QSENSEMOTION

Sensor Interfaces

Document Title:	QSense Motion - Sensor Interfaces
Company:	2M Engineering Ltd – John F Kennedylaan 3, 5555XC, Valkenswaard, The Netherlands
Project:	QSense Motion
Author:	Jose Lopez Hidalgo

Contents

1 Introduction

The QSense IMU Motion Sensing Platform is a wearable system intended to monitor the 3D orientation of human body segments from linear acceleration, angular velocity, and magnetometer measurements. It provides easy data collection for a wide range of applications including research, sports and exercise science, ergonomics and (virtual reality) games.

This guide is designed to assist you in integrating the QSense sensoe=r into your applications and services, unlocking its core functionalities and enabling efficient data management.

Whether you are building mobile apps, IoT solutions, or any other software that requires seamless connectivity and data exchange, this guide will provide you with the necessary tools and resources to access all the functionalities of the QSense sensor.

The QSense sensor can be accessed through two communication protocols: wireless BLE communication, and serial communication accessible through a micro-USB connector. The BLE communication provides access to the Core Interface that focuses on essential device configuration and data access. The serial communication is primarily used for direct PC connectivity and provides access to the Serial Interface, which extends the functionality of the Core Interface.

Throughout this document, you will find detailed explanations on how to:

- Discover and communicate with the QSense sensor over BLE and serial communication
- Configure communication settings to optimize performance
- Interact with the device's internal memory map
- Parse the data streamed by the QSense sensors.

Code examples are available in QSense-Examples.

NOTE: The interfaces described in this document are compatible with FW versions 2.0.0 and higher.

2 Communication protocols

To exchange data with the QSense sensor, you first need to establish a connection using one of its available communication protocols. Only after this connection is made can the interfaces be used to send commands or retrieve information from the device. Each protocol has its own specific characteristics, which are detailed in the following subsections.

2.1 BLE communication

The QSense sensor communicates over BLE using the Nordic UART Service (NUS) — a custom BLE service that emulates a serial port.

For more details, see Nordic Semiconductor's official NUS documentation.

QSense exposes a single GATT (Generic Attribute Profile) service. In BLE, the GATT protocol organizes data into services. Each service groups related characteristics. Characteristics represent readable, writable, or notifiable values — effectively, the interface for data exchange. You can read more about this in this link.

The QSense service is identified by the following 128-bit UUID: 6E400001-B5A3-F393-E0A9-E50E24DCCA9E

This UUID is required to discover and connect to the device's custom NUS implementation.

Within this service, QSense defines two key characteristics:

- RX Characteristic: used to write data to the device (e.g., Core Interface packets).
- TX Characteristic: used to receive data from the device via notifications.

To receive messages from QSense, notifications must be explicitly enabled on the TX characteristic.

Characteristic	UUID	Properties
RX Characteristic	6E400002-B5A3-F393-E0A9-E50E24DCCA9E	Write, Write Without Response
TX Characteristic	6E400003-B5A3-F393-E0A9-E50E24DCCA9E	Notify

Table 1. BLE characteristics.

When scanning for BLE devices, your application will receive two types of packets: the Advertisement packet and the Scan Response packet. These are broadcast by nearby devices and contain metadata that helps identify them before a connection is made.

The QSense sensor sends a custom advertisement that includes:

- The service UUID, used to identify its support for the Nordic UART Service (NUS).
- The device name, "QSense".

These two fields are sufficient for your application to reliably detect and filter QSense sensors in the scan results.

The Scan Response packet includes additional identifying information: the device's serial number, embedded in the Manufacturer Specific Data field. This serial number matches the QR code printed on the back of the sensor, allowing your application to implement device whitelisting and ensure only authorized QSense units are recognized and connected.

Byte Address	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0x4D	0x32	0x46	0x46	0x35	0x45	0x43	0x44	0x38	0x39	0x34	0x45	0x44	0x41	0x2D	0x45
16	0x37	0x36	0x33													

Table 2. Manufacturer Specific Data example. The first two bytes are constant and correspond to the company identifier.

The QSense sensor utilizes the 2 Mbps PHY introduced in BLE 5.0. To take full advantage of this increased throughput and ensure compatibility, we recommend using BLE 5.0 or higher on the central device.

To optimize performance, several BLE parameters should be considered:

- Connection Interval: Defines how often the peripheral and central can exchange data. It can range from 7.5 milliseconds to 4 seconds, in 1.25 milliseconds increments. Actual limits vary by operating system and BLE stack.
- ATT MTU (Attribute Maximum Transmission Unit): Increases the maximum size of each GATT packet, reducing the overhead for large messages.
- Data Length Extension (DLE): Expands the payload size beyond the default 27-byte BLE packet limit.

The connection interval directly affects how much time each QSense sensor has to communicate with the central.

Maximizing both ATT MTU and DLE helps ensure that most Core Interface requests fit within a single BLE transaction, improving throughput and reducing latency.

NOTE: It is important to carefully configure the connection interval when setting up the QSense sensor. Some configurations may result in data loss under high throughput. See section 6 for detailed recommendations.

2.2 Serial communication

To use the QSense sensor over a serial (USB) connection, start by connecting the device to one of your computer's USB ports using the provided cable.

Once connected, the device will appear in your system's list of serial ports. You can identify it by inspecting the USB device metadata — specifically the bus-reported device description and parent properties.

The relevant identifiers are:

- Product Name: "QSense"
- VID (Vendor ID): 16D0
- PID (Product ID): 06A1
- MAC Address: A unique hardware identifier that corresponds to the first part of the QR code printed on the back of the sensor

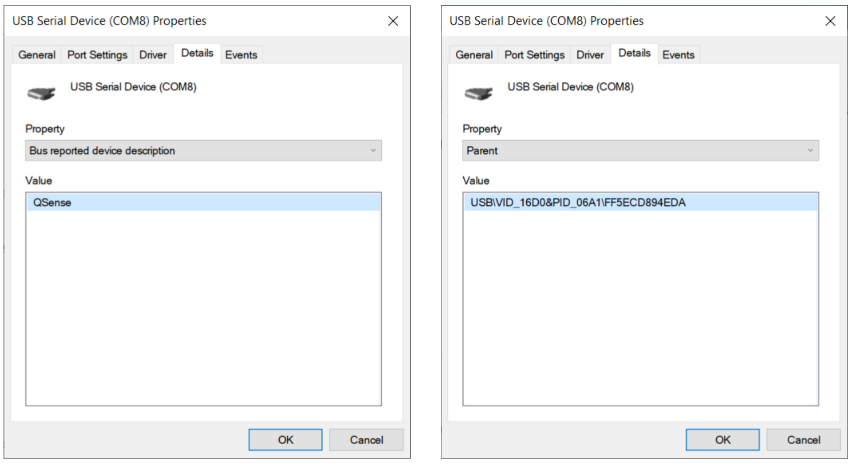
These properties allow your application to reliably detect and distinguish QSense sensors from other connected USB peripherals.

For a reliable connection, ensure you use a USB cable that supports data transfer—not just charging. Additionally, configure the serial port with the following settings for optimal communication:

- Baud rate: 460800
- Parity: None
- Stop bits: One

When opening the serial port, make sure to enable Data Terminal Ready (DTR) to properly initialize communication.

For more details on serial communication with QSense, please refer to the external documentation linked here.



USB Serial Device (COM8) Properties

USB Serial Device (COM8)

Property: Bus reported device description

Value: QSense

USB Serial Device (COM8) Properties

USB Serial Device (COM8)

Property: Parent

Value: USB\VID_16D0&PID_06A1\FF5ECD894EDA

Figure 1. Serial properties of the QSense sensor.

3 Core Interface

The Core Interface provides direct access to the device's memory through four distinct packet types, designed to simplify reading and writing memory blocks.

Each packet includes the following fields:

- Opcode: A single byte that specifies the packet type.
- Address: A 32-bit unsigned integer indicating the starting memory address for the read or write operation.
- Length: A 16-bit unsigned integer specifying how many bytes to read or write.
- Data: A byte array containing the payload, i.e., either the data to be read from the device or the data to be written to the device.

Byte Address	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**…**	**N**
0	Opcode	Address				Length		Data		

Table 3. Packet fields and byte order of the Core Interface packets.

Not all fields are present in every packet type. Some are optional and can be omitted depending on the operation. The table below outlines which fields are required for each packet type.

Packet type	**Fields**			
	Opcode	**Address**	**Length**	**Data**
Read	Used	Used	Used	Not Used
Data	Used	Used	Used	Used
Abort	Used	Not Used	Not Used	Not Used
Stream	Used	Used	Used	Not Used

Table 4. Format for each packet type.

NOTE: All multi-byte fields in the packets use little-endian format. Pay close attention to the examples below to see how values are encoded.

NOTE: Large Core Interface messages may be split across multiple smaller packets because of protocol size limits. Your application should concatenate these fragments and wait until the entire message is received before attempting to parse it.

NOTE: The QSense sensor supports only one operation at a time. You should always wait for the current data transaction to complete before initiating a new one. Avoid sending read or write packets while the device is actively streaming, as this can cause communication issues or data loss.

3.1 Read Packet (Opcode 1)

Read packets request a specific number of bytes from a designated memory address on the QSense sensor. Use these packets when you need to retrieve fixed blocks of data.

Byte Address	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0x01	0x10	0x00	0x00	0x00	0x04	0x00	

Table 5. Example of read packet to read 4 bytes from memory address 16.

3.2 Data Packet (Opcode 2)

Data packets are used to write data to the device, allowing you to override parts of its memory. Additionally, the device sends Data packets as responses to Read or Stream requests, delivering the requested data payload.

Byte Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0x02	0x10	0x00	0x00	0x00	0x04	0x00	0x01	0x00	0x00	0x00					

Table 6. Example of data packet to write 1 to a 32-bit register at address 16.

3.3 Abort Packet (Opcode 3)

Abort packets are single-byte commands that interrupt any ongoing data streaming. Use this packet to stop a continuous stream of data from the device immediately.

3.4 Stream Packet (Opcode 4)

Stream packets request continuous data streaming from the QSense sensor. Unlike Read packets, they cause the device to send new data continuously as it becomes available. This streaming is restricted to a specific region of the device's memory known as the stream memory.

Byte Address	0	1	2	3	4	5	6	7
0	0x04	0x00	0x01	0x00	0x00	0xED	0x00	

Table 7. Stream packet.

4 Serial Interface

The Serial Interface provides access to the Core Interface over the serial interface and wraps the Core Interface packet in an ASCII encoded message. Besides the Core Interface packet, additional packets can be transmitted and received. The Serial Interface messages follow the following structure:

- Prefix: A Dollar character '$' (hex value 0x24) that marks the start of the packet.
- Opcode: A single character indicating the Serial Interface packet type.
- Data: Contains the encapsulated Core Interface packet as a hex encoded byte array. Every byte consists of two characters between '0..9' or 'A…F', representing hexadecimal notation. The first 2 characters of the Data part in the message represent the 1st byte of the hex encoded byte array. The 3rd and 4th character represent the 2nd byte of the hex encoded byte array, etcetera.
- Postfix: A Line Feed character '\n' (hex value 0x0A) marks the end of the packet.

Byte Address	0	1	2	…	N-1	N
Field	Prefix	Opcode	Data (optional)			Postfix
Character	$	T/R/U				\n

Table 8. Packet fields and byte order of the Serial Interface packets.

4.1 Transmit

This packet allows you to send a Core Interface packet to the device via the serial interface. It uses opcode 'T' (hex value 0x54).

If a response is necessary, the device will reply with a Receive packet.

Byte Address	0	1	2	…	N-1	N
Field	0x24	0x54	Core Interface packet			0x0A
Character	$	T				\n

Table 9. Transmit packet.

4.2 Receive

This packet type is sent by the device in response to a Transmit packet. Its opcode 'R' (hex value 0x52).

Byte Address	0	1	2	…	N-1	N
Field	0x24	0x52	Core Interface packet			0x0A
Character	$	R				\n

Table 10. Receive packet.

4.3 Upgrade

This packet is used to put the device into bootloader mode, which enables firmware updates. The opcode for this packet is 'U' (hex value 0x55). This packet type does not include a data field.

Byte Address	0	1	2	3
Field	0x24	0x55	0x0A	
Character	$	U	\n	

Table 11. Upgrade packet.

5 Memory Map

The memory of the QSense sensor is accessible and modifiable through the Core Interface, as detailed in the previous section. Broadly, the memory includes read-only fields that expose device-specific information, current device status, and sensor data. It also contains registers that can be written to in order to trigger actions or configure device behaviour.
Each memory field has a defined address and size, outlined in the table below.

Name	Address	Size	Type	Read/Write	Protected
Who Am I	0	4	Char[]	read only	No
Id	4	8	UInt64	read only	No
Address	12	8	UInt64	read only	No
Version	20	4	UInt8[]	read only	No
Battery	24	1	UInt8	read only	No
Motion Level	25	1	UInt8	read only	No
Offset Compensated	26	1	UInt8	read only	No
Magnetic Field Mapped	27	1	UInt8	read only	No
Magnetic Field Mapping Progress	28	1	UInt8	read only	No
Connection Interval	29	1	UInt8	read only	No
Sync. Status	30	1	UInt8	read only	No
100Hz Ticks	31	1	UInt8	read only	No
Pin	32	4	UInt8[]	read/write	No
Time	36	4	UInt32	read/write	Yes
Annotation	40	1	UInt8	read/write	Yes
Device State	42	2	UInt16	read/write	Yes
UI animation	44	4	UInt32	write	Yes
Device Name	48	12	Char[]	read/write	Yes
Data Mode	60	1	UInt8	read/write	Yes
Timesync	61	1	UInt8	read/write	Yes
Algorithm Selection	63	1	UInt8	read/write	Yes
Stream Memory	256	237	UInt8[]	read only	Yes

Table 12. Memory map.

5.1 Who Am I

This register identifies the device as a QSense sensor. The current version contains the value "QSM2".

5.2 Id

A unique 64-bit unsigned integer that serves as the device’s identifier.

5.3 Address

This register holds the device’s unique 64-bit MAC address.

NOTE: The QR code on the back of each QSense sensor combines the address and the first 4 digits of the Id (in hexadecimal), separated by a dash (-).

5.4 Version

Represents the firmware version of the device. Details on how to read this version are provided in the table below.

Byte Address	0	1	2	3
0	Unused	Major version	Minor version	Build version

Table 13. Structure of the version register.

5.5 Battery

Indicates the current battery level of the device as a percentage of the full battery.

5.6 Motion level

This register reflects the amount of movement detected by the device during the last period, based on the standard deviation of accelerometer data. It's useful for identifying devices before or after placement on the body, for example, by tapping or moving one device at a time.

5.7 Offset compensated

Shows whether offset compensation has ever been performed. This byte is set to 1 when the offset compensation completes. While the process is running or after a device reset, the value remains 0.

5.8 Magnetic field mapped

Indicates whether magnetic field mapping has ever been completed. This byte is set to 1 once the mapping finishes. It remains 0 during the process or after a reset.

5.9 Connection interval

Exposes the BLE connection interval set by the central device upon establishing the connection. The value is measured in units of 1.25 milliseconds. Monitoring this register is important, as it directly affects device configurations. See section 6 for further details.

5.10 Sync. Status

Shows the status of time synchronization. This register is relevant only when the timesync feature is enabled and the device is operating as a slave. It will be 1 if communication with the master is adequate, or 0 otherwise.

5.11 100Hz ticks

Used alongside the Time register, this value represents ticks since the start of the current second, in increments of 10 milliseconds.

5.12 Pin

This register controls access to certain protected registers. Protected registers cannot be read out nor written to unless you first set the pin. To enable access, you must write the following specific sequence of bytes to this register:

Byte Address	**0**	**1**	**2**	**3**
0	0x6C	0x6F	0x76	0x65

Table 14. Pin register value.

5.13 Time

This register holds the current Unix time in seconds. If the device has been reset or unused for an extended period, it may be necessary to manually set the correct time.

5.14 Annotation

This register is used to embed custom metadata into the stream. By default, its value is 0, but you can write any value between 0 and 255. Since the annotation is included in the streamed data, it allows you to add context-specific markers in real time.

For example, you could assign annotation values to represent different body limbs and configure each device accordingly. This can simplify post-processing and data analysis. Alternatively, annotations can be used to tag specific events relevant to your application. The use cases are open-ended and highly flexible.

5.15 Device state

This 16-bit register reflects the current configuration and operational state of the device. It also allows you to control specific functions such as offset compensation and magnetic field mapping. These actions are controlled via dedicated bits: setting a bit to 1 starts the corresponding process, while setting it to 0 stops it.

Refer to the table below for a detailed breakdown of the bit fields and to section 6 for a more in-depth explanation of these features.

Bit Field	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Byte 0	Unused	Gyroscope range			Accelerometer range		Enable magnetic field mapping	Enable offset compensation
Byte 1	Buffering				Sampling rate			

Table 15. Structure of the device state register.

5.16 UI animation

This write-only register allows you to control the behaviour of the device's top LED for a duration of 3 seconds. Two animation modes are available:

- 0: Blinking
- 1: Fixed (solid colour)

In addition to the animation type, you can specify the LED colour using an RGB value. This feature is useful for visually identifying devices during deployment or debugging.

Byte Address	0	1	2	3
0	Animation	Blue	Green	Red

Table 16. Structure of the UI animation register.

5.17 Device name

This register allows you to assign a custom name to the QSense sensor. Custom naming helps distinguish between multiple devices during deployment or analysis, especially when working with several devices simultaneously.

NOTE: This register is not intended to store sensitive information, since it does not encrypt the data.

5.18 Data mode

This register defines the format of the data output by the device. You can change the data mode at any time. This setting is persistent.

To set the desired mode, write the corresponding mode ID to this register. Each data mode determines the structure and content of the streamed data. The number of samples per packet (denoted as N) is controlled by the buffering field in the Device State register.

Refer to the table below for a breakdown of each data mode and the data it includes. For detailed instructions on parsing stream packets for each mode, see section 7.

Name	Id	Accelerometer	Gyroscope	Quaternion	Free Acc.
Mixed	0	N	N	1	1
Raw	1	N	N	0	N
Quaternion	2	0	0	N	0
Optimized	3	N	N	N	0
Quat+Mag	4	0	0	N	N

Table 17. Number of samples per signal for each data mode.

5.19 Timesync

This register controls the timesync feature, which enables synchronization between multiple QSense sensors.

The most significant bit (MSB) determines the device role:

- 1: Device operates as master
- 0: Device operates as slave

The remaining 7 bits define the **network key**, a shared identifier used to group devices for synchronization. Only devices with matching network keys will synchronize. This allows for multiple independent synchronization groups to coexist.

To disable **timesync,** write either 0x00 or 0x80.

Bit Field	7	6	5	4	3	2	1	0
Byte 0	Is Master	Network key						

Table 18. Structure of the Timesync register.

NOTE: Timesync is disabled by default when the device powers up or wakes from sleep. If required, it must be re-enabled manually after each wake-up

NOTE: Once timesync is enabled, the devices require a brief period to fully synchronize. It is recommended to wait at least 1–2 minutes before starting a recording session or relying on synchronized timestamps.

5.20 Algorithm selection

This register lets you choose the algorithm used by the device to compute quaternions. Two options are currently supported:

- 0: 9DoF Madgwick, which fuses data from the accelerometer, gyroscope, and magnetometer.
- 1: 6DoF Madgwick, which excludes the magnetometer.

The 6DoF option is especially useful in environments with significant magnetic interference, such as near metal structures or electronic equipment, where magnetometer data may be unreliable.

5.21 Stream memory

This section of memory holds the most recent sensor data packet generated by the device. When a stream packet is issued using the correct address and size corresponding to this region, the device enters streaming mode and begins sending new data packets automatically as they become available.

The format and parsing details of these data packets are described in section 7.

6 Device Configuration

As detailed in section 5.15 , the QSense sensor allows you to configure key parameters that control how data is measured and transmitted. When setting these parameters, pay close attention to the BLE connection interval and the number of QSense sensors you plan to connect simultaneously.

The sampling rate and buffer size directly influence how many data packets each device streams per second. To help ensure reliable transmission, the following formula provides a useful approximation (with the connection interval expressed in milliseconds). If this guideline is not followed, packet loss may occur.

$$n.devices \leq \frac{1000 \cdot buffering}{connection\ interval \cdot sampling\ rate}$$

NOTE: This formula only addresses limitations due to configuration, ideal conditions. It does not account for external factors such as interference from other wireless devices or obstructions between QSense sensors and the BLE central device.

6.1 Accelerometer range

This register sets the accelerometer's measurement range. The table below lists supported options along with the corresponding values and conversion factors from raw counts to g-units.

Range	Value	Conversion (g/counts)
±2 g	0	0.000061
±4 g	2	0.000122
±8 g	3	0.000244
±16 g	1	0.000488

Table 19. Accelerometer range settings and their conversion factors.

6.2 Gyroscope range

This register controls the gyroscope's measurement range. The table below provides the available range settings, corresponding register values, and the conversion factors to degrees per second (dps).

Range	Value	Conversion (dps/counts)
±125 dps	1	0.004375
±250 dps	0	0.008750
±500 dps	2	0.0175
±1000 dps	4	0.035
±2000 dps	6	0.07

Table 20. Gyroscope range settings and their conversion factors.

6.3 Sampling rate

This register sets how frequently the device samples data. Supported sampling frequencies and their corresponding values are listed below.

Frequency	Value
1 Hz	0
2 Hz	1
4 Hz	2
5 Hz	3
10 Hz	4
20 Hz	5
25 Hz	6
50 Hz	7
100 Hz	8
200 Hz	9
400 Hz	10
800 Hz	11

Table 21. Supported sampling rates.

6.4 Buffering

This setting defines how many samples are grouped into a single stream packet. It directly affects the transmission rate. Larger buffers mean fewer packets per second, which can reduce overhead but increase latency.

7 Stream data packet

The QSense sensor transmits data in fixed-size packets of 237 bytes. Each packet begins with a 10-byte header, followed by a data payload. The structure of this payload depends on the selected data mode.

Byte Address	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	Header															
16	Payload															
…																
224																

Table 22. General layout of a QSense stream data packet.

7.1 Header

The 10-byte header of each stream data packet encodes several key fields.

Bit Field	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Byte 0	Sample count				Data mode			
Byte 1	Seconds							
…								
Byte 4								
Byte 5	800 Hz ticks							
Byte 6								
Byte 7	Battery level					Magnetic Interference		
Byte 8	Annotation							
Byte 9	Unused		Accelerometer range		Gyroscope range		Sync status	

Table 24. Structure of the device state register.

- Data mode: Indicates the data mode used for the packet. This matches the value set in the Data Mode register in the memory map.
- Sample count: Number of samples contained in the packet. This corresponds to the Buffering register in the memory map.
- Seconds: Unix timestamp (in seconds) of the last sample in the buffer.
- 800Hz ticks: Time offset (in 1.25 milliseconds units) from the start of the current Unix timestamp. To get the precise timestamp of the last sample, add this offset to the value in the Seconds field.
- Magnetic Interference: Indicates the level of magnetic distortion detected in the environment. Relevant only when using the 9DoF Madgwick algorithm. When 6DoF is selected, this field is always 0. The device also detects if the magnetic environment differs significantly from the one used during field mapping. In that case, it is recommended to perform the magnetic field mapping again.

Value	**Meaning**
0	Not relevant.
1	No interference
2	Soft-iron interference
3	Hard-iron interference
4	Change of environment detected

Table 23. Magnetic interference levels.

- Battery level: Remaining battery reported in 6.25% increments.
- Annotation: Mirrors the Annotation register in the memory map.

- Sync status: This bit is set when a slave device receives a sufficient number of sync messages from the master. If synchronization is poor, time sync is disabled, or the device is acting as the master, the bit will be 0.
- Gyroscope range: Same value as the Gyroscope Range register.
- Accelerometer range: Same value as the Accelerometer Range register.

7.2 Payload

The QSense sensor supports multiple data payload formats, each optimized for different use cases and trade-offs between precision, bandwidth, and processing requirements. Each mode balances data granularity and bandwidth differently. Choose the one that best fits your application's performance and accuracy needs.

7.2.1 Mixed mode

This mode combines accelerometer, gyroscope, and magnetometer data at the configured sampling rate, along with a single sample of quaternion and free acceleration data per packet. The raw sensor values are transmitted as 16-bit signed integers and must be converted from ADC counts to physical units.

- Magnetometer data uses a conversion factor of 0.0015 gauss/count.
- Gyroscope and accelerometer conversions depend on the selected range. See tables 19 and 20 for the corresponding factors.

In contrast, quaternion and free acceleration data are transmitted as 32-bit floats and are already in physical units, so no conversion is needed.

Byte Address	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	Quaternion.W				Quaternion.X				Quaternion.Y				Quaternion.Z			
16	FreeAcc.X				FreeAcc.Y				FreeAcc.Z				Acc[0].X		Acc[0].Y	
32	Acc[0].Z		Acc[1].X		Acc[1].Y		Acc[1].Z		Acc[2].X		Acc[2].Y		Acc[2].Z		Acc[3].X	
48	Acc[3].Y		Acc[3].Z		Acc[4].X		Acc[4].Y		Acc[4].Z		Acc[5].X		Acc[5].Y		Acc[5].Z	
64	Acc[6].X		Acc[6].Y		Acc[6].Z		Acc[7].X		Acc[7].Y		Acc[7].Z		Acc[8].X		Acc[8].Y	
80	Acc[8].Z		Acc[9].X		Acc[9].Y		Acc[9].Z		Gyr[0].X		Gyr[0].Y		Gyr[0].Z		Gyr[1].X	
96	Gyr[1].Y		Gyr[1].Z		Gyr[2].X		Gyr[2].Y		Gyr[2].Z		Gyr[3].X		Gyr[3].Y		Gyr[3].Z	
112	Gyr[4].X		Gyr[4].Y		Gyr[4].Z		Gyr[5].X		Gyr[5].Y		Gyr[5].Z		Gyr[6].X		Gyr[6].Y	
128	Gyr[6].Z		Gyr[7].X		Gyr[7].Y		Gyr[7].Z		Gyr[8].X		Gyr[8].Y		Gyr[8].Z		Gyr[9].X	
144	Gyr[9].Y		Gyr[9].Z		Mag[0].X		Mag[0].Y		Mag[0].Z		Mag[1].X		Mag[1].Y		Mag[1].Z	
160	Mag[2].X		Mag[2].Y		Mag[2].Z		Mag[3].X		Mag[3].Y		Mag[3].Z		Mag[4].X		Mag[4].Y	
176	Mag[4].Z		Mag[5].X		Mag[5].Y		Mag[5].Z		Mag[6].X		Mag[6].Y		Mag[6].Z		Mag[7].X	
192	Mag[7].Y		Mag[7].Z		Mag[8].X		Mag[8].Y		Mag[8].Z		Mag[9].X		Mag[9].Y		Mag[9].Z	
208	Unused															
224																

Table 25. Payload of a Mixed mode data packet.

7.2.2 Raw mode

Raw Mode includes only accelerometer, gyroscope, and magnetometer data. All values are transmitted as 16-bit signed integers and must be converted from ADC counts to physical units.

- Magnetometer data uses a conversion factor of 0.0015 gauss/count.
- Gyroscope and accelerometer conversions depend on the selected range. See tables 19 and 20 for the corresponding factors.

Byte Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Acc[0].X		Acc[0].Y		Acc[0].Z		Acc[1].X		Acc[1].Y		Acc[1].Z		Acc[2].X		Acc[2].Y	
16	Acc[2].Z		Acc[3].X		Acc[3].Y		Acc[3].Z		Acc[4].X		Acc[4].Y		Acc[4].Z		Acc[5].X	
32	Acc[5].Y		Acc[5].Z		Acc[6].X		Acc[6].Y		Acc[6].Z		Acc[7].X		Acc[7].Y		Acc[7].Z	
48	Acc[8].X		Acc[8].Y		Acc[8].Z		Acc[9].X		Acc[9].Y		Acc[9].Z		Gyr[0].X		Gyr[0].Y	
64	Gyr[0].Z		Gyr[1].X		Gyr[1].Y		Gyr[1].Z		Gyr[2].X		Gyr[2].Y		Gyr[2].Z		Gyr[3].X	
80	Gyr[3].Y		Gyr[3].Z		Gyr[4].X		Gyr[4].Y		Gyr[4].Z		Gyr[5].X		Gyr[5].Y		Gyr[5].Z	
96	Gyr[6].X		Gyr[6].Y		Gyr[6].Z		Gyr[7].X		Gyr[7].Y		Gyr[7].Z		Gyr[8].X		Gyr[8].Y	
112	Gyr[8].Z		Gyr[9].X		Gyr[9].Y		Gyr[9].Z		Mag[0].X		Mag[0].Y		Mag[0].Z		Mag[1].X	
128	Mag[1].Y		Mag[1].Z		Mag[2].X		Mag[2].Y		Mag[2].Z		Mag[3].X		Mag[3].Y		Mag[3].Z	
144	Mag[4].X		Mag[4].Y		Mag[4].Z		Mag[5].X		Mag[5].Y		Mag[5].Z		Mag[6].X		Mag[6].Y	
160	Mag[6].Z		Mag[7].X		Mag[7].Y		Mag[7].Z		Mag[8].X		Mag[8].Y		Mag[8].Z		Mag[9].X	
176	Mag[9].Y		Mag[9].Z													
192	Unused															
208																
224																

Table 26. Payload of a Raw mode data packet.

7.2.3 Quaternion mode

In this mode, each data packet can contain up to 10 quaternions, transmitted as 32-bit floating-point values. No conversion is necessary.

Byte Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Quaternion[0].W				Quaternion[0].X				Quaternion[0].Y				Quaternion[0].Z			
16	Quaternion[1].W				Quaternion[1].X				Quaternion[1].Y				Quaternion[1].Z			
32	Quaternion[2].W				Quaternion[2].X				Quaternion[2].Y				Quaternion[2].Z			
48	Quaternion[3].W				Quaternion[3].X				Quaternion[3].Y				Quaternion[3].Z			
64	Quaternion[4].W				Quaternion[4].X				Quaternion[4].Y				Quaternion[4].Z			
80	Quaternion[5].W				Quaternion[5].X				Quaternion[5].Y				Quaternion[5].Z			
96	Quaternion[6].W				Quaternion[6].X				Quaternion[6].Y				Quaternion[6].Z			
112	Quaternion[7].W				Quaternion[7].X				Quaternion[7].Y				Quaternion[7].Z			
128	Quaternion[8].W				Quaternion[8].X				Quaternion[8].Y				Quaternion[8].Z			
144	Quaternion[9].W				Quaternion[9].X				Quaternion[9].Y				Quaternion[9].Z			
160	Unused															
…																
224																

Table 27. Payload of a Quaternion mode data packet.

7.2.4 Optimized mode

This mode includes up to 10 samples of quaternion, accelerometer, and gyroscope data per packet. All values are transmitted as 16-bit signed integers.

- Quaternion data must be divided by 32767 to convert to normalized decimal values.
- Accelerometer and gyroscope data use the same conversion factors as defined in tables 19 and 20, based on the selected range settings.

Byte Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Quat[0].W		Quat[0].X		Quat[0].Y		Quat[0].Z		Quat[1].W		Quat[1].X		Quat[1].Y		Quat[1].Z	
16	Quat[2].W		Quat[2].X		Quat[2].Y		Quat[2].Z		Quat[3].W		Quat[3].X		Quat[3].Y		Quat[3].Z	
32	Quat[4].W		Quat[4].X		Quat[4].Y		Quat[4].Z		Quat[5].W		Quat[5].X		Quat[5].Y		Quat[5].Z	
48	Quat[6].W		Quat[6].X		Quat[6].Y		Quat[6].Z		Quat[7].W		Quat[7].X		Quat[7].Y		Quat[7].Z	
64	Quat[8].W		Quat[8].X		Quat[8].Y		Quat[8].Z		Quat[9].W		Quat[9].X		Quat[9].Y		Quat[9].Z	
80	Acc[0].X		Acc[0].Y		Acc[0].Z		Acc[1].X		Acc[1].Y		Acc[1].Z		Acc[2].X		Acc[2].Y	
96	Acc[2].Z		Acc[3].X		Acc[3].Y		Acc[3].Z		Acc[4].X		Acc[4].Y		Acc[4].Z		Acc[5].X	
112	Acc[5].Y		Acc[5].Z		Acc[6].X		Acc[6].Y		Acc[6].Z		Acc[7].X		Acc[7].Y		Acc[7].Z	
128	Acc[8].X		Acc[8].Y		Acc[8].Z		Acc[9].X		Acc[9].Y		Acc[9].Z		Gyr[0].X		Gyr[0].Y	
144	Gyr[0].Z		Gyr[1].X		Gyr[1].Y		Gyr[1].Z		Gyr[2].X		Gyr[2].Y		Gyr[2].Z		Gyr[3].X	
160	Gyr[3].Y		Gyr[3].Z		Gyr[4].X		Gyr[4].Y		Gyr[4].Z		Gyr[5].X		Gyr[5].Y		Gyr[5].Z	
176	Gyr[6].X		Gyr[6].Y		Gyr[6].Z		Gyr[7].X		Gyr[7].Y		Gyr[7].Z		Gyr[8].X		Gyr[8].Y	
192	Gyr[8].Z		Gyr[9].X		Gyr[9].Y		Gyr[9].Z		Unused							
208																
224																

Table 28. Payload of an Optimized mode data packet.

7.2.5 Quat+Mag mode

This mode streams up to 10 samples of quaternion and magnetometer data per packet. Both data types are transmitted as 16-bit signed integers.

- Quaternion values must be divided by 32767 to convert to normalized decimal values.
- Magnetometer data uses a conversion factor of 0.0015 gauss/count.

Byte Address	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Quat[0].W		Quat[0].X		Quat[0].Y		Quat[0].Z		Quat[1].W		Quat[1].X		Quat[1].Y		Quat[1].Z	
16	Quat[2].W		Quat[2].X		Quat[2].Y		Quat[2].Z		Quat[3].W		Quat[3].X		Quat[3].Y		Quat[3].Z	
32	Quat[4].W		Quat[4].X		Quat[4].Y		Quat[4].Z		Quat[5].W		Quat[5].X		Quat[5].Y		Quat[5].Z	
48	Quat[6].W		Quat[6].X		Quat[6].Y		Quat[6].Z		Quat[7].W		Quat[7].X		Quat[7].Y		Quat[7].Z	
64	Quat[8].W		Quat[8].X		Quat[8].Y		Quat[8].Z		Quat[9].W		Quat[9].X		Quat[9].Y		Quat[9].Z	
80	Mag[0].X		Mag[0].Y		Mag[0].Z		Mag[1].X		Mag[1].Y		Mag[1].Z		Mag[2].X		Mag[2].Y	
96	Mag[2].Z		Mag[3].X		Mag[3].Y		Mag[3].Z		Mag[4].X		Mag[4].Y		Mag[4].Z		Mag[5].X	
112	Mag[5].Y		Mag[5].Z		Mag[6].X		Mag[6].Y		Mag[6].Z		Mag[7].X		Mag[7].Y		Mag[7].Z	
128	Mag[8].X		Mag[8].Y		Mag[8].Z		Mag[9].X		Mag[9].Y		Mag[9].Z		Unused			
144																
…																
224																

Table 29. Payload of a Quat+Mag mode data packet.