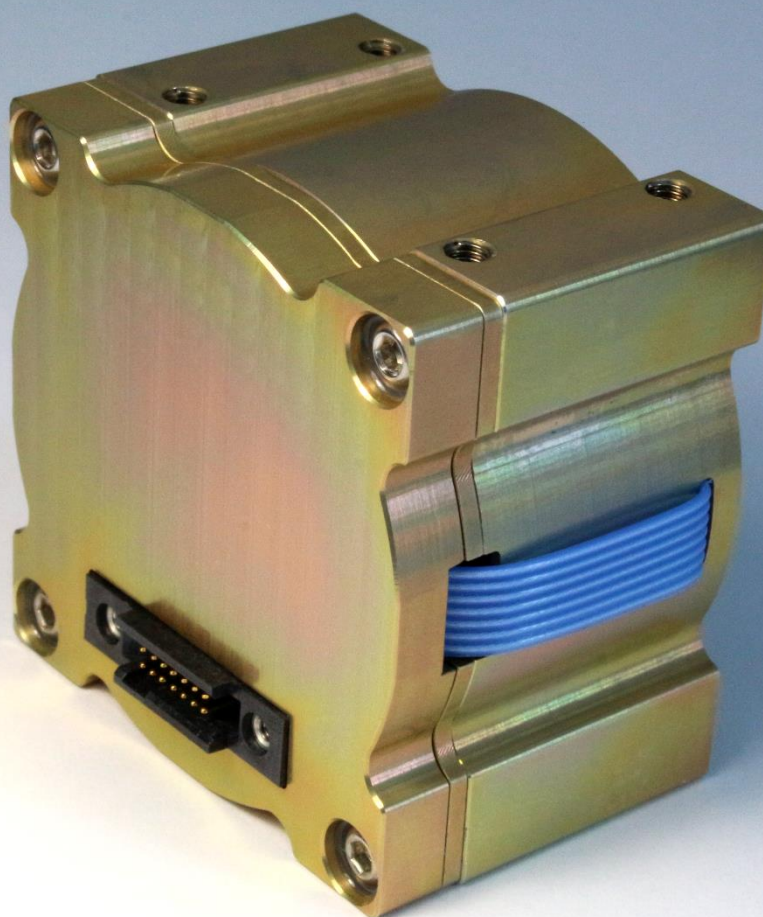




# CUBEWHEEL

MOMENTUM/REACTION WHEELS FOR NANOSATELLITES



## USER MANUAL

**ESL**  
Electronic Systems  
Laboratory

### Contact Us

Phone : +27 21 808 9499  
E-mail : [info@cubespace.co.za](mailto:info@cubespace.co.za)  
Web : [www.cubespace.co.za](http://www.cubespace.co.za)  
Facebook : /CubeSpaceADCS  
Twitter : @CubeSpace\_ADCS

### Physical Address

**CubeSpace**  
The LaunchLab  
Hammanhand Road  
Stellenbosch, 7600  
South Africa



UNIVERSITEIT  
STELLENBOSCH  
UNIVERSITY

# Table of Contents

<b>List of Acronyms/Abbreviations .....</b>	<b>3</b>
<b>List of Figures .....</b>	<b>4</b>
<b>List of Tables.....</b>	<b>4</b>
<b>1. Introduction.....</b>	<b>6</b>
<b>2. Functional Description.....</b>	<b>7</b>
2.1 Hardware.....	7
2.2 Software .....	7
2.3 Mounting .....	9
2.4 Communication interfaces .....	10
<b>3. Specifications.....</b>	<b>11</b>
<b>4. Getting Started .....</b>	<b>13</b>
4.1 Unpacking the CubeWheel package .....	13
4.2 Before getting started.....	13
4.3 CubeSupport software.....	13
4.4 Hardware setup .....	15
4.5 Getting to know the CubeWheel interface panel.....	16
4.6 Incoming health check.....	23
<b>5. Electrical Connection .....</b>	<b>24</b>
5.1 Physical connector.....	24
5.2 Power.....	24
5.3 Communication .....	25
<b>6. Communicating with a CubeWheel Unit .....</b>	<b>26</b>
6.1 UART interface .....	26
6.2 I <sup>2</sup> C interface.....	29
6.3 CAN interface .....	31
6.4 Telecommands and telemetry requests.....	33
<b>7. Document Version History .....</b>	<b>43</b>

## List of Acronyms/Abbreviations

ADC	Analog to Digital Converter
ADCS	Attitude and Determination Control System
BLDC	Brushless DC
DC	Direct Current
EOM	End-Of-Message
ESL	Electronic Systems Laboratory
FPGA	Field Programmable Gate Array
ICD	Interface Control Document
I <sup>2</sup> C	Inter- Integrated Circuit
MCU	Microcontroller Unit
OBC	On-board Computer
PBH	Piggyback Header
RTC	Real-time Clock
RW	Reaction Wheel
SOM	Start-Of-Message
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TBC	To Be Confirmed
TBD	To Be Determined
TC	Telecommand
TLM	Telemetry
UART	Universal Asynchronous Receiver/Transmitter

## List of Figures

Figure 1 – CubeWheel Small and Small Plus angular momentum vector .....	9
Figure 2 – CubeWheel Medium and Large angular momentum vector .....	9
Figure 3 – The General tab (top) and the Configuration tab (bottom) in CubeSupport.....	17
Figure 4 – The sections of the General tab.....	19
Figure 5 – The sections of the Configuration tab .....	20
Figure 6 – CubeWheel telemetry logs created by CubeSupport .....	22
Figure 7 – CubeWheel telemetry log example.....	22
Figure 8 – Suggested power connection to a single CubeWheel unit .....	24
Figure 9 – Suggested power connection to multiple CubeWheel units .....	25
Figure 10 – Suggested I <sup>2</sup> C connection to a CubeWheel unit .....	29
Figure 11 – Example of I <sup>2</sup> C telemetry request.....	30
Figure 12 – Example of I <sup>2</sup> C telecommand.....	30
Figure 13 – Suggested CAN connection to a CubeWheel unit.....	31
Figure 14 – CAN identification packet .....	31
Figure 15 – Example of CAN identification packet during telemetry request .....	32
Figure 16 – Example of CAN identification packet during telemetry response .....	32

## List of Tables

Table 1 – Small CubeWheel specifications .....	11
Table 2 – Small Plus CubeWheel specifications.....	11
Table 3 – Medium CubeWheel specifications .....	12
Table 4 – Large CubeWheel specifications.....	12
Table 5 – UART specifications for CubeWheel units.....	26
Table 6 – UART protocol character definition .....	26
Table 7 – Example of UART telemetry request .....	27
Table 8 – Example of UART telemetry response .....	27
Table 9 – Example of UART telecommand .....	28
Table 10 – UART telecommand response.....	28
Table 11 – Example of special character in UART data bytes.....	28
Table 12 – CAN message types .....	32
Table 13 – List of telecommands .....	33
Table 14 – List of telemetry requests.....	33
Table 15 – Reset command format .....	34
Table 16 – Wheel reference speed command format.....	34
Table 17 – Wheel commanded torque command format.....	34
Table 18 – Motor power state command format.....	34

Table 19 – Encoder power state command format.....	35
Table 20 – Hall power state command format .....	35
Table 21 – Control mode command format .....	35
Table 22 – Control mode enumeration values.....	35
Table 23 – Backup wheel mode command format.....	36
Table 24 – Clear errors command format.....	36
Table 25 – Set I <sup>2</sup> C address command format.....	36
Table 26 – Set CAN mask command format.....	36
Table 27 – Set PWM gain command format.....	37
Table 28 – Set main gain command format.....	37
Table 29 – Set backup gain command format.....	37
Table 30 – Identification telemetry format.....	38
Table 31 – Extended identification telemetry format.....	38
Table 32 – Wheel status telemetry format .....	39
Table 33 – Wheel speed telemetry format .....	39
Table 34 – Wheel reference telemetry format .....	40
Table 35 – Wheel current telemetry format.....	40
Table 36 – Wheel data telemetry format .....	40
Table 37 – Wheel data additional telemetry format.....	41
Table 38 – PWM gain telemetry format .....	41
Table 39 – Main gain telemetry format .....	41
Table 40 – Backup gain telemetry format.....	42
Table 41 – Status and error flags telemetry format.....	42

# 1. Introduction

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.

This document will aim to familiarize the user with CubeSpace's momentum/reaction wheel unit, the CubeWheel. A brief functional description of the unit will be followed by the specifications of the various sizes of CubeWheel units. An in-depth guide to getting started with a CubeWheel will also be provided in this document. This document is applicable to all the different sizes of CubeWheel units.



**The CubeWheel unit contains electrostatic sensitive components. Under no circumstances should the device be handled without anti-static protection.**



**The CubeWheel unit is a delicate mechanical assembly. Always handle with great care, preferably using gloves.**



**When handling the CubeWheel, always place the unit on a flat surface, preferably an anti-static mat. It is furthermore important to handle the unit in a clean environment, as required for flight-model components.**

## 2. Functional Description

This section will describe the hardware and software that are enveloped in a CubeWheel unit.

### 2.1 Hardware

#### 2.1.1 Mechanical assembly

The CubeWheel unit consists of a rotating flywheel attached to a brushless DC (BLDC) motor. A small PCB, which includes the necessary drive, control, and interface electronics, is attached to the bottom of the BLDC motor and is protected by an aluminium cover. The primary function of the electronics PCB is to measure and control the speed of the motor. The top cover of the CubeWheel unit supplies the necessary mounting holes for the user to mount the CubeWheel in any one of its 3 axes. The mounting of a CubeWheel will be discussed in Section 2.3.

#### 2.1.2 Size

Three sizes of CubeWheels are available: Small, Small Plus, Medium, and Large. Small and Small Plus CubeWheel units are supplied with a 14-way wire harness terminated in a female connector, whereas the Medium and Large units are supplied with a 14-way screw-down connector. Refer to the CubeWheel Interface Control Document (ICD) for more information regarding the harnesses and connectors.

#### 2.1.3 Speed measurement

The speed of the motor is measured by two independent sources. The primary source of speed measurements is a magnetic encoder and the secondary source is the internal Hall sensors of the motor. Independent switches control the power to each one of these measurement sources. The speed control algorithms use the encoder measurements by default, but it can also use the Hall sensor measurements as a backup. Both measurements are available as telemetry.

### 2.2 Software

The electronics PCB houses a microcontroller unit (MCU) that is responsible for commanding the motor driver, measuring the wheel's speed, performing speed control algorithms, measuring the motor's current, toggling power switches, and responding to telemetry requests and telecommands.

Two main aspects of the MCU's software are applicable to the user: the control state and backup mode.

### 2.2.1 Control state

The software on the MCU is governed by various control states. The MCU toggles power switches and commands the motor driver based on the current control state. There are four control states: (1) *Idle*, (2) *No Control*, (3) *Duty Cycle*, and (4) *Speed Controller*. The CubeWheel unit will respond to telemetry requests and execute telecommands in any control state.

The *Idle* state is 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. Note that during this state, the motor current measurements are invalid.

During the *No Control* state, 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.

A duty cycle command will activate the *Duty Cycle* state. This state will maintain a reference control signal to the motor.

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.

### 2.2.2 Backup mode

Redundancy is incorporated through the availability of two speed measurements (from the encoder, at 10 Hz, and from the Hall sensors, at 1 Hz). 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. Backup mode is disabled by default. It should be noted that the default settling time of the backup mode speed controller is roughly double that of the encoder-based speed controller. The backup mode cannot control the wheel speed at low speeds and only react to speed commands larger than 100 RPM.

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.

## 2.3 Mounting

Various sets of mounting holes can be found on the cover of the CubeWheel unit, allowing the unit to be mounted in any of the 3 axes. Refer to the CubeWheel ICD for the mechanical interface definition of the mounting holes.

A positive rotation (resulting from a positive wheel speed reference or duty cycle command) can be translated to an angular momentum vector pointing out of the bottom of the CubeWheel unit. The aforementioned is illustrated in Figure 1.

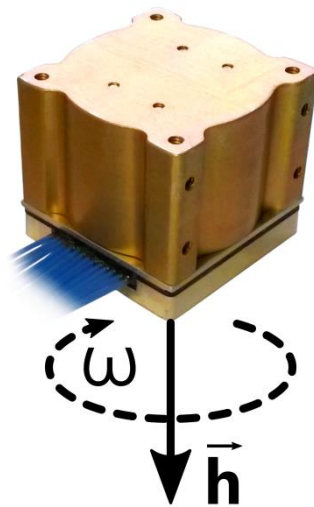


Figure 1 – CubeWheel Small and Small Plus angular momentum vector

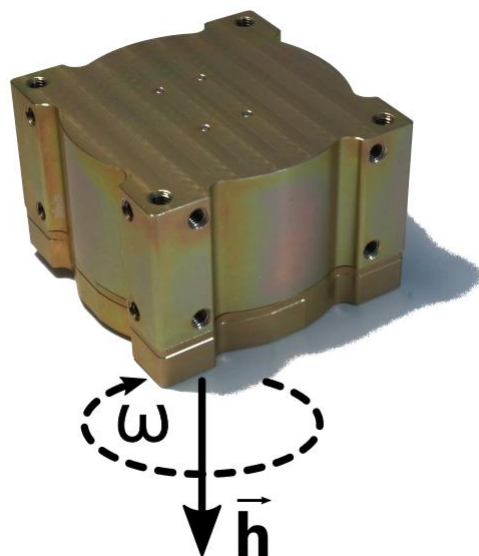


Figure 2 – CubeWheel Medium and Large angular momentum vector

## 2.4 Communication interfaces

All CubeWheel units can be configured to interface via I<sup>2</sup>C, UART, or CAN. The specifications of each of these three protocols can be found in Section 6 of this document. It should be noted that multiple communication interfaces can be used simultaneously.

A complete list of telemetry requests and telecommands can be found in Section 6.4 of this document.

### 3. Specifications

Table 1 – Small CubeWheel specifications

Specification	Value		Notes
Dimensions	28 x 28 x 26.2 mm		Excl. internal harnesses, see ICD
Mass	±60 g		Depends on harness length
Nominal momentum	1.7 mNms		@ 8000 rpm
Nominal torque	0.23 mNm		
Supply voltage	3.3 V / (6.5 V – 16.0 V)		
Operating temperature	-10°C to 60°C		
Static imbalance	<0.003 g·cm		
Dynamic imbalance	<0.005 g·cm <sup>2</sup>		
Power consumption	3.3 V (mW)	V <sub>battery</sub> (mW)	
<ul style="list-style-type: none"> <li>V<sub>battery</sub> = 8.0 V</li> <li>CAN electronics populated</li> </ul>	50	0	Idle state
	75	26	All electronics on, 0 rpm
	75	50	@ 2000 rpm
	75	640	Maximum torque

Table 2 – Small Plus CubeWheel specifications

Specification	Value		Notes
Dimensions	33.4 x 33.4 x 29.7 mm		Excl. internal harnesses, see ICD
Mass	±90 g		Depends on harness length
Maximum momentum	3.6 mNms		@ 6000 rpm
Nominal torque	2.3 mNm		
Supply voltage	3.3 V / (6.5 V – 16.0 V)		
Operating temperature	-10°C to 60°C		
Static imbalance	<0.004 g·cm		
Dynamic imbalance	<0.014 g·cm <sup>2</sup>		
Power consumption	3.3 V (mW)	V <sub>battery</sub> (mW)	
<ul style="list-style-type: none"> <li>V<sub>battery</sub> = 8.0 V</li> <li>CAN electronics populated</li> </ul>	50	0	Idle state
	75	12	All electronics on, 0 rpm
	75	64	@ 2000 rpm
	75	2300	Maximum torque

Table 3 – Medium CubeWheel specifications

Specification	Value		Notes
Dimensions	46 x 46 x 31.5 mm		Excl. internal harnesses, see ICD
Mass	150 ± 2 g		
Maximum momentum	10.8 mNms		@ 6000 rpm
Nominal torque	1.0 mNm		
Supply voltage	3.3 V / (6.5 V – 16.0 V)		
Operating temperature	-10°C to 60°C		
Static imbalance	<0.004 g.cm		
Dynamic imbalance	<0.014 g.cm <sup>2</sup>		
Power consumption	3.3 V (mW)	V <sub>battery</sub> (mW)	
• V <sub>battery</sub> = 8.0 V CAN electronics populated	50	0	Idle state
	75	12	All electronics on, 0 rpm
	75	130	@ 2000 rpm
	75	2300	Maximum torque

Table 4 – Large CubeWheel specifications

Specification	Value		Notes
Dimensions	57 x 57 x 31.5 mm		Excl. internal harnesses, see ICD
Mass	±225 g		
Nominal momentum	30 mNms		@ 6000 rpm
Nominal torque	2.3 mNm		
Supply voltage	3.3 V / (6.5 V – 16.0 V)		
Operating temperature	-10°C to 60°C		
Static imbalance	<0.006 g.cm		
Dynamic imbalance	<0.05 g.cm <sup>2</sup>		
Power consumption	3.3 V (mW)	V <sub>battery</sub> (mW)	
• V <sub>battery</sub> = 8.0 V CAN electronics populated	50	0	Idle state
	75	12	All electronics on, 0 rpm
	75	280	@ 2000 rpm
	75	4500	Maximum torque

## 4. Getting Started

This section will provide an in-depth guide to getting started with a CubeWheel unit. The aim of this section is to familiarise the user with the CubeWheel unit and the supporting software, CubeSupport.

**It is important to read every instruction in this section carefully and to perform the instructions sequentially.**

### 4.1 Unpacking the CubeWheel package

The received Peli-Case contains the following items:

- CubeWheel unit(s)
- Support PCB
- UART-to-USB cable
- CubeSpace flash drive

The included items will allow the user to interface with a CubeWheel unit without an OBC.

### 4.2 Before getting started

The following additional items are required before the user can get started with the CubeWheel:

- Adjustable bench power supply (capable of  $V_{\max} > 16 \text{ V}$ ,  $I_{\max} > 1 \text{ A}$ )
- DC power plug (inner diameter = 2.1 mm, outer diameter = 5.5 mm, inside-positive)
- Multi-meter or oscilloscope
- CubeSupport software (see Section 4.3)
- Computer with an open USB port (running Windows 7 or later)

### 4.3 CubeSupport software

#### 4.3.1 What is CubeSupport?

CubeSupport allows the user to interface with a CubeWheel via the UART connection. No additional software or hardware (except the items mentioned in Section 4.2) is required, which means that the CubeWheel can act as a standalone module.

The software enables the user to request telemetry and to send telecommands from/to a CubeWheel. The user can also set a CubeWheel's I<sup>2</sup>C address and CAN mask with the CubeSupport software.

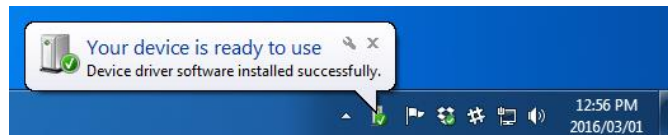
### 4.3.2 Connecting to the UART cable

Follow the instructions below to verify the connection between CubeSupport and the UART cable:

- Plug the UART-to-USB cable into the computer's open USB port.
- The computer should detect the cable and install the drivers by itself.




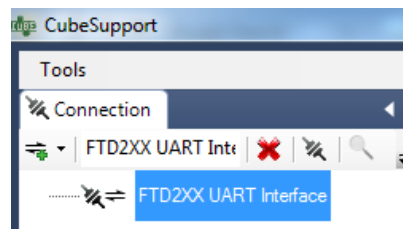
- After a couple of minutes, a message indicating the successful installation of the drivers should appear.




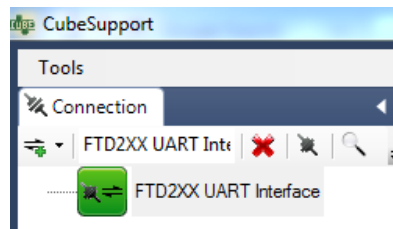
- Browse to the folder containing CubeSupport and launch the application.


Name	Date modified	Type	Size
CubeSupport	2016/02/09 08:03 ...	Application	354 KB
CubeSupport.vshost	2016/02/09 08:03 ...	Application	23 KB
CubeSat Support Software Library.dll	2016/02/09 08:03 ...	Application extens...	1 652 KB
FTD2XX_NET.dll	2015/03/02 08:37 ...	Application extens...	69 KB
CubeSupport.vshost.exe.manifest	2010/03/17 10:39 ...	MANIFEST File	1 KB
CubeSat Support Software Library	2016/02/09 08:03 ...	Program Debug D...	3 874 KB
CubeSupport	2016/02/09 08:03 ...	Program Debug D...	58 KB
FTD2XX_NET	2015/03/02 08:37 ...	XML Document	106 KB

- Add a new UART interface by clicking on the  button (not on the drop-down arrow). A new interface should appear, as shown below.



- Connect to the UART cable by clicking on the  button.
- The connection status should turn green if the connection was **successful**, as illustrated below.



- Disconnect from the UART-to-USB cable by clicking on the  button.

## 4.4 Hardware setup



### 4.4.1 Connecting power to the support PCB

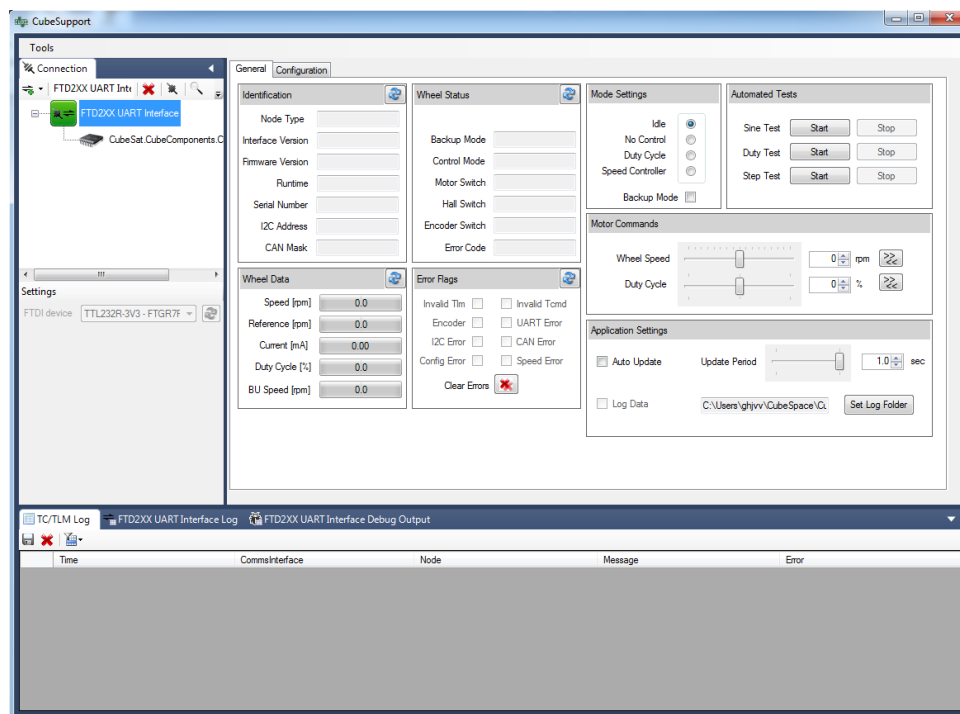
Follow the instructions below to set up the support PCB:

- Set the voltage on the power supply to the desired bus battery voltage (e.g. 8 V, up to a maximum of 16.0 V).
- Set the current limit on the power supply to 1 A.
- Plug the DC power cable into the support PCB.
- Connect the wheel harness to the support PCB.
- Switch on the support PCB and check that both LED light up.

## 4.4.2 Connecting the CubeWheel to CubeSupport

Follow the instructions below to connect the CubeWheel to the CubeSupport software:

- Plug the supplied UART-to-USB cable into the support PCB (the header is labelled as "UART" on the PCB). Plug the UART-to-USB cable into the computer (if it is not plugged in already).
- Open the CubeSupport application (if it is not open already).
- Add a new UART interface by clicking on the  button.
- Connect to the CubeWheel by clicking on the  button.
- A successful connection should yield an interface panel, as shown below. Note that most of the fields will still be empty.



- If the connection is not successful, check the following:
  - ✓ Verify that the power supply is switched on and that the current limit has not been reached.
  - ✓ Verify that the CubeWheel is plugged into the support PCB.
  - ✓ Verify that the UART connection cables are plugged in correctly.
  - ✓ Probe the pins on the support PCB header that are marked 3V3, Vbat and EN.

## 4.5 Getting to know the CubeWheel interface panel

The CubeWheel interface panel in the CubeSupport software provides a user-friendly environment from which telemetry can be requested and telecommands can be sent. It is

important for the user to become familiarized with the interface panel before attempting to use it. The various sections of the software will be explained in this section.

The interface panel consists of two tabs: *General* and *Configuration*, as illustrated in Figure 3.

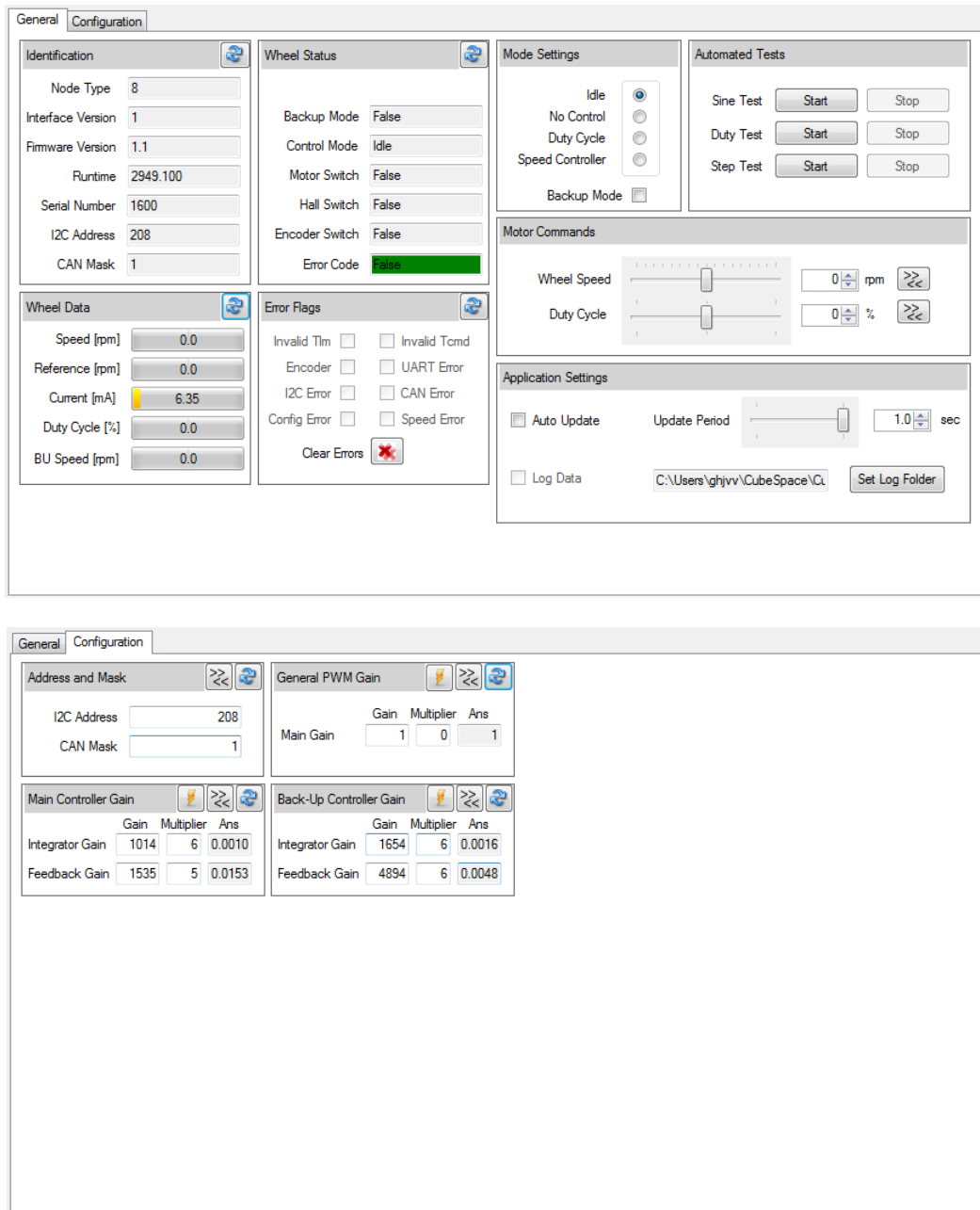


Figure 3 – The General tab (top) and the Configuration tab (bottom) in CubeSupport

The *General* tab contains display fields for the following:

- wheel identification

- wheel status
- error flags
- wheel speed and current

The *Configuration* tab, which will seldom be used, enables the user to change the following:

- the I<sup>2</sup>C address
  - CAN mask
  - gains of the speed controllers running on the CubeWheel's MCU
- \* The correct functioning of the CubeWheel is heavily dependent on the controller gains. Consult CubeSpace before changing the controller gains.***

#### 4.5.1 General tab

Figure 4 illustrates the various sections (or blocks) of the *General* tab. Each block is briefly described below:

1. **Identification** – The identification telemetry of the CubeWheel is displayed in this block.
2. **Wheel Data** – The wheel speed reference and measurements, the wheel current, and the commanded duty cycle is displayed in this block.
3. **Wheel Status** – The current control mode, the status of the backup mode, the state of the various switches, and the combined error flag is displayed in this block.
4. **Error Flags** – The individual error flags are displayed in this block.
5. **Mode Settings** – The current control mode can be selected and the backup mode can be enabled/disabled using this block.
6. **Automated Tests** – Standard pre-defined tests can be performed using this block.
7. **Motor Commands** – The reference wheel speed and the commanded duty cycle can be set using this block.
8. **Application Settings** – This block houses the auto-update and data logging settings.

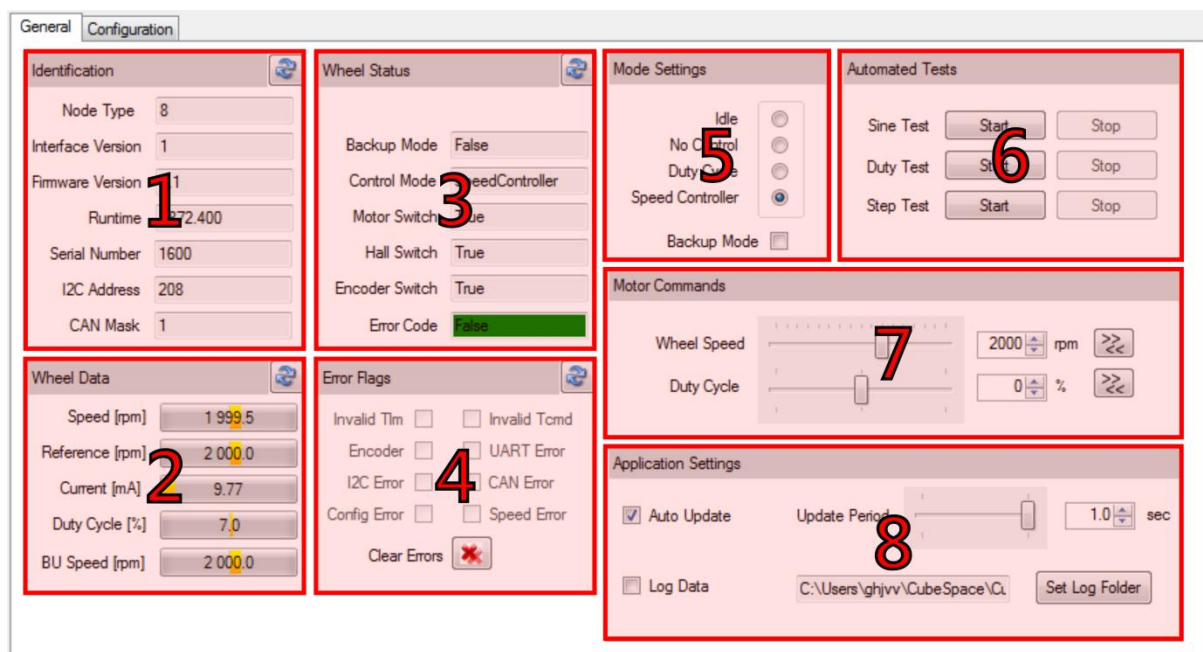


Figure 4 – The sections of the General tab

## 4.5.2 Configuration tab

Figure 5 illustrates the various sections (or blocks) of the *Configuration* tab. Each block is briefly described below:

1. **Address and Mask** – The I<sup>2</sup>C address and CAN mask of the CubeWheel can be read and configured using this block.
2. **General PWM Gain** – The gain of the PWM register that commands the motor driver can be read and configured using this block.
3. **Main Controller Gain** – The gains of the main speed controller (i.e. when backup mode IS NOT enabled) can be read and configured using this block.
4. **Backup Controller Gain** – The gains of the backup speed controller (i.e. when backup mode IS enabled) can be read and configured using this block.

**Do not use blocks 2, 3, and 4 without consulting CubeSpace.**

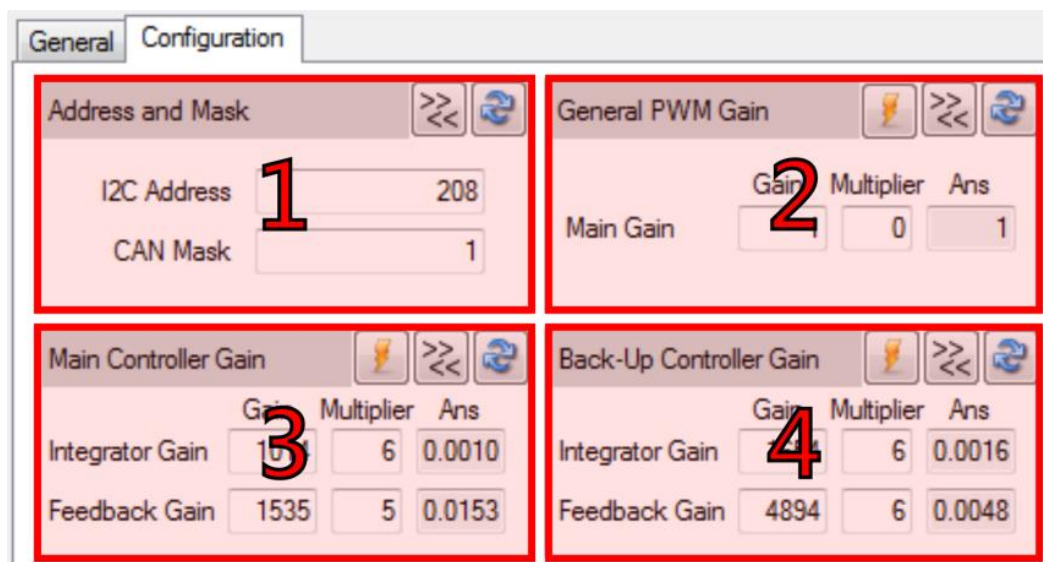




Figure 5 – The sections of the Configuration tab

## 4.5.3 Requesting telemetry

The request button in the CubeWheel interface panel is denoted by the  symbol. Clicking on this button will send a single request to the CubeWheel for the relevant telemetry.

## 4.5.4 Sending a telecommand

The transmit button in the interface panel is denoted by the  symbol. Clicking on this button will send the relevant telecommand to the CubeWheel once.

#### 4.5.5 Auto Update and the update period

CubeSupport has the ability to automatically update several blocks under the *General* tab by requesting telemetry from the CubeWheel unit at a specified sample period. *Auto Update* can be enabled by checking relevant box in the *Application Settings* block.

#### 4.5.6 Creating a telemetry log

The CubeSupport software provides the user with the capability to log CubeWheel telemetry. To create a telemetry log, simply check the *Log Data* box in the *Application Settings* block under the *General* Tab. Note that *Auto Update* must be enabled before telemetry can be logged.

CubeSupport generates the log as a .txt file in CSV format. The filename of the log will include the CubeWheel's serial number and a timestamp. The following telemetry data is logged:

- Runtime in seconds (including the millisecond telemetry)
- Backup mode flag
- Control mode
- Motor switch state
- Hall sensor switch state
- Encoder switch state
- Error code as a decimal value (which contains all the error flags)
- Wheel speed measured by magnetic encoder in rpm
- Wheel speed measured by Hall sensors (backup speed measurement) in rpm
- Reference wheel speed in rpm
- Duty cycle (including direction) of PWM control signal expressed as a percentage
- Motor current (drawn directly from power supply) in mA

Figure 6 illustrates how the logs are created in the relevant folder. Note that while CubeSupport is busy logging telemetry, the size of the log will be given as 0 KB (as is the case with the last log file in Figure 6). **Do not open a log file while CubeSupport is busy logging to that file.** To view a log, first ensure that *Log Data* on the *General* tab under the *Application Settings* block is not selected before attempting to open the file.













Name	Date modified	Type	Size
 CubeSat Support Software Library.dll	2016/03/03 10:50 ...	Application extens...	1 653 KB
 CubeSat Support Software Library	2016/03/03 10:50 ...	Program Debug D...	4 152 KB
 CubeSupport	2016/03/03 10:59 ...	Application	420 KB
 CubeSupport	2016/03/03 10:59 ...	Program Debug D...	62 KB
 CubeSupport.vshost	2016/03/03 11:02 ...	Application	24 KB
 CubeSupport.vshost.exe.manifest	2013/03/18 05:00 ...	MANIFEST File	1 KB
 CW1600_2016-3-3_15h32m56s	2016/03/03 03:33 ...	Text Document	3 KB
 CW1600_2016-3-3_15h33m32s	2016/03/03 03:34 ...	Text Document	3 KB
 CW1600_2016-3-3_15h34m14s	2016/03/03 03:35 ...	Text Document	5 KB
 CW1600_2016-3-3_15h35m21s	2016/03/03 03:35 ...	Text Document	0 KB
 FTD2XX_NET.dll	2016/03/03 09:06 ...	Application extens...	69 KB
 FTD2XX_NET	2016/03/03 09:05 ...	XML Document	106 KB

Figure 6 – CubeWheel telemetry logs created by CubeSupport

An example of a CubeWheel telemetry log file is shown in Figure 7.

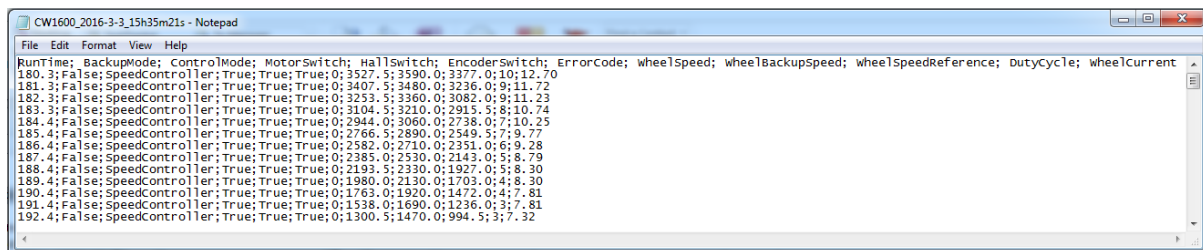


Figure 7 – CubeWheel telemetry log example

Note that the runtime in seconds is stored as a 16-bit unsigned integer on the CubeWheel MCU, which means that it will overflow and start at 0 again when it increments from 65535.

## 4.6 Incoming health check

It is necessary to perform an incoming health check as soon as possible after receiving a CubeWheel unit(s). This section will guide the user through the health check, which is identical to the outgoing health check which is performed at CubeSpace before delivery.

**Do not attempt to perform the incoming health check before working through Section 4.1 to Section 4.5 of this document.**

### 4.6.1 Before the incoming health check

Follow the instructions below to set up the incoming health check:

- Unpack the contents of the CubeWheel package in a clean environment (Section 4.1).
- Connect power to the support PCB (Section 4.4.1).
- Connect the CubeWheel to CubeSupport (Section 4.4.2).
- Keep a copy of the **CubeWheel Health Check** document nearby, as it must be filled in during the incoming health check.

### 4.6.2 Performing the incoming health check

The procedures of the incoming health check can be found in the CubeWheel Health Check document. Measured and observed results must also be captured onto this document.

### 4.6.3 After the incoming health check

Follow the instructions below to end the incoming health check:

- Disconnect the UART connection from the support PCB.
- Disconnect the power from the support PCB.
- Disconnect the CubeWheel from the support PCB.
- Carefully place the CubeWheel unit and the support PCB back into their respective anti-static bags and place the bags into the supplied Peli-Case for storage.
- **Sign the CubeWheel Health Check document (on every page with a signature line) and send the document to CubeSpace as soon as possible.**

## 5. Electrical Connection

### 5.1 Physical connector

The CubeWheel unit has a single 14-way connection which contains all the required electrical connections to power and to communicate with the unit. Refer to the CubeWheel ICD for the pin definition of the connector.

The Small Plus wheel has a harness soldered in on the wheel side that terminates in a 14-way Samtec SFSDT-series screw down connector. The Small wheel has an additional Molex header on the wheel itself to allow the user to use a different harness. The Medium and Large CubeWheels house a 14-way screw-down connector from the Samtec TFM-series, which mates with a Samtec SFSDT-series wired connector. All the Samtec connectors have a current rating of at least 2.9 A and an operating temperature of  $-40^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$ . For more information regarding the connectors, refer to [www.samtec.com](http://www.samtec.com).

### 5.2 Power

A CubeWheel unit requires 3.3 V and the battery voltage ( $V_{\text{battery}} > 6.5 \text{ V}$ ) to operate. The digital electronics are powered by 3.3 V, whereas the motor driver (i.e. the motor) runs off the battery voltage. CubeWheel contains power switches for both the 3.3V and for VBat supplies. The power switch for the 3.3V supply is toggled using the externally controlled ENABLE input and is default off. ENABLE must be pulled high to switch the wheel on. The power switch for the battery supply is controlled by the CubeWheel itself and is switched on whenever CubeWheel gets a speed/PWM command.

A simplified suggested electrical power diagram to a single CubeWheel unit and to multiple CubeWheel units are shown in Figure 8 and Figure 9 respectively.

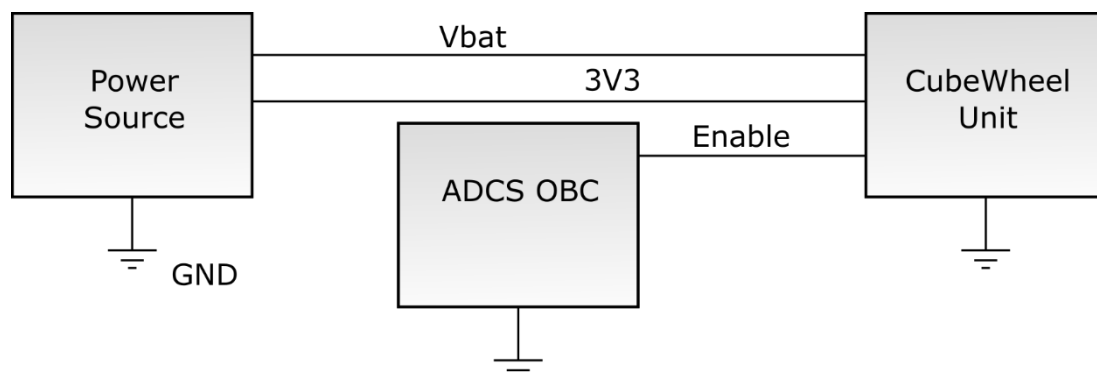


Figure 8 – Suggested power connection to a single CubeWheel unit

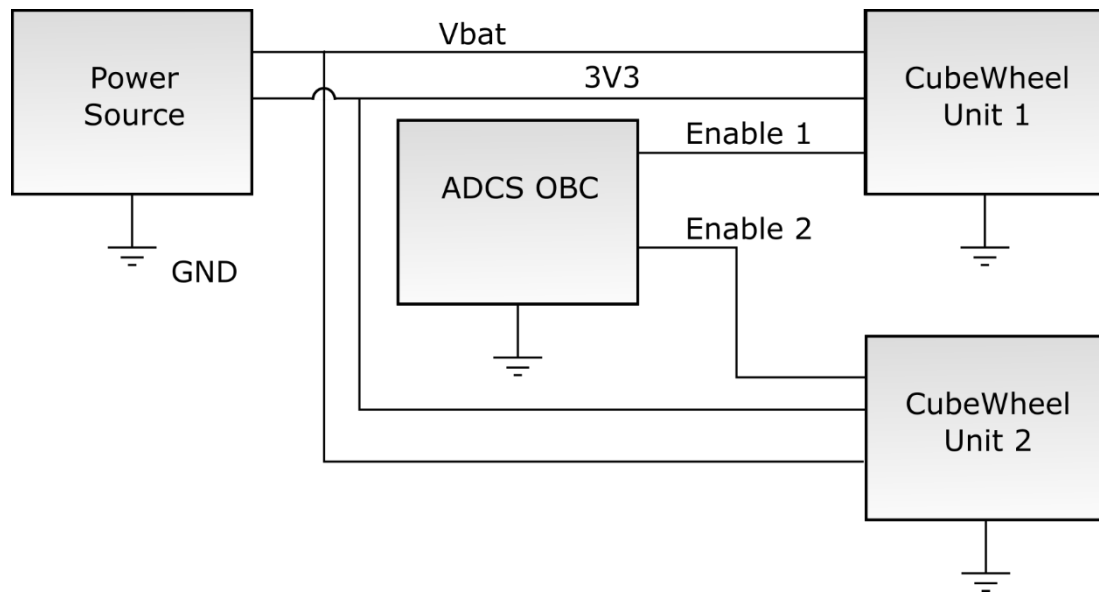


Figure 9 – Suggested power connection to multiple CubeWheel units

### 5.3 Communication

The various communication interfaces on a CubeWheel unit and their respective protocols are discussed in detail in Section 6 of this document.

## 6. Communicating with a CubeWheel Unit

### 6.1 UART interface

#### 6.1.1 Buffer

The CubeWheel unit contains a UART buffer to isolate the unit. The logic of the unit operates at 3.3 V. **Using a higher voltage than 3.3 V may result in damage to the electronics.**

#### 6.1.2 Interface specifications

The specifications of the CubeWheel's UART interface are listed in Table 5.

Table 5 – UART specifications for CubeWheel units

Parameter	Value
Baud Rate	115200
Data Bits	8
Parity	None
Stop Bits	1

#### 6.1.3 Protocol

Any telemetry (TLM) or telecommand (TC) packet sent to/from the CubeWheel unit is initiated by a unique ID. In the case of a TC or a TLM reply, the additional data bytes also form part of the TLM/TC packet. An additional UART protocol is furthermore added which encapsulates the TLM/TC packet.

A UART transmission to/from a CubeWheel unit starts with a special character followed by a start-of-message (SOM) character. At the end of the UART packet, the special character is sent again followed by an end-of-message (EOM) character. If the UART packet itself contains a special character, that special character needs to be followed by another special character. The CubeWheel unit will only respond if this protocol is followed by the master and will return TLM or a TC acknowledge response with the same protocol. A UART error flag will be set by the CubeWheel if an incorrect protocol is received.

The characters relevant to the UART protocol are listed in Table 6.

Table 6 – UART protocol character definition

Character	Value
Special character	0x1F
SOM	0x7F
EOM	0xFF

A message will therefore begin with the sequence 0x1F, 0x7F and end with the sequence 0x1F, 0xFF. Table 7 illustrates the format of a TLM request to the CubeWheel unit via UART, whereas Table 8 depicts the format of the unit's response to the request.

Table 7 – Example of UART telemetry request

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

Table 8 – Example of UART telemetry response

0x1F	0x7F	TLM byte 0	TLM byte 1	...	0x1F	0xFF
Start-of-message					End-of-message	

The CubeWheel unit will acknowledge the receipt of a TC by responding with either a 0 (if the TC was received successfully) or a 1 (invalid TC identifier). **A telecommand via UART must be populated with the correct amount of data bytes.** Failure to do so may cause unexpected behaviour from the CubeWheel unit. Table 9 illustrates the format of a TC to the CubeWheel unit via UART and Table 10 shows the format of the unit's response to the TC.

Table 9 – Example of UART telecommand

0x1F	0x7F	TC ID	TC byte 1	TC byte 2	...	0x1F	0xFF
Start-of-message						End-of-message	

Table 10 – UART telecommand response

0x1F	0x7F	TLM frame ID	0x1F	0xFF
Start-of-message		0x00 or 0x01	End-of-message	

Whenever a data byte matches the special character, it will be replaced with the sequence 0x1F, 0x1F. This process enables the receiver of the message to differentiate between a data byte containing 0x1F and the start-of-message and end-of-message identifiers. Table 11 illustrates how a data byte matching the special character is handled.

Table 11 – Example of special character in UART data bytes

0x1F	0x7F	...	Byte <i>i-1</i>	Byte <i>i</i>	Extra byte	Byte <i>i+1</i>	...	0x1F	0xFF
Start-of-message			0x??	0x1F	0x1F	0x??		End-of-message	

## 6.2 I<sup>2</sup>C interface

### 6.2.1 Buffer and pull-up resistors



**Pull-up resistors on the I<sup>2</sup>C bus need to be supplied by the master.**

The CubeWheel unit contains an I<sup>2</sup>C buffer to isolate the unit from the rest of the I<sup>2</sup>C bus. Pull-up resistors are required on the data and clock lines between the unit and the master. The I<sup>2</sup>C logic of the unit operates at 3.3 V. **Using a higher bus voltage than 3.3 V may result in damage to the electronics.** Refer to Figure 10 for the suggested I<sup>2</sup>C connection between a CubeWheel and an I<sup>2</sup>C master.

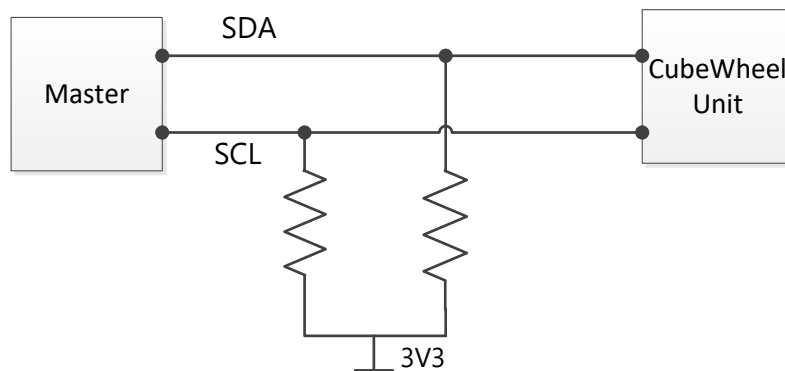


Figure 10 – Suggested I<sup>2</sup>C connection to a CubeWheel unit

### 6.2.2 Addressing

The CubeWheel unit acts as an I<sup>2</sup>C slave node with 7-bit addressing. The 8-bit read and write addresses of the unit are saved within the MCU's EEPROM and can be changed through the appropriate TC or by using the CubeSupport software.

The default I<sup>2</sup>C write address of a CubeWheel unit is 0xD0 (or 208 in decimal). The I<sup>2</sup>C read address of the CubeWheel is simply the write address plus 1. For example, if the I<sup>2</sup>C write address of a unit is 0xD0, then its I<sup>2</sup>C read address is 0xD1 (or 209). Note that the I<sup>2</sup>C write address of a CubeWheel unit must be an even number.

If multiple CubeWheel units are to be used on the same I<sup>2</sup>C bus, then their I<sup>2</sup>C write addresses must be separated by at least 2. For example, if 3 units are used, their I<sup>2</sup>C write addresses can be 208 (or 0xD0), 210 (or 0xD2), and 212 (or 0xD4).

The I<sup>2</sup>C address of a CubeWheel unit can be changed using the appropriate TC. The unit's I<sup>2</sup>C address will instantly change and will be written to the on-board EEPROM. **Wait 5 seconds**

**after sending the new I<sup>2</sup>C address to ensure the writing to EEPROM is complete.** When the unit is power cycled or manually reset (via TC) the new address will be read out of EEPROM. An I<sup>2</sup>C error flag will be set by the CubeWheel unit if an error occurred during an I<sup>2</sup>C transaction.

### 6.2.3 Protocol

Telemetry is requested from a CubeWheel unit on the I<sup>2</sup>C bus by performing a combined read-write operation. The first byte following the start condition is the write address of CubeWheel. The write address is followed by the telemetry identifier, followed by another start condition (without a preceding stop condition).

After the second start condition, the I<sup>2</sup>C read address is written by the master. The master then issues a number of read cycles depending on the length of the TLM frame. An example of the telemetry request process explained above is depicted in Figure 11. Note that an "S" indicates a start condition and a "P" indicates a stop condition.

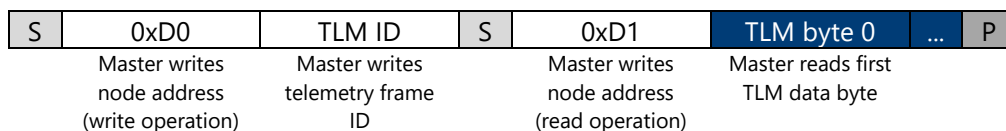


Figure 11 – Example of I<sup>2</sup>C telemetry request

Telecommands are sent by performing a master write to the CubeWheel unit. The first data byte (after the address byte) is the TC identifier, followed by the TC parameters (or data bytes). An example of the TC sending process explained above is shown in Figure 12.

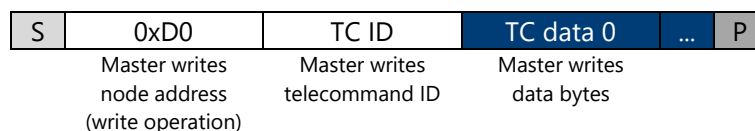


Figure 12 – Example of I<sup>2</sup>C telecommand

## 6.3 CAN interface

### 6.3.1 Bus configuration



**The CubeWheel unit does NOT have a termination resistor between the CANL and CANH lines. Clearly specify at time of placing order if termination resistor should be added.**

The CubeWheel unit contains a CAN transceiver module to isolate the unit from the rest of the CAN bus. The combination of the transceiver and an on-board CAN controller module allows the CubeWheel unit to interface with CAN bus voltage levels of 3.3 V or 5 V. There is no termination resistor populated between the CANH and CANL lines by default.

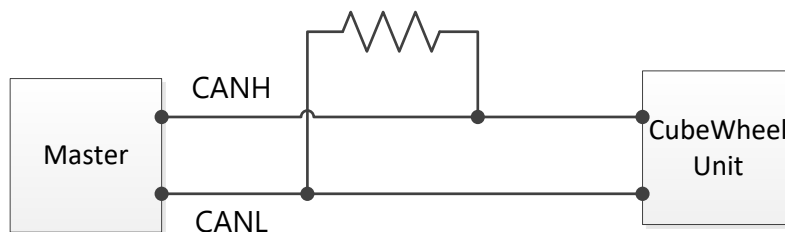


Figure 13 – Suggested CAN connection to a CubeWheel unit

The CubeWheel unit acts as a slave on a **1 Mbps** CAN bus.

### 6.3.2 Addressing

The CAN address (or mask) of a CubeWheel unit can be changed using the appropriate TC. The unit's CAN mask will instantly change and will be written to the on-board EEPROM. **Wait 5 seconds after sending the new mask to ensure the writing to EEPROM is complete.** When the unit is power cycled or manually reset (via TC) the new mask will be read out of EEPROM. A CAN error flag will be set by the CubeWheel unit if an error occurred during CAN transaction.

### 6.3.3 Protocol

The CubeWheel unit will only respond to a CAN message if the destination mask in the CAN identification packet is matched with the mask of the unit. The unit makes use of the extended identification packet (29-bit ID) and can be broken into the segments illustrated in Figure 14.

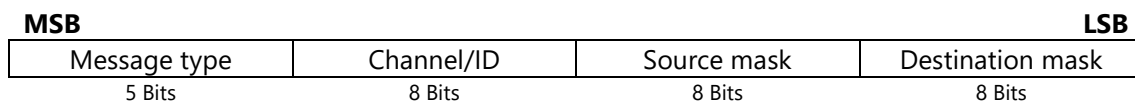


Figure 14 – CAN identification packet

The *Channel/ID* byte in Figure 14 is the specific TC or TLM ID, as defined in Section 0 of this document. The *Message Type* byte is defined in Table 12.

Table 12 – CAN message types

Message type	Identifier	Description
Telecommand Request	0x01	Command/Request
Telecommand Response	0x02	Acknowledgement
Telecommand Not Acknowledge	0x03	Command Failure
Telemetry Request	0x04	Request
Telemetry Response	0x05	Response
Telemetry No Acknowledge	0x06	Request Failure

For example, if the standard identification telemetry (ID 0x80) is requested by the master (with address/mask 0x01) from a CubeWheel unit (with address/mask 0x03) then the CAN identification packet will be [0x04, 0x80, 0x01, 0x03], as illustrated in Figure 15.

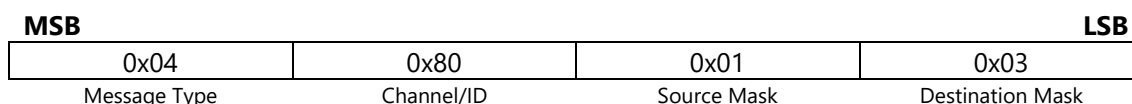


Figure 15 – Example of CAN identification packet during telemetry request

The identification packet of the CubeWheel's response to the above-mentioned telemetry request will be [0x05, 0x80, 0x03, 0x01], as illustrated in Figure 16.

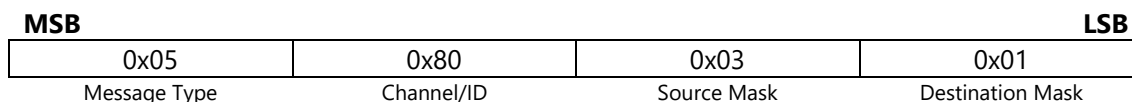


Figure 16 – Example of CAN identification packet during telemetry response

The identification packet from Figure 16 will be followed by the rest of the CAN header and the 8 byte TLM response. Telecommands and telemetry failures should only exist when supplying an invalid TC/TLM ID or not sending the correct data in the case of telecommands and will result in a CAN reply containing either a 0x03 or 0x06 message type.

## 6.4 Telecommands and telemetry requests

This section will list all the available telecommands and telemetry requests to a CubeWheel unit. Specific detail (e.g. length, bit description, etc.) regarding each TC and TLM request will also be given. Note that the tables in this section were automatically generated.

Table 13 – List of telecommands

ID	Name	Description	Length (bytes)
1	Reset	Perform a microcontroller reset	1
2	Wheel Reference Speed	Set momentum wheel reference speed	2
3	Wheel Commanded Duty	Set momentum wheel commanded duty cycle	2
7	Motor Power State	Turn motor power on/off	1
8	Encoder Power State	Turn encoder power on/off	1
9	Hall Power State	Turn Hall sensors power on/off	1
10	Control Mode	Set the motor control mode	1
12	Backup Wheel Mode	Set the back-up wheel mode	1
20	Clear Errors	Clear the processor error flags	1
31	Set I <sup>2</sup> C Address	Set I <sup>2</sup> C address	1
32	Set CAN Mask	Set CAN mask	1
33	Set PWM Gain	Set general PWM proportional gain	3
34	Set Main Gain	Set gain of main speed controller	6
35	Set Backup Gain	Set gain of backup speed controller	6

Table 14 – List of telemetry requests

ID	Name	Description	Length (bytes)
128	Identification	Identification information for this node	8
129	Extended Identification	Extended Identification information on this node	4
130	Wheel Status	Current status telemetry of wheel electronics	8
133	Wheel Speed	Wheel speed measurement	2
134	Wheel Reference	Wheel reference speed	2
135	Wheel Current	Wheel current measurement	2
137	Wheel Data	Complete wheel data	6
138	Wheel Data Additional	Additional wheel data	4
139	PWM Gain	General PWM gain	3
140	Main Gain	Main speed controller gain values	6
141	Backup Gain	Backup speed controller gain values	6
145	Status and Error Flags	Processor status and error flags	1

Table 15 – Reset command format

ID	1		Parameters (bytes)	Length	1
Description	Perform a microcontroller reset				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	8	Reset Parameter	UINT	Reset parameter. This has to be set to 85 (decimal) to perform the reset.

Table 16 – Wheel reference speed command format

ID	2	Parameters length (bytes)		2	
Description	Set momentum wheel reference speed				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Reference Speed	INT	Wheel reference speed. Raw parameter value is obtained using the formula: (raw parameter) = ((formatted value))*2.0 (formatted value is in [RPM] units)

Table 17 – Wheel commanded duty command format

ID	3		Parameters length (bytes)		2
Description	Set momentum wheel commanded duty cycle				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Duty Cycle	INT	Duty cycle of motor PWM control signal. This is the open-loop signal provided to the motor. (Unit of measure is [%])

Table 18 – Motor power state command format

ID	7		Parameters (bytes)	Length	1
Description	Turn motor power on/off				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	1	Motor Enabled State	BOOL	Enabled state of battery power to motor

Table 19 – Encoder power state command format

ID	8		Parameters length (bytes)	1	
Description	Turn encoder power on/off				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	1	Encoder Enabled State	BOOL	Enabled state of Encoder sensor

Table 20 – Hall power state command format

ID	9	Parameters length (bytes)			1
Description	Turn Hall sensors power on/off				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	1	Hall Sensor Enabled State	BOOL	Enabled state of Hall sensors

Table 21 – Control mode command format

ID	10	Parameters length (bytes)			1
Description	Set the motor control mode				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	8	Control Mode	ENUM	Control mode of motor. Possible values are in Table 22 – Control mode enumeration values

Table 22 – Control mode enumeration values

Numeric value	Name	Description
0	Idle	Idle mode
1	No Control	No control mode
2	Duty Cycle Input	Duty cycle input mode
3	Speed Controller	Speed controller mode

Table 23 – Backup wheel mode command format

ID	12		Parameters (bytes)	length	1
Description	Set the back-up wheel mode				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	1	BackupMode	BOOL	Enable the back-up mode

Table 24 – Clear errors command format

ID	20	Parameters length (bytes)			1
Description	Clear the processor error flags				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	8	Clear Error Parameter	UINT	Clear error flags parameter. This has to be set to 85 (decimal) to clear the flags



**Wait 5 seconds after setting any address or gain before sending any other commands or switching off the wheel. This ensures it is correctly written to EEPROM.**

Table 25 – Set I<sup>2</sup>C address command format

ID	31		Parameters length (bytes)		1
Description	Set I <sup>2</sup> C address				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	8	I <sup>2</sup> C Address	UINT	I <sup>2</sup> C address

Table 26 – Set CAN mask command format

ID	32	Parameters length (bytes)			1
Description	Set CAN mask				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	8	CANAddress	UINT	CAN address

Table 27 – Set PWM gain command format

ID	33	Parameters length (bytes)		3	
Description	Set general PWM proportional gain				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	K	INT	Main gain
	16	8	KMultiplier	UINT	Multiplier for main gain

Table 28 – Set main gain command format

ID	34		Parameters length (bytes)	6	
Description	Set gain of main speed controller				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Ki	UINT	Integrator gain
	16	8	KiMultiplier	UINT	Multiplier for integrator gain
	24	16	Kd	UINT	Feedback gain
	40	8	KdMultiplier	UINT	Multiplier for feedback gain

Table 29 – Set backup gain command format

ID	35		Parameters length (bytes)	6	
Description	Set gain of backup speed controller				
Parameters	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Ki	UINT	Integrator gain
	16	8	KiMultiplier	UINT	Multiplier for integrator gain
	24	16	Kd	UINT	Feedback gain
	40	8	KdMultiplier	UINT	Multiplier for feedback gain

Table 30 – Identification telemetry format

ID	128	Frame length (bytes)		8	
Description	Identification information for this node				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	8	Node type	UINT	Node type identifier. For CubeControl Motor, this field will always have the value 8
	8	8	Interface version	UINT	Interface version. This field should have a value of 1
	16	8	Firmware version (Major)	UINT	Firmware major version
	24	8	Firmware version (Minor)	UINT	Firmware minor version
	32	16	Runtime (seconds)	UINT	Number of seconds since processor start-up. (Unit of measure is [s])
	48	16	Runtime (milliseconds)	UINT	Number of milliseconds (after the integer second) since processor start-up. (Unit of measure is [ms])

Table 31 – Extended identification telemetry format

ID	129		Frame length (bytes)	4	
Description	Extended Identification information on this node				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Serial Number	UINT	Serial number of this unit
	16	8	I <sup>2</sup> C Address	UINT	I <sup>2</sup> C address of this unit
	24	8	CAN Address	UINT	CAN mask of this unit

Table 32 – Wheel status telemetry format

ID	130	Frame length (bytes)		8	
Description	Current status telemetry of wheel electronics				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Runtime (seconds)	UINT	Number of seconds since processor start-up. (Unit of measure is [s])
	16	16	Runtime (milliseconds)	UINT	Number of milliseconds (after the integer second) since processor start-up. (Unit of measure is [ms])
	32	16	Reserved	UINT	Shows 0 value
	48	8	Motor control mode	ENUM	Current control mode. Possible values are in Table 22 – Control mode enumeration values
	56	1	Backup-mode state	BOOL	Backup-mode state
	57	1	Motor switch state	BOOL	Motor switch state
	58	1	Hall sensor switch state	BOOL	Hall sensor switch state
	59	1	Encoder switch state	BOOL	Encoder switch state
	60	1	Error Flag	BOOL	Indicates whether an error has occurred

Table 33 – Wheel speed telemetry format

ID	133	Frame length (bytes)	2		
Description	Wheel speed measurement				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Wheel Speed	INT	Wheel speed measurement in rpm. Formatted value is obtained using the formula: (formatted value) [RPM] = (raw channel value)/2.0

Table 34 – Wheel reference telemetry format

ID	134	Frame length (bytes)		2	
Description	Wheel reference speed				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Wheel Reference	INT	Wheel reference speed in rpm. Formatted value is obtained using the formula: (formatted value) [RPM] = (raw channel value)/2.0

Table 35 – Wheel current telemetry format

ID	135	Frame length (bytes)	2		
Description	Wheel current measurement				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Wheel Current	UINT	Wheel current measurement. Formatted value is obtained using the formula: (formatted value) [mA] = (raw channel value)*0.48828125

Table 36 – Wheel data telemetry format

ID	137	Frame length (bytes)		6	
Description	Complete wheel data				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Wheel Speed	INT	Wheel speed measurement in rpm. Formatted value is obtained using the formula: (formatted value) [RPM] = (raw channel value)/2.0
	16	16	Wheel Reference	INT	Wheel reference speed in rpm. Formatted value is obtained using the formula: (formatted value) [RPM] = (raw channel value)/2.0
	32	16	Wheel Current	UINT	Wheel current measurement. Formatted value is obtained using the formula: (formatted value) [mA] = (raw channel value)*0.48828125

Table 37 – Wheel data additional telemetry format

ID	138	Frame length (bytes)		4	
Description	Additional wheel data				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Wheel Duty	INT	Current duty cycle command to motor
	16	16	Wheel Speed Backup	INT	Backup wheel speed measurement in rpm. Formatted value is obtained using the formula: (formatted value) [RPM] = (raw channel value)/2.0

Table 38 – PWM gain telemetry format

ID	139		Frame length (bytes)	3	
Description	General PWM gain				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	K	INT	Main gain
	16	8	Kmultiplier	UINT	Multiplier for main gain

Table 39 – Main gain telemetry format

ID	140		Frame length (bytes)	6	
Description	Main speed controller gain values				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Ki	UINT	Integrator gain
	16	8	KiMultiplier	UINT	Multiplier for integrator gain
	24	16	Kd	UINT	Feedback gain
	40	8	KdMultiplier	UINT	Multiplier for feedback gain

Table 40 – Backup gain telemetry format

ID	141		Frame length (bytes)	6	
Description	Backup speed controller gain values				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	16	Ki	UINT	Integrator gain
	16	8	KiMultiplier	UINT	Multiplier for integrator gain
	24	16	Kd	UINT	Feedback gain
	40	8	KdMultiplier	UINT	Multiplier for feedback gain

Table 41 – Status and error flags telemetry format

ID	145		Frame length (bytes)		1
Description	Processor status and error flags				
Channels	Offset (bits)	Length (bits)	Name	Data type	Description
	0	1	Invalid Telemetry	BOOL	An invalid telemetry request was received
	1	1	Invalid Telecommand	BOOL	An invalid telecommand was received
	2	1	Encoder Error	BOOL	Encoder indicates an error
	3	1	UART Error	BOOL	Error in UART protocol
	4	1	I <sup>2</sup> C Error	BOOL	Error in I <sup>2</sup> C protocol
	5	1	CAN Error	BOOL	Error in CAN protocol
	6	1	Configuration Error	BOOL	Configuration load error
	7	1	Speed Error	BOOL	Speed measurements indicates an error

## 7. Document Version History

Version	Author(s)	Pages	Date	Description of change
0.1	HWJ	ALL	04/12/2015	First draft
1.0	HWJ	ALL	08/12/2015	Added Medium and Large Wheel info
1.1	GJVV	ALL	14/03/2016	Several major updates
1.2	GJVV	8	18/03/2016	Very important change to angular momentum vector
1.3	GJVV	9,19	15/06/2016	Specifications updated Updated details for incoming health check
1.4	GJVV	ALL	26/07/2016	Updated template Updated wheel dimensions
1.5	GJVV	10,24, 25	17/08/2016	Updated power specifications Additional UART protocol information
1.6	GJVV	10	17/08/2016	Updated masses based on accurate CADs
1.7	DGS	ALL	22/11/2016	Added changes for new support PCB
1.8	MK	10	13/02/2017	Update Medium wheel specs
1.9	DGS	ALL	15/05/2017	Update for new PCB enable line
1.10	MK	ALL	22/06/2017	Removed imbalance spec; General grammar
1.11	MK	10	27/06/2017	Added new static imbalance spec
1.12	DGS	8,31, 32	12/07/2017	Reset TCMD fixed and explained currents measured during idle and no-control modes
1.13	DGS	10	25/10/2017	New power specs for vacuum bearing
1.14	DGS	All	18/07/2019	Added Small Plus descriptions
1.15	DGS	12	18/09/2019	Changed specifications table
1.16	DGS	33, 34	22/06/2021	Changed references torque