data
Wheel Speed 0.0 RPM
Wheel Reference 0.0 RPM
Wheel Current 0.00000000 mA

and refreshing the Telemetry Request Message and noting the "**Wheel current**" while "**Wheel Reference**" values are the values in RPM in Table 10.

9. Also note the corresponding **"Wheel Speed"** for the commanded Wheel Reference Speed (commanded in step 7), by using the same "Wheel Data" telemetry request


Wheel Data
Wheel Speed 0.0 RPM
Wheel Reference 0.0 RPM
Wheel Current 0.00000000 mA

and noting the "**Wheel Speed**" in Table 11 while "**Wheel Reference**" values are the values in RPM in Table 10.

10. Repeat steps 7-9 for all the reference speeds in Table 10. Verify and note the current values and the Wheel speeds as a result of the various reference speeds commanded.

Table 10: CubeWheel Health Check: Expected (additional) Current Consumption

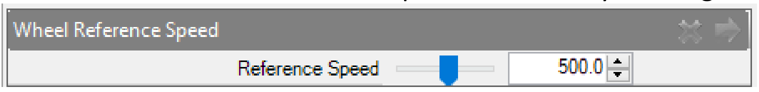
FIELD	EXPECTED RESULT	ACTUAL RESULT	COMMENTS	P/F
Idle CubeWheel current – POWER ON, not spinning	All Sizes: ~32mA		Failed if current: > 45 mA	
Active CubeWheel battery current – POWER ON, spinning at 500 rpm	CW0162: ~7 mA CW0057 ~7mA CW0017 ~4mA		After allowing the wheel to reach constant speed.	
Active CubeWheel battery current – POWER ON, spinning at 2000 rpm	CW0162: ~20mA CW0057 ~20mA CW0017 ~8mA		After allowing the wheel to reach constant speed.	
Active CubeWheel battery current – POWER ON, spinning at 4000 rpm	CW0162: ~36mA CW0057 ~36mA CW0017 ~13mA		After allowing the wheel to reach constant speed.	
Active CubeWheel battery current – POWER ON, spinning at 6000 rpm	CW0162: ~56mA CW0057 ~56mA CW0017 ~18mA		After allowing the wheel to reach constant speed.	

11. Verify that the wheels measure approximately the commanded speed, as shown in Table 11.

Table 11: CubeWheel Health Check: Expected Actual speeds.

FIELD	EXPECTED RESULT	ACTUAL RESULT	COMMENTS	P / F
CubeWheel measured speed	500		±5 rpm	
CubeWheel measured speed	-1000		±5 rpm	
CubeWheel measured speed	1500		±5 rpm	
CubeWheel measured speed	-2000		±5 rpm	

12. Command all the wheels' speed back to 0 by locating



and, using either the slider or the text field, entering "0" in the "**Reference Speed**" field and sending the telecommand by pressing the button.

13. Switch off all the CubeWheel.

Once the values in the above tables have been confirmed, the CubeWheel is considered to be in working condition. If any result obtained does not match the expected result, please contact CubeSpace.

5.5 Known issues.

This section is for recording known issues that were encountered during the Health Check but were not considered to be blockers for proceeding with the order. **Note**: This section might be left blank if no issues were encountered.