CubeRover, Flight Software Design Specifications
<Document number>

Rev.	Author	Description of changes	Date
A	C.Corpa	Initial Release	TBD

Table of Contents

1 Scope

This document presents the software design specifications of the CubeRover flight software.

2 References

- <TBD> CubeRover Requirements

3 Definitions and acronyms

4 CubeRover flight software design specifications

4.1 Software architecture overview

<insert architecture diagram, present high level>

4.2 Definition of operating modes

4.3 Software elements design specifications

4.3.1 Watchdog MCU

4.3.1.1 Description

The Watchdog MCU is responsible of sanity check of the CubeRover and is responsible of sending low-level telemetric data. The Watchdog MCU has the capacity to reset other MCU in case the latter don't respond in time to watchdog requirement.

4.3.1.2 State machine

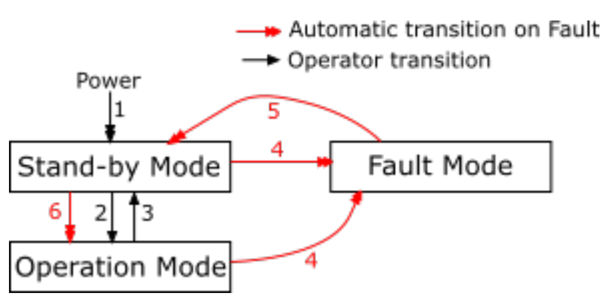


Table 1 Watchdog MCU State Machine state definition

Mode	Description
Stand-by Mode	The Wachdog MCU hold in reset all other MCUs except Primary Flight MCU. The RS422 communication interface to the lander is active but restricted to small data packets (<10 B/s) as heartbeat interval (1Hz). Heater and temperature sensing are active. The Watchdog MCU can exit the Stand-by on Operator request.
Operation Mode	On state entry, the Watchdog MCU enables the Primary Flight MCU and the watchdog monitoring for that device. The SPI interface to the Primary Flight MCU is active. The RS422 communication interface may or may not be active depending on mission progress. If it is enabled, the bandwidth is increased to 20KB/s. Watchdog of MCUs are enabled as they get activated by Primary Flight MCU.
Fault Mode	The Fault mode logs fault and do a power cycle of the Watchdog MCU and entire system.

Table 2 Watchdog MCU State Machine transition definition

#	Name	Actions
1	Power-Up	After power is applied, the Watchdog performs a POST and transition to Stand-by Mode if no fault is detected.
2	Go To Operation	On Operator request the Watchdog MCU exits the Stand-by Mode and reaches the Operation Mode.
3	Go To Stand-by	On Operator request, the Watchdog MCU exits the Operation state and return to Stand-by Mode.
4	Go To Fault	When a system fault is detected, the Watchdog MCU updates the Fault register and transits automatically to Fault Mode.
5	Power Cycle	The Watchdog hold in reset all MCUs and perform a self-reset.
6	Go To Operation	After a fault is detected and CubeRover is not attached to Lander then the MCU watchdog places the CubeRover is Operation Mode automatically.

4.3.1.3 Interfaces software to software

The Watchdog MCU has two communication interfaces to the Lander and Primary Flight MCU. The communication interface to the Lander is wired and functional until release of CubeRover from the Lander. The interface to the Primary flight is always enabled during all phases of the mission.

Figure 1 Watchdog MCU relationship between sofware components

4.3.1.3.1 Interface to Lander

During the flight and mission preparation, the CubeRover communicates to Earth using a low bandwidth RS-422 wired serial interface to perform some basic telemetry of the CubeRover. The Lander RS-422 connection is interfaced with a transceiver that interface directly the Watchdog MCU. The RS-422 connection is full duplex and can support speed up to 500KB/s.

The serial interface uses a SLIP (Serial Line Internet Protocol[1]). This interface is an encapsulation of the Internet Protocol design to work over serials ports.

[1] https://en.wikipedia.org/wiki/Serial_Line_Internet_Protocol

4.3.1.3.1.1 Heartbeat

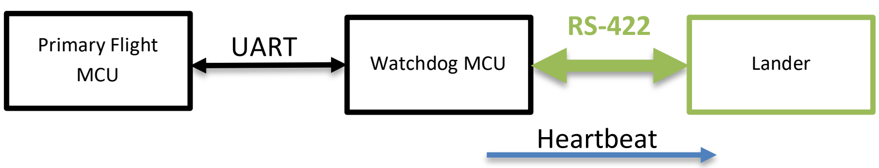
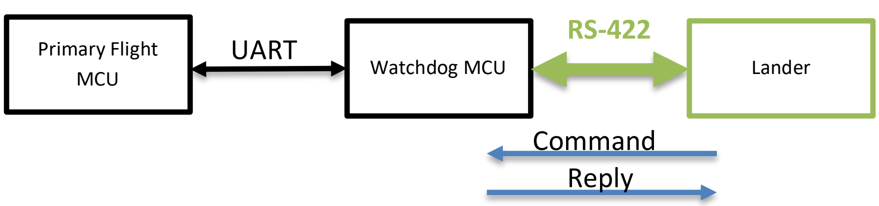


Figure 2 Software interface to support heartbeat feature to wired interface

The heartbeat is a unidirectional data transfer initiated automatically by CubeRover and transferred to PMCC during cruise and Lunar orbit are restricted to 10B/s. The purpose of the command is to provide heartbeat signal and for basic system monitoring.

The heartbeat packet data format is defined in section 12 Heartbeat Packet Format Definition.

4.3.1.3.1.2 Command / reply



The command/reply data transfers are used to perform some configuration and maintenance operation during the flight and mission preparation. Comparably to heartbeat the interface uses the RS-422 interface and SLIP Internet protocol to talk to the Lander.

The data format of a transmission (PMCC ←→ Watchdog MCU) is as follow:

Name	Header	OpCode	Command	Parameter	R/W, Counter	Data size	Data MSB	Data n	Data LSB	CRC8
Size (B)	1	1	1	1	1	4	1	n	1	1

Header:

- Command: 0001 0101b (0x15)
- Reply: 0010 0011b (0x23)

Opcode, Command and Parameter:

5 See section Header Definitions

The header indicates the provenance and destination of the packet and if the packet is a command/reply or a heartbeat message. The header is a bitwise register stored on 1 byte and is constructed as follow:

Lander Communication	Provenance	Destination	Hearbeat / Command
0b: wired 1b: wireless	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU 1xxb: reserved	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU 1xxb: reserved	0b: heartbeat 1b: command

Opcode, Command and Parameter Definitions for complete list of Opcode. The supported Opcodes by the Watchdog MCU are:

Supported OpCodes	
Telemetry Settings	0x04
Fault Masking	0x05
Fault Register	0x06
State Machines	0x07
Telemetry Data	0x08
Lander Deployment	0x09
CubeRover. Power Input Selection	0x0A
System Reset	0x0B

R/W and command counter:

It is a bitwise register that specifies if the command is a read or write. It also keeps track of message pairing through a command cyclic counter.

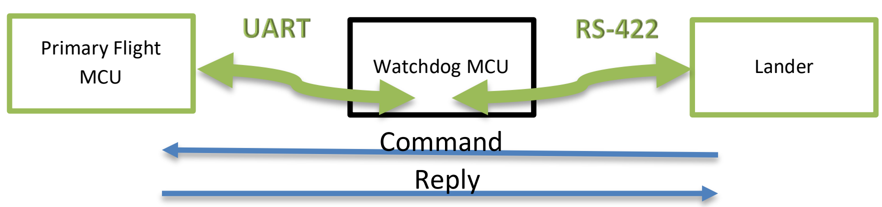
R/W, Counter	
Bit 7 .. 1	**Bit 0**
Command counter	Read (0b) Write (1b)

CRC8:

The message is terminated by a checksum byte calculated with a 0xD5 polynomial. It includes every bytes of the message.

5.1.1.1.1 Communication bridge between Lander and Primary Flight MCU

The interface to the Primary Flight MCU is done through a UART interface.



The data format of a transmission (PMCC ←→ Primary Flight MCU) is as follow:

Name	Header	OpCode	Command	Parameter	R/W, Counter	Data size	Data MSB	Data n	Data LSB	CRC8
Size (B)	1	1	1	1	1	4	1	n	1	1

Header:

Command: 0001 0111b (0x13)

Reply: 0011 0011b (0x33)

Opcode, Command and Parameter:

6 See section Header Definitions

The header indicates the provenance and destination of the packet and if the packet is a command/reply or a heartbeat message. The header is a bitwise register stored on 1 byte and is constructed as follow:

Lander Communication	**Provenance**	**Destination**	**Hearbeat / Command**
0b: wired 1b: wireless	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU 1xxb: reserved	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU 1xxb: reserved	0b: heartbeat 1b: command

Opcode, Command and Parameter Definitions for complete list of Opcode. The supported Opcodes by the Primary Flight MCU are:

Supported OpCodes	
Imaging	0x02
IMU settings	0x03
Driving	0x01
State Machines	0x07
IMU	0x03
State Machines	0x07
Motor Control Configuration	0x0C
Radio Configuration	0x0D

R/W and command counter:

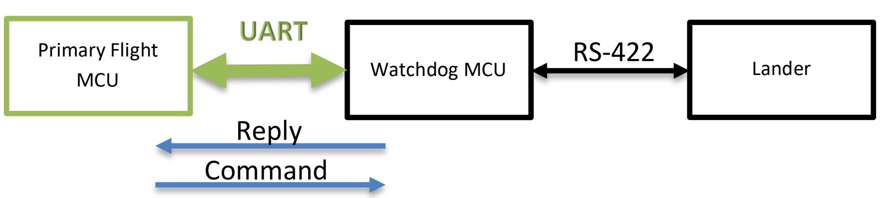
It is a bitwise register that specifies if the command is a read or write. It also keeps track of message pairing through a command cyclic counter.

R/W, Counter	
Bit 7 .. 1	**Bit 0**
Command counter	Read (0b) Write (1b)

CRC8:

The message is terminated by a checksum byte calculated with a 0xD5 polynomial. It includes every bytes of the message.

6.1.1.1.1 Interface to Primary Flight MCU



The data format of a transmission (PMCC ←→ Primary Flight MCU) is as follow:

Name	**Header**	**OpCode**	**Command**	**Parameter**	**R/W, Counter**	**Data size**	**Data MSB**	**Data n**	**Data LSB**	**CRC8**
Size (B)	1	1	1	1	1	4	1	n	1	1

Header:

Command: 0011 0101b (0x35)

Reply: 0010 0111b (0x27)

Opcode, Command and Parameter:

7 See section Header Definitions

The header indicates the provenance and destination of the packet and if the packet is a command/reply or a heartbeat message. The header is a bitwise register stored on 1 byte and is constructed as follow:

Lander Communication	Provenance	Destination	Hearbeat / Command
0b: wired 1b: wireless	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU 1xxb: reserved	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU 1xxb: reserved	0b: heartbeat 1b: command

Opcode, Command and Parameter Definitions for complete list of Opcode. The supported Opcodes by the Watchdog MCU are:

Supported OpCodes	
Telemetry Settings	0x04
Fault Masking	0x05
Fault Register	0x06
State Machines	0x07
Telemetry Data	0x08
Lander Deployment	0x09
CubeRover. Power Input Selection	0x0A
System Reset	0x0B

R/W and command counter:
It is a bitwise register that specifies if the command is a read or write. It also keeps track of message pairing through a command cyclic counter.

R/W, Counter	
Bit 7 .. 1	**Bit 0**
Command counter	Read (0b) Write (1b)

CRC8:
The message is terminated by a checksum byte calculated with a 0xD5 polynomial. It includes every bytes of the message.

7.1.1.1 Interfaces software to hardware

7.1.1.1.1 Internal and peripheral watchdogs

The Watchdog MCU verifies its own activity and other MCUs. It is capable of resetting devices that don't service the Watchdog MCU watchdog timer within a 5-second period. The service of the watchdog is done on falling or rising edge on the reserved GPIOs (see schematics for specific GPIOs). The list of monitored MCUs are the following:

- Primary Flight MCU

- Camera FPGA
- Radio module

In case of peripheral watchdog timeout failure, the Watchdog MCU sets and logs the Fault register accordingly (see section Fault Register Definition) and transits to Fault Mode.
In case of internal watchdog failure, the Watchdog MCU sets and logs Fault register then performs a system reset of CubeRover.

7.1.1.1.2 Voltage monitoring

The Watchdog MCU monitors the status of the SBC power regulators by monitoring "power good" output signals. A falling edge of these signals (see schematics for specific GPIOs) corresponds to a voltage failure. On failure, the Watchdog MCU sets and logs the fault, disable the faulty supply rail (when possible) and transits to Fault Mode.
The "power good" signals are reported to telemetry as defined in section 12 Heartbeat Packet Format Definition.

7.1.1.1.3 Heater control, thermistors and temperature protection

The Watchdog MCU controls the Heater with two modes: thermostat and manual override. In thermostat mode, the heater is controlled by the Watchdog MCU with temperature controller algorithm to maintain a minimum temperature of -20C.
The temperature is collected from 16 thermistors that are connected to ADC module. The voltage level is converted to degree celsius from a look-up table that characterize the relationship between these two properties. The temperature is logged at a rate defined by the Telemetry Settings register defined in section Opcode, Command and Parameter Definitions.
The Watchdog MCU detect failure of the thermistor by checking for short circuit and open circuit conditions. On failure, the Watchdog MCU set the Fault register accordingly, logs the Fault and transits to Safe Mode.
If one of the measured temperature exceeds the maximum allowed threshold on a given thermistor then the Watchdog MCU sets the Fault register accordingly, logs the Fault and transits to Safe Mode. The maximum temperature level are set as follow:

ADC#	Thermistor	Temperature Threshold (Celsius)
0	Temperature Sensor Primary Flight MCU (Celsius)	TBD
1	Temperature Battery Pack (Celsius)	TBD
2	Temperature Radio Module (Celsius)	TBD
3	Temperature Camera 0 (Celsius)	TBD
4	Temperature Camera 1 (Celsius)	TBD
5	Temperature Motor 0 (Celsius)	TBD
6	Temperature Motor 1 (Celsius)	TBD
7	Temperature Motor 2 (Celsius)	TBD
8	Temperature Motor 3 (Celsius)	TBD
9	Reserved	
10	Reserved	
11	Reserved	
12	Reserved	
13	Reserved	
14	Reserved	
15	Reserved	

7.1.1.1.4 Deployment control

The Watchdog MCU controls the release mechanism of CubeRover from the Lander. To perform that operation, the PMCC operator need to write to the "Lander Release" register with a password defined in the section Opcode, Command and Parameter Definitions. A release delay can be added to the release if necessary, via another register. The release mechanism is an active high signal.

7.1.1.2 Error handling

7.1.1.3 Boundary conditions

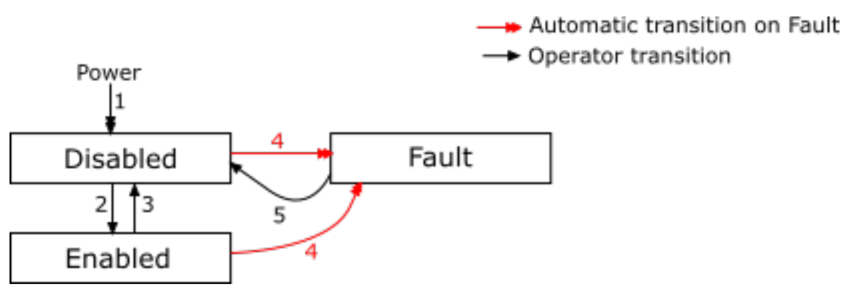
7.1.1.4 Software maintenance and revision control

7.1.2 Motor controller MCU

7.1.2.1 Description

The Motor Controller MCU is responsible to drive one DC brushless motor that drives one of the CubeRover wheel. The CubeRover SBC contains 4 Motor Controllers. They are connected to the Primary Flight MCU over a I2C interface. Each Motor Controller has a unique I2C address set by two resistors connected the MCU inputs.

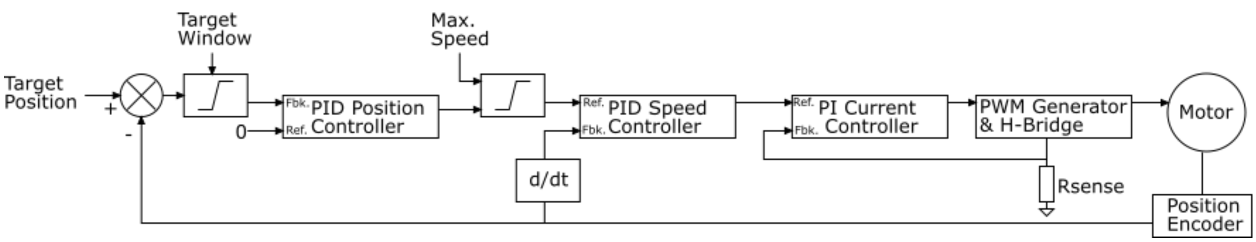
7.1.2.2 State machine



State	Description
Disabled	In Disabled state the Motor Controller can configure the motor controller parameters (PID coefficients, etc.) and motor driver is not powered.
Enabled	In Enabled state the motor driver is powered and motor controller can take actuation commands.
Fault	In Fault state the motor driver is not powered. The motor controller remains in Fault state until the fault is cleared.

#	Name	Actions
1	Power-Up	At power-up the MCU performs a POST then transitions to disabled state automatically. If a fault is present during POST, the MCU transits to Fault. The Motor Controller MCU performs the following tasks: 1. POST (clock verification, memory integrity check, DRV8304 check), 2. DRV8304 current calibration.
2	Enable Motor Driver	On Operator request, the MCU transits to Enabled state and enable motor driver.
3	Disable Motor Driver	On Operator request, the MCU transits to Disabled state and disable motor driver.
4	Go To Fault State	Under fault condition, the MCU transits automatically to Fault state. The Fault register is set accordingly.
5	Clear Fault	On Operator request, the MCU transits to Disabled state after fault is cleared successfully. If fault cannot be cleared, the MCU remains in Fault state.

7.1.2.3 Motor controller topology



The motor controller topology consists of cascaded PID/PI controllers to perform position control of the motor. The Target Position input is conditioned with a target window that normalizes and sets the position beyond which the input reaches its saturated state. The PID position controller tacks that the difference between the target position and current position is zero. The PID speed controller tracks that the speed feedback tracks the target speed. The speed feedback is calculated by deriving the position over time. Finally, the PI current controller tracks torque demand to achieve desired speed. The current feedback is measured by a current sense measuring the total current from the H-bridge circuit.

7.1.2.4 Interfaces software to software

The Motor Controller MCU interfaces the Primary Flight MCU with an I2C interface. The details on the I2C protocol are in section 9.1.1.1.1 Interface to Motor Controllers.

7.1.2.5 Interfaces software to hardware

7.1.2.5.1 Internal Watchdog

The Motor Controller MCU monitors its own activity using an internal watchdog timer. The internal watchdog timer is set to 5 seconds. If the timer triggers, the Fault register is accordingly after the Motor Controller MCU performs a reset.

7.1.2.5.2 Definition of I2C registers for hardware configuration

The Motor Controller MCU has registers that are accessible from the I2C communication interface. These registers are used from telemetric and configuration purposes and are identified with a unique identifier.

Register I2C ID	Register Name	Size (B)	Format
0x01	Motor controller I2C address	1	Uint8 (RO)

Register I2C ID	Register Name	Size (B)	Format
0x02	Relative Target Position	4	Int32 (ticks) (RW)
0x03	Direction	1	Uint8 (RW) Forward: 0 Reverse: 1
0x04	Target Speed	1	Uint8 (0-100%) (RW)
0x05	Motor Current Position	4	Int32 (ticks) (RW)
0x06	Rotor motor position	4	Int32 (ticks) (RW)
0x07	Motor Current	2	Int16 (mA) (RO)
0x08	P Current	2	Linear Format (RW)
0x09	I Current	2	Linear Format (RW)
0x0A	P Velocity	2	Linear Format (RW)
0x0B	I Velocity	2	Linear Format (RW)
0x0C	D Velocity	2	Linear Format (RW)
0x0D	P Position	2	Linear Format (RW)
0X0E	I Position	2	Linear Format (RW)
0x0F	D Position	2	Linear Format (RW)
0x10	Acceleration	2	Uint16 (tick*s-2) (RW)
0x11	Deceleration	2	Uint16 (tick*s-2) (RW)
0x12	Execute Command	1	Uint8 (RW) – write command executes new target position
0x13	Current Velocity	2	Uint16 (ticks*s-1)
0x14	Enable Driver	1	Uint8 (RW) – write command enables driver.
0x15	Disable Driver	1	Uint8 (RW) – write command disables driver.
0x16	Reset Controller	1	Uint8 (RW) – write command to reset controller.
0x17	Fault Register	1	Uint16 bitwise register (RW)
0x18	Clear Fault	1	Uint8 (RW) – write command to reset fault.
0x19	Status Register	1	Uint16 bitwise register (RW)
0x1A	Position Sensor Current Combination	1	Uint8 bitwise register (RO): Bit 0: hall sensor 1 Bit 1: hall sensor 2 Bit 2: hall sensor 3 Bit 3..7: reserved

Motor Controller Fault Register	
Bit #	**Description**
0	Driver fault/overcurrent
1	Motor stall
2	Bad position sensor
3	I2C bad parameter
4	Unexpected fault
5	Watchdog
6..7	Reserved

Motor Controller Status Register	
Bit #	**Description**
0..1	Current State: 00b: Disabled 01b: Enabled 10b: Fault
2	Target position reached (0b: not reached, 1b reached)
3..7	Reserved

7.1.2.5.3 Interface to motor driver (DRV8304)

The Motor Controller MCU interfaces a H-bridge driver DRV8304. The driver has some built-in features to protect the H-bridge and help reduces the number of discrete components by integrating some current sense circuitry.

7.1.2.5.3.1 Current Calibration

The driver DRV8304 has 3 current sense amplifiers that are used to monitor the current of each motor phases. These current amplifiers can be calibrated to consider any deviation of the common mode voltage. The calibration is done through the CAL pin. The calibration of the current sense circuitry is done at power-up of the Motor Controller MCU before entering the Disabled state.

7.1.2.5.3.2 PWM interface

The driver is configured for 6x PWM mode. Each half-bridge supported three output states: low, high and high-impedance. In that configuration, the PWM signals control the MOSFET gates and sources of the H-Bridge as follow:

INLx	INHx	GLx	GHx	SHx
0	0	L	L	Hi-Z
0	1	L	H	H
1	0	H	L	L
1	1	L	L	Hi-Z

7.1.2.5.3.3 DRV8304 fault output

The DRV8304 has a fault output (nFAULT) to notify the Motor Controller MCU of a general error (over-current, low voltage, etc). More details can be retrieved over SPI if necessary. The fault is cleared by pulsing the ENABLE to GND.

7.1.2.6 Error handling

In case of fault, the Motor Controller MCU state machine transits to Fault state. The Fault register is set accordingly and disables power output of the DRV8304. The state machine remains in Fault state until the error is cleared by the Primary Flight MCU over the I2C interface.

7.1.2.7 Boundary conditions

I2C parameters that are set by the master device are checked for boundary conditions. If a given parameter is outside acceptable range, the state machine transits to Fault state and Fault register is set accordingly.

7.1.2.8 Software maintenance and revision control

TBD

7.1.3 Camera FPGA

7.1.3.1 Description

7.1.3.2 State machine

7.1.3.3 Interfaces software to software

7.1.3.4 Interfaces software to hardware

7.1.3.5 Error handling

7.1.3.6 Boundary conditions

7.1.3.7 Software maintenance and revision control

7.1.4 Primary Flight MCU

7.1.4.1 Description

7.1.4.2 State machine

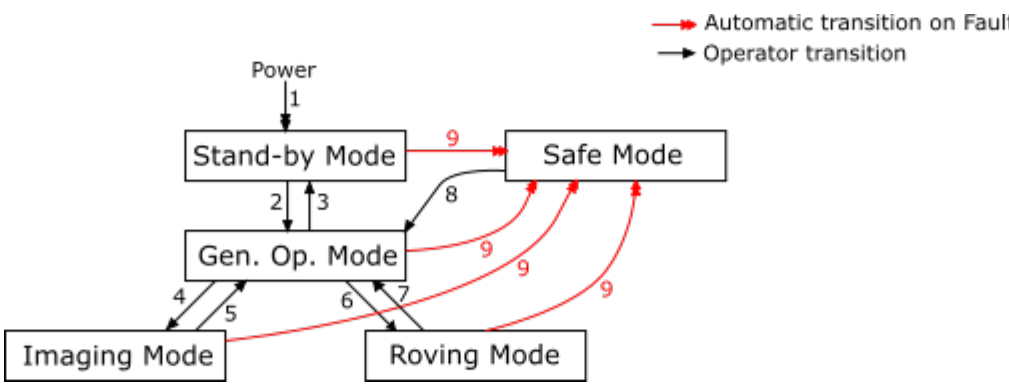


Figure 3 State machine diagram of Operational Modes

Mode	Description
Stand-by Mode	The Stand-by Mode is a mode that set all components to a low-power mode. The watchdog MCU verifies the Primary Flight MCU periodically. The Primary Flight MCU can be brought out of Stand-by Mode upon operator request.
General Op. Mode	The General Operation Mode is a mode that let the operator configures the CubeRover and retrieve some data (telemetric and payload)
Imaging Mode	The Imaging Mode is a mode that allows the CubeRover to control the camera of the CubeRover and perform imaging tasks such as take, retrieve, erase pictures from camera(s) and configure it.
Roving Mode	The Roving Mode is a mode that allows the CubeRover to actuate the motors to rove on the surface of a planet.
Safe Mode	The Safe Mode is a mode that is triggered by a CubeRover software or hardware event that requires operator intervention before returning to other modes. In Safe Mode, the operator can access all log data and fault registers.

Table 3 State machine transitions

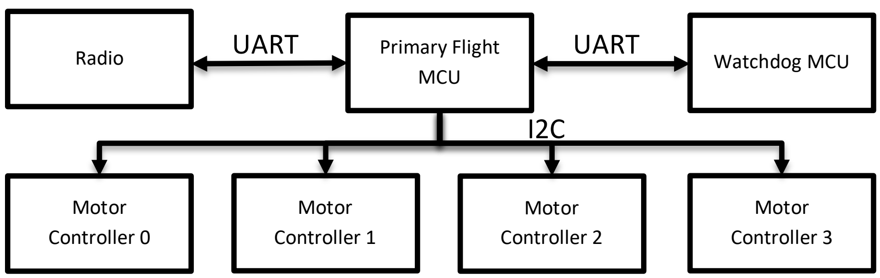
#	Name	Actions
1	Power-Up	At power-up CubeRover performs a POST then transition to Stand-by Mode automatically. The transition turns off voltage rails that are not required to support that mode and put the Watchdog MCU and all other IC in a low power mode.
2	Go To Operational	On Operator request, the CubeRover transits to General Operation Mode. The transition enables all features required to support configuration.
3	Go To Stand-by	On Operator request, the CubeRover transits to Stand-by Mode. The transition turns off voltage rails that are not required to support that mode and put the Watchdog MCU and all other IC in a low power mode.
4	Start Imaging	On Operator request, the CubeRover transits to Imaging Mode and enable camera(s) modules to perform imaging tasks.
5	Stop Imaging	On Operator request, the CubeRover transits to Imaging Mode and disable camera(s) modules.
6	Start Roving	On Operator request, the CubeRover transits to Roving Mode and enable motor controller(s).
7	Stop Roving	On Operator request, the CubeRover transits to Roving Mode and disable motor controller(s).
8	Exit Safe Mode	On Operator request, the CubeRover exits the Safe Mode. The Operator can have the option to mask failures to not revert to Safe Mode automatically.
9	Go To Safe Mode	On failure or other event detection, the CubeRover enters Safe Mode and remain until Operator requests exit of that state.

Mode	Hardware components status				
	Watchdog MCU	**Radio Module**	**Primary Flight MCU**	**Motor Controller**	**Payload- Camera / FPGA**
Stand-by Mode	Low Power	If connected to Lander: OFF. If not connected to Lander: ON	ON	MCU: OFF Pre-Driver: OFF	FPGA: OFF Camera: OFF
General Op. Mode	Low Power	ON	ON	MCU: Low Power Pre-Driver: OFF	FPGA: OFF Camera: OFF
Imaging Mode	Low Power	ON	ON	MCU: Low Power Pre-Driver: OFF	FPGA: ON Camera: ON
Roving Mode	Low Power	ON	ON	MCU: ON Pre-Driver: ON	FPGA: OFF Camera: OFF
Safe Mode	Low Power	If connected to Lander: OFF. If not connected to Lander: ON	ON	MCU: Low Power Pre-Driver: OFF	FPGA: OFF Camera: OFF

Mission phases	Description	Supported CubeRover Modes
Service	CubeRover is integrated and tested with the Peregrine Lander	Stand-by Mode General Op. Mode Imaging Mode Roving Mode
Sleep	CubeRover remains power down.	Stand-by Mode
Transit Status Check	Powers on and performs status check.	General Op. Mode

Mission phases	Description	Supported CubeRover Modes
Heartbeat	Remain in a low power state, transmits heartbeat and maintains temperature.	Stand-by Mode
Sleep (landing)	Remains power down.	Stand-by Mode
Mission Wired Status Check and charge state	Powers on and performs status check including Wi-fi and charge state.	Stand-by Mode General Op. Mode Imaging Mode Roving Mode
Deployment	Drops from Lander on signal from Astrobotic.	Stand-by Mode General Op. Mode
Mission Wireless Status Check	Performs system and mobility status check	Stand-by Mode General Op. Mode
Exploration	Awaits commands / perform mission tasks.	Stand-by Mode General Op. Mode Imaging Mode Roving Mode

7.1.4.3 Interfaces software to software



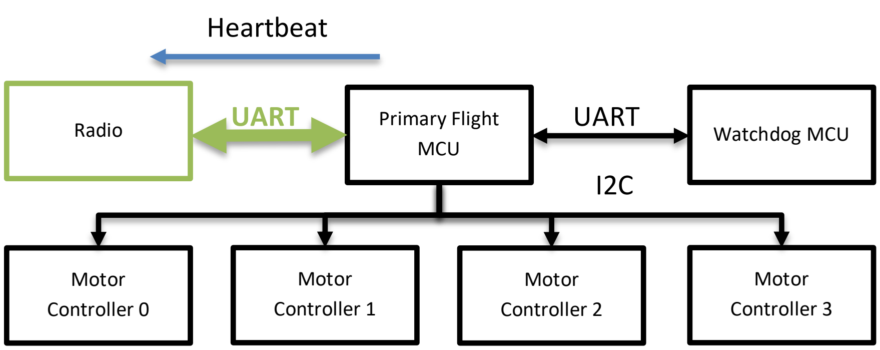
7.1.4.3.1 Interface to Watchdog MCU

Interface to Watchdog MCU is detailed in the section 6.1.1.1.1.

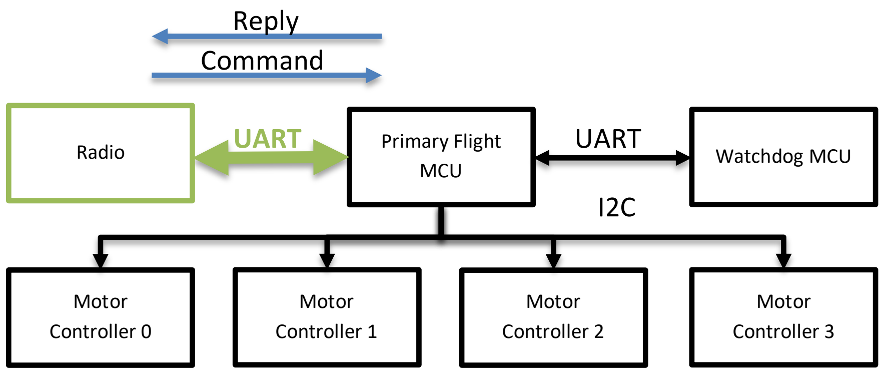
7.1.4.3.2 Interface using Radio Communication

7.1.4.3.2.1 Heartbeat

When the radio is enabled, the Primary Flight MCU transmits heartbeat at periodic interval. The periodic interval is defined by the minimum period set by the telemetric settings register. The data format of the heartbeat packet is defined in 12 Heartbeat Packet Format Definition.



7.1.4.3.2.2 Command / reply



The data format of a transmission (PMCC ←→ Watchdog MCU) is as follow:

Name	Header	OpCode	Command	Parameter	R/W, Counter	Data size	Data MSB	Data n	Data LSB	CRC8
Size (B)	1	1	1	1	1	4	1	n	1	1

Header:

Command: 1001 0111 (0x97)

Reply: 1011 0011 (0xB3)

Opcode, Command and Parameter:

8 See section Header Definitions

The header indicates the provenance and destination of the packet and if the packet is a command/reply or a heartbeat message. The header is a bitwise register stored on 1 byte and is constructed as follow:

Lander Communication	Provenance	Destination	Hearbeat / Command
0b: wired 1b: wireless	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU 1xxb: reserved	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU 1xxb: reserved	0b: heartbeat 1b: command

Opcode, Command and Parameter Definitions for complete list of Opcode. The supported Opcodes by the Primary Flight MCU are:

Supported OpCodes	
Imaging	0x02
IMU settings	0x03
Driving	0x01
State Machines	0x07
IMU	0x03
State Machines	0x07
Motor Control Configuration	0x0C
Radio Configuration	0x0D

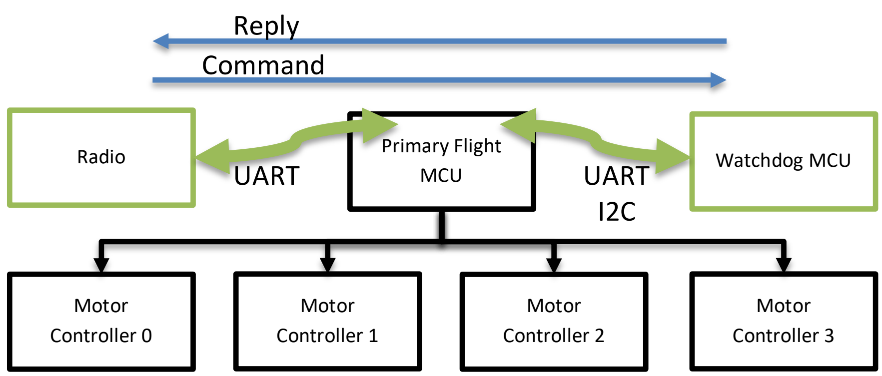
R/W and command counter:

It is a bitwise register that specifies if the command is a read or write. It also keeps track of message pairing through a command cyclic counter.

R/W, Counter	
Bit 7 .. 1	**Bit 0**
Command counter	Read (0b) Write (1b)

CRC8:
The message is terminated by a checksum byte calculated with a 0xD5 polynomial. It includes every bytes of the message.

8.1.1.1.1 Communication bridge between Radio and Watchdog MCU



The data format of a transmission (PMCC ←→ Watchdog MCU) is as follow:

Name	Header	OpCode	Command	Parameter	R/W, Counter	Data size	Data MSB	Data n	Data LSB	CRC8
Size (B)	1	1	1	1	1	4	1	n	1	1

Header:
Command: 1001 0101 (0x95)
Reply: 1010 0011 (0xA3)

Opcode, Command and Parameter:

9 See section Header Definitions

The header indicates the provenance and destination of the packet and if the packet is a command/reply or a heartbeat message. The header is a bitwise register stored on 1 byte and is constructed as follow:

Lander Communication	Provenance	Destination	Hearbeat / Command
0b: wired 1b: wireless	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU	0b: heartbeat 1b: command

	1xxb: reserved	1xxb: reserved	

Opcode, Command and Parameter Definitions for complete list of Opcode. The supported Opcodes by the Primary Flight MCU are:

Supported OpCodes	
Telemetry Settings	0x04
Fault Masking	0x05
Fault Register	0x06
State Machines	0x07
Telemetry Data	0x08
Lander Deployment	0x09
CubeRover. Power Input Selection	0x0A
System Reset	0x0B

R/W and command counter:

It is a bitwise register that specifies if the command is a read or write. It also keeps track of message pairing through a command cyclic counter.

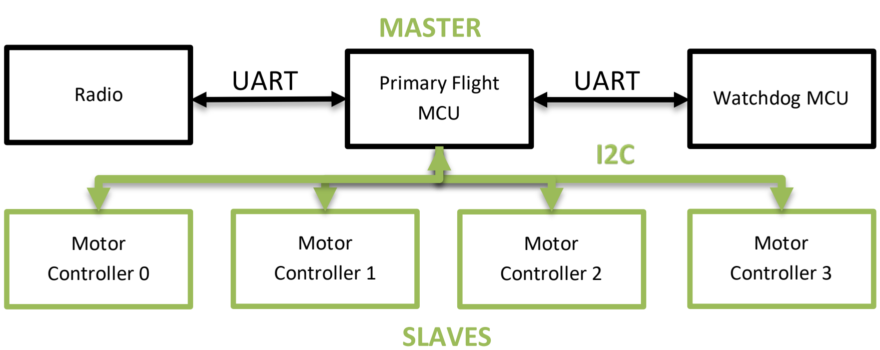
R/W, Counter	
Bit 7 .. 1	**Bit 0**
Command counter	Read (0b) Write (1b)

CRC8:

The message is terminated by a checksum byte calculated with a 0xD5 polynomial. It includes every bytes of the message.

9.1.1.1.1 Interface to Motor Controllers

The Primary Flight MCU interfaces the Motor Controllers over I2C interface. The Primary Flight MCU is configured as master device whereas the Motor Controllers are configured as slave devices.



9.1.1.1.1.1 I2C address definition

The last two digit of the I2C address are defined by a set of two resistors connected to the MCU. See SBC schematics for actual address. The address is: 0110 0xx where xx represents the address of the slave device.

9.1.1.1.1.2 I2C packet format

The I2C packets are constructed as follow:

Function	Fault & Register ID	Data Size	Data	CRC8
Size (B)	1	1	n	1

Fault &Register ID:

The register ID is a bitwise register is defined in section Motor controller MCU. They are used to access the Motor Controller MCU parameters required to perform motor control tasks and to check if the Motor Controller MCU is in a fault state. The register is defined as follow:

Function	Fault	Register ID
Bit #	7 0b: no fault 1b: fault state	6..0

Data Size:

The Data Size is 1-byte and describe the size of the Data to read or write.

Data:

The Data to read or write. For I2C Read, the data byte(s) is/are ignored by the MCU Motor Controller.

CRC8:

The message is terminated by a checksum byte calculated with a 0xD5 polynomial. It includes every bytes of the message.

9.1.1.2 Interfaces software to hardware

9.1.1.2.1 Current monitoring

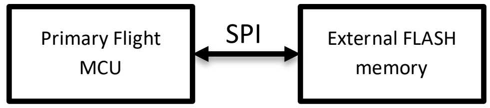
The Primary Flight MCU monitors the current of the critical components with current monitor circuits. The current circuits are connected the ADC channels as follow:

ADC#	Current monitor
0	TBD
1	
2	
3	
4	
5	
6	
7	
8	

9.1.1.2.2 External FLASH memory and telemetric data storage

The Primary Flight MCU has an external FLASH memory (S25FL064L) to store telemetric data. The communication interface to that device use a SPI interface. The communication protocol to interface that device is defined in the datasheet[2].

[2] https://www.cypress.com/file/316661/download



Each telemetric data element is stored in FLASH memory in the following format:

Data	Telemetry ID	Telemetry Data	Timestamp	CRC8
Size (B)	1	2	3	1

9.1.1.3 Error handling

In case I2C communication failure the Primary Flight MCU behaves as follow:

- I2C timeout error (timeout = 1s); the MCU retries 3 times to initiate the same command. If all retries fails, the Primary Flight MCU goes to the Safe Mode and its Fault register is set accordingly.
- Checksum error; the MCU retries 3 times to initiate the same command. If all retries fails, the Primary Flight MCU goes to Safe Mode and its Fault register is set accordingly.

With each I2C reply, the Motor Controller MCU includes a fault bit (see “Fault & Register ID” register definition).

9.1.1.4 Boundary conditions

The parameters received are checked for boundary condition. If data read is out-of-range then the Primary Flight MCU goes to Safe Mode and its fault register is set accordingly.

9.1.1.5 Software maintenance and revision control

10 Header Definitions

The header indicates the provenance and destination of the packet and if the packet is a command/reply or a heartbeat message. The header is a bitwise register stored on 1 byte and is constructed as follow:

Lander Communication	Provenance	Destination	Hearbeat / Command
0b: wired 1b: wireless	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU 1xxb: reserved	001b: PMCC 010b: Watchdog MCU 011b: Primary Flight MCU 1xxb: reserved	0b: heartbeat 1b: command

11 Opcode, Command and Parameter Definitions

Name	Op. Code	Command	Command Name	Param	Parameter name	Data size (B)	Format / permission	Value range
Driving	0x01	0x00	Forward Configuration	0x00	Distance	1	uint8 (RW)	0-255 cm
				0x01	Speed	1	uint8 (RW)	0-100 % (100 = 4cm/s)
		0x01	Reverse Configuration	0x00	Distance	1	uint8 (RW)	0-255 cm
				0x01	Speed	1	uint8 (RW)	0-100 % (100 = 4cm/s)
		0x02	Left Configuration	0x00	Angle	1	uint8 (RW)	0-180 degree
				0x01	Speed	1	uint8 (RW)	0-100% (4cm/s)
		0x03	Right Configuration	0x00	Angle	1	uint8 (RW)	0-180 degree
				0x01	Speed	1	uint8 (RW)	0-100% (4cm/s)
		0x04	Execute driving	0x00	Direction	1	uint8 (RW)	0: FWD 1: REV 2: LFT 3: RGT
Imaging	0x02	0x00	Image Compression (front)	0x01	Image Compression	1	uint8 (RW)	0: None 1: 20% 2: 50%
		0x01	Image Sensor Configuration (front)	0x00 ~ 0xFF	See MT9P031 Reference Registers	2	Uint16 (see datasheet)	See MT9P031 Reference Registers
		0x02	Image Compression (rear)	0x01	Image Compression	1	uint8 (RW)	0: None 1: 20% 2: 50%
		0x03	Image Sensor Configuration (rear)	0x00 ~ 0xFF	Direct access-See MT9P031 Reference Registers	2	Uint16 (see datasheet)	Direct access-See MT9P031 Reference Registers
		0x04	Image Transfer	0x00	Image Transfer (front)	1	Uint8 (RW)	0: start transfer 1: abort transfer
				0x01	Transfer Image (back)	1	Uint8 (RW)	0: start transfer 1: abort transfer
		0x05	Image Capture	0x00	Capture Image (front)	1	Uint8 (RW)	0: capture image
				0x01	Capture Image (back)	1	Uint8 (RW)	0: capture image
IMU	0x03	0x00	Gyrometer Direct Configuration	0x00 ~ 0xFF	Direct access – see L3GD20H datasheet	1	Uint8 (RW)	Direct access – see L3GD20H datasheet
		0x01	Accelerometer Direct Configuration	0x00 ~ 0xFF	Direct access – see ADXL312 datasheet	1	Uint8 (RW)	Direct access – see ADXL312 datasheet
		0x03	Gyrometer Data	0x00	X-axis	2	int16 (RO)	0-65535 (degree/s)
				0x01	Y-axis	2	int16 (RO)	0-65535 (degree/s)
				0x02	Z-axis	2	int16 (RO)	0-65535 (degree/s)

Name	Op. Code	Command	Command Name	Param	Parameter name	Data size (B)	Format / permission	Value range
		0x04	Accelerometer Data	0x00	X-axis	2	int16 (RO)	0-65535 (m.s-2)
				0x01	Y-axis	2	int16 (RO)	0-65535 (m.s-2)
				0x02	Z-axis	2	int16 (RO)	0-65535 (m.s-2)
Telemetry Settings	0x04	0x00	Gyro Telemetry Frequency	0x00	Active Frequency	1	Uint8 (RW)	0-255 (Hz)
				0x01	Stand-by Frequency	1	Uint8 (RW)	0-255 (Hz)
		0x01	Accelerometer Telemetry Frequency	0x00	Active Frequency	1	Uint8 (RW)	0-255 (Hz)
				0x01	Stand-by Frequency	1	Uint8 (RW)	0-255 (Hz)
		0x02	Thermistors Telemetry Frequency	0x00	Active Frequency	1	Uint8 (RW)	0-255 (Hz)
				0x01	Stand-by Frequency	1	Uint8 (RW)	0-255 (Hz)
		0x03	Motors Current Telemetry Frequency	0x00	Active Frequency	1	Uint8 (RW)	0-255 (Hz)
				0x01	Stand-by Frequency	1	Uint8 (RW)	0-255 (Hz)
		0x04	Motors Velocity Telemetry Frequency	0x00	Active Frequency	1	Uint8 (RW)	0-255 (Hz)
				0x01	Stand-by Frequency	1	Uint8 (RW)	0-255 (Hz)
		0x05	Rails Voltages Telemetry Frequency	0x00	Active Frequency	1	Uint8 (RW)	0-255 (Hz)
				0x01	Stand-by Frequency	1	Uint8 (RW)	0-255 (Hz)
		0x06	Rails Currents Telemetry Frequency	0x00	Active Frequency	1	Uint8 (RW)	0-255 (Hz)
				0x01	Stand-by Frequency	1	Uint8 (RW)	0-255 (Hz)
		0x07	Radio Telemetry Frequency	0x00	Active Frequency	1	Uint8 (RW)	0-255 (Hz)
				0x01	Stand-by Frequency	1	Uint8 (RW)	0-255 (Hz)
		0x08	Heater Activity Telemetry Frequency	0x00	Active Frequency	1	Uint8 (RW)	0-255 (Hz)
				0x01	Stand-by Frequency	1	Uint8 (RW)	0-255 (Hz)
		0x09	Thermistors Telemetry Frequency	0x00	Active Frequency	1	Uint8 (RW)	0-255 (Hz)
				0x01	Stand-by Frequency	1	Uint8 (RW)	0-255 (Hz)
		0x0A	Motors Position Telemetry Frequency	0x00	Active Frequency	1	Uint8 (RW)	0-255 (Hz)
				0x01	Stand-by Frequency	1	Uint8 (RW)	0-255 (Hz)
		0xFF	Reset parameters	-	-	-	-	-
Fault Masking	0x05	0x00	Fault Mask	0x00	Fault Mask	2	Uint16 (RW)	Bitwise masking. See Fault Table. 0: fault not masked. 1: fault masked
Fault Register	0x06	0x00	Fault	0x00	Fault Status	2	Uint16 (RO)	Bitwise register. See Fault Table.
				0x01	Clear Fault	2	Uint16 (RW)	Bitwise register. See Fault Table.
State Machines	0x07	0x00	Watchdog MCU state machine	0x00	State	1	Uint8 (RW)	0: Stand-by 1: Operation

Name	Op. Code	Command	Command Name	Param	Parameter name	Data size (B)	Format / permission	Value range
		0x01	Primary Flight MCU state machine	0x00	State	1	Uint8 (RW)	0: Stand-by 1: Gen Operation Mode 2: Imaging Mode 3: Roving Mode 4: Safe Mode
Telemetry Data	0x08	0x00	Data	Data ID 0~255	See Telemetry Table	5	RO	Telemetry Data Encoded in linear format. See Telemetry Table. Date Appended to each data.
Lander Release	0x09	0x00	Release	0x00	Trigger Release	2	Uint16 (RW)	Must be equal to: 0xA5A5 to initiate detachment.
				0x01	Deployment Delay	1	Uint8 (RW)	0-255 (s)
CubeRov. Power Input Selection	0x0A	0x00	CubeRover Power Selection	0x00	Power Selection	1	Uint8 (RW)	0: Battery 1: Lander
System Reset	0x0B	0x00	Reset Selection	0x00	Reset Selection	1	Uint8 (RW)	0: release from reset. 1: hold in reset. Bitwise register: Bit 0: Watchdog MCU Bit 1: Pri. Flight MCU Bit 2: Radio Module Bit 3: FPGA Bit 4: Motor Ctr MCU 0 Bit 5: Motor Ctr MCU 1 Bit 6: Motor Ctr MCU 2 Bit 7: Motor Ctr MCU 3
Motor Control Configuration	0x0C	0x00	Motor Control Parameters	0x00	P Current	2	Uint16(RW)	0~65535
				0x01	I Current	2	Uint16(RW)	0 ~ 65535
				0x02	P Speed	2	Uint16(RW)	0 ~ 65535
				0x03	I Speed	2	Uint16(RW)	0 ~ 65535
				0x03	D Speed	2	Uint16(RW)	0 ~ 65535
				0x04	P Speed	2	Uint16 (RW)	0 ~ 65535
				0x05	P Position	2	Uint16 (RW)	0 ~ 65535
				0x06	I Position	2	Uint16 (RW)	0 ~ 65535
				0x07	D Position	2	Uint16 (RW)	0 ~ 65535
		0x01	Acceleration /Deceleration profile	0x00	Acceleration	2	Uint16 (RW)	0 ~ 65535 cm/s2
				0x01	Deceleration	2	Uint16 (RW)	0 ~ 65535 cm/s2
		0x02	Stall Detection	0x00	Disable / Enable	1	Uint8 (RW)	0: Stall detect. disabled 1: Stall detect. Enabled
		0x03	Position Counter	0x00	Reset Counter Position to 0.	1	Uint8 (RW)	Writing reset to 0.
Radio Configuration	0x0D	0x00 ~ 0xFF	See WF-121 Datasheet for message class	0x00 ~ 0xFF	See WF-121 Datasheet for message ID	See WF-121 Datasheet for data length[3]	See WF-121 Datasheet for message ID	See WF-121 Datasheet for message ID
Heater Configuration	0x0E	0x01	Temperature Control	0x00	Temperature Heater ON	1	Int8	-128 ~ +127 (Celsius)
				0x01	Temperature Heater OFF	1	Int8	-128 ~ +127 (Celsius)

[3] https://www.silabs.com/documents/public/reference-manuals/Bluegiga-WiFi-Software-3.0-API-RM.pdf

Name	Op. Code	Command	Command Name	Param	Parameter name	Data size (B)	Format / permission	Value range
				0x02	Manual /Thermostat Selection	1	Uint8	0x00: manual control 0x01: thermostat control
				0x03	Manual Control	1	Uint8	0x00: Heater OFF 0x01: Heater ON
System Log	0x0F							

12 Heartbeat Packet Format Definition

The data format is as follow:

Name	Header	Fault Reg.	Data ID	Telemetry Data + Time	CRC8
Size (B)	1	4	1	5	1

Header:
Wired heartbeat from watchdog MCU: 0010 0010b (0x22)
Wireless heartbeat from primary Flight MCU: 1011 0010b (0xB2)

Fault register: The Fault register is stored in the Watchdog MCU. It is a bitwise register informing of fault of CubeRover (see section Fault Register Definition).

Data ID: identifier that specify what data is being returned. Data ID is incremented for each communication cycle then circle back to ID #0 when it reaches max data ID.

Telemetry Data + Time: The Time of the telemetric data collection is appended to the data in 3 bytes format. It is encoded in the following format: hhmmss in a decimal format.

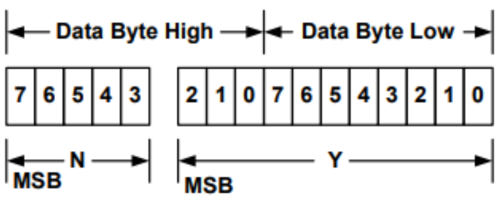
Telemetry ID	Telemetry Data	Data Size (B)	Time	Time size (B)
0	Temperature Sensor Primary MCU (Celsius)			
	Temperature Battery Pack (Celsius)			
	Temperature Radio Module (Celsius)			
	Temperature Camera 0 (Celsius)			
	Temperature Camera 1 (Celsius)			
	Temperature Motor 0 (Celsius)			
	Temperature Motor 1 (Celsius)			
	Temperature Motor 2 (Celsius)			
	Temperature Motor 3 (Celsius)			
	Temperature FPGA (Celsius)			
	Temperature Watchdog MCU (Celsius)	2	hhmmss	3
	Bitwise register: Bit 0: Power Good 5V0 Bit 1: Power Good 3V3 Bit 2: Power Good 2V8 Bit 3: Power Good 2V5 Bit 4: Power Good 1V8 Bit 5: Power Good 1V2 Bit 6..15: reserved			
	Battery Remaining Level (percentage)			
	Heater Status (bool)			
Telemetry parameter below only available over wi-fi (released from lander) or through command/reply				
	Current 5V0 (mA)			
	Current 3V3 (mA)			
	Current 2V8 (mA)			
	Current 2V5 (mA)			
	Current 1V8 (mA)			
	Current 1V2 (mA)			
	Revolution Rotor Counter Motor 0 (wheel rev.)			
	Revolution Rotor Counter Motor 1 (wheel rev.)			
	Revolution Rotor Counter Motor 2 (wheel rev.)			

Telemetry ID	Telemetry Data	Data Size (B)	Time	Time size (B)
	Revolution Rotor Counter Motor 3 (wheel rev.)			
	Velocity Rotor Counter Motor 0 (RPM)			
	Velocity Rotor Counter Motor 1 (RPM)			
	Velocity Rotor Counter Motor 2 (RPM)			
	Velocity Rotor Counter Motor 3 (RPM)			

The data is encoded with two bytes in a linear data format with:

- An 11 bit, two's complement mantissa and,
- A 5 bit, two's complement exponent (scaling factor)

The format of the two data bytes is illustrated below:



The relation between Y, N and the real world value is = $X = Y2^N$ where X is the real world value; Y is an 11 bit, two's complement integer; and N is a 5 bit, two's complement integer.

CRC8: The message is terminated by a checksum byte calculated with a 0xD5 polynomial. It includes every bytes of the message.

13 Fault Register Definition

The Fault register is stored in the Watchdog MCU. It is a bitwise register informing of fault of CubeRover.

Bit#	Fault
0..3	**Temperature Sensor Failure:** 0000b: no error 0001b: Primary MCU 0010b: Battery 0011b: Radio 0100b: Camera 0 0101b: Camera 1 0110b: Motor 0 0111b: Motor 1 1000b: Motor 2 1001b: Motor 3 1010b: FPGA 1011b: Watchdog MCU 1100b: reserved 1101b: reserved 1110b: reserved 1111b: reserved

Bit#	Fault
4..7	**Watchdog Timeout:** 0000b: no error 0001b: Motor Controller 0 0010b: Motor Controller 1 0011b: Motor Controller 2 0100b: Motor Controller 3 0101b: Primary Flight MCU 0110b: FPGA 0111b: Watchdog MCU 1000b: Radio Module
8..11	**Temperature Out-Of-Range:** 0000b: no error 0001b: Primary MCU 0010b: Battery 0011b: Radio 0100b: Camera 0 0101b: Camera 1 0110b: Motor 0 0111b: Motor 1 1000b: Motor 2 1001b: Motor 3 1010b: FPGA 1011b: Watchdog MCU 1100b: reserved 1101b: reserved 1110b: reserved 1111b: reserved
12..14	**Voltage failure:** 000b: no error 001b: 5V0 010b: 3V3 011b: 2V8 100b: 2V5 101b: 1V8 110b: 1V2 111b: reserved
15..17	**Current failure:** 000b: no error 001b: 5V0 010b: 3V3 011b: 2V8 100b: 2V5 101b: 1V8 110b: 1V2 111b: reserved
18	Motor 0 stall
19	Motor 1 stall
20	Motor 2 stall
21	Motor 3 stall
22	Prohibited Op Code
23	Motor Controller I2C communication error
24..31	Reserved