Head Tracker 1 protocol
General information

This revision was published on 25th October 2023. The gesture flags have been revised, and some readback messages have been added, to ensure compatibility with firmware version 0.69.

Form of messages

Supperware Head Tracker 1 communicates using MIDI System Exclusive messages:

```
        f0  00 21 42  <message type>  [other data]  f7
```

Messages directed to the head tracker have message types below 0x40; responses from it have message types of 0x40 or more.

Although the head tracker's firmware is user-upgradeable, the management of this relies on universal System Exclusive messages, which are out of the scope of this document.

Instant gratification

All sensors and data output are turned off by default. Most users may simply want a continual output of head orientation at around 50Hz. The transactions required to do this are as follows:

Reset; turn on all sensors and processing; enable yaw, pitch, and roll output at 50Hz:

```
        f0  00 21 42  00   00 48 01 01  f7
(       240 0  33 66  0    0  72 1  1   247          )
```

As above, but enable quaternion output instead:

```
        f0  00 21 42  00   00 48 01 05  f7
(       240 0  33 66  0    0  72 1  5   247          )
```

Here is a slightly more involved example, which enables yaw, pitch, and roll output at 100Hz, turns off the compass, and turns on shake-to-zero:

```
        f0  00 21 42  00   00 68  03 20 04 18 01 01  f7
(       240 0  33 66  0    0  104  3 32  4 24  1  1  247    )
```

The head tracker will immediately start sending data. Zero it (when ready):

```
        f0  00 21 42  01   00 01  f7
(       240 0  33 66  1    0  1   247        )
```

14-bit fixed-point numbers

Wherever used, 14-bit numbers are two's complement signed and given in big-endian form. There is one sign bit, two integer bits and eleven signed bits (Q2.11), so that a number between –4 and almost +4 can be expressed to an accuracy of about ±0.0005.

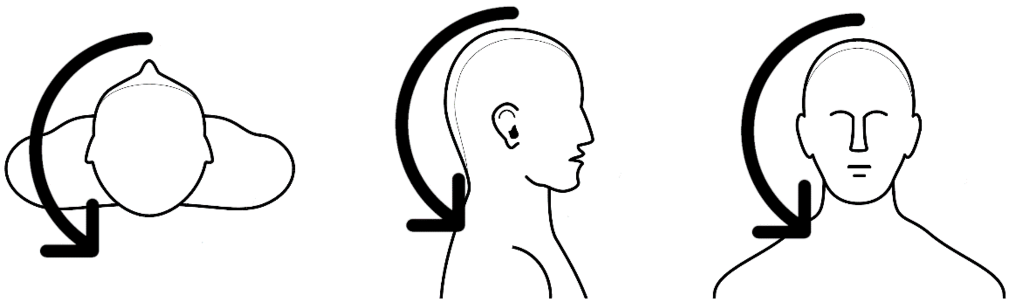
First byte	`0MXX XXXX`
Second byte	`0XXX XXXL`

(M and L denote the most and least significant bits.) These numbers can be converted to a floating-point number using the following code. Brackets are added for clarity:

```
int i = (128 * first_byte) + second_byte;
if (i >= 8192) { i -= 16384; }
float f = i / 2048.0f;
```

Tait-Bryan angles

Tait-Bryan angles, used by the head tracker to provide yaw, pitch, and roll, are the ZXY class of Euler angle. Precise head orientation is obtained by applying the three rotations in the following order. The arrows indicate the direction of a positive rotation:



Successive rotations are fixed with respect to the head, not the world. The order is important, because 'pitch 45 degrees then roll 45 degrees' (as employed by the tracker) places one's head in a different orientation from 'roll 45 degrees then pitch 45 degrees.'

Handling 3D rotations is mathematically equally burdensome in any format, but Tait-Bryan angles present the most intuitive output. However, they should be used only if the head tracker is not expected to deal with a very wide field of movement. Gimbal lock seriously affects readings around ±90 degrees of pitch. In other words, as you look at the ceiling or floor, yaw and roll become exactly the same rotation. The remaining degree of freedom available to your head is impossible to express. No other output format has this drawback.

Head tracker protocol
Messages in detail

Message 0 : Configure sensors and processing pipeline

```
       f0  00 21 42  00  <parameter> <value> [<parameter> <value> ...]  f7
(      240 0  33 66  0   <parameter> <value> [<parameter> <value> ...]  247     )
```

Summary of Message 0 parameters

parameter	function
0	Sensor setup
1	Data output and formatting
2	Calibrate gyroscope / Save data to non-volatile storage
3	Compass control
4	Gestures / Chirality
5	New state (readback only)

Message 0 / Parameter 0 : Sensor setup

6	5	4	3	2	1	0
RESET	RATE		SENSORS_ON	0	0	0

Writing to this address will always reinitialise the sensors, regardless of their current and intended states.

bit 6	RESET 1 : reset the head tracker to its power-on (or saved) defaults
bits 5:4	RATE : sensor output rate 00 : 50-52Hz (default) 01 : 25-26Hz 10 : 100-104Hz 11 : reserved (do not use)
bit 3	SENSORS_ON 0: reset/turn off all sensors (default) 1: reset/activate all sensors
bits 2:0	Reserved (set to 0)

Message 0 / Parameter 1 : Data output and formatting

6	5	4	3	2	1	0
0	RAW_FORMAT		ORIENT_FORMAT		TRACKING	

bit 6	Reserved (set to 0)
bits 5:4	RAW_FORMAT : output raw sensor data
	00 : do not output raw data (default)
	01 : output raw sensor data, applying compass calibration
	10 : output raw sensor data, without compass calibration
	11 : reserved (do not use)
bits 3:2	ORIENT_FORMAT : output format for orientation data
	00 : Tait-Bryan angles
	01 : Quaternions
	10 : Orthogonal matrix
	11 : reserved (do not use)
bits 1:0	TRACKING : enable head tracking
	00 : head tracking output is turned off (default)
	01 : head tracking is turned on
	10-11 : reserved (do not use)

Message 0 / Parameter 2 : Calibrate gyroscope / Reset non-volatile storage

6	5	4	3	2	1	0
PATTERN						

This message requires the writing of a specific pattern.

bits 6-0	**function**
0x3C (60)	One-time gyroscope calibration, and save. (Sensors must be enabled for this to work).
0x5A (90)	Reset all saveable register values to factory defaults, and save.

Any other value written to this parameter will be ignored.

Message 0 / Parameter 3 :
Compass control

6	5	4	3	2	1	0
VERBOSE	COMPASS_MODE			FORCE_CAL	COMPASS_STATE	

Values written to the underlined registers are preserved in non-volatile memory.

bit 6	VERBOSE When set, any change in status sends out Readback message 5. Default is 0.
bits 5:3	COMPASS_MODE 000-011: ignore this message. Otherwise: Bit 4: 1 to turn compass on; 0 to turn compass off. (If the data from the compass is invalid, it will be turned off anyway.) Bit 3: Amount of yaw correction to use when compass is off: 0: slow central pull (0.3 degrees per second) 1: use no correction
bit 2	FORCE_CAL 1 : force a recalibration of the compass over the next 6 seconds. The new calibration data is automatically saved. Always read as 0.
bits 1:0	COMPASS_STATE (readback only: writes are ignored) 00 : compass is turned off 01 : compass is on but poorly calibrated, so is not being used 10 : compass is being used and is calibrated 11 : compass is currently in calibration mode

Note that SENSORS_ON (Message 0 Parameter 0) must be set in order for this command to function correctly.

Message 0 / Parameter 4 :
Gestures
Chirality

6	5	4	3	2	1	0
0	0	GESTURES_ON			RIGHT_EAR	

Values written to the underlined registers are preserved in non-volatile memory.

bits 6:4	Reserved (set to 0)
bits 4:2	GESTURES_ON 000-011 : preserve the current value of GESTURES-ON 100 : gestures are turned off (default) 101 : reserved 110 : shake-to-zero gesture on 111 : reserved
bits 1:0	RIGHT_EAR 00-01 : preserve the current value of RIGHT_EAR 10 : Power cable is positioned over left ear (default) 11 : Power cable is positioned over right ear

Setting RIGHT_EAR effectively inverts the polarity of pitch and roll rotation, or its equivalent in other modes, while yaw remains the same.

The shake-to-zero gesture can be activated by a double-shake of the head, or any other repeated reciprocal movement (such as a nod or roll) will work. Shaking the head is easiest to do consistently, though. Readback message 5 in verbose mode will output data about the progression of this gesture:

0	The head is still
1	Forward movement 1 detected
2	Reverse movement 1 detected
3	Forward movement 2 detected
4	Reverse movement 2 detected; head tracker about to zero

A successful gesture will cause the counter to advance from 0 to 4; an unsuccessful gesture will abort because of insufficient speed, precision, or range of head movement before the counter reaches 4.

Message 0 / Parameter 5 :
New state (readback only)

6	5	4	3	2	1	0
0	0	0	NEW_STATE			

This parameter is sent by the head tracker as a readback message whenever its state has changed, only if the VERBOSE bit (bit 6 of Parameter 3) is set.

bits 6:4	Always return 0
bits 3:0	NEW_STATE

0000 :	Zero. Head tracker has just zeroed.
0001 :	Compass calibration started.
0010 :	Compass calibration successful.
0011 :	Compass calibration finished unsuccessfully. (Head tracker will revert to the last good calibration).
0100 :	Compass data has become bad. (Head tracker will stop using compass data: This is the same as COMPASS_STATE 01).
0101 :	Compass data has recovered. (Head tracker will start using compass data: This is the same as COMPASS _STATE 10).
0110 :	Gyroscope calibration has finished.
1000 – 1100 :	Stages 0–4 of shake-to-zero gesture.

Message 1 : Control during use

```
      f0  00 21 42 01  <parameter> <value> [<parameter> <value> ...]  f7
(     240 0  33 66 1   <parameter> <value> [<parameter> <value> ...]  247      )
```

Summary of Message 1 parameters

parameter	function
0	Zeroing
1	Travel mode

Message 1 / Parameter 0 :
Zeroing

6	5	4	3	2	1	0
0	0	0	0	0	0	ZERO

bit 0	ZERO : Set zero point 0 : No action 1 : Use most recent stable data as zero (level, straight ahead)

Message 1 / Parameter 1 :
Travel mode

6	5	4	3	2	1	0
0	0	0	0	TRAVEL		

bit 6:3	Reserved (set to 0)
bits 2:0	TRAVEL : automatic yaw correction when travelling 000-011 : preserve the current value of TRAVEL 100 : travel mode off (default) 101 : reserved (do not use) 110 : slow correction 111 : fast correction (preferred)

Message 2 : Readback

Read back calibration and configuration data.

```
       f0  00 21 42 02  <parameter> [<parameter> ...]  f7
(      240 0  33 66 2   <parameter> [<parameter> ...]  247       )
```

Parameter	Meaning
0x00	Sensor setup (Message 0 / Parameter 0) readback
0x01	Data output and formatting (Message 0 / Parameter 1) readback
0x03	Compass (Message 0 / Parameter 3) readback
0x04	Gesture and chirality (Message 0 / Parameter 4) readback
0x05	Change in status readback. Not requested directly, but given in verbose mode (see Message 0 / Parameter 3)
0x11	Travel mode (Message 1 / Parameter 1) readback

Message 64 : Head tracker data responses

Message 64 / Parameter 0 :
Orientation in Tait-Bryan Angles

```
       f0  00 21 42  40 00  <14-bit yaw> <14-bit pitch> <14-bit roll>  f7
(      240 0  33 66  64 0   <14-bit yaw> <14-bit pitch> <14-bit roll>  247      )

                                                                  [13 bytes]
```

The data is given in radians in the Q2.11 format described in the 'Number Formats' section earlier. This gives a maximum accuracy of 1/2048 radians, and the range is fixed to ±π radians so that numbers cannot overflow. For example:

180 degrees → π radians → 0x1922

Example: to obtain the data in degrees from the raw 14-bit input,

```
// convert signed 14-bit data (see page 1):
int i = (128 * d[0]) + d[1];
if (i >= 8192) { i -= 16384; }

// the head tracker produces its output in radians:
float f_radian = i / 2048.0f;

// but if you need degrees, the multiplier is 180/(pi*2048)
float f_degrees = i * 0.027976f;
```

Message 64 / Parameter 1 :
Orientation in Quaternions

```
       f0  00 21 42  40 01  <14-bit w> <14-bit x> <14-bit y> <14-bit z>  f7
(      240 0  33 66  64 1   <14-bit w> <14-bit x> <14-bit y> <14-bit z>  247    )

                                                                  [15 bytes]
```

Quaternion coefficients are sent in the Q2.11 format, such that:

```
R  =  w  +  i . x  +  j . y  +  k . z
```

Message 64 / Parameter 2 : Orientation in Orthogonal Matrix

```
f0  00 21 42  40 02  <14-bit xx> <14-bit xy> <14-bit xz>
                     <14-bit yx> <14-bit yy> <14-bit yz>
                     <14-bit zx> <14-bit zy> <14-bit zz>  f7     [25 bytes]
```

This is a 3×3 rotation matrix that can be multiplied by a 3D coordinate to change its frame of reference. Reading each row as a 3D unit vector for X, Y, and Z, it gives the body-to-world transformation.

Because this is an orthogonal matrix, the inverse matrix (for world-to-body transformation) is the transpose of this one, found by reading each column as a 3D unit vector.

Message 65 : Raw data response

```
       f0  00 21 42 41 <sensor id> <timestamp>  <16-bit x> <16-bit y> <16-bit z>  f7
(      240 0  33 66 65 <sensor id> <timestamp>  <16-bit x> <16-bit y> <16-bit z> 247  )

                                                                  [17 bytes]
```

Sent asynchronously as soon as data is ready. The timestamp increments every millisecond, so it wraps every 128ms. This is frequent enough that there will be no more than one wrap-around between sensor readings.

sensor ID	device	sensor
0	near-end sensor	accelerometer
1		gyroscope
2	top sensor	accelerometer
3		compass
4	far-end sensor	accelerometer
5		gyroscope

Signed 16-bit data is arranged as follows:

First byte	`0000 00MX`
Second byte	`0XXX XXXX`
Third byte	`0XXX XXXL`

These numbers can be converted to a floating-point number using the following code. Brackets are added for clarity:

```
int i = (16384 * first_byte) + (128 * second_byte) + third_byte;
if (i >= 32768) { i -= 65536; }
float f = i / 32768.0f;
```

Message 66 : Readback data response

```
       f0   00  21  42  42  <parameter> <value>  f7
 (     240  0   33  66  66  <parameter> <value>  247       )
```

Sent in response to a Message 2 readback request. Responses to a string of requests are sent individually.

Appendix: MIDI Device Inquiry

The head tracker responds to a standard MIDI Device Inquiry message with a 17-byte reply, which may be used to determine its hardware revision and firmware version.

In response to:

```
    f0   7e   01  06  01  f7
(   240  126  1   6   1   247   )
```

or:

```
    f0   7e   7f   06  01  f7
(   240  126  127  6   1   247   )
```

an attached head tracker responds:

```
    f0 7e 01 06 02
       00 21 42            : Supperware System Exclusive manufacturer ID
       00 00               : Device family code
       [hw] 00             : [hw] = Device’s hardware revision code
       [min] [maj] 00 00   : [min], [maj] = Firmware’s minor and major revision numbers
       f7

(   240 126 1 6 2
    0 33 66                : Supperware System Exclusive manufacturer ID
    0 0                    : Device family code
    [hw] 0                 : [hw] = Device’s hardware revision code
    [min] [maj] 0 0        : [min], [maj] = Firmware’s minor and major revision numbers
    247
)
```

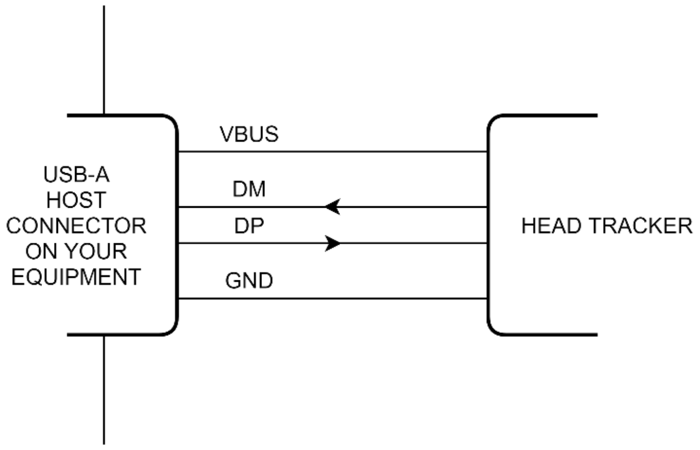
For example, version 0.10 of firmware installed on Issue 2 hardware will return the following:

```
    f0  7e  01  06  02    00  21  42    00  00    02  00    0a  00  00  00    f7
(   240 127 1   6   2     0   33  66    0   0     2   0     10  0   0   0     247   )
```

Appendix: Using the head tracker with a UART

The head tracker will connect either to a USB host or a UART (*universal asynchronous receiver/transmitter*). The latter is more convenient for embedded projects. UART mode cannot be used to perform firmware upgrades (which are beyond the scope of this document anyway) but all other features are available.

The head tracker should be connected as follows:



- A USB-A connector is be placed on your equipment to accept an ordinary USB cable.
- The USB 'DM' (or D–) pin connects to your equipment's receiver line, to carry data from the head tracker.
- The USB 'DP' (or D+) pin connects to your equipment's transmitter line, to carry data to the head tracker.

A more comprehensive schematic appears at the end of this section.

VBUS, GND, and chassis

To power the head tracker, VBUS must provide between 4 and 6 volts. The maximum current draw will be 30mA. Your microcontroller must be able to turn this VBUS on or off remotely, as the timing of power-on and data transmission is very important.

The recommended schematic delivers up to 30mA through a current-limiting load switch. If a user might expect to charge their phone over the socket, you should consider embedding a USB power delivery controller, and then you might as well be speaking USB.

The connector's chassis should not be left floating, but ideally should connect to your own chassis net, and be kept away from sensitive electronics. If there is no suitable chassis, connect the USB chassis to signal ground via a 10k resistor as shown. This does not provide a shield against electrostatic interference, but will protect your equipment from static discharges by gently ensuring equipotential at each end of the cable before more sensitive lines are connected.

Handshaking

The head tracker is designed to work in UART mode in response to the following procedure:

1. Apply power to the head tracker, and wait 200ms for it to wake up.
2. Send a System Exclusive message to make the head tracker work in the intended way. Wait 100ms for a reply.
3. If no reply is received, go back to step 2. Retry up to 15 times.
4. After 15 failed attempts, power down, wait 150ms, and go back to step 1.

The head tracker will start to respond at the intended interval, usually on around the tenth attempt after power on. It is very important to follow a procedure similar to the one above, because the head tracker will give up attempting to communicate with a UART, and will default to USB mode, unless both of these occur:

- The DP and DM lines are read high within 50ms of powering up;
- It receives a complete message on the UART within 250ms of powering up.

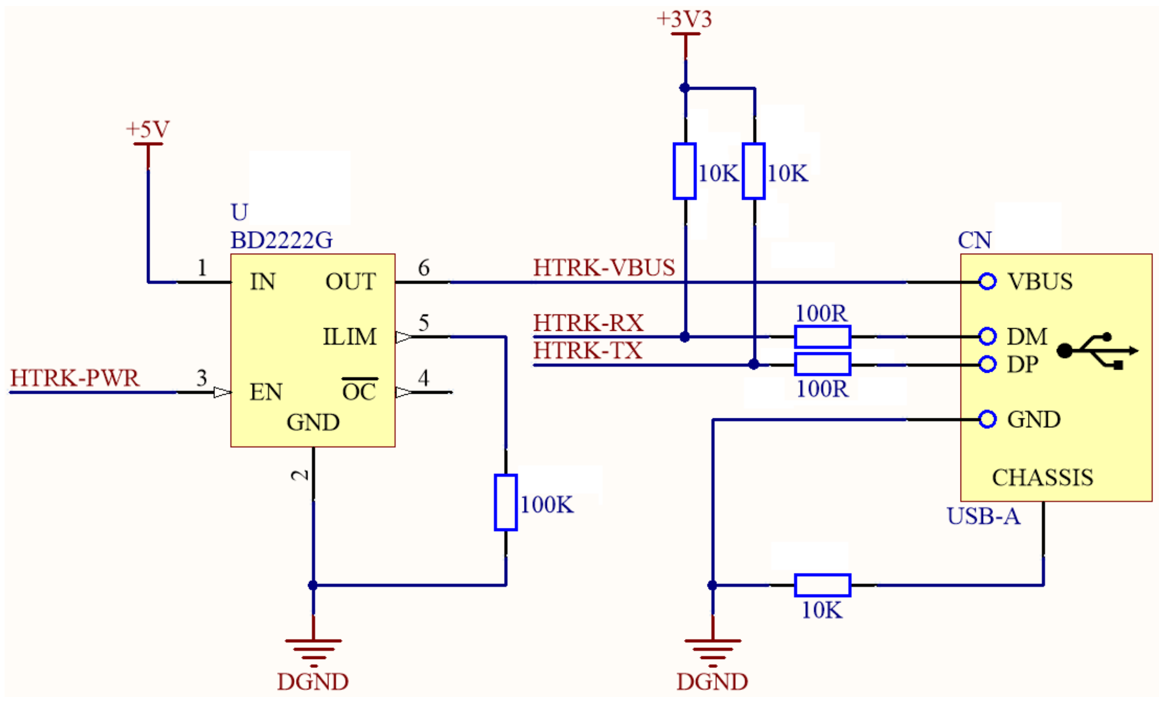
DM and DP

The UART is intended to work at 3.3V, but is 5V-tolerant. Driving more than 5V or less than 0V onto the DM or DP lines may permanently damage the head tracker.

The UART runs at 115 200 baud, with one start bit and one stop bit. Your device must provide pull-up resistors if your microcontroller doesn't provide them itself. Values of around 10-20 kiloohms, attached to a 3.3V rail, are recommended.

Messages sent and received in UART mode are the same as standard MIDI, albeit at a higher rate. The head tracker's own baud rate is accurate to ±1%. This may inform your design approach if your UART operates on a marginal tolerance.

As there is no flow control, your equipment should wait around 1ms between transmitting each System Exclusive message to give the head tracker time to process it. This delay may be added to a precalculated block of set-up data by padding with twelve zero bytes between System Exclusive frames. As these bytes are not part of a valid MIDI message, they will take up time on the bus, but the head tracker will ignore them.



Recommended schematic for interfacing your own equipment
to the head tracker in UART mode