IMU Configuration Manual

Applicable products: CH0X0 / HI14 / CH10X series (V1.6.6)

Official website: https://sealandtech.com.tw

Classification: Public

Purchasing / Support: sltech@ms28.hinet.net

IMU Configuration Manual

1. Module Configuration Overview

The default configuration of the product can satisfy the needs of most users. Therefore, before using the product, please read this section carefully and, based on your application requirements, decide whether user configuration is needed.

1.1. Geomagnetic Assistance Scenario (AHRS / 9-axis mode)

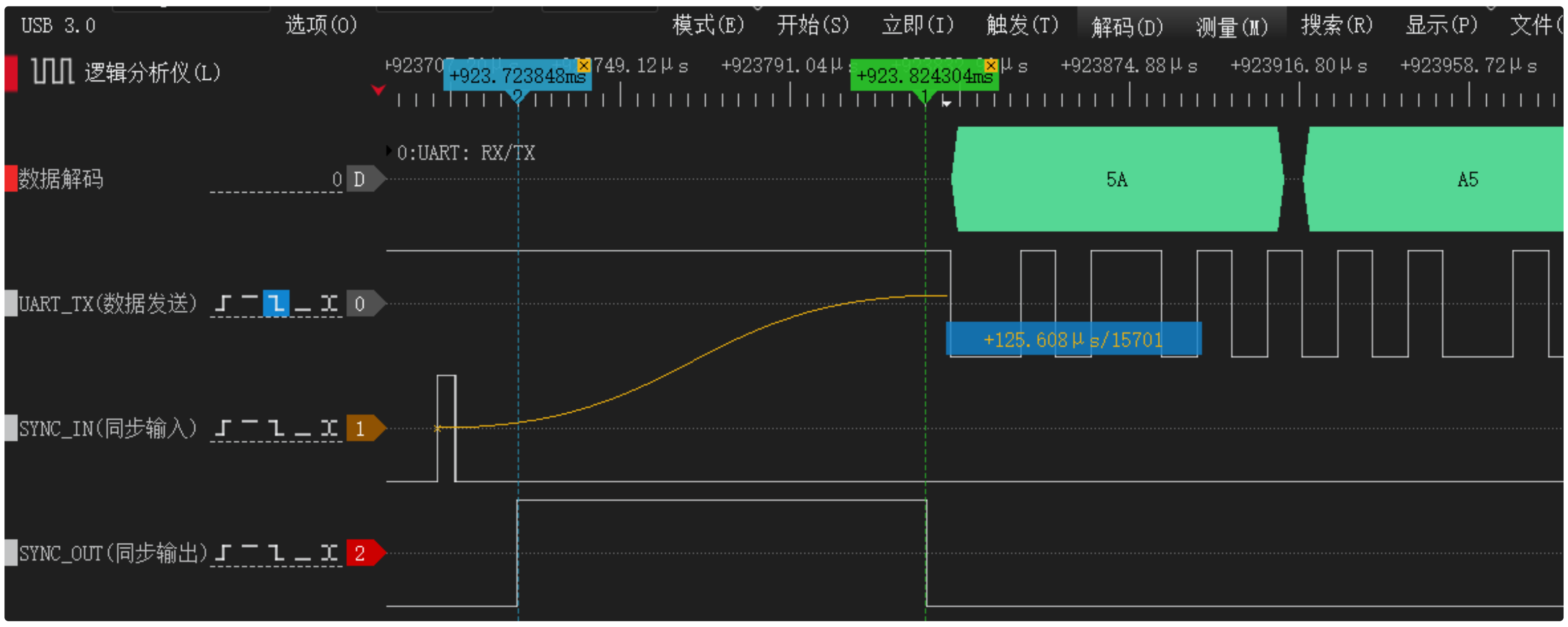
In the vast majority of cases (robots and indoor environments), AHRS (9-axis) mode is easily affected by magnetic interference, causing heading (yaw) errors. In a small number of open environments with little magnetic disturbance, you can try using the geomagnetic assistance mode, for example on UAVs. Before use, configure the module to geomagnetic assistance mode and perform magnetometer calibration. See the **Magnetometer Calibration** section for details.

> For serial interface configuration, refer to **CONFIG – Mode Configuration**.

1.2. Sync Input & Sync Output (Time Synchronization)

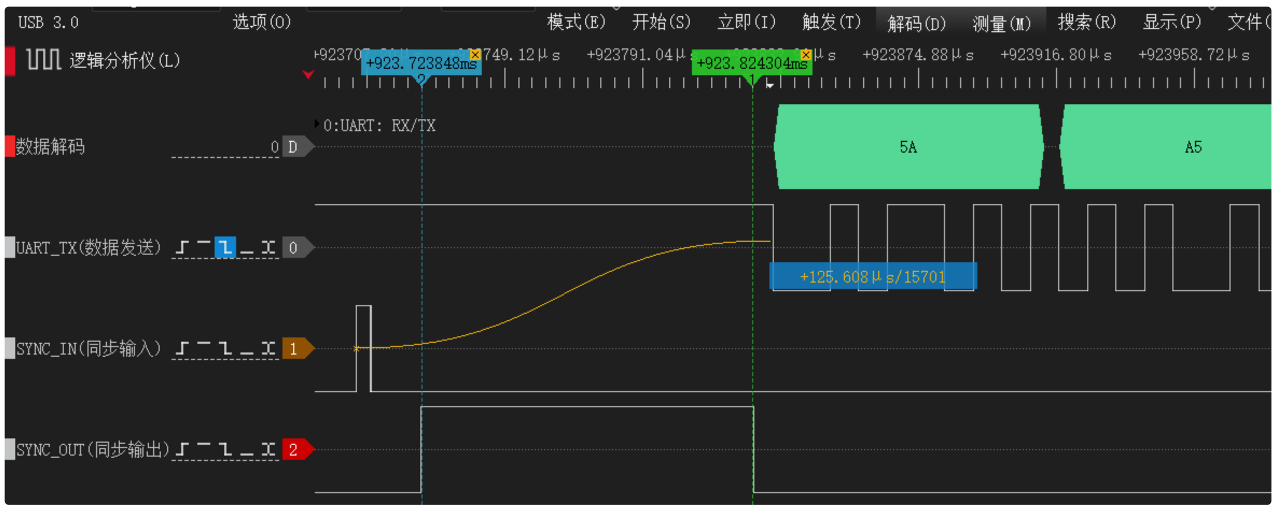
1.2.1. Data Sync Trigger (SIN)

Some products provide a sync-input pin (SIN) for IMU time synchronization. It may be left floating when not used. When an output frame is configured for sync triggering (ONMARK trigger; see the `LOG` command), the module outputs one frame of that data each time a rising edge is detected on the SIN pin. This feature is mainly used to receive a high-precision square-wave pulse generated by the host controller to trigger high-rate, synchronous IMU data. The delay from the SIN rising edge to the start of frame transmission is **125 µs** (see figure below).



1.2.2. Sync Output (SOUT)

- Data sync output (SYNC_OUT): output pin. It stays low when there is no data output (idle). Before a frame starts transmitting, the module outputs a high pulse. Immediately after the pulse ends (falling edge), the data output begins, as shown below.



> - When using the sync input feature, first disable all non-synchronous periodic outputs, then use `LOG <MSG> ONMARK 1` to enable sync-triggered output. See the `LOG` command section for details.

2. Module Configuration Commands

The module uses ASCII string commands for configuration. Each command must end with a carriage return + line feed `\r\n` (similar to AT commands) to be recognized by the system.

2.1. Command Summary

Command	Function	Notes
REBOOT	Reset the module	Equivalent to power-cycle
SAVECONFIG	Save all configuration parameters	Takes effect immediately
SERIALCONFIG	Set baud rate	Takes effect immediately
CONFIG	Set user parameters and modes	Takes effect immediately
LOG	Print module info or configure data output	Takes effect immediately
FRESET	Restore factory defaults	Takes effect immediately

All configuration commands take effect only after a reset or power-cycle.

2.2. Command Details

2.2.1. `REBOOT`

Reset the module. Takes effect immediately and is equivalent to a power cycle.

2.2.2. `SAVECONFIG`

Save all user configuration parameters to Flash.

2.2.3. `SERIALCONFIG`

Set the serial port baud rate. Supported values: 9600 / 115200 / 256000 / 460800 / 921600

Example: set baud rate to 115200
`SERIALCONFIG 115200`
`SAVECONFIG`

Use this command with caution. Setting an incorrect baud rate may prevent communication with the module.

2.2.4. `CONFIG`

Used to configure module operating parameters. Most commands take effect immediately. Use `SAVECONFIG` to persist changes across power-off.

Attitude mode configuration: 6-axis or 9-axis (geomagnetic assisted) mode

- `CONFIG ATT MODE 0` Configure the module to 6DOF mode
- `CONFIG ATT MODE 1` Configure the module to AHRS (9-axis) mode

Leveling / Horizon calibration

- `CONFIG ATT RST 2` Set relative level (set relative zero): set the current Pitch/Roll angles to zero.
- `CONFIG ATT RST 5` Cancel leveling: clear the relative pitch/roll set by `CONFIG ATT RST 2`.
- `CONFIG ATT RST 3` Auto leveling (for robot IMU products):
  If the current pitch/roll angles are close to 0°,0° (placed flat, face up), auto calibrate to 0°,0°.
  If the current pitch/roll angles are close to 0° or 180° (placed flat, face down), auto calibrate to 0°,180°.
  “Close to” is defined as **both** Pitch and Roll < 5°.

- When executing `CONFIG ATT RST` commands, keep the module stationary. Performing the command while the module is moving may cause significant leveling error.
- Differences between `ATT RST 2` and `ATT RST 3`:

- `ATT RST 3` checks whether the module is only slightly tilted; if the tilt is large, it will abort. `RST 2` performs no pre-check.
- `ATT RST 3` also calibrates accelerometer output during leveling so that X and Y accelerations are close to 0.
- After `ATT RST 3`, it cannot be canceled by `ATT RST 5`; it is permanent.

Manually set heading (yaw)

Manually set yaw: `CONFIG YAW <MODE> <VAL>`
MODE: 0 = relative mode, 1 = absolute mode
VAL: yaw setpoint, range (-180 to 180), unit: deg
This command takes effect immediately and is not saved after power-off.

- `CONFIG YAW 1 16` Absolute set: set yaw to 16 deg
- `CONFIG YAW 0 1.5` Relative set. Example: if current yaw is 30 deg, after setting it becomes 30 + 1.5 = 31.5 deg

Coordinate rotation (change mounting orientation)

`CONFIG IMU URFR C00,C01,C02,C10,C11,C12,C20,C21,C22`

Where each C_{nn} supports floating-point values.

$$\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_B = \begin{bmatrix} C00 & C01 & C02 \\ C10 & C11 & C12 \\ C20 & C21 & C22 \end{bmatrix} \cdot \begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_U$$

$\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_U$ is the sensor data under the **rotated** sensor coordinate system, and $\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_B$ is the sensor data under the **original** sensor coordinate system.

Common rotation examples:

- New sensor coordinate system = rotate about original X axis by −90° (**vertical mounting with +Y pointing downward**). Command:
  `CONFIG IMU URFR 1,0,0,0,0,1,0,-1,0`
- New sensor coordinate system = rotate about original X axis by +90° (**vertical mounting with +Y pointing upward**). Command:
  `CONFIG IMU URFR 1,0,0,0,0,-1,0,1,0`
- New sensor coordinate system = rotate about original X axis by 180°. Command:
  `CONFIG IMU URFR 1,0,0,0,-1,0,0,0,-1`
- New sensor coordinate system = rotate about original Y axis by +90° (**vertical mounting with +X pointing upward**). Command:
  `CONFIG IMU URFR 0,0,-1,0,1,0,1,0,0`
- New sensor coordinate system = rotate about original Y axis by −90° (**vertical mounting with +X pointing downward**). Command:
  `CONFIG IMU URFR 0,0,1,0,1,0,-1,0,0`
- New sensor coordinate system = rotate about original Y axis by 180°. Command:
  `CONFIG IMU URFR -1,0,0,0,1,0,0,0,-1`
- New sensor coordinate system = rotate about original Z axis by +90°. Command:
  `CONFIG IMU URFR 0,-1,0,1,0,0,0,0,1`
- New sensor coordinate system = rotate about original Z axis by −90°. Command:
  `CONFIG IMU URFR 0,1,0,-1,0,0,0,0,1`
- Level mounting, Z axis up (default):
  `CONFIG IMU URFR 1,0,0,0,1,0,0,0,1`

> - After setting URFR, a software reset or power cycle is required for it to take effect. You do not need to send this command on every power-up.
> - How to determine URFR parameters (example: rotate about original X axis by −90°, i.e., vertical mounting with +Y pointing downward). The relationship between the transformed axes and original axes:
>   - $X_U = X_B$
>   - $Y_U = -Z_B$
>   - $Z_U = Y_B$
>
> Therefore, the transformation matrix is:
>
> 1 0 0 (new X = original X)
> 0 0 -1(new Y = − original Z)
> 0 1 0(new Z = original Y)

> According to the URFR definition above, URFR expects the transpose of this matrix:
>
> $$\begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{matrix}$$

Multi-function IO multiplexing

The module provides multiple multi-function pins: IO1–IO9. These pins can be assigned to different functions and switched via configuration commands.

`CONFIG <PMUX> <IO>`

- PMUX: multiplex function: PMUX1 – PMUX5
- IO: pin number: IO1 – IO9

Example:

- Assign IO2 to LED (PMUX3): `CONFIG PMUX3 IO2`

PMUX No.	Function name	Dir	Description	Default IO
PMUX1	SIN	I	Sync pulse input (SIN/PPS): input pin. See the sync section.	IO1
PMUX2	SOUT	O	Sync output: idle low. A high pulse (80 µs) is output before each frame to sync data.	IO2
PMUX3	LED	O	Output, status indicator LED	IO5

> - Not all IO pins are brought out on every product. Refer to the hardware section of the product user manual.

User-level gyroscope calibration

Gyroscope accuracy may be affected by aging (gradual performance drift), mounting stress (PCB soldering deformation), temperature variation, and mechanical stress (vibration/shock). This product supports independent X/Y/Z axis calibration to improve angular measurement accuracy (after calibration, scale-factor error can be guaranteed within 0.1%). Calibration parameters are stored permanently in Flash and include real-time monitoring to ensure calibration validity.

Calibration methods:

- Z-axis calibration: can be performed by placing the module on a level surface without special fixtures. Suitable for AGV/large machines where fixtures are hard to build. Do not calibrate by hand or with a manual turntable; this cannot guarantee accuracy.
- X/Y-axis calibration: requires a precision rotation device (e.g., professional turntable) to ensure the rotation axis is aligned with the axis to be calibrated. If you do not have such equipment, X/Y calibration is not recommended.
- The three axes are independent; you may calibrate any one axis. No need to specify which axis is rotating—the firmware detects it automatically. After a successful calibration, it takes effect immediately and is saved across power-off.

Calibration procedure:

1. Preparation
   - For Z-axis calibration: place the module on a flat, level surface.
   - For X/Y-axis calibration: mount the module on the rotation device and align the rotation axis to the axis under calibration.
   - Keep ambient temperature stable and avoid strong vibration.
2. Send the start command: `CONFIG USRCAL START <ANGLE>`
   - <ANGLE>: calibration angle (720–1800 degrees, i.e., 2–5 turns). More turns yield higher accuracy.
   - Example: `CONFIG USRCAL START 720` starts a 2-turn calibration (X/Y/Z axis can be any one axis).
3. Perform the rotation
   - Speed requirement: 20–100 deg/s (recommended 50 deg/s, ~5–6 s per revolution; either direction is fine)
   - Rotate at a constant speed using the turntable/device.
   - Keep speed uniform; avoid pauses, sudden acceleration/deceleration, etc.
4. Send the stop command: `CONFIG USRCAL STOP`
   - Returns "OK": calibration successful; new parameters are automatically saved and applied.
   - Returns "ERR": calibration failed; original parameters remain in use.

Common reasons for calibration failure:

1. Excessive angle deviation: the actual rotation angle differs from the set value by more than 5%.
2. Non-standard rotation:
   - Z-axis: ensure the module is stable and remains level during rotation.

   - X/Y-axis: ensure accurate alignment between the rotation axis and the axis being calibrated.

3. Improper operation: rotating too fast/too slow, pausing midway, etc.
4. Environmental interference: avoid operation near vibrating machinery (e.g., excessive motor vibration).
5. Error remains or worsens after calibration: most likely due to non-standard operation (see 1–4). Calibration is a precision process. For example, if the target is 720° but the actual rotation is 725°, the scale factor error becomes 5/720 = 0.6%, which exceeds factory accuracy. Ensure proper fixtures and precise rotation; otherwise, do not perform user calibration.

2.2.5. `LOG`

ENABLE / DISABLE: Globally enable/disable data output

`LOG ENABLE` Globally enable frame output (default)
`LOG DISABLE` Globally disable frame output

VERSION: Show firmware version information

`LOG VERSION` Print firmware version info

COMCONFIG: Show serial port configuration information

`LOG COMCONFIG` Print serial port and output protocol configuration info

Configure frame output type and rate

`LOG <MSG> <TYPE> <VALUE>`

Periodic output

- **MSG**: HI91, HI92
- **TYPE**: fixed as `ONTIME`
- **VALUE**: output period in seconds, valid range: 1 (1 Hz), 0.5 (2 Hz), 0.1 (10 Hz), 0.02 (50 Hz), 0.01 (100 Hz), 0.005 (200 Hz), 0.002 (500 Hz), etc.

Examples:

- `LOG HI91 ONTIME 0.01` Set HI91 output period to 0.01 s (100 Hz) on the current serial port
- `LOG HI92 ONTIME 0.05` Set HI92 output period to 0.05 s (20 Hz)
- `LOG HI91 ONTIME 0` Disable HI91 output

Synchronous (external trigger) output

- **MSG**: HI91, HI92
- **TYPE**: fixed as `ONMARK`
- **VALUE**: fixed as `1`

Examples:

- `LOG HI91 ONMARK 1` Configure HI91 as sync-triggered output. Each pulse on SIN/PPS triggers one frame.
- `LOG HI91 ONMARK ONCE` Manually trigger one output (same effect as one SIN pulse)

> When the output rate is high (e.g., 500 Hz), the default 115200 baud may not provide enough bandwidth. Increase the module baud rate (e.g., to 921600) to output correctly. After setting the baud rate, save to Flash and reset the module to take effect. The host/PC baud rate must be updated accordingly.

2.2.6. `FRESET`

Restore factory defaults.

3. RS-232 / TTL / USB Data Protocol (Custom Binary)

This is a private binary protocol that can output all sensor information. Supported interfaces: RS-232 / TTL / USB (virtual COM port). The default serial format is N-8-N-1 (8 data bits, 1 stop bit, no parity).

3.1. Frame Format

After power-up, the module outputs frames at the default rate (100 Hz). Frame format:

Field name	Value	Length (bytes)	Description
Header	5A A5	2	Frame header
Data length	1–512	2	Length of the payload data field in the frame, LSB first. Length is payload length (excluding header, frame type, length, CRC).
CRC	-	2	16-bit CRC of all bytes except CRC itself (header, frame type, length, payload). LSB first.
Payload (data field)	-	1–512	Data carried by one frame, composed of multiple sub-packets. Each sub-packet includes a tag and data. The tag determines type and length.

3.2. Factory Default Output

Factory default output: floating-point IMU data frame (HI91)

3.3. Payload Content

3.3.1. Floating-point IMU frame (HI91)

Payload length: 76 bytes. Includes module ID, temperature, raw IMU data, magnetometer, barometer, fused attitude, etc.

Enable example: `LOG HI91 ONTIME 1`. See the configuration commands section for details.

Byte offset	Name	Data type	Size (Byte)	Unit	Scale	Description
0	tag	uint8_t	1	-	-	Packet tag: 0x91
1	status	uint16_t	2	-	-	Status word, reserved
3	temperature	int8_t	1	°C	1	Module average temperature
4	air_pressure	float	4	Pa	1	Air pressure
8	system_time	uint32_t	4	ms	1	If GPS time is not synchronized, this is the local timestamp (ms since boot). If GPS time is synchronized, this is UTC time.
12	acc_b	float	4*3	g	1	Factory-calibrated acceleration, XYZ order. 1 g is local gravity acceleration, approximately 9.8 m/s^2.
24	gyr_b	float	4*3	deg/s (dps)	1	Factory-calibrated angular rate, XYZ order
36	mag_b	float	4*3	uT	1	Magnetic field strength, XYZ order
48	roll	float	4	deg	1	Roll angle
52	pitch	float	4	deg	1	Pitch angle
56	yaw	float	4	deg	1	Yaw / heading angle
60	quat	float	4*4	-	-	Quaternion, WXYZ order

3.3.2. Integer IMU frame (HI92)

Total length: 48 bytes (smaller than HI91). Enable example: `LOG HI92 ONTIME 1`. See the configuration commands section for details.

Byte offset	Name	Data type	Size (Byte)	Unit	Scale	Description
0	tag	uint8_t	1	-	-	Packet tag: 0x92
1	status	uint16_t	2	-	-	Status word, reserved
3	temperature	int8_t	1	°C	1	System average temperature
4	reserved	uint16_t	2	-	-	Reserved
6	air_pressure	int16_t	2	Pa	1	Air pressure + 100000 Pa. Example: 2000 means 102000 Pa
8	heave	in16_t	2	m	0.01	Vessel heave (vertical displacement), marine products only
10	gyr_b	int16_t*3	6	rad/s	0.001	IMU angular rate X,Y,Z (factory calibrated)
16	acc_b	int16_t*3	6	m/s^(2)	0.0048828	IMU acceleration X,Y,Z (factory calibrated)
22	mag_b	int16_t*3	6	uT	0.030517	IMU magnetic field X,Y,Z (factory calibrated)
28	roll	int32_t	4	deg	0.001	Roll angle
32	pitch	int32_t	4	deg	0.001	Pitch angle
36	yaw	int32_t	4	deg	0.001	Yaw / heading angle
40	quat	int16_t*4	8	-	0.0001	Quaternion, WXYZ order

3.4. CRC

16-bit CRC implementation example:

```
/*
    currectCrc: previous crc value, set 0 if it's first section
    src: source stream data
    lengthInBytes: length
*/
static void crc16_update(uint16_t *currectCrc, const uint8_t *src, uint32_t lengthInBytes)
{
    uint32_t crc = *currectCrc;
    uint32_t j;
    for (j=0; j < lengthInBytes; ++j)
    {
        uint32_t i;
        uint32_t byte = src[j];
        crc ^= byte << 8;
        for (i = 0; i < 8; ++i)
        {
            uint32_t temp = crc << 1;
            if (crc & 0x8000)
            {
                temp ^= 0x1021;
            }
            crc = temp;
        }
    }
    *currectCrc = crc;
}
```

3.5. Frame Structure Example (HI91)

Capture one frame (HI91) with a serial assistant. Total length is 82 bytes: the first 6 bytes are header, length, and CRC; the remaining 76 bytes are payload. Assume the frame is stored in a C array `buf` as follows:

5A A5 4C 00 14 BB 91 08 15 23 09 A2 C4 47 08 15 1C 00 CC E8 61 BE 9A 35 56 3E 65 EA 72 3F 31 D0 7C BD 75 DD C5 BB 6B D7 24 BC 89 88 FC 40 01 00 6A 41 AB 2A 70 C2 96 D4 50 41 ED 03 43 41 41 F4 F4 C2 CC CA F8 BE 73 6A 19 BE F0 00 1C 3D 8D 37 5C 3F

Field name	Type	Raw value	Parsed value	Description
Header	/	5A A5	-	Frame header
Payload length	/	4C 00	76	Payload length = 76 bytes
CRC	/	14 BB	BB14	CRC value
tag	/	91	91	0x91 packet (payload starts from the next field)
Status (reserved)	uint16_t	08 15	5384	Reserved
temperature	int8_t	23	35	Temperature: °C
air_pressure	float	09 A2 C4 47	100676	Air pressure, Pa
system_time	uint32_t	08 15 1C 00	0x001C1508 = 1840392	Timestamp, ms
acc_b_x	float	CC E8 61 BE	-0.220615	Acceleration X, g
acc_b_y	float	9A 35 56 3E	0.209189	Acceleration Y, g
acc_b_z	float	65 EA 72 3F	0.948889	Acceleration Z, g
gyr_b_x	float	31 D0 7C BD	-0.061722	Angular rate X, dps
gyr_b_y	float	75 DD C5 BB	-0.00603836	Angular rate Y, dps
gyr_b_z	float	6B D7 24 BC	-0.0100611	Angular rate Z, dps
mag_b_x	float	89 88 FC 40	7.89167	Magnetic field X, uT
mag_b_y	float	01 00 6A 41	14.625	Magnetic field Y, uT
mag_b_z	float	AB 2A 70 C2	-60.0417	Magnetic field Z, uT
roll	float	96 D4 50 41	13.0519	Roll, deg
pitch	float	ED 03 43 41	12.1885	Pitch, deg
yaw	float	41 F4 F4 C2	-122.477	Yaw, deg
q_w	float	CC CA F8 BE	-0.485922	Quaternion W
q_x	float	73 6A 19 BE	-0.14982	Quaternion X
q_y	float	F0 00 1C 3D	0.0380868	Quaternion Y
q_z	float	8D 37 5C 3F	0.860223	Quaternion Z

3.6. C Parsing Code Example (HI91)

1. CRC check

```c
uint16_t payload_len;
uint16_t crc;
crc = 0;
payload_len = buf[2] + (buf[3] << 8);

/* calulate 5A A5 and LEN filed crc */
crc16_update(&crc, buf, 4);

/* calulate payload crc */
crc16_update(&crc, buf + 6, payload_len);
```

The CRC value is 0x516C, which matches the CRC carried in the frame; CRC check passes.

2. Define receive structure

The payload starts at `0x91`. Define the data structure and common conversion macros:

```c
#include "stdio.h"
#include "string.h"
/* common type conversion */
#define U1(p) (*((uint8_t *)(p)))
#define I1(p) (*((int8_t  *)(p)))
#define I2(p) (*((int16_t  *)(p)))
static uint16_t U2(uint8_t *p) {uint16_t u; memcpy(&u,p,2); return u;}
static uint32_t U4(uint8_t *p) {uint32_t u; memcpy(&u,p,4); return u;}
static int32_t  I4(uint8_t *p) {int32_t  u; memcpy(&u,p,4); return u;}
static float    R4(uint8_t *p) {float    r; memcpy(&r,p,4); return r;}
typedef struct
{
    uint8_t     tag;                /* item tag: 0x91        */
    float       acc[3];             /* acceleration          */
    float       gyr[3];             /* angular velocity      */
    float       mag[3];             /* magnetic field        */
    float       eul[3];             /* attitude: eular angle */
    float       quat[4];            /* attitude: quaternion  */
    float       pressure;           /* air pressure          */
    uint32_t    timestamp;
}imu_data_t;
```

3. Receive data (payload begins at buf[6] = 0x91)

```c
    imu_data_t i0x91 = {0};
    int offset = 6; /* payload strat at buf[6] */
    i0x91.tag =             U1(buf+offset+0);
    i0x91.pressure =        R4(buf+offset+4);
    i0x91.timestamp =       U4(buf+offset+8);
    i0x91.acc[0] =          R4(buf+offset+12);
    i0x91.acc[1] =          R4(buf+offset+16);
    i0x91.acc[2] =          R4(buf+offset+20);
    i0x91.gyr[0] =          R4(buf+offset+24);
    i0x91.gyr[1] =          R4(buf+offset+28);
    i0x91.gyr[2] =          R4(buf+offset+32);
    i0x91.mag[0] =          R4(buf+offset+36);
    i0x91.mag[1] =          R4(buf+offset+40);
    i0x91.mag[2] =          R4(buf+offset+44);
    i0x91.eul[0] =          R4(buf+offset+48);
    i0x91.eul[1] =          R4(buf+offset+52);
    i0x91.eul[2] =          R4(buf+offset+56);
    i0x91.quat[0] =         R4(buf+offset+60);
    i0x91.quat[1] =         R4(buf+offset+64);
    i0x91.quat[2] =         R4(buf+offset+68);
    i0x91.quat[3] =         R4(buf+offset+72);
```

4. Print received data

```c
    printf("%-16s0x%X\r\n",                 "tag:",             i0x91.tag);
    printf("%-16s%8.4f %8.4f %8.4f\r\n",    "acc(G):",          i0x91.acc[0], i0x91.acc[1], 
i0x91.acc[2]);
    printf("%-16s%8.3f %8.3f %8.3f\r\n",    "gyr(deg/s):",      i0x91.gyr[0], i0x91.gyr[1], 
i0x91.gyr[2]);
    printf("%-16s%8.3f %8.3f %8.3f\r\n",    "mag(uT):",         i0x91.mag[0], i0x91.mag[1], 
i0x91.mag[2]);
    printf("%-16s%8.3f %8.3f %8.3f\r\n",    "eul(deg):",        i0x91.eul[0], i0x91.eul[1], 
i0x91.eul[2]);
    printf("%-16s%8.3f %8.3f %8.3f %8.3f\r\n",    "quat:",            i0x91.quat[0], i0x91.quat[1], 
i0x91.quat[2], i0x91.quat[3]);
    printf("%-16s%8.3f\r\n",                      "presure(pa):",     i0x91.pressure);
    printf("%-16s%d\r\n",                         "timestamp(ms):",   i0x91.timestamp);
```

3.7. Maximum Throughput

Protocol	Bytes	9600 bps	115200 bps	230400 bps	256000 bps	460800 bps	921600 bps
91	76	10 Hz	100 Hz	250 Hz	250 Hz	500 Hz	1000 Hz
92	48	10 Hz	200 Hz	250 Hz	250 Hz	500 Hz	1000 Hz

4. RS-485 Output Protocol (Modbus)

- Supported interface: RS-485
- Default serial configuration: 115200 – N8N1
- Modbus:
  - RS-485 communication follows the Modbus RTU specification. Data is sent/received in registers. Each register is 2 bytes, big-endian (high byte first).
  - Write: 0x06 (Write Single Register): write one register (each Modbus register is 2 bytes)
  - Read: 0x03 (Read Holding Registers): read one or multiple registers
  - Custom function code: 0x50, used for automatic Modbus ID assignment (mass production deployment), firmware upgrade, etc.
- Modbus device address is configurable. Factory default: 80 (0x50)

4.1. Frame Format

4.1.1. Read registers (0x03)

Master request:

Field	Value	Description
ID	1–0xFF	Modbus device address
FUN_CODE	0x03	Function code
ADDR_H	-	Register address high 8 bits
ADDR_L	-	Register address low 8 bits
LEN_H	-	Length high 8 bits (number of registers)
LEN_L	-	Length low 8 bits (number of registers)
CRC_L	-	CRC low 8 bits
CRC_H	-	CRC high 8 bits

Slave (module) response:

Field	Value	Description
ID	1–0xFF	Modbus device address
FUN_CODE	0x03	Function code
LEN	-	Length of returned data in bytes (excluding ID, FUN_CODE, LEN, CRC)
DATAH	-	Returned data high 8 bits
DATAL	-	Returned data low 8 bits
----	-	Returned data high 8 bits
----	-	Returned data low 8 bits
CRC_L	-	CRC low 8 bits
CRC_H	-	CRC high 8 bits

4.1.2. Write register (0x06)

Field	Value	Description
ID	1–0xFF	Modbus device address
FUN_CODE	0x06	Function code
ADDR_H	-	Register address high 8 bits
ADDR_L	-	Register address low 8 bits
DATA_H	-	Data high 8 bits
DATA_L	-	Data low 8 bits
CRC_L	-	CRC low 8 bits
CRC_H	-	CRC high 8 bits

Slave response:

Field	Value	Description
ID	1–0xFF	Modbus device address
FUN_CODE	0x06	Function code
ADDR_H	-	Register address high 8 bits
ADDR_L	-	Register address low 8 bits
DATA_H	-	Data high 8 bits
DATA_L	-	Data low 8 bits
CRC_L	-	CRC low 8 bits
CRC_H	-	CRC high 8 bits

4.1.3. CRC

- Online CRC calculator: https://www.23bei.com/tool/59.html
- C code:

```c
static const uint16_t modbus_crc_table[256] = {
    0x0000, 0xc0c1, 0xc181, 0x0140, 0xc301, 0x03c0, 0x0280, 0xc241,
    0xc601, 0x06c0, 0x0780, 0xc741, 0x0500, 0xc5c1, 0xc481, 0x0440,
    0xcc01, 0x0cc0, 0x0d80, 0xcd41, 0x0f00, 0xcfc1, 0xce81, 0x0e40,
    0x0a00, 0xcac1, 0xcb81, 0x0b40, 0xc901, 0x09c0, 0x0880, 0xc841,
    0xd801, 0x18c0, 0x1980, 0xd941, 0x1b00, 0xdbc1, 0xda81, 0x1a40,
    0x1e00, 0xdec1, 0xdf81, 0x1f40, 0xdd01, 0x1dc0, 0x1c80, 0xdc41,
    0x1400, 0xd4c1, 0xd581, 0x1540, 0xd701, 0x17c0, 0x1680, 0xd641,
    0xd201, 0x12c0, 0x1380, 0xd341, 0x1100, 0xd1c1, 0xd081, 0x1040,
    0xf001, 0x30c0, 0x3180, 0xf141, 0x3300, 0xf3c1, 0xf281, 0x3240,
    0x3600, 0xf6c1, 0xf781, 0x3740, 0xf501, 0x35c0, 0x3480, 0xf441,
    0x3c00, 0xfcc1, 0xfd81, 0x3d40, 0xff01, 0x3fc0, 0x3e80, 0xfe41,
    0xfa01, 0x3ac0, 0x3b80, 0xfb41, 0x3900, 0xf9c1, 0xf881, 0x3840,
    0x2800, 0xe8c1, 0xe981, 0x2940, 0xeb01, 0x2bc0, 0x2a80, 0xea41,
    0xee01, 0x2ec0, 0x2f80, 0xef41, 0x2d00, 0xedc1, 0xec81, 0x2c40,
    0xe401, 0x24c0, 0x2580, 0xe541, 0x2700, 0xe7c1, 0xe681, 0x2640,
    0x2200, 0xe2c1, 0xe381, 0x2340, 0xe101, 0x21c0, 0x2080, 0xe041,
    0xa001, 0x60c0, 0x6180, 0xa141, 0x6300, 0xa3c1, 0xa281, 0x6240,
    0x6600, 0xa6c1, 0xa781, 0x6740, 0xa501, 0x65c0, 0x6480, 0xa441,
    0x6c00, 0xacc1, 0xad81, 0x6d40, 0xaf01, 0x6fc0, 0x6e80, 0xae41,
    0xaa01, 0x6ac0, 0x6b80, 0xab41, 0x6900, 0xa9c1, 0xa881, 0x6840,
    0x7800, 0xb8c1, 0xb981, 0x7940, 0xbb01, 0x7bc0, 0x7a80, 0xba41,
    0xbe01, 0x7ec0, 0x7f80, 0xbf41, 0x7d00, 0xbdc1, 0xbc81, 0x7c40,
    0xb401, 0x74c0, 0x7580, 0xb541, 0x7700, 0xb7c1, 0xb681, 0x7640,
    0x7200, 0xb2c1, 0xb381, 0x7340, 0xb101, 0x71c0, 0x7080, 0xb041,
    0x5000, 0x90c1, 0x9181, 0x5140, 0x9301, 0x53c0, 0x5280, 0x9241,
    0x9601, 0x56c0, 0x5780, 0x9741, 0x5500, 0x95c1, 0x9481, 0x5440,
    0x9c01, 0x5cc0, 0x5d80, 0x9d41, 0x5f00, 0x9fc1, 0x9e81, 0x5e40,
   0x5a00, 0x9ac1, 0x9b81, 0x5b40, 0x9901, 0x59c0, 0x5880, 0x9841,
   0x8801, 0x48c0, 0x4980, 0x8941, 0x4b00, 0x8bc1, 0x8a81, 0x4a40,
   0x4e00, 0x8ec1, 0x8f81, 0x4f40, 0x8d01, 0x4dc0, 0x4c80, 0x8c41,
   0x4400, 0x84c1, 0x8581, 0x4540, 0x8701, 0x47c0, 0x4680, 0x8641,
   0x8201, 0x42c0, 0x4380, 0x8341, 0x4100, 0x81c1, 0x8081, 0x4040
};

 uint16_t modbus_crc_calc(uint8_t *buf, uint16_t len)
{
    uint16_t crc = 0xFFFFU;
    uint8_t nTemp;

    while (len--)
    {
        nTemp = *buf++ ^ crc;
        crc >>= 8;
        crc  ^= modbus_crc_table[(nTemp & 0xFFU)];
    }

    return(crc);
}
```

4.2. Register List

Address (Hex)	Address (Dec)	Name	Type	Function	R/W	Description
0x00	0	CTRL	u16	Control	W	See the Modbus configuration section
0x04	4	UART1_BAUD	u16	Baud rate	R/W	Serial port baud rate
0x05	5	MD_ID	u16	Modbus ID	R/W	Modbus ID valid range: 1–128
0x06	6	HEADING_MODE	u16	Heading mode	R/W	0: 6-axis mode (relative heading, yaw = 0 at power-up). 1: 9-axis mode (mag fusion, absolute heading)
0x34	52	ACCX	i16	Acceleration X	R	Unit: g (1 g = gravity). Scale: 0.00048828
0x35	53	ACCY	i16	Acceleration Y	R	Unit: g. Scale: 0.00048828
0x36	54	ACCZ	i16	Acceleration Z	R	Unit: g. Scale: 0.00048828
0x37	55	GYRX	i16	Angular rate X	R	Unit: deg/s. Scale: 0.061035
0x38	56	GYRY	i16	Angular rate Y	R	Unit: deg/s. Scale: 0.061035
0x39	57	GYRZ	i16	Angular rate Z	R	Unit: deg/s. Scale: 0.061035
0x3A	58	MAGX	i16	Magnetic field X	R	Unit: uT. Scale: 0.030517
0x3B	59	MAGY	i16	Magnetic field Y	R	Unit: uT. Scale: 0.030517
0x3C	60	MAGZ	i16	Magnetic field Z	R	Unit: uT. Scale: 0.030517
0x3D	61	R_H	i32	Roll high 16 bits	R	Unit: deg. Scale: 0.001
0x3E	62	R_L	-	Roll low 16 bits	R	Unit: deg. Scale: 0.001
0x3F	63	P_H	i32	Pitch high 16 bits	R	Unit: deg. Scale: 0.001
0x40	64	P_L	-	Pitch low 16 bits	R	Unit: deg. Scale: 0.001
0x41	65	Y_H	i32	Yaw high 16 bits	R	Unit: deg. Scale: 0.001
0x42	66	Y_L	-	Yaw low 16 bits	R	Unit: deg. Scale: 0.001
0x43	67	TEMP	i16	Temperature	R	Unit: °C. Scale: 0.01
0x44	68	PRS_H	i32	Pressure high 16 bits	R	Unit: Pa. Scale: 0.01
0x45	69	PRS_L	-	Pressure low 16 bits	R	Unit: Pa. Scale: 0.01
0x46	70	Q0	u16	Quaternion QW	R	Quaternion. Scale: 0.0001
0x47	71	Q1	u16	Quaternion QX	R	Quaternion. Scale: 0.0001
0x48	72	Q2	u16	Quaternion QY	R	Quaternion. Scale: 0.0001
0x49	73	Q3	u16	Quaternion QZ	R	Quaternion. Scale: 0.0001
0x4A	74	INCLI_X	i16	Inclinometer X angle	R	Dual-axis products: X angle ±180 deg, scale 0.011 Single-axis products: X angle 0–360 deg, scale 0.011
0x4B	75	INCLI_Y	i16	Inclinometer Y angle	R	Dual-axis: Y angle ±90 deg, scale 0.011 Single-axis: reserved
0x4E	78	HEVAE	i16	Heave	R	Vessel heave displacement, unit m, scale 0.01
0x51	81	HEAVE_PERIOD	i16	Heave period	R	Heave period, unit s, scale 0.001

Address (Hex)	Address (Dec)	Name	Type	Function	R/W	Description
0x70-0x77	112-119	PNAME	u16	Device name	R	Device name string in ASCII, occupies 8 registers
0x78	120	SW_VERSION	u16	Software version	R	Software version
0x79	121	BL_VERSION	u16	Bootloader version	R	Bootloader version
0x7F-0x82	127-130	SN	u16	Unique serial number	R	Unique serial number, occupies 4 registers
0xA5	165	SET_LV	u16	Auto leveling	W	3: execute auto leveling once: if pitch/roll close to 0°,0° (flat, face up), calibrate to 0°,0°. If close to 0° or 180° (flat, face down), calibrate to 0°,180°. For robot environments. “Close to” means Pitch & Roll < 15°. 5: cancel auto leveling, restore absolute angle output. Other values: invalid
0xA6	166	URFR	u16	Mounting setting	W	0: level mounting (default) 1: vertical, +Y downward 2: vertical, +Y upward 3: vertical, +X upward 4: vertical, +X downward

4.3. Common Configuration Examples

All examples below assume Modbus ID = 0x50 (factory default). If the Modbus ID has been changed, update the ID field and CRC accordingly.

4.3.1. Control register (0x00)

Command	Value written to CTRL	Command (Hex) ID=0x50 (factory default)
Save all configuration to Flash	0x0000	50 06 00 00 00 00 84 4B
Restore factory defaults	0x0001	50 06 00 00 00 01 45 8B
Reset	0x00FF	50 06 00 00 00 FF C4 0B

4.3.2. Configure baud rate (0x04)

Target baud rate	Command (Hex) ID=0x50 (factory default)
4800	50 06 00 04 00 00 C5 8A
9600	50 06 00 04 00 01 04 4A
19200	50 06 00 04 00 02 44 4B
38400	50 06 00 04 00 03 85 8B
57600	50 06 00 04 00 04 C4 49
115200	50 06 00 04 00 05 05 89
230400	50 06 00 04 00 06 45 88
460800	50 06 00 04 00 07 84 48
921600	50 06 00 04 00 08 C4 4C

4.3.3. Configure node ID (0x05)

[CUR_ID] 06 00 05 00 [NEW_ID] CRC(2 bytes)

- CUR_ID: current Modbus ID
- NEW_ID: new ID to set

Examples (current ID = 0x50):

- Set NEW_ID to 0x50: 50 06 00 05 00 50 94 76
- Set NEW_ID to 0x51: 50 06 00 05 00 51 55 B6
- Set NEW_ID to 0x52: 50 06 00 05 00 52 15 B7
- Set NEW_ID to 0x53: 50 06 00 05 00 53 D4 77

Note: After a successful change, the Modbus address takes effect immediately. Update the CUR_ID field in subsequent Modbus requests. If you are not familiar with Modbus, it is recommended to use the PC host software.

4.3.4. Set mounting orientation (0xA6)

Target mounting	Command (Hex) ID=0x50 (factory default)
0: level mounting (default)	50 06 00 A6 00 00 64 68
1: vertical, +Y downward	50 06 00 A6 00 01 A5 A8
2: vertical, +Y upward	50 06 00 A6 00 02 E5 A9
3: vertical, +X upward	50 06 00 A6 00 03 24 69
4: vertical, +X downward	50 06 00 A6 00 04 65 AB

4.3.5. Leveling (0xA5)

- Enable auto leveling: 50 06 00 A5 00 02 15 A9
- Cancel auto leveling: 50 06 00 A5 00 05 54 6B

4.3.6. Set 6-axis or 9-axis mode (0x06)

- Set to 6-axis mode: 50 06 00 06 00 00 64 4A
- Set to 9-axis mode: 50 06 00 06 00 01 A5 8A

4.4. Read module version information (0x70–0x82)

Read product name, software version, and SN:

Request frame `50 03 00 70 00 14 49 9F`

Field	Value	Description
Device address	0x50	Module address
Function code	0x03	Read holding registers
Start address	0x0070	Product info start address
Length	0x0014	Read 20 registers
CRC	0x9F49	-

Response frame: `50 03 28 48 49 31 34 52 32 4E 2D 34 38 35 2D 30 30 30 00 00 98 00 6B 00 00 00 00 00 00 00 00 00 00 04 7D 95 5F 8D 2A 17 08 00 00 4D 0C`

Field	Data	Description
Product name	48 49...30 30	CH10x(M)
Software version	0x98	V1.52
Bootloader version	0x6B	V1.07
Serial number	047D955F8D2A1708	SN

4.5. Read sensor data (0x34–0x4B)

Request frame `50 03 00 34 00 18 09 8F`

Field	Value	Description
Device address	0x50	Module address
Function code	0x03	Read holding registers
Start address	0x0034	Sensor data start address
Length	0x0018	Read 24 registers
CRC	0x8F09	-

Response frame: `50 03 30 FF 01 03 B0 06 50 FC C9 FF 7C 00 91 01 D5 FD DB FD 27 00 00 21 FF 00 00 7F F6 FF FD 73 E7 00 00 00 00 00 00 10 A6 0D 59 DD 4E 86 A8 06 30 17 82 1E CE`

Acceleration (unit: g, can be converted using 9.8 m/s^(2)):

Axis	Register value (HEX)	Raw value (DEC)	Scale	Physical value
X	FF 01	-255	0.00048828	-0.1245
Y	03 B0	944	0.00048828	0.4609
Z	06 50	1616	0.00048828	0.7891

Angular rate (unit: deg/s)

Axis	Register value (HEX)	Raw value (DEC)	Scale	Physical value
X	FC C9	-823	0.061035	-50.2318
Y	FF 7C	-132	0.061035	-8.0566
Z	00 91	145	0.061035	8.8501

Magnetic field (unit: uT)

Axis	Register value (HEX)	Raw value (DEC)	Scale	Physical value
X	01 D5	469	0.030517	14.3125
Y	FD DB	-549	0.030517	-16.7538
Z	FD 27	-729	0.030517	-22.2469

Euler angles (unit: deg)

Axis	Register value (HEX)	Raw value (DEC)	Scale	Physical value
Roll (Roll)	00 00 21 FF	8703	0.001	8.703
Pitch (Pitch)	00 00 7F F6	32758	0.001	32.758
Yaw (Yaw)	FF FD 73 E7	-166937	0.001	-166.937

5. CAN Data Protocol (CANopen)

The CAN interface complies with the CANopen protocol. All communication uses **standard data frames** and transmits data via **TPDO1–TPDO7**. The device **does not receive or send Remote Frames** and **does not use Extended Frames**. All TPDOs use an **asynchronous time-triggered** transmission mode.

5.1. CANopen Default Settings

Default Item	Value
CAN baud rate	500 kbit/s
Node ID	8
Initial state	Operational
TPDO output rate	1 Hz – 200 Hz (per TPDO)

5.2. CANopen TPDOs

Channel	COB-ID (Frame ID)	DLC	Transmission type	Output rate (Hz)	Data	Description
TPDO1	0x180 + ID	6	Asynchronous time (0xFE)	100	Acceleration	Type: **int16**, **little-endian**. 2 bytes per axis, total 6 bytes. Order: X, Y, Z acceleration. Unit: **mG (0.001 G)**.
TPDO2	0x280 + ID	6	Asynchronous time (0xFE)	100	Angular rate	Type: **int16**, **little-endian**. 2 bytes per axis, total 6 bytes. Order: X, Y, Z angular rate. Unit: **0.1 dps (°/s)**.
TPDO3	0x380 + ID	6	Asynchronous time (0xFE)	100	Euler angles	Type: **int16**, **little-endian**. 2 bytes per axis, total 6 bytes. Order: Roll, Pitch, Yaw. Unit: **0.01°**.
TPDO4	0x480 + ID	8	Asynchronous time (0xFE)	100	Quaternion	Type: **int16**, **little-endian**. 2 bytes per element, total 8 bytes. Order: q_w q_x q_y q_z. The unit quaternion is scaled by **10000** before output. For example, quaternion (1,0,0,0) outputs (10000,0,0,0).
TPDO6	0x680 + ID	4	Asynchronous time (0xFE)	20	Pressure	Type: **int32**, total 4 bytes. Unit: **Pa**.
TPDO7	0x780 + ID	8	Asynchronous time (0xFE)	100	Inclinometer angles	Type: **int32**, **little-endian**. 4 bytes per axis, total 8 bytes. Order: X, Y. Unit: **0.01°**.

Example decoding (Acceleration and Angular Rate)

Acceleration CAN frame: ID=0x188, DATA = 4A 00 1F 00 C8 03

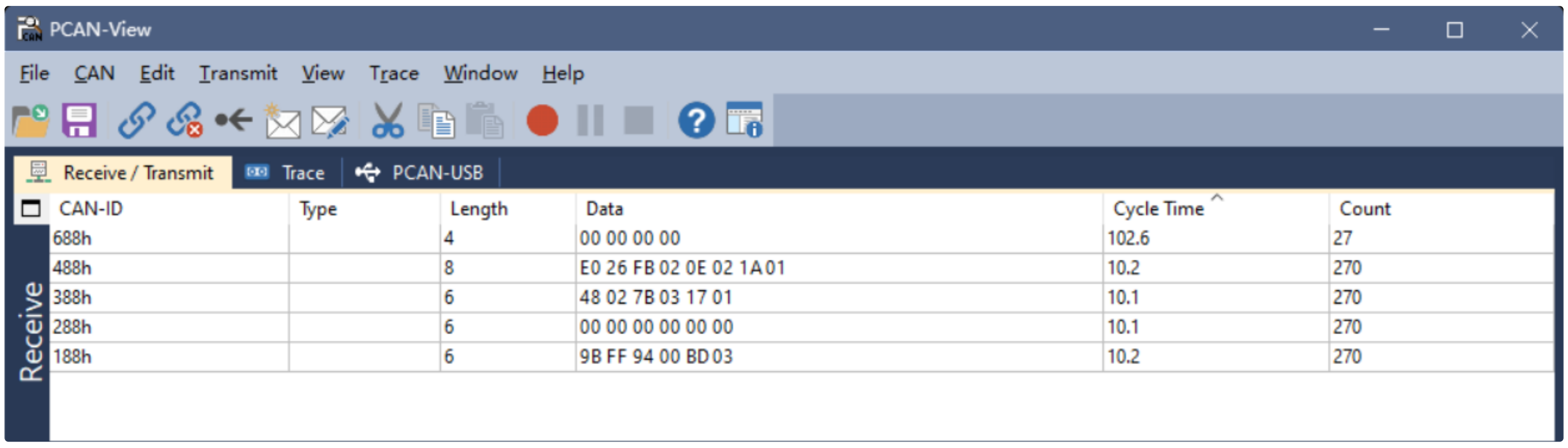
- ID=0x188: acceleration frame sent by the device with node ID = 8
- Acc X = 0x004A = 74 → **74 mG**
- Acc Y = 0x001F = 31 → **31 mG**
- Acc Z = 0x03C8 = 968 → **968 mG**

Angular rate CAN frame: ID=0x288 · DATA = 15 00 14 01 34 00

- ID=0x288: angular-rate frame sent by the device with node ID = 8
- Gyro X = 0x0015 = 21 → **2.1 dps**
- Gyro Y = 0x0114 = 276 → **27.6 dps**
- Gyro Z = 0x0034 = 52 → **5.2 dps**

5.3. Connecting to the CAN Device Using PC Software

Using the **PCAN-View** tool together with PCAN, you can display received CAN messages and frame rate in the Rx Message window, as shown below:



CAN-ID	Type	Length	Data	Cycle Time	Count
688h		4	00 00 00 00	102.6	27
488h		8	E0 26 FB 02 0E 02 1A 01	10.2	270
388h		6	48 02 7B 03 17 01	10.1	270
288h		6	00 00 00 00 00 00	10.1	270
188h		6	9B FF 94 00 BD 03	10.2	270

5.4. Configuration Commands (SDO Protocol)

All configuration commands use **Expedited SDO** writes. After any configuration changes, you must send the **Save-to-Flash** command to store the parameters in Flash.

5.4.1. SDO (Service Data Object) Protocol

Expedited SDO format:

Host sends an SDO command to the slave:

CAN_ID	CS (1B)	Object index (2B)	Sub-index (1B)	Data (4B)
0x600+ID	0x23 (write 4B)	little-endian	sub-index	data, little-endian

Slave replies to the host:

CAN_ID	SDO cmd (1B)	Object index (2B)	Sub-index (1B)	Data (4B)
0x580+ID	0x60 (write OK response)	little-endian	sub-index	reserved

Change Node ID (0x20A0)

`ID=0x608 · DATA=23,A0,20,00,[ID],00,00,00`

Node ID range: **1–127**. After changing the ID, you must **save to Flash** and **reset (or power-cycle)** for it to take effect.

Save Configuration to Flash (0x2000)

`ID=0x608 · DATA=23,00,20,00,00,00,00,00`

Reset (0x2000)

`ID=0x608 · DATA=23,00,20,00,FF,00,00,00`

Restore Factory Defaults (0x2000)

`ID=0x608 · DATA=23 00 20 00 01 00 00 00`

Restores all parameters (including baud rate, node ID, etc.) to factory defaults. Takes effect after power-cycle. Use with caution.

Change CAN Baud Rate (0x209A)

`ID=0x608 · DATA=23,9A,20,00,[ID]`

After changing, you must **save to Flash** and **reset (or power-cycle)** for it to take effect.

- Set CAN baud rate to **1000 kbit/s**: `ID=0x608 · DATA=23,9A,20,00,00,00,00,00`
- Set CAN baud rate to **500 kbit/s**: `ID=0x608 · DATA=23,9A,20,00,02,00,00,00`
- Set CAN baud rate to **250 kbit/s**: `ID=0x608 · DATA=23,9A,20,00,03,00,00,00`
- Set CAN baud rate to **125 kbit/s**: `ID=0x608 · DATA=23,9A,20,00,04,00,00,00`

All configuration operations below write the CANopen object dictionary via expedited SDO. The TPDO channels and their corresponding communication parameter index are:

Channel	COB-ID (Frame ID)	Parameter index	Description
TPDO1	0x180+ID	0x1800	Acceleration
TPDO2	0x280+ID	0x1801	Angular rate
TPDO3	0x380+ID	0x1802	Euler angles
TPDO4	0x480+ID	0x1803	Quaternion
TPDO6	0x680+ID	0x1804	Pressure
TPDO7	0x780+ID	0x1805	Inclinometer output

Change / Disable / Enable Output Rate (0x1800–0x1805)

This setting takes effect immediately.

- ID=0x608 · DATA=2B,00,18,05,00,00,00,00 Disable acceleration output (1800.5 = 0)
- ID=0x608 · DATA=2B,00,18,05,05,00,00,00 Acceleration 200 Hz (1800.5 = 5)
- ID=0x608 · DATA=2B,00,18,05,0A,00,00,00 Acceleration 100 Hz (1800.5 = 10)
- ID=0x608 · DATA=2B,00,18,05,14,00,00,00 Acceleration 50 Hz (1800.5 = 20)
- ID=0x608 · DATA=2B,00,18,05,32,00,00,00 Acceleration 20 Hz (1800.5 = 50)
- ID=0x608 · DATA=2B,00,18,05,64,00,00,00 Acceleration 10 Hz (1800.5 = 100)
- ID=0x608 · DATA=2B,01,18,05,00,00,00,00 Disable angular rate output (1801.5 = 0)
- ID=0x608 · DATA=2B,01,18,05,05,00,00,00 Angular rate 200 Hz (1801.5 = 5)
- ID=0x608 · DATA=2B,01,18,05,0A,00,00,00 Angular rate 100 Hz (1801.5 = 10)
- ID=0x608 · DATA=2B,01,18,05,14,00,00,00 Angular rate 50 Hz (1801.5 = 20)
- ID=0x608 · DATA=2B,01,18,05,32,00,00,00 Angular rate 20 Hz (1801.5 = 50)
- ID=0x608 · DATA=2B,01,18,05,64,00,00,00 Angular rate 10 Hz (1801.5 = 100)
- ID=0x608 · DATA=2B,02,18,05,00,00,00,00 Disable Euler angle output (1802.5 = 0)
- ID=0x608 · DATA=2B,02,18,05,05,00,00,00 Euler angles 200 Hz (1802.5 = 5)
- ID=0x608 · DATA=2B,02,18,05,0A,00,00,00 Euler angles 100 Hz (1802.5 = 10)
- ID=0x608 · DATA=2B,02,18,05,14,00,00,00 Euler angles 50 Hz (1802.5 = 20)
- ID=0x608 · DATA=2B,02,18,05,32,00,00,00 Euler angles 20 Hz (1802.5 = 50)
- ID=0x608 · DATA=2B,02,18,05,64,00,00,00 Euler angles 10 Hz (1802.5 = 100)
- ID=0x608 · DATA=2B,03,18,05,00,00,00,00 Disable quaternion output (1803.5 = 0)
- ID=0x608 · DATA=2B,03,18,05,05,00,00,00 Quaternion 200 Hz (1803.5 = 5)
- ID=0x608 · DATA=2B,03,18,05,0A,00,00,00 Quaternion 100 Hz (1803.5 = 10)
- ID=0x608 · DATA=2B,03,18,05,14,00,00,00 Quaternion 50 Hz (1803.5 = 20)
- ID=0x608 · DATA=2B,03,18,05,32,00,00,00 Quaternion 20 Hz (1803.5 = 50)
- ID=0x608 · DATA=2B,03,18,05,64,00,00,00 Quaternion 10 Hz (1803.5 = 100)
- ID=0x608 · DATA=2B,04,18,05,00,00,00,00 Disable pressure output (1804.5 = 0)
- ID=0x608 · DATA=2B,04,18,05,05,00,00,00 Pressure 200 Hz (1804.5 = 5)
- ID=0x608 · DATA=2B,04,18,05,0A,00,00,00 Pressure 100 Hz (1804.5 = 10)
- ID=0x608 · DATA=2B,04,18,05,14,00,00,00 Pressure 50 Hz (1804.5 = 20)
- ID=0x608 · DATA=2B,04,18,05,32,00,00,00 Pressure 20 Hz (1804.5 = 50)
- ID=0x608 · DATA=2B,04,18,05,64,00,00,00 Pressure 10 Hz (1804.5 = 100)

Example: Set TPDO1 (acceleration) output rate to 100 Hz (one output every 10 ms).
0x2B indicates an expedited SDO write of 2 bytes. 0x00, 0x18 means index 0x1800, and 0x05 is the sub-index. 0x00, 0x0A = (0x00<<8) + 0x0A = 10 (unit: ms). Pad remaining bytes with 0.

Set Inclinometer Output Sign (0x209E)

- ID=0x608 · DATA=23,9E,20,00,00,00,00,00 X-axis sign uses factory default direction
- ID=0x608 · DATA=23,9E,20,00,01,00,00,00 Invert X-axis sign
- ID=0x608 · DATA=23,9F,20,00,00,00,00,00 Y-axis sign uses factory default direction
- ID=0x608 · DATA=23,9F,20,00,01,00,00,00 Invert Y-axis sign

Set Inclinometer Zero Offset (0x20A5)

- ID=0x608 · DATA=23,A5,20,00,02,00,00,00 After writing, the current orientation is set as zero output (X=0, Y=0)
- ID=0x608 · DATA=23,A5,20,00,05,00,00,00 After writing, cancel zero-offset configuration and output true X/Y angles (equivalent to X/Y offset = 0)

5.4.2. Synchronization

In compliance with CANopen, the module can configure TPDOs to **synchronous mode**: it stops asynchronous time-triggered transmission and waits for the CANopen **SYNC** frame. When a SYNC frame arrives, it transmits one TPDO frame.

Configure a TPDO for Synchronous Mode

Set the desired TPDO's transmission type via the TPDO communication parameter object **[0x180x.2] (Transmission type)** to **0x01** for synchronous mode. Refer to the CANopen specification for the detailed meaning. Example for TPDO1 (acceleration):

ID=0x608 · DATA=2F,00,18,02,01,00,00,00 Write [0x1800.2 (US8)] = 1 to set TPDO1 to synchronous mode

ID=0x608 · DATA=2F,00,18,02,FF,00,00,00 Write [0x1800.2 (US8)] = 0xFF to set TPDO1 to asynchronous mode (factory default)

Send a CANopen SYNC Frame

Send a CANopen SYNC frame: ID:80 · DATA:Empty
After receiving the SYNC frame, the module transmits one frame for each TPDO configured in synchronous mode, achieving synchronization.

Set Heartbeat Producer Time

Configure the heartbeat via **[0x1017.0 (US16)]**. Valid range: 0–65535, unit: ms. A value of 0 disables heartbeat.

ID=0x608 · DATA=2B,17,10,00,64,00,00,00 Set heartbeat period to 100 ms.

6. CAN Data Protocol (J1939)

The module's default CAN output protocol is **CANopen**. If you require the **SAE J1939** protocol, please contact our company.

Item	Description
Communication mode	Broadcast
Default transmit interval	100 ms
Data length	8 bytes per PGN
PF (PDU format)	0xFF
PS (PDU specific)	When PF > 0xF0, PS is the **Group Extension (GE)** of the PGN; otherwise PS is the **Destination Address (DA)**
Priority	3
Default J1939 source address	0x08
Data format	All frames use **LSB-first**. Unless otherwise specified, values are **signed integers**.

6.0.3. PGN 65327 (0xFF2F) Time Information

CANID = `0x0CFF2F08`

SPN Name	SPN Byte Position	Description
UTC Year	0	20 represents 2020, and so on. If UTC time cannot be obtained, this byte is 20
UTC Month	1	If UTC month cannot be obtained, this byte is 0
UTC Day	2	If UTC date cannot be obtained, this byte is 0
UTC Hour	3	
UTC Minute	4	
UTC Second	5	
UTC Millisecond	6–7	Unit: ms, scale factor: 1

6.0.4. PGN 65332 (0xFF34) Acceleration

CANID = `0x0CFF3408`

Name	Byte position	Description
Acceleration X	0–1	Unit: **g** (1 g = standard gravity), scale: **0.00048828**
Acceleration Y	2–3	Unit: **g** (1 g = standard gravity), scale: **0.00048828**
Acceleration Z	4–5	Unit: **g** (1 g = standard gravity), scale: **0.00048828**
Reserved	6–7	-

6.0.5. PGN 65335 (0xFF37) Angular Rate

CANID = `0x0CFF3708`

Name	Byte position	Description
Angular rate X	0–1	Unit: **deg/s**, scale: **0.061035**
Angular rate Y	2–3	Unit: **deg/s**, scale: **0.061035**
Angular rate Z	4–5	Unit: **deg/s**, scale: **0.061035**
Reserved	6–7	-

6.0.6. PGN 65341 (0xFF3D) Pitch & Roll

CANID = `0x0CFF3D08`

SPN name	Byte position	Description
Roll	0–3	Unit: °, scale: **0.001**
Pitch	4–7	Unit: °, scale: **0.001**

6.0.7. PGN 65345 (0xFF41) Yaw / Heading

CANID = `0x0CFF4108`

SPN name	Byte position	Description
Yaw (Heading)	0–3	Range: **0–360**; unit: °; scale: **0.001**; **clockwise is positive**
Reserved	4–7	-

6.0.8. PGN 65354 (0xFF4A) Inclinometer Output

CANID = `0x0CFF4A08` *(only for inclinometer products outputting J1939)*

Name	Byte position	Description
X tilt angle	0–3	Range: **0–360** or **±180**; unit: **deg**; scale: **0.001**
Y tilt angle	4–7	Range: **0–360** or **±90**; unit: **deg**; scale: **0.001**

6.1. Configuration Commands

6.1.1. Command Format

Host sends: `ADDR + CMD + STATUS + VAL`
Device responds: `ADDR + CMD + STATUS + VAL`

Field	Size (bytes)	Description
ADDR	2	Register address
CMD	1	`0x06` : write, `0x03` : read
STATUS	1	Reserved
VAL	4	Write: value to write; Read: reserved

6.1.2. Configure the Module

29-bit extended ID	Data	Description	Notes
0x0CEF08xx	`34 01 06 00 [VAL]`	VAL: 4 bytes	PGN FF34 (acceleration) transmit interval in **ms**, range: **5–1000**
0x0CEF08xx	`37 01 06 00 [VAL]`	VAL: 4 bytes	PGN FF37 (angular rate) transmit interval in **ms**, range: **5–1000**
0x0CEF08xx	`3D 01 06 00 [VAL]`	VAL: 4 bytes	PGN FF3D (roll/pitch) transmit interval in **ms**, range: **5–1000**
0x0CEF08xx	`41 01 06 00 [VAL]`	VAL: 4 bytes	PGN FF41 (yaw) transmit interval in **ms**, range: **5–1000**
0x0CEF08xx	`4A 01 06 00 [VAL]`	VAL: 4 bytes	PGN FF4A (inclinometer) transmit interval in **ms**, range: **5–1000**
0x0CEF08xx	`9D 00 06 00 01 00 00 00`	-	Globally **enable** node data output
0x0CEF08xx	`9D 00 06 00 00 00 00 00`	-	Globally **disable** node data output *(default)*
0x0CEF08xx	`00 00 06 00 00 00 00 00`	-	Save all configuration parameters to **Flash**
0x0CEF08xx	`00 00 06 00 01 00 00 00`	-	Restore factory defaults
0x0CEF08xx	`00 00 06 00 FF 00 00 00`	-	Reset
0x0CEF08xx	`9A 00 06 00 [VAL]`	VAL: 4 bytes	Set CAN baud rate *(save + reset required)* : `0` =1000K, `1` =800K, `2` =500K, `3` =250K, `4` =125K
0x0CEF08xx	`9C 00 06 00 [VAL]`	VAL: 4 bytes	Set J1939 node ID: **1–128**
0x0CEF08xx	`A5 00 06 00 [VAL]`	VAL: 4 bytes	Set zero position: `0x02` =set current position as zero; `0x05` =cancel zeroing and output absolute physical angle
0x0CEF08xx	`9E 00 06 00 [VAL]`	VAL: 4 bytes	Set X-axis direction: `0` =default, `1` =invert
0x0CEF08xx	`9F 00 06 00 [VAL]`	VAL: 4 bytes	Set Y-axis direction: `0` =default, `1` =invert

In the ID field, `xx` is the J1939 source address and may be any byte.
In the data field, `xx` means any byte.
Example: `ID=0x0CEF0855, DATA = 37 01 06 00 64 00 00 00`
Set PGN FF37 to a 100 ms period (10 Hz).

7. CAN Data Protocol (NMEA2000)

Custom protocol. If you need NMEA2000 protocol, please contact us.

8. Magnetometer Calibration

8.1. Calibration Prerequisites

Using **9-axis mode** (magnetometer-assisted absolute heading) requires:

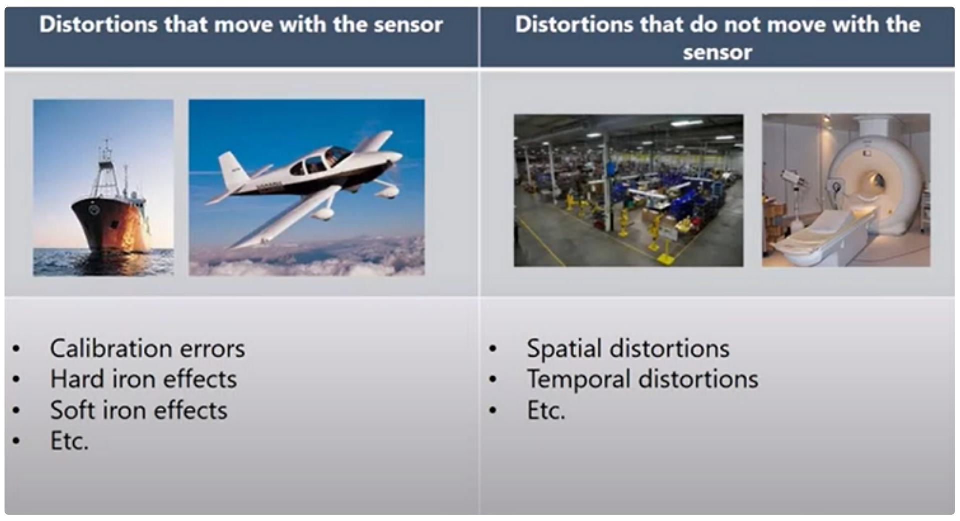
1. The first time you use 9-axis mode, you must perform **at least one user magnetometer calibration**.
2. During operation there must be **no spatial magnetic-field disturbance** (this is difficult to guarantee indoors where magnetic environments are complex).

Only when **both** conditions are met can the heading accuracy in 9-axis mode reach the specification stated in the manual.

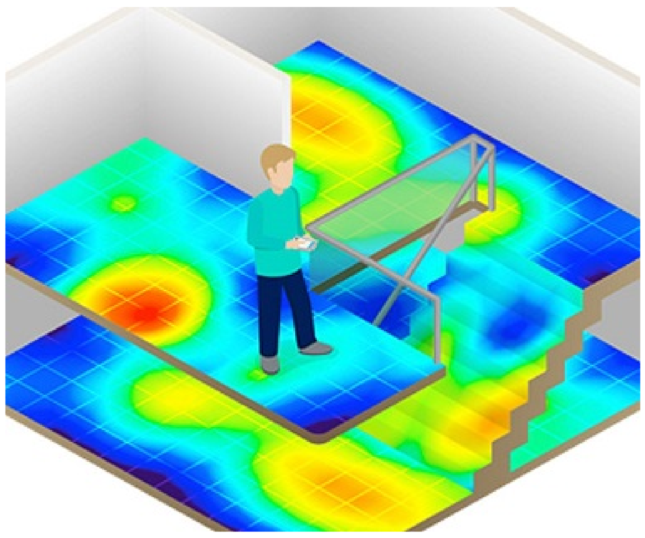
8.1.1. User Magnetometer Calibration

When you first use the module and need AHRS (9-axis) mode, perform the following:

1. Switch the module to **9-axis mode**.
2. In an environment with minimal magnetic interference (preferably outdoors), move the module (and any rigidly attached structure) slowly through as many orientations as possible. A slow **figure-8 motion** for **1–2 minutes** is recommended.
3. Use `LOG MAGCONFIG` to check that calibration has completed (for example, `MAG_BIAS=Y` indicates that the magnetometer bias parameters are available).



Magnetic disturbance type	**Distortion that moves with the sensor (hard-iron / soft-iron distortion)**	**Spatial magnetic-field disturbance (does not move with the sensor)**
Characteristics	The disturbance source moves together with the sensor	The disturbance source does **not** move with the sensor
Typical sources	PCB mounted with the module, metallic enclosure, UAV/robot structure rigidly attached to the module	Furniture, home appliances, cables, rebar in buildings, etc. (cannot move with the sensor). Also humanoid robots: the robot itself acts as a magnetic rigid body whose field distribution changes while walking.
Can it be calibrated out?	**Yes**	**No**
Mitigation	Can be eliminated through user magnetometer calibration	Avoid the disturbed region; otherwise heading will have significant error. Spatial disturbances are especially severe indoors near desks/chairs/appliances. The figure below shows a typical indoor spatial magnetic disturbance map: blue = weak disturbance, red = strong disturbance.



8.1.2. About the Module's Magnetometer Calibration Algorithm

The module includes an **active magnetometer calibration system**. No user command is required. The system automatically collects magnetometer data over a period of time in the background, analyzes it, rejects outliers, and estimates calibration parameters.

For automatic calibration to work, the module must experience **sufficient attitude changes** (slowly rotate through as many orientations as possible). Calibration **cannot** be performed while the module remains stationary.

The module also provides an interface for checking the current calibration status. Automatic calibration requires that the module has sufficient attitude variation during the sampling window.

8.2. Configuring Magnetometer Calibration (Manual / One-time Calibration)

By default, when set to 9-axis mode the module continuously attempts to estimate magnetometer calibration parameters in the background. The product also provides APIs to enable/disable this behavior:

- `CONFIG IMU EN_MCAL 1` : Enable real-time background magnetometer calibration (factory default).
- `CONFIG IMU EN_MCAL 0` : Disable real-time background magnetometer calibration.

If you want to perform magnetometer calibration **once** and then prevent further background calibration attempts, do the following:

1. In a low-interference environment (outdoors), move the module (including any rigidly attached carrier such as a robot/UAV) slowly in a figure-8 motion for 1–2 minutes to complete an automatic calibration. Use `LOG MAGCONFIG` to confirm calibration is complete.
2. Enter `CONFIG IMU EN_MCAL 0` and then `SAVECONFIG` to save the setting to non-volatile memory. After that, even after reboot, the module will no longer attempt automatic magnetometer calibration in the background.

8.3. Emphasis (Very Important)

Indoors, spatial magnetic-field disturbances can be extremely severe, and **cannot** be eliminated by calibration. Although the module includes homogeneous-field detection and masking mechanisms, magnetometer-assisted heading can still degrade significantly indoors. If the indoor magnetic environment is poor (e.g., near server rooms, laboratories, workshops, underground parking garages, etc.), the heading accuracy after calibration may be worse than 6-axis mode and may even exhibit large errors.

The module's automatic magnetometer calibration system can only compensate for **fixed** disturbances that are rigidly attached to the module installation. If the disturbance changes (e.g., the module moves relative to the carrier after calibration), calibration must be performed again.

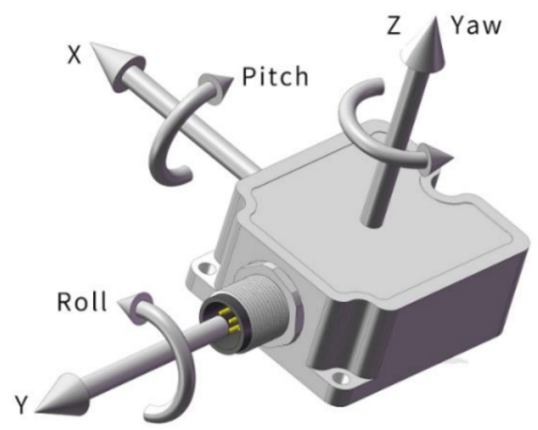
9. Appendix 1 Quaternion / Euler Angles / Rotation Matrix Conversions

9.1. Quaternion to Rotation Matrix

Given quaternion $[Q_{b2n} = [q_0, q_1, q_2, q_3]^T]$, the direction cosine matrix is:

$$C_{b2n} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

9.2. Quaternion to Euler Angles – ENU-312 (rotate about Z, then X, then Y)



Given quaternion $Q_{b2n} = [q_0, q_1, q_2, q_3]^T$, where q_0 is the scalar part and $[q_1, q_2, q_3]$ is the vector part. Q_{b2n} represents the rotation quaternion from frame **b** to frame **n**, where:

- pitch (θ): rotation about the X-axis, range $[-\frac{\pi}{2}, \frac{\pi}{2}]$
- roll (ϕ): rotation about the Y-axis, range $[-\pi, \pi]$
- yaw (ψ): rotation about the Z-axis, range $[-\pi, \pi]$

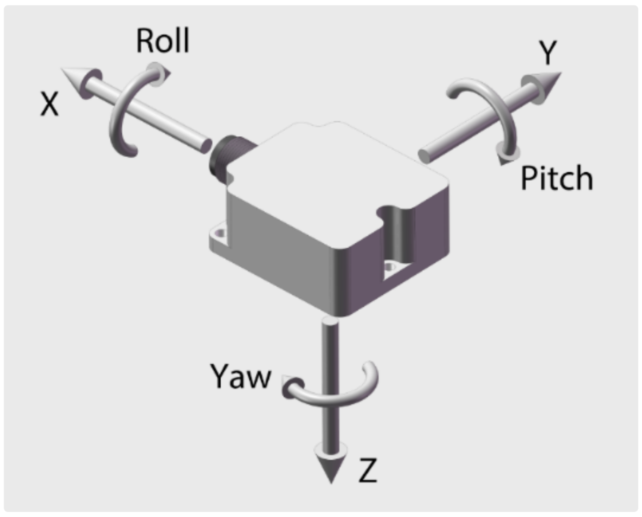
Quaternion → Euler angles:

$$pitch = \arcsin(2(q_0q_1 + q_2q_3))$$
$$roll = -\arctan 2(2(q_1q_3 - q_0q_2), q_0^2 - q_1^2 - q_2^2 + q_3^2)$$
$$yaw = -\arctan 2(2(q_1q_2 - q_0q_3), q_0^2 - q_1^2 + q_2^2 - q_3^2)$$

Euler angles → quaternion:

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\text{pitch}}{2})\cos(\frac{\text{roll}}{2})\cos(\frac{\text{yaw}}{2}) - \sin(\frac{\text{pitch}}{2})\sin(\frac{\text{roll}}{2})\sin(\frac{\text{yaw}}{2}) \\ \cos(\frac{\text{roll}}{2})\cos(\frac{\text{yaw}}{2})\sin(\frac{\text{pitch}}{2}) - \cos(\frac{\text{pitch}}{2})\sin(\frac{\text{roll}}{2})\sin(\frac{\text{yaw}}{2}) \\ \cos(\frac{\text{pitch}}{2})\cos(\frac{\text{yaw}}{2})\sin(\frac{\text{roll}}{2}) + \cos(\frac{\text{roll}}{2})\sin(\frac{\text{pitch}}{2})\sin(\frac{\text{yaw}}{2}) \\ \cos(\frac{\text{pitch}}{2})\cos(\frac{\text{roll}}{2})\sin(\frac{\text{yaw}}{2}) + \sin(\frac{\text{pitch}}{2})\sin(\frac{\text{roll}}{2})\cos(\frac{\text{yaw}}{2}) \end{bmatrix}$$

9.3. Quaternion to Euler Angles – NED-321 (rotate about Z, then Y, then X)



Given quaternion $Q_{b2n} = [q_0, q_1, q_2, q_3]^T$, where q_0 is the scalar part and $[q_1, q_2, q_3]$ is the vector part. Q_{b2n} represents the rotation quaternion from frame **b** to frame **n**, where:

- pitch (θ): rotation about the Y-axis, range $[-\frac{\pi}{2}, \frac{\pi}{2}]$
- roll (ϕ): rotation about the X-axis, range $[-\pi, \pi]$
- yaw (ψ): rotation about the Z-axis, range $[-\pi, \pi]$

Quaternion → Euler angles:

$$\begin{gathered} roll = \arctan 2(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)) \\ pitch = \arcsin(2(q_0q_2 - q_1q_3)) \\ yaw = \arctan 2(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)) \end{gathered}$$

Euler angles → quaternion:

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\text{roll}}{2})\cos(\frac{\text{pitch}}{2})\cos(\frac{\text{yaw}}{2}) + \sin(\frac{\text{roll}}{2})\sin(\frac{\text{pitch}}{2})\sin(\frac{\text{yaw}}{2}) \\ \sin(\frac{\text{roll}}{2})\cos(\frac{\text{pitch}}{2})\cos(\frac{\text{yaw}}{2}) - \cos(\frac{\text{roll}}{2})\sin(\frac{\text{pitch}}{2})\sin(\frac{\text{yaw}}{2}) \\ \cos(\frac{\text{roll}}{2})\sin(\frac{\text{pitch}}{2})\cos(\frac{\text{yaw}}{2}) + \sin(\frac{\text{roll}}{2})\cos(\frac{\text{pitch}}{2})\sin(\frac{\text{yaw}}{2}) \\ \cos(\frac{\text{roll}}{2})\cos(\frac{\text{pitch}}{2})\sin(\frac{\text{yaw}}{2}) - \sin(\frac{\text{roll}}{2})\sin(\frac{\text{pitch}}{2})\cos(\frac{\text{yaw}}{2}) \end{bmatrix}$$