x-IMU3 User Manual

v1.1

April 6, 2023

x-io Technologies

Contents

1 Overview

The x-IMU3 is x-io Technologies' third generation Inertial Measurement Unit (IMU). It is a high-performance and versatile measurement device designed to accommodate a wide range of data logging and real-time applications including biomechanics, motion-capture, virtual reality, drones, robotics, and industrial.

USB, Wi-Fi and Bluetooth provide connectivity for mobile and desktop devices while serial communication supports embedded and industrial systems. An on-board microSD card allows the x-IMU3 to function as a stand-alone data logger with the ability to download files by USB and Wi-Fi. Multiple x-IMU3s operating together on the same wireless network will automatically synchronise to stream or log synchronised measurements.

Sensors

- Gyroscope, ±2000 °/s, 400 Hz
- Accelerometer, ±24 g, 400 Hz
- Magnetometer, ±2.5 uT, 20 Hz
- High-g accelerometer, ±200 g, 1600 Hz
- Temperature sensor[1]

Calibration

- 15-parameter calibration for: axis sensitivity, axis offset, inter-axis misalignment, and package misalignment.
- Hard-iron and soft-iron calibration
- On-board gyroscope bias correction algorithm

AHRS

- Algorithm outputs:
  - Quaternion
  - Rotation matrix
  - Euler angles
  - Linear acceleration
  - Earth acceleration
- Linear acceleration rejection
- Magnetic distortion rejection
- 400 Hz update rate
- Static accuracy:
  - 0.5 ° RMS inclination
  - 1 ° RMS heading

Communication

- USB (CDC)
- Serial, 3.3V UART
- TCP (Wi-Fi)
- UDP (Wi-Fi)
- Bluetooth[2]

Wi-Fi

- Client and AP mode
- Dual band (2.4 GHz, 5 GHz)
- WPA/WPA2-Personal
- WPA/WPA2-Enterprise[3]

Data logging

- Supports microSD cards up to 32 GB[4]
- Start/stop logging remotely
- USB download
- Wi-Fi download
- CSV output

Serial accessories

- Receive data from external sensors and user electronics, e.g. GPS, analogue/digital inputs, application-specific sensors.
- 3.3 V output to power external electronics

Battery

- Internal battery charged by USB
- 20 hours data logging
- 15 hours Bluetooth
- 12 hours Wi-Fi client 2.4 GHz
- 8 hours Wi-Fi client 5 GHz

Housing

- IP67
- Wearable strap or chassis mount

Software GUI

- Real-time data plots and 3D view
- Log real-time data to CSV
- Forward real-time data to other applications
- Windows and macOS

Software API

- Rust, C, C++, C#, Python
- Code examples for other languages available

[1] The temperature sensor is used for calibration and is not intended to provide an accurate measurement of ambient temperature.
[2] Bluetooth support is currently in development and not yet supported.
[3] WPA/WPA2-Enterprise security is currently in development and not yet supported. Will only be supported in client mode.
[4] The product is supplied with an 8 GB microSD card that can be upgraded by the user.

2 Hardware

2.1 Board

Board components are annotated in Figure 1. A detailed mechanical drawing describing the board dimensions and locations of key components is available on the product web page.

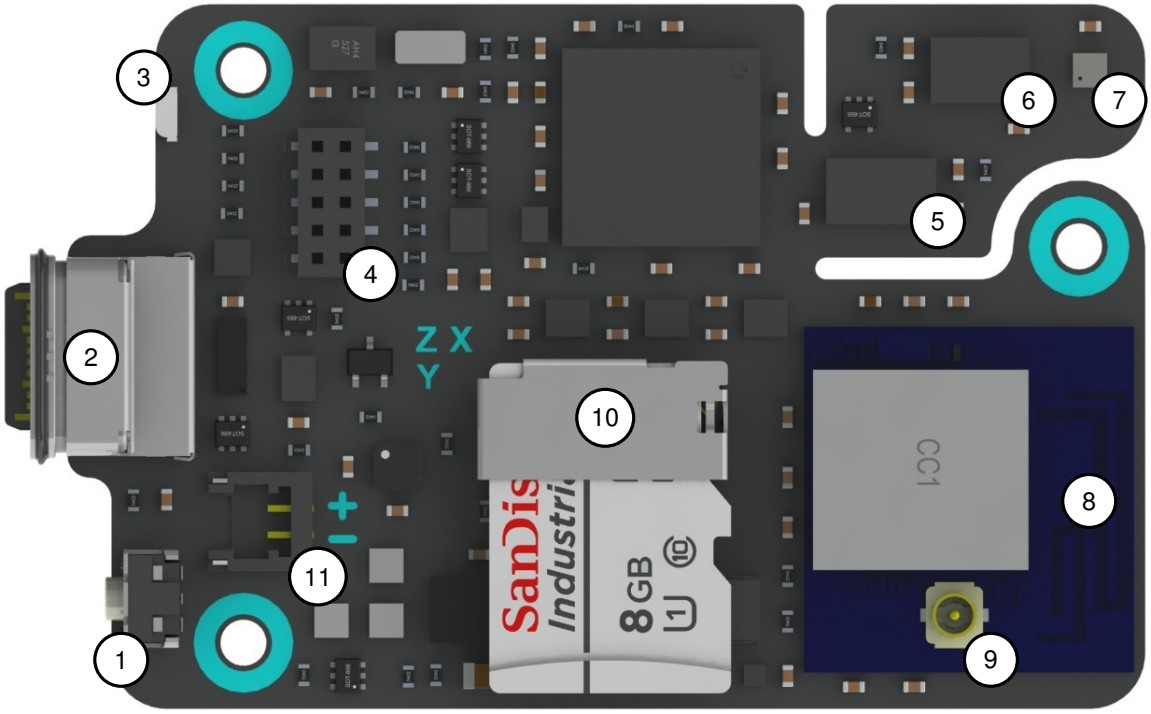

Figure 1: Board

1. Power button
2. USB-C connector
3. LED
4. Serial header
5. High-g accelerometer
6. Inertial sensor (gyroscope and accelerometer)
7. Magnetometer
8. Wireless antennae
9. U.FL connector for external wireless antennae
10. microSD card socket
11. Battery connector

2.2 Housing

The housing interfaces are annotated in Figure 2. A detailed mechanical drawing describing the housing dimensions is available on the product web page.

Figure 2: Housing

1. Power button
2. USB-C connector
3. LED

2.2.1 IP67 rating

The Ingress Protection 67 (IP67) rating is an international standard that describes the ability of the housing to protect against the ingress of solid particles and water. The first digit, 6 indicates complete protection against dust and solid particles. The second digit, 7 indicates protection from water for a maximum depth of 1 meter for up to 30 minutes.

In practical terms, this means that the housing can be used outdoors in all weather conditions and that it will survive accidental or temporary submersion in water. The housing should not be used in underwater applications.

3 Technical specification

3.1 Temperature

3.1.1 No battery

Characteristic	Value	Notes
Operating	-40 °C to 85 °C	1, 2
Storage	-40 °C to 105 °C	-

Table 1: Temperature specification (no battery)

Notes

1. The temperature of the device will always be greater than the surroundings due to heat generated by electronics.
2. The specified accuracy of the device is not achieved over the full operating temperature range. See Section 4 on page 12 for more information.

3.1.2 With battery

Characteristic	Value	Notes
Operating (discharging)	-20 °C to 60 °C	1, 2
Operating (charging)	0 °C to 45 °C	1, 2, 3
Storage	-20 °C to 25 °C	-

Table 2: Temperature specification (with battery)

Notes

1. The temperature of the device will always be greater than the surroundings due to heat generated by electronics.
2. The specified accuracy of the device is not achieved over the full operating temperature range. See Section 4 on page 12 for more information.
3. Charging at temperatures below 0 °C will reduce the capacity and cycle life of the battery.

3.2 Sensors

3.2.1 Gyroscope

Characteristic	Value	Notes
Range	±2000 °/s	-
Resolution	16-bit, 0.061 °/s	-
Sample rate	400 Hz ±0.3%	1
Bandwidth	47 Hz	2
Noise	0.014 °/s/$\sqrt{}$Hz	-

Table 3: Gyroscope specification

Notes

1. Each sample includes a timestamp for a reliable measurement of time independent of the sample rate error. See Section 9 on page 31 for more information.
2. The maximum bandwidth is achieved when the message rate is equal to the sample rate. If the message rate is less than the sample rate then samples are averaged. See Section 9 on page 31 for more information.

3.2.2 Accelerometer

Characteristic	Value	Notes
Range	±24 g	-
Resolution	16-bit, 732 µg	-
Sample rate	400 Hz ±0.3%	1
Bandwidth	145 Hz	2
Noise	160 µg/$\sqrt{\text{Hz}}$ (X, Y), 190 µg/$\sqrt{\text{Hz}}$ (Z)	-
Accuracy at 1 g	TBC	3, 4

Table 4: Accelerometer specification

Notes

1. Each sample includes a timestamp for a reliable measurement of time independent of the sample rate error. See Section 9 on page 31 for more information.
2. The maximum bandwidth is achieved when the message rate is equal to the sample rate. If the message rate is less than the sample rate then samples are averaged. See Section 9 on page 31 for more information.
3. The accuracy at 1 g is evaluated as the deviation of the measured magnitude of gravity for a 360° rotation around the X, Y, and Z axis aligned to the horizontal axis.
4. Accuracy is specified for the calibrated temperature only. See Section 4 on page 12 for more information.

3.2.3 Magnetometer

Characteristic	Value	Notes
Range	±1300 µT (X, Y), ±2500 µT (Z)	-
Sample rate	20 Hz ±8%	1
Noise	0.3 µT	-
Accuracy at 1 a.u.	TBC	2, 3, 4

Table 5: Magnetometer specification

Notes

1. Each sample includes a timestamp for a reliable measurement of time independent of the sample rate error. See Section 9 on page 31 for more information.
2. The calibrated magnetometer units are arbitrary units (a.u.). 1 a.u. is equal to the magnitude of the ambient magnetic field during calibration, approximately 50 µT.
3. The accuracy at 1 a.u. is evaluated as the deviation of the measured magnitude of the ambient magnetic field for a 360° rotation around the X, Y, and Z axis aligned to the vertical axis.
4. Accuracy is specified for the calibrated temperature only. See Section 4 on page 12 for more information.

3.2.4 High-g accelerometer

Characteristic	Value	Notes
Range	±200 g	-
Resolution	16-bit, 6.1 mg	-
Sample rate	1600 Hz ±2%	1
Bandwidth	800 Hz	2
Noise	5 mg/$\sqrt{}$Hz	-
Accuracy at 1 g	TBC	3, 4

Table 6: High-g accelerometer specification

Notes

1. Each sample includes a timestamp for a reliable measurement of time independent of the sample rate error. See Section 9 on page 31 for more information.
2. The maximum bandwidth is achieved when the message rate is equal to the sample rate. If the message rate is less than the sample rate then samples are averaged. See Section 9 on page 31 for more information.
3. The accuracy at 1 g is evaluated as the deviation of the measured magnitude of gravity for a 360° rotation around the X, Y, and Z axis aligned to the horizontal axis.
4. Accuracy is specified for the calibrated temperature only. See Section 4 on the next page for more information.

3.2.5 Temperature sensor

Characteristic	Value	Notes
Range	-104 ℃ to 150 ℃	1
Sample rate	5 Hz ±0.3%	2, 3
Accuracy	±1 ℃ at 25 ℃	-

Table 7: Temperature sensor specification

Notes

1. The temperature sensor measurement range exceeds the device operating temperature range. See Section 3.1 on page 9 for more information.
2. Each sample includes a timestamp for a reliable measurement of time independent of the sample rate error. See Section 9 on page 31 for more information.
3. The temperature sensor is oversampled and the result decimated to the specified sample rate.

3.3 Data logger capacity

3.3.1 8 GB microSD card

Condition	Value	Notes
Default message rates	485 hours	1
Maximum message rates	37 hours	1

Table 8: Data logger capacity for 8 GB microSD card

Notes

1. The data logging capacity is indicated for binary data messages. Capacity will be reduced for ASCII data messages.

3.3.2 32 GB microSD card

Condition	Value	Notes
Default message rates	1941 hours	1, 2
Maximum message rates	149 hours	1, 2

Table 9: Data logger capacity for 32 GB microSD card

Notes

1. The data logging capacity is indicated for binary data messages. Capacity will be reduced for ASCII data messages.
2. The product is supplied with an 8 GB microSD card that can be upgraded by the user. The maximum compatible microSD card capacity is 32 GB.

4 Calibration

Each device is calibrated during production to achieve the specified accuracy. The calibration process uses specialist equipment and propriety algorithms to calculate calibration parameters specific to each device. These parameters are used by the calibration models described in the following sections to compensate for the characteristics of each measurement source. Calibration is performed at room temperature and accuracy will be reduced for operating temperatures that deviate from this temperature. Please refer to the calibration certificate for specific temperature values.

4.1 Inertial sensors

The inertial sensors are the gyroscope, accelerometer, and high-g accelerometer. Each inertial sensor is calibrated for axis sensitivity, axis offset, inter-axis misalignment, and package misalignment. The inertial calibration model is described by Equation (1) where i_c is the calibrated inertial measurement obtained from the uncalibrated inertial measurement, i_u, given the misalignment matrix, M, the sensitivity diagonal matrix, s, and the offset vector, b. The inertial calibration model is expanded as Equation (2) on the next page to express the model as 15 scalar quantities. The units of i_c, i_u, and b are degrees per second for the gyroscope, and g for the accelerometer and high-g accelerometer. M and s are ratios and therefore have no units. The calibration parameters M, s, and b for each inertial sensor can be accessed as device settings.

$$i_c = Ms(i_u - b) \tag{1}$$

$$\begin{bmatrix} i_{cx} \\ i_{cy} \\ i_{cz} \end{bmatrix} = \begin{bmatrix} m_{xx} & m_{xy} & m_{xz} \\ m_{yx} & m_{yy} & m_{yz} \\ m_{zx} & m_{zy} & m_{zz} \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \left(\begin{bmatrix} i_{ux} \\ i_{uy} \\ i_{uz} \end{bmatrix} - \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \right) \tag{2}$$

4.2 Magnetometer

The magnetometer is calibrated for soft iron and hard-iron characteristics. Soft iron characteristics are distortions that alter the intensity and direction of the magnetic field as measured by the magnetometer. Soft iron calibration also accounts for magnetometer axis sensitivity, inter-axis misalignment, and package misalignment. Hard iron characteristics are unintended magnetic fields generated by the device that offset magnetometer measurements. Hard iron calibration also accounts for magnetometer axis offset.

The magnetometer calibration model is described by Equation (3) where m_c is the calibrated magnetometer measurement obtained from the uncalibrated magnetometer measurement, m_u, given the soft iron matrix, S, the hard iron vector, h. The magnetometer calibration model is expanded as Equation (4) to express the model as 12 scalar quantities. The units of m_c, m_u, and h are a.u.. S is a ratio and therefore has no units. The calibration parameters S, h can be accessed as device settings.

$$m_c = Sm_u - h \tag{3}$$

$$\begin{bmatrix} m_{cx} \\ m_{cy} \\ m_{cz} \end{bmatrix} = \begin{bmatrix} s_{xx} & s_{xy} & s_{xz} \\ s_{yx} & s_{yy} & s_{yz} \\ s_{zx} & s_{zy} & s_{zz} \end{bmatrix} \begin{bmatrix} m_{ux} \\ m_{uy} \\ m_{uz} \end{bmatrix} - \begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} \tag{4}$$

4.3 Calibration certificate

Each device is supplied with a calibration certificate unique to that device. The calibration certificate details all calibration parameters, the calibration date, the ambient temperature and device temperature during calibration, and any equipment used during the calibration process. The certificate also includes graphs demonstrating the accuracy of the device over the measurement range. Calibration certificates are provided as a Portable Document Format (PDF) file. There are three ways to access the calibration certificate for a device:

1. Scan the Quick Response (QR) code on the back of the device.
2. Open the “Calibration Certificate.html” file stored on the micro Secure Digital (microSD) card.
3. Enter the device serial number on the calibration certificate web page.

5 Power button

Pressing the power button while the device is switched off will switch the device on. Pressing and holding the power button for two seconds while the device is switched on will switch the device off.

A timestamped notification message containing the string, ”Button pressed.” is sent each time the button is pressed. This allows the button to function as a basic user input for real-time applications or as a means for marking events during data logging. If the button is used in this way then the user must be careful not to hold the button for too long otherwise the device may be switched off unintentionally.

6 LED

The Light-Emitting Diode (LED) indicates the mode and status of the device using different colours and flashing behaviours.

6.1 Wireless disabled (green)

A green LED, as shown in Figure 3 indicates that the device is switched on and that the wireless mode is disabled.

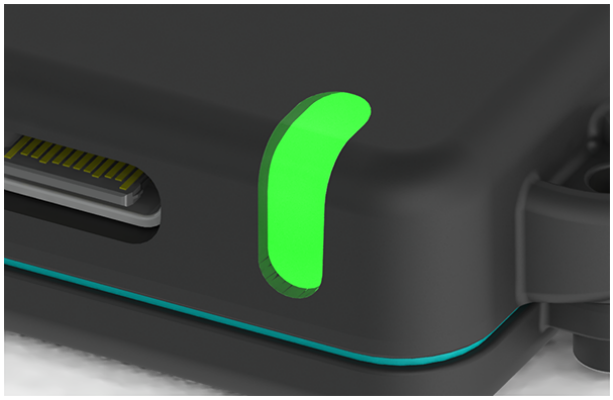

Figure 3: Green LED indicating that the device is switched on and that the wireless mode is disabled

6.2 Wi-Fi client (cyan)

A cyan LED, as shown in Figure 4 indicates that the device is switched on and in Wi-Fi client mode. Slow flashing (once per second) indicates that device is not connected to an Access Point (AP), fast flashing (five times per second) indicates that the device is connected to an AP but has not yet obtained an Internet Protocol (IP) address, and a solid LED indicates that the device is connected to an AP and has an IP address.

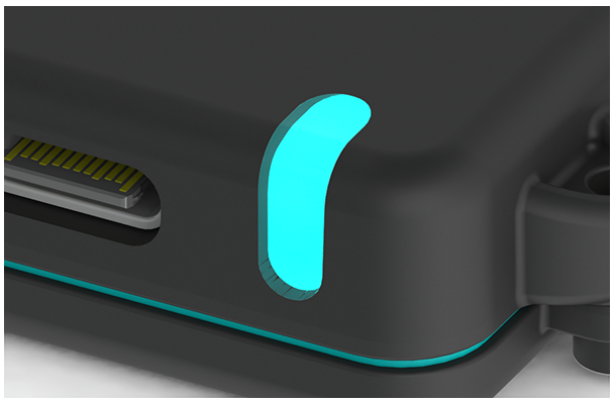

Figure 4: Cyan LED indicating that the device is switched on and in Wi-Fi client mode

6.3 Wi-Fi AP (magenta)

A magenta LED, as shown in Figure 5 on the following page indicates that the device is switched on and in Wi-Fi AP mode. The LED will flash during the initialisation of the Wi-Fi network. Once the network has been created, the LED will remain solid.

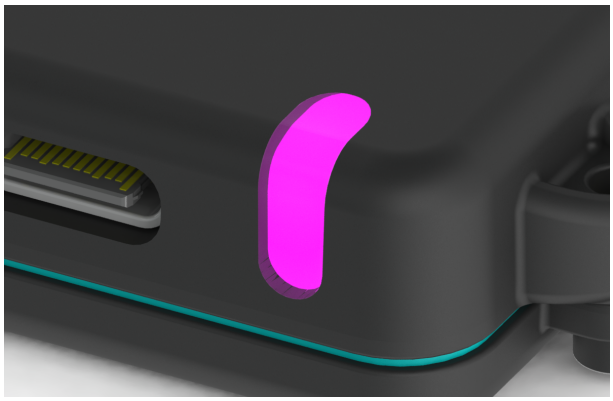

Figure 5: Magenta LED indicating that the device is switched on and in Wi-Fi AP mode

6.4 Bluetooth (blue)

A blue LED, as shown in Figure 6 indicates that the device is switched on and in Bluetooth mode. Slow flashing (once per second) indicates that Bluetooth is not connected and the device is not discoverable, fast flashing (five times per second) indicates that Bluetooth is not connected and the device is discoverable, and a solid LED indicates that Bluetooth is connected. The device is not discoverable while Bluetooth is connected.

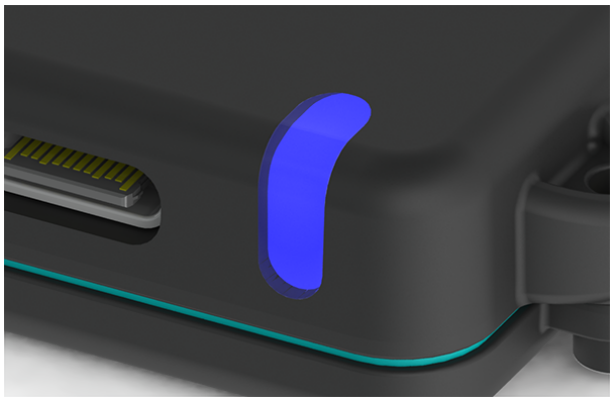

Figure 6: Blue LED indicating that the device is switched on and in Bluetooth mode

6.5 Error (red)

A red LED, as shown in Figure 7 on the following page indicates an error. The LED will interrupt it's normal behaviour to blink red each time an error message is sent by the device.

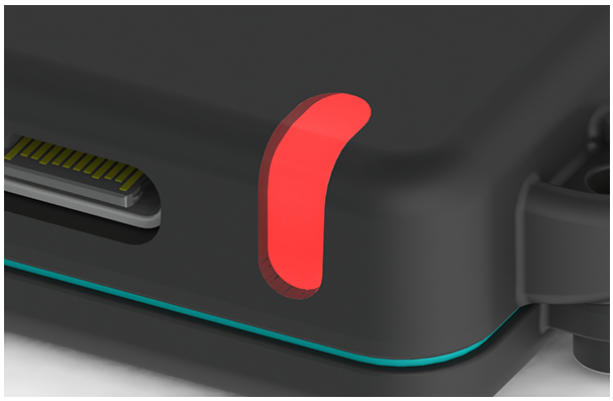

Figure 7: Red LED indicating an error

6.6 Low battery and charging (orange)

An orange LED, as shown in Figure 8 indicates either a low battery the device is switched on, or the charging status if the device is switched off. The LED will interrupt it's normal behaviour to blink orange once a second to indicate that the battery is low. If the device is switched off and USB power is connected then the LED will remain solid while the device is charging and blink once every four seconds once charging is complete.

Figure 8: Orange LED indicating low battery or charging status

6.7 User control

The LED can be controlled by the user using the strobe and colour commands. See Section 8.1.12 on page 20 and Section 8.1.13 on page 20 for more information.

7 Data logger

The device can function as a stand-alone data logger by streaming real-time data to a file on the microSD card. Files created by the data logger use the .ximu3 extension and can be downloaded from the device to be converted to Comma-Separated Values (CSV) files using the product software.

The data logger will create a new file in the "Data Logger" directory on the microSD card each time logging starts. The data logger will never overwrite data. If the micro microSD card becomes full then the data logger will stop and the device will indicate an error.

7.1 Start and stop

The data logger is enabled or disabled in the device settings. If the data logger is enabled then logging will start when the device is switched on and stop when the device is switched off. Alternatively, an application can start and stop logging remotely by enabling and disabling the data logger while the device is switched on.

Logging will stop automatically when a Universal Serial Bus (USB) host is connected or when a Hypertext Transfer Protocol (HTTP) client connects. The data logger will start again when the USB host is disconnected or when the HTTP client disconnects. Connecting USB power alone will not stop logging.

7.2 File name

The file name format is “prefix_YYYY-MM-DD_hh-mm-ss_CCCC.ximu3” where where ”prefix” is a user-defined label configured in the device settings, “YYYY-MM-DD_hh-mm-ss” is the time that the file was created, and “CCCC” is a counter. If the prefix is left blank then the device serial number will be used with the format “XXXX-XXXX-XXXX-XXXX”. The time and counter parts of the file name can be individually enabled or disabled in the device settings. For example, if the counter was disabled and the prefix left blank for a device with the serial number “0123-4567-89AB-CDEF” then a file created at 3.30 p.m. on January 20, 2025 would have the name “0123-4567-89AB-CDEF_2025-01-20_15-30-00.ximu3”.

The counter is a four digit number between 0000 and 9999 that increments each time it is used. If a file name using the counter already exists then the counter will increment until the file name is available. Incrementing beyond 9999 will cause the counter to wraparound to 0000. If the counter part of the file name is disabled and the file name already exists then the counter will used automatically to create an available file name.

7.3 File contents

The contents of the file is a byte stream as per the communication protocol described in Section 8. Each file starts with a preamble of the following messages, in order.

1. Ping response
2. Write time command
3. Write setting command for each setting

8 Communication protocol

All communication interfaces use the same communication protocol. The byte stream is therefore identical for USB, serial, Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Bluetooth, and the files created by the data logger. The communication protocol consists of two message types:

- Command messages
- Data messages

All messages are terminated by a Line Feed (LF) control character. This termination byte will not appear anywhere else in a message and so can be used to divide a byte stream into individual messages. Some messages are terminated with an additional Carriage Return (CR) control character. Table 10 describes the different ways that the control characters LF and CR may be referred to throughout this document.

Control character	Abbreviation	String	Hex	Decimal
Line Feed	LF	“\n”	0x0A	10
Carriage Return	CR	“\r”	0x0D	13

Table 10: Control characters LF and CR representations

The first byte of a message indicates the message type. Command messages start with the character “{” (0x7B in hex, 123 in decimal). Data messages start with either an uppercase character or a byte value

greater than 0x80 (128 in decimal) depending on the message.

8.1 Command messages

Command messages are sent to the device to read and write settings and execute commands. All command messages are a JavaScript Object Notation (JSON) object containing a single key/value pair, terminated by the control character sequence: CR, LF. The control character CR is optional. The control character LF must not appear anywhere else in a command message. The device will acknowledge each received command message by sending a command message with the same key to the host.

The key used by command messages sent to the device is not case sensitive and can use non-alphanumeric characters arbitrarily. For example, “serialNumber”, “Serial Number”, and “serial_number” are all valid keys for a command message to read the device serial number.

8.1.1 Read setting command

The read setting command is sent to the device to read a setting value. The key is the setting key and the value is null. See Section 11.1 on page 33 for a complete list of settings. The device will acknowledge a read setting command by sending a write setting command to the host.

Example: `{"serialNumber":null}\r\n`

8.1.2 Write setting command

The write setting command is sent to the device to write a setting value, or sent from the device to the host in response to a read setting command. The key is the setting key and the value is the setting value. See Section 11.1 on page 33 for a complete list of settings. The device will acknowledge a write setting command by sending a setting write command back to the host, indicating the new settings value. The device will not apply new settings until two seconds after the most recent write setting command or default command was received.

Example: `{"deviceName":"x-IMU3"}\r\n`

8.1.3 Default command

The default command is sent to the device to set all settings to default values. The key is “default” and the value is null. The device will not apply new settings until two seconds after the most recent write setting command or default command was received.

Example: `{"default":null}\r\n`

8.1.4 Apply command

The apply command is sent to the device to apply all settings. The key is “apply” and the value is null. This command can be sent after a write setting or default command to apply settings immediately instead of after a two second delay.

Example: `{"apply":null}\r\n`

8.1.5 Save command

The save command is sent to the device to save all settings to Electrically Erasable Programmable Read-Only Memory (EEPROM). The key is “save” and the value is null. The command acknowledgement will not be sent

until the save is complete. This may take up to 300 milliseconds. The save command is unnecessary in most applications because the device will automatically save all settings on shutdown.

Example: `{"save":null}\r\n`

8.1.6 Read time command

The read time command is sent to the device to read the date and time of the Real-Time Clock (RTC). The key is “time” and the value is null. The device will acknowledge a read time command by sending a write time command to the host.

Example: `{"time":null}\r\n`

8.1.7 Write time command

The write time command is sent to the device to write the date and time of the RTC, or sent from the device to the host in response to a read time command. The key is “time” and the value is a string expressing the date and time in the format “YYYY-MM-DD hh:mm:ss” where each delimiter can be any non-numerical character. The device will acknowledge a write time command by sending a write time command back to the host, indicating the new date and time.

Example: `{"time":"2020-01-01 00:00:00"}\r\n`

8.1.8 Ping command

The ping command is sent to the device to trigger a ping response. The key is “ping” and the value is null. The device will acknowledge a ping command by sending a ping response to the host.

Example: `{"ping":null}\r\n`

8.1.9 Ping response

The ping response is sent from the device to the host in response to the ping command. The key is “ping” and the value is a JSON object containing three key/value pairs indicating the communication interface, device name, and device serial number. The keys are “interface”, “deviceName”, and “serialNumber”, respectively and all values are string types.

Example*:

```
{
  "ping": {
    "interface":     "USB",
    "deviceName":    "x-IMU3",
    "serialNumber":  "0123-4567-89AB-CDEF"
  }
}\r\n
```

* The actual JSON will not include any whitespace.

8.1.10 Reset command

The reset command is sent to the device to reset the device. The key is “reset” and the value is null. A reset is equivalent to switching the device off and then on again. The device will reset two seconds after receiving this command.

Example: {"reset":null}\r\n

8.1.11 Shutdown command

The shutdown command is sent to the device to switch the device off. The key is “shutdown” and the value is null. The device will shutdown two seconds after receiving this command.

Example: {"shutdown":null}\r\n

8.1.12 Strobe command

The strobe command is sent to the device to strobe the LED bright white for 5 seconds. The key is “strobe” and the value is null. This command can be used to quickly find a specific device when using multiple devices.

Example: {"strobe":null}\r\n

8.1.13 Colour command

The colour command is sent to the device to set the LED colour. The key is “colour” or “color” and the value is either a Red Green Blue (RGB) hex triplet expressed as a string, or null. Setting the colour will override the normal LED behaviour. A value of null will restore the normal behaviour.

Example: {"colour":"FFFFFF"}\r\n

8.1.14 Heading command

The heading command is sent to the device to set the heading of the orientation measurement provided by the Attitude Heading Reference System (AHRS) algorithm. The key is “heading” and the value is a number equal to the heading in degrees. The heading command can only be used if the magnetometer is ignored in the AHRS settings.

Example: {"heading":0}\r\n

8.1.15 Serial accessory command

The serial accessory command is sent to the device to transmit data to a serial accessory when the serial interface is in serial accessory mode. The key is “accessory” and the value is the data expressed as a string of up to 256 characters. Longer strings will be truncated to the maximum size. The string escape sequence “\u” can be used to express any byte value as per the JSON specification.

Example: {"accessory":"hello \u0077\u006F\u0072\u006C\u0064"}\r\n

8.1.16 Note command

The note command is sent to the device to generate a timestamped notification message containing a user-defined string. The key is “note” and the value is the string of up to 127 characters. Longer strings will be truncated to the maximum size. This command can be used to create timestamped notes of events during data logging.

Example: {"note":"Something happened."}\r\n

8.1.17 Format command

The format command is sent to the device to format the microSD card. The key is “format” and the value is null. The command acknowledgement will not be sent until the format is complete. This will take approximately 3 seconds for an 8 GB SD card. Larger SD cards will take longer to format. Formatting will erase all data on the SD card.

Example: `{"format":null}\r\n`

8.1.18 Self-test command

The self-test command is sent to the device to perform a self-test. The key is “test” and the value is null. The device will acknowledge a self-test command by sending a self-test response to the host once the self-test is complete. This may take up to 5 seconds. The device must be stationary during the self-test.

Example: `{"test":null}\r\n`

8.1.19 Self-test response

The self-test response is sent from the device to the host in response to the self-test command. The key is “test” and the value is a JSON object containing multiple key/value pairs. Each key/value pair indicates the result of a test. Each key is the test name and the value is the string ”Passed” if the test was passed.

Example*:

```
{
  "test": {
    "EEPROM":                "Passed",
    "RTC":                   "Passed",
    "Inertial":              "Passed",
    "Magnetometer":          "Passed",
    "High-g Accelerometer":  "Passed",
    "Battery":               "Passed",
    "SD Card":               "Passed",
    "Wireless":              "Passed",
  }
}\r\n
```

* The actual JSON will not include any whitespace.

8.1.20 Bootloader command

The bootloader command is sent to the device to put the device in bootloader mode. The key is “bootloader” and the value is null. The device will enter bootloader mode two seconds after receiving this command.

Example: `{"bootloader":null}\r\n`

8.1.21 Factory command

The factory command is sent to the device to enable factory mode. The key is “factory” and the value is null. In factory mode, read-only settings can be written using the write setting command, the save command will succeed when the EEPROM Cyclic Redundancy Check (CRC) has failed, and the erase command will be enabled.

Example: `{"factory":null}\r\n`

> **Warning** - Incorrect use of this command may permanently damage the device. Do not use this command unless instructed by customer support.

8.1.22 Erase command

The erase command is sent to the device to erase the EEPROM. The key is “erase” and the value is null. The command acknowledgement will not be sent until the erase is complete. This will take approximately 700 milliseconds. This command can only be used after the factory command.

Example: `{"erase":null}\r\n`

> **Warning** - Incorrect use of this command may permanently damage the device. Do not use this command unless instructed by customer support.

8.2 Data messages

Data messages are sent from the device to the host to provide timestamped measurements, serial accessory data, notifications, and error messages. Data messages will be either American Standard Code for Information Interchange (ASCII) or binary, depending on the device settings.

ASCII data messages consist of multiple comma-separated values terminated by the control character sequence: CR, LF. The first value is a single uppercase character indicating the message type. The second value is the timestamp in microseconds. The remaining values are arguments specific to the message type.

Binary data messages are a sequence of bytes terminated by the control character LF. The first byte of the sequence indicates the message type. The value of this byte is equal to 0x80 plus the first character of the equivalent ASCII message. The next eight bytes are the timestamp in microseconds expressed as a 64-bit unsigned integer. The remaining bytes are arguments specific to the message type. Numerical types use little-endian ordering. Byte stuffing is used to remove all occurrences of the control character LF prior to the termination byte.

8.2.1 Byte stuffing

Byte stuffing ensures that the termination byte value, 0x0A, only occurs at the end of a binary data message. This is achieved by replacing all occurrences of the termination byte prior to termination with an escape sequence. This process is identical to Serial Line Internet Protocol (SLIP) except that the “END” byte value is defined as 0x0A. Table 11 lists the values used by the byte stuffing process.

Hex	Decimal	Name	Description
0x0A	10	END	Message termination
0xDB	219	ESC	Message escape
0xDC	220	ESC_END	Transposed message termination
0xDD	221	ESC_ESC	Transposed message escape

Table 11: Values used by the byte stuffing process

Byte stuffing is achieved by the following:

- Replace each occurrence of END in the original message with the two byte sequence: ESC, ESC_END.
- Replace each occurrence of ESC in the original message with the two byte sequence: ESC, ESC_ESC.

The byte stuffing process will not modify the END that terminates the message. Table 12 demonstrates byte stuffing for example byte sequences terminated as binary data messages.

Example	Before byte stuffing	After byte stuffing
1	45 58 41 4D 50 4C 45 0A	45 58 41 4D 50 4C 45 0A
2	45 0A 41 4D 50 4C 45 0A	45 DB DC 41 4D 50 4C 45 0A
3	45 58 DB 4D 50 4C 45 0A	45 58 DB DD 4D 50 4C 45 0A
4	45 58 41 4D 50 DB 0A 0A	45 58 41 4D 50 DB DD DB DC 0A

Table 12: Byte stuffing examples

8.2.2 Inertial message

The inertial message provides timestamped gyroscope and accelerometer measurements. Inertial messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character "I" and the arguments are six numerical values expressed to four decimal places. The first byte of a binary message is 0xC9 (equal to 0x80 + "I") and the arguments are six contiguous 32-bit floats. The message arguments are described in Table 13.

Argument	Description
1	Gyroscope X axis in degrees per second
2	Gyroscope Y axis in degrees per second
3	Gyroscope Z axis in degrees per second
4	Accelerometer X axis in g
5	Accelerometer Y axis in g
6	Accelerometer Z axis in g

Table 13: Inertial message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Gyroscope X axis = 0
2. Gyroscope Y axis = 0
3. Gyroscope Z axis = 0
4. Accelerometer X axis = 0
5. Accelerometer Y axis = 0
6. Accelerometer Z axis = 1

ASCII example: `I,1000000,0.0000,0.0000,0.0000,0.0000,0.0000,1.0000\r\n`

Binary example: `C9 40 42 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 3F 0A`

8.2.3 Magnetometer message

The magnetometer message provides timestamped magnetometer measurements. Magnetometer messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character "M" and the arguments are three numerical values expressed to four decimal places.

The first byte of a binary message is 0xCD (equal to 0x80 + “M”) and the arguments are three contiguous 32-bit floats. The message arguments are described in Table 14.

Argument	Description
1	Magnetometer X axis in a.u.
2	Magnetometer Y axis in a.u.
3	Magnetometer Z axis in a.u.

Table 14: Magnetometer message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Magnetometer X axis = 1
2. Magnetometer Y axis = 0
3. Magnetometer Z axis = 0

ASCII example: `M,1000000,1.0000,0.0000,0.0000\r\n`

Binary example: `CD 40 42 0F 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 0A`

8.2.4 Quaternion message

The quaternion message provides timestamped measurements of the orientation of the device relative to the Earth. Quaternion messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character “Q” and the arguments are four numerical values expressed to four decimal places. The first byte of a binary message is 0xD1 (equal to 0x80 + “Q”) and the arguments are four contiguous 32-bit floats. The message arguments are described in Table 15.

Argument	Description
1	Quaternion W element
2	Quaternion X element
3	Quaternion Y element
4	Quaternion Z element

Table 15: Quaternion message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Quaternion W element = 1
2. Quaternion X element = 0
3. Quaternion Y element = 0
4. Quaternion Z element = 0

ASCII example: `Q,1000000,1.0000,0.0000,0.0000,0.0000\r\n`

Binary example: `D1 40 42 0F 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 00 00 00 00 0A`

8.2.5 Rotation matrix message

The rotation matrix message provides timestamped measurements of the orientation of the device relative to the Earth. Rotation matrix messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character “R” and the arguments are nine numerical values expressed to four decimal places. The first byte of a binary message is 0xD2 (equal to 0x80 + “R”) and the arguments are nine contiguous 32-bit floats. The message arguments are described in Table 16.

Argument	Description
1	Rotation matrix XX element
2	Rotation matrix XY element
3	Rotation matrix XZ element
4	Rotation matrix YX element
5	Rotation matrix YY element
6	Rotation matrix YZ element
7	Rotation matrix ZX element
8	Rotation matrix ZY element
9	Rotation matrix ZZ element

Table 16: Rotation matrix message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Rotation matrix XX element = 1
2. Rotation matrix XY element = 0
3. Rotation matrix XZ element = 0
4. Rotation matrix YX element = 0
5. Rotation matrix YY element = 1
6. Rotation matrix YZ element = 0
7. Rotation matrix ZX element = 0
8. Rotation matrix ZY element = 0
9. Rotation matrix ZZ element = 1

ASCII example:

```
R,1000000,1.0000,0.0000,0.0000,0.0000,1.0000,0.0000,0.0000,0.0000,1.0000\r\n
```

Binary example:

```
D2 40 42 0F 00 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 3F 0A
```

8.2.6 Euler angles message

The Euler angles message provides timestamped measurements of the orientation of the device relative to the Earth. Euler angles messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character “A” and the arguments are three numerical values expressed to four decimal places. The first byte of a binary message is 0xC1 (equal to 0x80 + “A”) and the arguments are three contiguous 32-bit floats. The message arguments are described in Table 17 on the following page.

Argument	Description
1	Roll angle in degrees
2	Pitch angle in degrees
3	Yaw angle in degrees

Table 17: Euler angles message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Roll angle = 0
2. Pitch angle = 0
3. Yaw angle = 0

ASCII example: `A,1000000,0.0000,0.0000,0.0000\r\n`

Binary example: `C1 40 42 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A`

8.2.7 Linear acceleration message

The linear acceleration message provides timestamped measurements of linear acceleration and the orientation of the device relative to the Earth. Linear acceleration messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character "L" and the arguments are seven numerical values expressed to four decimal places. The first byte of a binary message is 0xCC (equal to 0x80 + "L") and the arguments are seven contiguous 32-bit floats. The message arguments are described in Table 18.

Argument	Description
1	Quaternion W element
2	Quaternion X element
3	Quaternion Y element
4	Quaternion Z element
5	Linear acceleration X axis in g
6	Linear acceleration Y axis in g
7	Linear acceleration Z axis in g

Table 18: Linear acceleration message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Quaternion W element = 1
2. Quaternion X element = 0
3. Quaternion Y element = 0
4. Quaternion Z element = 0
5. Linear acceleration X axis = 0
6. Linear acceleration Y axis = 0
7. Linear acceleration Z axis = 0

ASCII example: `L,1000000,1.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000\r\n`

Binary example: `CC 40 42 0F 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A`

8.2.8 Earth acceleration message

The Earth acceleration message provides timestamped measurements of Earth acceleration and the orientation of the device relative to the Earth. Earth acceleration messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character "E" and the arguments are seven numerical values expressed to four decimal places. The first byte of a binary message is 0xC5 (equal to 0x80 + "E") and the arguments are seven contiguous 32-bit floats. The message arguments are described in Table 19.

Argument	Description
1	Quaternion W element
2	Quaternion X element
3	Quaternion Y element
4	Quaternion Z element
5	Earth acceleration X axis in g
6	Earth acceleration Y axis in g
7	Earth acceleration Z axis in g

Table 19: Earth acceleration message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Quaternion W element = 1
2. Quaternion X element = 0
3. Quaternion Y element = 0
4. Quaternion Z element = 0
5. Earth acceleration X axis = 0
6. Earth acceleration Y axis = 0
7. Earth acceleration Z axis = 0

ASCII example: `E,1000000,1.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000\r\n`

Binary example: `C5 40 42 0F 00 00 00 00 00 00 80 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A`

8.2.9 High-g accelerometer message

The High-g accelerometer message provides timestamped high-g accelerometer measurements. High-g accelerometer messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character "H" and the arguments are three numerical values expressed to four decimal places. The first byte of a binary message is 0xC8 (equal to 0x80 + "H") and the arguments are three contiguous 32-bit floats. The message arguments are described in Table 20 on the next page.

Argument	Description
1	High-g accelerometer X axis in g
2	High-g accelerometer Y axis in g
3	High-g accelerometer Z axis in g

Table 20: High-g accelerometer message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. High-g accelerometer X axis = 0
2. High-g accelerometer Y axis = 0
3. High-g accelerometer Z axis = 1

ASCII example: `H,1000000,0.0000,0.0000,1.0000\r\n`

Binary example: `C8 40 42 0F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 3F 0A`

8.2.10 Temperature message

The temperature message provides timestamped temperature measurements. Temperature messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character “T” and the argument is a numerical value expressed to four decimal places. The first byte of a binary message is 0xD4 (equal to 0x80 + “T”) and the argument is a 32-bit float. The message arguments are described in Table 21.

Argument	Description
1	Temperature in degrees Celsius

Table 21: Temperature message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Temperature = 25

ASCII example: `T,1000000,25.0000\r\n`

Binary example: `D4 40 42 0F 00 00 00 00 00 00 00 41 C8 0A`

8.2.11 Battery message

The battery message message provides timestamped measurements of the battery level, voltage, and charger status. Battery message messages are sent continuously at the message rate configured in the device settings. The first value of an ASCII message is the character “B” and the arguments are four numerical values expressed to four decimal places. The first byte of a binary message is 0xC2 (equal to 0x80 + “B”) and the arguments are four contiguous 32-bit floats. The message arguments are described in Table 22 on the next page.

Argument	Description
1	Battery percentage
2	Battery voltage in volts
3	Charging status (See Table 23)

Table 22: Battery message arguments

Charging status	Description
0	Not connected
1	Charging
2	Charging complete

Table 23: Charging status enumeration

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Percentage = 100
2. Voltage = 4.2
3. Charging status = 2

ASCII example: `B,1000000,100.0000,4.2000,2.0000\r\n`

Binary example: `C2 40 42 0F 00 00 00 00 00 00 00 C8 42 66 66 86 40 00 00 00 40 0A`

8.2.12 RSSI message

The Received Signal Strength Indicator (RSSI) message provides timestamped Wi-Fi RSSI measurements. RSSI messages are sent continuously at the message rate configured in the device settings. RSSI messages will only be sent if the device is configured as a Wi-Fi client. The first value of an ASCII message is the character “W” and the arguments are two numerical values expressed to four decimal places. The first byte of a binary message is 0xD7 (equal to 0x80 + “W”) and the arguments are two contiguous 32-bit floats. The message arguments are described in Table 24.

Argument	Description
1	RSSI percentage
2	RSSI power in dBm

Table 24: RSSI message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. RSSI percentage = 100
2. RSSI power = -50

ASCII example: `W,1000000,100.000,-50.0000\r\n`

Binary example: `D7 40 42 0F 00 00 00 00 00 00 00 C8 42 00 00 48 C2 0A`

8.2.13 Serial accessory message

The serial accessory message provides timestamped received serial accessory data. Serial accessory messages are sent as serial accessory data is received as configured in the device settings. The first value of an ASCII message is the character “S” and the argument is the received data. Received byte values less than 0x20 or greater than 0x7E will be replaced with the character “?” so that the argument is a string of printable characters. The string is not null-terminated. The first byte of a binary message is 0xD3 (equal to 0x80 + “S”) and the argument is the unmodified received data. The message arguments are described in Table 25.

Argument	Description
1	Received serial accessory data

Table 25: Serial accessory message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. Data = 0x61 0x62 0x63 0x31 0x32 0x33 0xF1 0xF2 0xF3

ASCII example: `S,1000000,abc123???\r\n`

Binary example: `D3 40 42 0F 00 00 00 00 00 61 62 63 31 32 33 F1 F2 F3 0A`

8.2.14 Notification message

The notification message provides timestamped notifications of system events. Notification messages may be sent by the device at any time and cannot be disabled. The first value of an ASCII message is the character “N”. The first byte of a binary message is 0xCE (equal to 0x80 + “N”). The argument of both ASCII and binary messages is a string of printable characters. The string is not null-terminated. The message arguments are described in Table 26.

Argument	Description
1	Notification string

Table 26: Notification message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. String = This is a notification message.

ASCII example: `N,1000000,This is a notification message.\r\n`

Binary example: `CE 40 42 0F 00 00 00 00 00 54 68 69 73 20 69 73 20 61 20 6E 6F 74 69 66 69 63 61 74 69 6F 6E 20 6D 65 73 73 61 67 65 2E 0A`

8.2.15 Error message

The error message provides timestamped notifications of errors. Error messages may be sent by the device at any time and cannot be disabled. The first value of an ASCII message is the character “F”. The first byte of a binary message is 0xC6 (equal to 0x80 + “F”). The argument of both ASCII and binary messages is a string of printable characters. The string is not null-terminated. The message arguments are described in Table 27 on the following page.

Argument	Description
1	Error string

Table 27: Notification message arguments

The following message examples are for a timestamp of 1 second (1,000,000 microseconds) and argument values of:

1. String = This is an error message.

ASCII example: `F,1000000,This is an error message.\r\n`

Binary example: `C6 40 42 0F 00 00 00 00 00 54 68 69 73 20 69 73 20 61 6E 20 65 72 72 6F 72 20 6D 65 73 73 61 67 65 2E 0A`

9 Sample rates, message rates, and timestamps

This section describes the relationship between sample rates and message rates, and the role of timestamps in synchronisation.

9.1 Sample rates

The sample rate is the rate at which measurements are sampled by a measurement source. For example, an Analog-to-Digital Converter (ADC). All sample rates are fixed and cannot be adjusted by the user. Each measurement source has an independent sample clock. The sample rate and associated data messages for each measurement source are listed in Table 28.

Measurement source	Sample rate	Sample rate error	Data messages
Inertial sensor	400 Hz	±0.3%	Inertial, Quaternion, Rotation matrix Euler angles, Linear acceleration, Earth acceleration, Temperature
Magnetometer	20 Hz	±8%	Magnetometer
High-g accelerometer	1600 Hz	±5%	High-g accelerometer
Battery voltage	5 Hz	-	Battery
RSSI	1 Hz	-	RSSI

Table 28: Sample rate and associated data messages for each measurement source

9.2 Message rates

The message rate is the rate at which a data message is sent. The message rate of each data message type is configured by a separate message rate divisor in the device settings. A message rate divisor is a positive integer that a fixed sample rate is divided by to obtain the message rate. For example, if the inertial message rate divisor is 4 then the inertial message rate will be 100 messages per second. A message rate divisor of 0 will disable the sending of that data message type. See Section 11.1.73 on page 49 to Section 11.1.79 on page 52 for a detailed description and examples for each message rate divisor setting.

9.3 Sample averaging

If a message rate divisor is greater than 1 then the measurements in each data message will be the average of the most recent *n* samples where *n* equal to the message rate divisor. The timestamp of the data message will

be that of the most recent sample. For example, if the inertial message rate divisor is 8 then the measurements in each inertial message will be the average of 8 samples and the timestamp of the message will be that of the 8^{th} sample.

9.4 Timestamps

The timestamp of a data message indicates the time at which a measurement was obtained. For example, when an ADC conversion completes. Timestamps are therefore not affected by the sample rate error or the latency of a communication interface. Applications that involve time-dependent calculations such as numerical integration or interpolation should not infer timing from the nominal sample rate and should instead use the timestamp of each measurement. A timestamp is the number of microseconds since device start up with a resolution of one microsecond.

9.5 Synchronisation

Multiple devices operating on the same Wi-Fi network will automatically synchronise so that the timestamps from all devices become the number of microseconds since the device start up of the device that was switched on first. The sample clocks of synchronised devices will remain asynchronous. If an application requires synchronous sampling then this must be achieved in post-processing through interpolation and resampling.

10 Network announcement message

The network announcement message is used by a host to discover and connect to devices on the same network. The message is continuously broadcast by the device on UDP port 10000 at a fixed rate of one message per second. Each message provides the device name, serial number, Wi-Fi and battery status, as well as the device settings required for a host to establish a TCP or UDP connection. The message is a single JSON object. The key/value pairs are described in Table 29.

Key	Value type	Description
“sync”	number	Used for synchronisation
“name”	string	Device name
“sn”	string	Device serial number
“ip”	string	Device IP address
“port”	number	TCP port
“send”	number	UDP send port (device sends to this port)
“receive”	number	UDP receive port (device receives on this port)
“rssi”	number	RSSI percentage (-1 in Wi-Fi AP mode)
“battery”	number	Battery percentage
“status”	number	Charging status (See Table 23 on page 29)

Table 29: Network announcement message key/value pairs

Example*:

```
{
  "sync":     0,
  "name":     "x-IMU3",
  "sn":       "0123-4567-89AB-CDEF",
  "ip":       "192.168.1.1",
  "port":     7000,
  "send":     8000,
  "receive":  9000,
  "rssi":     100,
  "battery":  100,
  "status":   2
}
```

* The actual JSON will not include any whitespace.

11 Device settings

11.1 Individual settings

11.1.1 Calibration date (read-only)

Description: Calibration date. See Section 4 on page 12 for more information.

JSON key: `"calibrationDate"`

JSON value type: string

Default value: “Unknown”

11.1.2 Gyroscope misalignment (read-only)

Description: Gyroscope misalignment matrix (in row-major order) used for inertial sensor calibration. See Section 4.1 on page 12 for more information.

JSON key: `"gyroscopeMisalignment"`

JSON value type: array of 9 numbers

Default value: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]

11.1.3 Gyroscope sensitivity (read-only)

Description: Gyroscope sensitivity vector used for inertial sensor calibration. See Section 4.1 on page 12 for more information.

JSON key: `"gyroscopeSensitivity"`

JSON value type: array of 3 numbers

Default value: [1.0, 1.0, 1.0]

11.1.4 Gyroscope offset (read-only)

Description: Gyroscope offset vector used for inertial sensor calibration. See Section 4.1 on page 12 for more information.

JSON key: `"gyroscopeOffset"`

JSON value type: array of 3 numbers

Default value: [0.0, 0.0, 0.0]

11.1.5 Accelerometer misalignment (read-only)

Description: Accelerometer misalignment matrix (in row-major order) used for inertial sensor calibration. See Section 4.1 on page 12 for more information.

JSON key: `"accelerometerMisalignment"`

JSON value type: array of 9 numbers

Default value: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]

11.1.6 Accelerometer sensitivity (read-only)

Description: Accelerometer sensitivity vector used for inertial sensor calibration. See Section 4.1 on page 12 for more information.

JSON key: `"accelerometerSensitivity"`

JSON value type: array of 3 numbers

Default value: [1.0, 1.0, 1.0]

11.1.7 Accelerometer offset (read-only)

Description: Accelerometer offset vector used for inertial sensor calibration. See Section 4.1 on page 12 for more information.

JSON key: `"accelerometerOffset"`

JSON value type: array of 3 numbers

Default value: [0.0, 0.0, 0.0]

11.1.8 Soft iron matrix (read-only)

Description: Soft iron matrix (in row-major order) used for magnetometer calibration. See Section 4.2 on page 13 for more information.

JSON key: `"softIronMatrix"`

JSON value type: array of 9 numbers

Default value: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]

11.1.9 Hard iron offset (read-only)

Description: Hard iron offset vector used for magnetometer calibration. See Section 4.2 on page 13 for more information.

JSON key: `"hardIronOffset"`

JSON value type: array of 3 numbers

Default value: [0.0, 0.0, 0.0]

11.1.10 High-g accelerometer misalignment (read-only)

Description: High-g accelerometer misalignment matrix (in row-major order) used for inertial sensor calibration. See Section 4.1 on page 12 for more information.

JSON key: `"highGAccelerometerMisalignment"`

JSON value type: array of 9 numbers

Default value: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]

11.1.11 High-g accelerometer sensitivity (read-only)

Description: High-g accelerometer sensitivity vector used for inertial sensor calibration. See Section 4.1 on page 12 for more information.

JSON key: `"highGAccelerometerSensitivity"`

JSON value type: array of 3 numbers

Default value: [1.0, 1.0, 1.0]

11.1.12 High-g accelerometer offset (read-only)

Description: High-g accelerometer offset vector used for inertial sensor calibration. See Section 4.1 on page 12 for more information.

JSON key: `"highGAccelerometerOffset"`

JSON value type: array of 3 numbers

Default value: [0.0, 0.0, 0.0]

11.1.13 Device name

Description: User-defined device name up to 31 characters long.

JSON key: `"deviceName"`

JSON value type: string

Default value: “x-IMU3”

11.1.14 Serial number (read-only)

Description: Unique 64-bit serial number expressed as a string of 16 hexadecimal digits in the format “XXXX-XXXX-XXXX-XXXX”.

JSON key: `"serialNumber"`

JSON value type: string

Default value: “Unknown”

11.1.15 Firmware version (read-only)

Description: Firmware version.

JSON key: `"firmwareVersion"`

JSON value type: string

Default value: “Unknown”

11.1.16 Bootloader version (read-only)

Description: Bootloader version.

JSON key: `"bootloaderVersion"`

JSON value type: string

Default value: “Unknown”

11.1.17 Hardware version (read-only)

Description: Hardware version.

JSON key: `"hardwareVersion"`

JSON value type: string

Default value: “Unknown”

11.1.18 Serial mode

Description: Serial mode.

JSON key: `"serialMode"`

JSON value type: number

Default value: 0

11.1.19 Serial baud rate

Description: Serial baud rate.

JSON key: `"serialBaudRate"`

JSON value type: number

Default value: 115200

11.1.20 Serial RTS/CTS enabled

Description: Serial Request To Send (RTS)/Clear To Send (CTS) enabled.

JSON key: `"serialRtsCtsEnabled"`

JSON value type: true or false

Default value: false

11.1.21 Serial accessory number of bytes

Description: Serial accessory number of bytes.

JSON key: `"serialAccessoryNumberOfBytes"`

JSON value type: number

Default value: 1024

11.1.22 Serial accessory termination byte

Description: Serial accessory termination byte.

JSON key: `"serialAccessoryTerminationByte"`

JSON value type: number

Default value: 10

11.1.23 Serial accessory timeout

Description: Serial accessory timeout.

JSON key: `"serialAccessoryTimeout"`

JSON value type: number

Default value: 100

11.1.24 Wireless mode

Description: Configures the wireless mode. The possible values are listed in Table 30. The current wireless mode is indicated by the LED colour. See Section 6 on page 13 for more information.

Value	Mode
0	Disabled
1	Wi-Fi client
2	Wi-Fi AP
3	Bluetooth

Table 30: Wireless modes

JSON key: `"wirelessMode"`

JSON value type: number

Default value: 2

11.1.25 Wireless firmware version (read-only)

Description: Current wireless firmware version.

JSON key: `"wirelessFirmwareVersion"`

JSON value type: string

Default value: “Unknown”

11.1.26 External antennae enabled

Description: Enables (true) or disables (false) the external antennae. An antennae must be connected to the U.FL connector if the external antennae is enabled. The internal antennae will be used if the external antennae is disabled.

JSON key: `"externalAntennaeEnabled"`

JSON value type: true or false

Default value: false

11.1.27 Wi-Fi region

Description: Configures the region for Wi-Fi operation. The device will operate according to the regulations of the selected region. See Section 11.1.30 for a list of channels available in each region. The possible values are listed in Table 31.

Value	Region
1	United States (US)
2	Europe (EU)
3	Japan (JP)

Table 31: Wi-Fi regions

JSON key: `"wiFiRegion"`

JSON value type: number

Default value: 2

11.1.28 Wi-Fi MAC address (read-only)

Description: Wi-Fi Media Access Control (MAC) address.

JSON key: `"wiFiMacAddress"`

JSON value type: string

Default value: 0

11.1.29 Wi-Fi IP address (read-only)

Description: Current IP address of the device. A value of 0.0.0.0 indicates that the device does not yet have an IP address.

JSON key: `"wiFiIPAddress"`

JSON value type: string

Default value: 0

11.1.30 Wi-Fi client SSID

Description: Configures the Service Set Identifier (SSID) in Wi-Fi client mode. This is the name of the router that the device will connect to. The SSID may be up to 31 characters long.

JSON key: `"wiFiClientSsid"`

JSON value type: string

Default value: “x-IMU3 Network”

11.1.31 Wi-Fi client key

Description: Configures the security key in Wi-Fi client mode. This is the password for the router that the device will connect to. If the router does not require a password then this setting will be ignored. The key may be up to 31 characters long.

JSON key: `"wiFiClientKey"`

JSON value type: string

Default value: “xiotechnologies”

11.1.32 Wi-Fi client channel

Description: Configures the channel in Wi-Fi client mode. This is the channel of the router that the device will connect to. Setting the correct channel will decrease the time taken to connect. If the channel is incorrect then the device will search all channels and then update this setting for the correct channel. The possible channels are listed in Table 32.

Channel	Band	Notes
0	-	All channels
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	2.4 GHz	-
12, 13	2.4 GHz	Invalid for US
14	2.4 GHz	Invalid for US and EU
36, 40, 44, 48	5 GHz	-
52, 56, 60, 64, 100, 104, 108, 112, 116	5 GHz, DFS	-
120, 124, 128	5 GHz, DFS	Invalid for US
132, 136, 140	5 GHz, DFS	-
149, 153, 157, 161, 165	5 GHz	Invalid for EU and JP

Table 32: Wi-Fi client channels

JSON key: `"wiFiClientChannel"`

JSON value type: number

Default value: 0

11.1.33 Wi-Fi client DHCP enabled

Description: Enables (true) or disables (false) Dynamic Host Configuration Protocol (DHCP) in Wi-Fi client mode. If DHCP is enabled then the device will be assigned an IP address by the router. This is an automatic process and the recommend mode of operation. If DHCP is disabled then the IP address, netmask, and gateway must be configured by the user.

JSON key: `"wiFiClientDhcpEnabled"`

JSON value type: true or false

Default value: true

11.1.34 Wi-Fi client IP address

Description: Configures the device IP address in Wi-Fi client mode when DHCP is disabled. This setting will be ignored if DHCP is enabled.

JSON key: `"wiFiClientIPAddress"`

JSON value type: string

Default value: “192.168.1.2”

11.1.35 Wi-Fi client netmask

Description: Configures the netmask in Wi-Fi client mode when DHCP is disabled. This setting will be ignored if DHCP is enabled.

JSON key: `"wiFiClientNetmask"`

JSON value type: string

Default value: “255.255.255.0”

11.1.36 Wi-Fi client gateway

Description: Configures the gateway in Wi-Fi client mode when DHCP is disabled. This setting will be ignored if DHCP is enabled.

JSON key: `"wiFiClientGateway"`

JSON value type: string

Default value: “192.168.1.1”

11.1.37 Wi-Fi AP SSID

Description: Configures the SSID of the device in Wi-Fi AP mode. This is the name of the network created by the device. The SSID may be up to 31 characters long. If the SSID is left blank then the device will use an SSID made up of the product name and the serial number in the format “x-IMU3_XXXX-XXXX-XXXX-XXXX”.

JSON key: `"wiFiAPSsid"`

JSON value type: string

Default value: “”

11.1.38 Wi-Fi AP key

Description: Configures the security key in Wi-Fi AP mode. This is the password required to connect to the network created by the device. The key may be up to 31 characters long. If the key is left blank then the network will not use security and a password would not be required to connect to the device.

JSON key: `"wiFiAPKey"`

JSON value type: string

Default value: “”

11.1.39 Wi-Fi AP channel

Description: Configures the channel of the device in Wi-Fi AP mode. This is the channel of the network created by the device. The possible channels are listed in Table 33.

Channel	Band	Notes
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	2.4 GHz	-
12, 13	2.4 GHz	Invalid for US
14	2.4 GHz	Invalid for US and EU
36, 40, 44, 48	5 GHz	-
149, 153, 157, 161, 165	5 GHz	Invalid for EU and JP

Table 33: Wi-Fi AP channels

JSON key: `"wiFiAPChannel"`

JSON value type: number

Default value: 36

11.1.40 TCP port

Description: TCP port.

JSON key: `"tcpPort"`

JSON value type: number

Default value: 7000

11.1.41 UDP IP address

Description: UDP IP address.

JSON key: `"udpIPAddress"`

JSON value type: string

Default value: 0

11.1.42 UDP send port

Description: UDP send port.

JSON key: `"udpSendPort"`

JSON value type: number

Default value: 0

11.1.43 UDP receive port

Description: UDP receive port.

JSON key: `"udpReceivePort"`

JSON value type: number

Default value: 9000

11.1.44 Synchronisation enabled

Description: Enables (true) or disables (false) synchronisation. See Section 9.5 on page 32 for more information.

JSON key: `"synchronisationEnabled"`

JSON value type: true or false

Default value: true

11.1.45 Synchronisation network latency

Description: The network latency (in microseconds) used by the synchronisation algorithm.

JSON key: `"synchronisationNetworkLatency"`

JSON value type: number

Default value: 1500

11.1.46 Bluetooth address (read-only)

Description: Device Bluetooth address.

JSON key: `"bluetoothAddress"`

JSON value type: number

Default value: 0

11.1.47 Bluetooth name

Description: Configures the name of the device in Bluetooth mode. The name may be up to 31 characters long. If the name is left blank then the device will use a name made up of the product name and the serial number in the format “x-IMU3_XXXX-XXXX-XXXX-XXXX”.

JSON key: `"bluetoothName"`

JSON value type: string

Default value: “”

11.1.48 Bluetooth pin code

Description: Configures the Bluetooth pin code. The pin code must be between 1 and 15 characters long.

JSON key: `"bluetoothPinCode"`

JSON value type: string

Default value: “1234”

11.1.49 Bluetooth discovery mode

Description: Configures the Bluetooth discovery mode. The possible values are listed in Table 34.

Value	Mode
0	Disabled
1	Enabled
2	Limited

Table 34: Bluetooth discovery modes

JSON key: `"bluetoothDiscoveryMode"`

JSON value type: number

Default value: 2

11.1.50 Bluetooth paired address (read-only)

Description: Bluetooth address of the paired device.

JSON key: `"bluetoothPairedAddress"`

JSON value type: number

Default value: 0

11.1.51 Bluetooth paired link key (read-only)

Description: Bluetooth link key of the paired device.

JSON key: `"bluetoothPairedLinkKey"`

JSON value type: number

Default value: 0

11.1.52 Data logger enabled

Description: Data logger enabled.

JSON key: `"dataLoggerEnabled"`

JSON value type: true or false

Default value: false

11.1.53 Data logger file name prefix

Description: Data logger file name prefix.

JSON key: `"dataLoggerFileNamePrefix"`

JSON value type: string

Default value: “”

11.1.54 Data logger file name time enabled

Description: Data logger file name time enabled.

JSON key: `"dataLoggerFileNameTimeEnabled"`

JSON value type: true or false

Default value: true

11.1.55 Data logger file name counter enabled

Description: Data logger file name counter enabled.

JSON key: `"dataLoggerFileNameCounterEnabled"`

JSON value type: true or false

Default value: false

11.1.56 Data logger max file size

Description: Data logger max file size.

JSON key: `"dataLoggerMaxFileSize"`

JSON value type: number

Default value: 0

11.1.57 Data logger max file period

Description: Data logger max file period.

JSON key: `"dataLoggerMaxFilePeriod"`

JSON value type: number

Default value: 0

11.1.58 Axes alignment

Description: Axes alignment describing the sensor axes relative to the body axes. For example, if the body X axis is aligned with the sensor Y axis and the body Y axis is aligned with sensor X axis but pointing the opposite direction then alignment is +Y-X+Z. The possible values are listed in Table 35.

Value	Axes alignment
0	+X+Y+Z
1	+X-Z+Y
2	+X-Y-Z
3	+X+Z-Y
4	-X+Y-Z
5	-X+Z+Y
6	-X-Y+Z
7	-X-Z-Y
8	+Y-X+Z
9	+Y-Z-X
10	+Y+X-Z
11	+Y+Z+X
12	-Y+X+Z
13	-Y-Z+X
14	-Y-X-Z
15	-Y+Z-X
16	+Z+Y-X
17	+Z+X+Y
18	+Z-Y+X
19	+Z-X-Y
20	-Z+Y+X
21	-Z-X+Y
22	-Z-Y-X
23	-Z+X-Y

Table 35: Axes alignments

JSON key: `"axesAlignment"`

JSON value type: number

Default value: 0

11.1.59 Gyroscope offset correction enabled

Description: Gyroscope offset correction enabled.

JSON key: `"gyroscopeOffsetCorrectionEnabled"`

JSON value type: true or false

Default value: true

11.1.60 AHRS axes convention

Description: Configures the Earth axes convention used by the AHRS algorithm. The possible values are listed in Table 36.

Value	Axes Convention
0	North-West-Up (NWU)
1	East-North-Up (ENU)
2	North-East-Down (NED)

Table 36: Axes conventions

JSON key: `"ahrsAxesConvention"`

JSON value type: number

Default value: 0

11.1.61 AHRS gain

Description: AHRS gain.

JSON key: `"ahrsGain"`

JSON value type: number

Default value: 0.5

11.1.62 AHRS ignore magnetometer

Description: AHRS ignore magnetometer.

JSON key: `"ahrsIgnoreMagnetometer"`

JSON value type: true or false

Default value: false

11.1.63 AHRS acceleration rejection enabled

Description: AHRS acceleration rejection enabled.

JSON key: `"ahrsAccelerationRejectionEnabled"`

JSON value type: true or false

Default value: true

11.1.64 AHRS magnetic rejection enabled

Description:	AHRS magnetic rejection enabled.
JSON key:	`"ahrsMagneticRejectionEnabled"`
JSON value type:	true or false
Default value:	true

11.1.65 Binary mode enabled

Description:	Enables (true) or disables (false) binary data messages. If binary data messages are disabled then data messages will be ASCII. See Section 8.2 on page 22 for more information.
JSON key:	`"binaryModeEnabled"`
JSON value type:	true or false
Default value:	true

11.1.66 USB data messages enabled

Description:	Enables (true) or disables (false) the sending of data messages for the USB communication interface. The sending of notification and error messages cannot be disabled.
JSON key:	`"usbDataMessagesEnabled"`
JSON value type:	true or false
Default value:	true

11.1.67 Serial data messages enabled

Description:	Enables (true) or disables (false) the sending of data messages for the serial communication interface when it is in normal mode. The sending of notification and error messages cannot be disabled.
JSON key:	`"serialDataMessagesEnabled"`
JSON value type:	true or false
Default value:	true

11.1.68 TCP data messages enabled

Description:	Enables (true) or disables (false) the sending of data messages for the TCP communication interface. The sending of notification and error messages cannot be disabled.
JSON key:	`"tcpDataMessagesEnabled"`
JSON value type:	true or false
Default value:	true

11.1.69 UDP data messages enabled

Description: Enables (true) or disables (false) the sending of data messages for the UDP communication interface. The sending of notification and error messages cannot be disabled.

JSON key: `"udpDataMessagesEnabled"`

JSON value type: true or false

Default value: true

11.1.70 Bluetooth data messages enabled

Description: Bluetooth data messages enabled.

JSON key: `"bluetoothDataMessagesEnabled"`

JSON value type: true or false

Default value: true

11.1.71 Data logger data messages enabled

Description: Enables (true) or disables (false) the sending of data messages for the on-board data logging. The sending of notification and error messages cannot be disabled.

JSON key: `"dataLoggerDataMessagesEnabled"`

JSON value type: true or false

Default value: true

11.1.72 AHRS message type

Description: Configures the AHRS message type. The possible values are listed in Table 37.

Value	Message type
0	Quaternion
1	Rotation matrix
2	Euler angles
3	Linear acceleration
4	Earth acceleration

Table 37: AHRS message types

JSON key: `"ahrsMessageType"`

JSON value type: number

Default value: 0

11.1.73 Inertial message rate divisor

Description: Configures the inertial message rate as the fixed sample of 400 Hz divided by the the message rate divisor. A message rate divisor of zero will disable the messages. Example message rates are listed in Table 38. See Section 9 on page 31 for more information about messages rates.

Divisor	Message rate
0	Disabled
1	400 messages/s
2	200 messages/s
3	133.33 messages/s
4	100 messages/s
...	...

Table 38: Example inertial message rates

JSON key: `"inertialMessageRateDivisor"`

JSON value type: number

Default value: 8

11.1.74 Magnetometer message rate divisor

Description: Configures the magnetometer message rate as the fixed sample of 20 Hz divided by the the message rate divisor. A message rate divisor of zero will disable the messages. Example message rates are listed in Table 39. See Section 9 on page 31 for more information about messages rates.

Divisor	Message rate
0	Disabled
1	20 messages/s
2	10 messages/s
3	6.67 messages/s
4	5 messages/s
...	...

Table 39: Example magnetometer message rates

JSON key: `"magnetometerMessageRateDivisor"`

JSON value type: number

Default value: 1

11.1.75 AHRS message rate divisor

Description: Configures the AHRS message rate as the fixed sample of 400 Hz divided by the the message rate divisor. A message rate divisor of zero will disable the messages. Example message rates are listed in Table 40. See Section 9 on page 31 for more information about messages rates.

Divisor	Message rate
0	Disabled
1	400 messages/s
2	200 messages/s
3	133.33 messages/s
4	100 messages/s
...	...

Table 40: Example AHRS message rates

JSON key: `"ahrsMessageRateDivisor"`

JSON value type: number

Default value: 8

11.1.76 High-g accelerometer message rate divisor

Description: Configures the high-g accelerometer message rate as the fixed sample of 1600 Hz divided by the the message rate divisor. A message rate divisor of zero will disable the messages. Example message rates are listed in Table 41. See Section 9 on page 31 for more information about messages rates.

Divisor	Message rate
0	Disabled
1	1600 messages/s
2	800 messages/s
3	533.33 messages/s
4	400 messages/s
...	...

Table 41: Example high-g accelerometer message rates

JSON key: `"highGAccelerometerMessageRateDivisor"`

JSON value type: number

Default value: 32

11.1.77 Temperature message rate divisor

Description: Configures the temperature message rate as the fixed sample of 5 Hz divided by the the message rate divisor. A message rate divisor of zero will disable the messages. Example message rates are listed in Table 42. See Section 9 on page 31 for more information about messages rates.

Divisor	Message rate
0	Disabled
1	5 messages/s
2	2.5 messages/s
3	1.67 messages/s
4	1.25 messages/s
...	...

Table 42: Example temperature message rates

JSON key: `"temperatureMessageRateDivisor"`

JSON value type: number

Default value: 5

11.1.78 Battery message rate divisor

Description: Configures the battery message rate as the fixed sample of 5 Hz divided by the the message rate divisor. A message rate divisor of zero will disable the messages. Example message rates are listed in Table 43. See Section 9 on page 31 for more information about messages rates.

Divisor	Message rate
0	Disabled
1	5 messages/s
2	2.5 messages/s
3	1.67 messages/s
4	1.25 messages/s
...	...

Table 43: Example battery message rates

JSON key: `"batteryMessageRateDivisor"`

JSON value type: number

Default value: 5

11.1.79 RSSI message rate divisor

Description: Configures the RSSI message rate as the fixed sample of 1 Hz divided by the the message rate divisor. A message rate divisor of zero will disable the messages. Example message rates are listed in Table 44. See Section 9 on page 31 for more information about messages rates.

Divisor	Message rate
0	Disabled
1	1 messages/s
2	0.5 messages/s
3	0.33 messages/s
4	0.25 messages/s
...	...

Table 44: Example RSSI message rates

JSON key: `"rssiMessageRateDivisor"`

JSON value type: number

Default value: 1

Glossary

a.u. arbitrary units . . . 10

ADC Analog-to-Digital Converter . . . 31

AHRS Attitude Heading Reference System . . . 20

AP Access Point . . . 14

API Application Programming Interface

ASCII American Standard Code for Information Interchange . . . 22

CDC Communications Device Class

CR Carriage Return . . . 17

CRC Cyclic Redundancy Check . . . 21

CSV Comma-Separated Values . . . 16

CTS Clear To Send . . . 36

DFS Dynamic Frequency Selection

DHCP Dynamic Host Configuration Protocol . . . 39

EU Europe . . . 38

EEPROM Electrically Erasable Programmable Read-Only Memory . . . 18

GPS Global Positioning System

GUI Graphical User Interface

HTTP Hypertext Transfer Protocol . . . 17

IMU Inertial Measurement Unit . . . 6

IP Internet Protocol . . . 14

IP67 Ingress Protection 67 . . . 8

JP Japan 38

JSON JavaScript Object Notation 18

LED Light-Emitting Diode 13

LF Line Feed 17

MAC Media Access Control 38

PDF Portable Document Format 13

QR Quick Response 13

RGB Red Green Blue 20

RMS Root Mean Square

RSSI Received Signal Strength Indicator 29

RTC Real-Time Clock 19

RTS Request To Send 36

microSD micro Secure Digital 13

SLIP Serial Line Internet Protocol 22

SSID Service Set Identifier 38

TCP Transmission Control Protocol 17

UDP User Datagram Protocol 17

UART Universal Asynchronous Receiver-Transmitter

US United States 38

USB Universal Serial Bus 17

Document version history

Version	Date	Changes
v0.0	April 7, 2020	• Advanced release
v0.1	April 14, 2020	• Add overview and hardware sections • Remove hyphenations • Add missing acronyms to glossary • Move glossary, disclaimer, and document version history to end of document
v0.2	April 15, 2020	• Add missing hexadecimal notation • Add linear acceleration and Earth acceleration data messages • Remove pressure data message
v0.3	April 29, 2020	• Add self-test commands • Add technical specification section • Add message rates section • Add axes alignment and data message setting descriptions
v0.4	May 20, 2020	• Add Wi-Fi setting descriptions • Add Bluetooth as product feature
v0.5	Jul 7, 2020	• Add instructions for updating device firmware
v0.6	Sep 7, 2020	• Update board image • Update technical specification • Add factory and erase commands • Add Sample rates, message rates, and timestamps section
v0.7	Feb 9, 2021	• Add charger status enumeration • Add power button section • Add annotated housing image
v0.8	Apr 15, 2021	• Add format command • Remove result command • Add colour command • Add IP67 section • Update Euler angles message

v0.9	Aug 01, 2021	• Add C# to overview • Update board image
v0.10	Jan 06, 2022	• Update device settings • Add temperature technical specification • Remove serial accessory rate divisor setting • Add calibration section
v0.11	Apr 06, 2022	• Use consistent numerical formatting in examples • Add LED section • Change serial accessory command key • Update battery life in overview • Add data logger capacity section • Use consistent terminology for high-g accelerometer • Update data logger device setting keys • Remove Bluetooth section • Add data logger section • Update network announcement message • Add AHRS device settings • Update images
v1.0	Sep 26, 2022	• Express sensitivity as a diagonal matrix • Update default clock calibration settings • Update synchronisation settings • Update factory command • Update static accuracy • Remove IP67 footnote
v1.1	Apr 06, 2023	• Update network announcement message • Add heading command • Add axes convention setting • Fix stylisation of microSD • Clarify that CR in is optional in commands messages • Remove system clock and RTC and battery voltmeter calibration • Fix high-g accelerometer sample rate error • Remove sentence stating that response is always in camel case • Remove serial accessory transmit enabled setting • Remove update device settings section

Disclaimer

The information in this document pertains to information related to x-io Technologies products. This information is provided as a service to our customers, and may be used for information purposes only.

x-io Technologies assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at x-io Technologies' sole discretion without any prior notice to anyone. x-io Technologies is not committed to updating this document in the future.