CH10X/HI14 Configuration Manual

Applicable products: CH0X0 / HI14 / CH10X series (V1.3.7)

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

Attribute: public

Purchase and support: sltech@ms28.hinet.net

1. User Configuration Overview

The default configuration of the product can meet the needs of most users. Therefore, you need to read this chapter carefully before using the product and determine whether user configuration is required based on your own usage needs.

1.1. Geomagnetic auxiliary scene (AHRS/9-axis mode)

In most cases, such as robots and indoor environments, the AHRS (9-axis) mode is easily disturbed and causes errors in the heading angle. In a few open environments without magnetic field interference, you can try to use the geomagnetic assist mode, such as a drone. Before use, you need to configure the module to the geomagnetic assist mode and perform geomagnetic calibration. See the geomagnetic calibration chapter for details.

> Please refer to CONFIG-mode configuration for serial interface configuration.

1.2. Synchronous input and synchronous output

- Sync pulse input (PPS_SYNC_PIN/SIN): The pin is in pull-up input mode, and the idle state is high level. When the module detects the falling edge, it will output a frame of data (if ONMARK is triggered). At the same time, some protocols can output the PPS synchronization timestamp. The PPS synchronization timestamp refers to the time elapsed from when the module detects the latest falling edge signal to the current frame of data sampling. This pin can be left floating if not used.
- Data synchronous output (SOUT): The pin is in output mode. It is high level (idle) when there is no data output. It becomes low level when a frame of data starts to be sent. After a frame of data is sent, it returns to high level (idle). Example: In the default state, the module outputs 100Hz data, and this pin outputs a 100Hz pulse with a duty cycle of 50%. It needs to be left floating when not in use.

> When using the synchronous input function, you must first cancel all scheduled outputs through the serial port input command UNLOGALL.

2. Module configuration instructions

Module configuration uses ASCII string commands. Each command must end with carriage return and line feed (\r\n) (similar to AT commands) in order to be recognized by the system.

2.1. Command list

Command	Function	Remarks
REBOOT	Reset module	Effective immediately
SAVECONFIG	Save all configuration parameters	Effective immediately
SERIALCONFIG	Baud rate setting	Restart to take effect
CONFIG	Set user parameters and modes	Restart to take effect
LOG	Print module information or configuration output data	Effective immediately
UNLOGALL	Cancel all message output	Effective immediately
FRESET	Restore factory settings	Effective immediately

> All configuration instructions need to be reset or powered on again to take effect.

2.2. Detailed explanation of instructions

2.2.1. REBOOT

Resetting the module takes effect immediately and has the same effect as powering on again.

2.2.2. SAVECONFIG

Save all user configurations to Flash.

2.2.3. SERIALCONFIG

Set the serial port baud rate, optional values: 9600/115200/256000/460800/921600

Configure the serial port baud rate to 115200

SERIALCONFIG 115200
SAVECONFIG

> Special attention is required when using this command. Entering an incorrect baud rate will result in the inability to communicate with the module.

2.2.4. CONFIG

Used to configure module working parameters. All configurations need to be SAVECONFIG and reset to take effect.

Mode configuration (6-axis, 9-axis)

CONFIG ATT MODE 0 Configure the module to 6DOF mode

CONFIG ATT MODE 1 Configure the module to AHRS (9-axis) mode

Horizontal Calibration

CONFIG ATT RST 3 Automatic calibration: If the current pitch angle/roll angle is close to 0°, 0° (horizontal front placement), it will automatically calibrate to 0,0. If the current pitch angle/roll angle is close to 0° or 180° (horizontal inversion), it will automatically calibrate to 0°, 180°. Suitable for robot installation environment. Among them, "close" is defined as Pitch Roll is less than 15°

CONFIG ATT RST 5 Cancel horizontal leveling: Clear the current pitch and roll angle leveling settings (restore to default)

> When executing the CONFIG ATT RST command, the module needs to remain stationary. If the module executes this command while in motion, it may cause a large calibration error.

Coordinate system rotation (change installation method)

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

Among them C_{nn} supports floating point numbers

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

Where $\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_U$ is the rotated sensor coordinates Set the sensor data, $\begin{Bmatrix} X \\ Y \\ Z \end{Bmatrix}_B$ is before rotation Sensor data in sensor coordinate system

Here are some examples of commonly used rotations:

- The new sensor coordinate system is rotated -90° around the X-axis of the original coordinate system (vertical installation with positive Y** axis facing downward**), configuration command:

  CONFIG IMU URFR 1,0,0,0,0,1,0,-1,0

- The new sensor coordinate system is rotated 90° around the X-axis of the original coordinate system (**Vertical installation with positive Y-axis facing upward**), configuration command:

  CONFIG IMU URFR 1,0,0,0,0,-1,0,1,0

- The new sensor coordinate system is rotated 180° around the X-axis of the original coordinate system. Configuration command:

  CONFIG IMU URFR 1,0,0,0,-1,0,0,0,-1

- The new sensor coordinate system is rotated 90° around the Y-axis of the original coordinate system (**Vertical installation with positive X-axis pointing upward**), configure the command

  CONFIG IMU URFR 0,0,-1,0,1,0,1,0,0

- The new sensor coordinate system is rotated -90° around the Y-axis of the original coordinate system (**X-axis is installed vertically with the positive direction downward**), configuration command:

  CONFIG IMU URFR 0,0,1,0,1,0,-1,0,0

- The new sensor coordinate system is rotated 180° around the Y-axis of the original coordinate system. Configuration command:

  CONFIG IMU URFR -1,0,0,0,1,0,0,0,-1

- The new sensor coordinate system is rotated 90° around the Z-axis of the original coordinate system. Configuration command:

  CONFIG IMU URFR 0,-1,0,1,0,0,0,0,1

- The new sensor coordinate system is rotated -90° around the Z-axis of the original coordinate system. Configuration command:

  CONFIG IMU URFR 0,1,0,-1,0,0,0,0,1

- Restore factory defaults:

  CONFIG IMU URFR 1,0,0,0,1,0,0,0,1

  - After setting URFR, you need to reset the software or power on again to take effect. There is no need to send this command every time you power on.
  - How to determine the URFR parameters: (Take a vertical installation rotated -90° around the X-axis of the original coordinate system (the positive direction of the Y-axis facing downward) as an example). The relationship between the coordinates after conversion and the coordinates before conversion can be written:
  - $X_U = X_B$
  - $Y_U = -Z_B$
  - $Z_U = Y_B$

  So we can write the transformation matrix =

  1 0 0 (X after conversion = X before conversion)
  0 0 -1 (Y after conversion = -Z before conversion)
  0 1 0 (Z after conversion = Y before conversion)

  According to the above URFR definition formula, what the URFR parameter requires is actually the transpose of the above matrix, that is:

  1 0 0
  0 0 1
  0 -1 0

2.2.5. LOG

Turn on/off data output globally

LOG ENABLE Enable data frame output globally (default)
LOG DISABLE Disable data frame output globally

Display module version information

LOG VERSION Print firmware version information

Display user configuration information

LOG USRCONFIG Prints user configuration information to check whether the configuration has been written successfully.

```
ATT_MODE: 0 /* Working mode: 0:6 axis, 1:9 axis */
...
```

Display serial port configuration information

LOG COMCONFIG Print serial port and output protocol configuration information

Configure message output (set frequency)

LOG <MSG> <TYPE> <PERIOD>

- MSG: IMU91, HI91 (same as IMU91, firmware version >=1.5.0 required), HI92
- TYPE: ONTIME: timing output, ONMARK: external trigger synchronous output
- PERIOD: Output frame period, unit is s, value range: 1(1Hz), 0.5(2Hz), 0.1(10Hz), 0.02(50Hz), 0.01(100Hz), and so on

Example (timing 100Hz output):

- LOG IMU91 ONTIME 0.01 Set the current serial port's 91 data packet output cycle to 0.01s (100Hz)
- SAVECONFIG Save settings
- REBOOT Restart takes effect

Example (turn off output):

- LOG IMU91 ONTIME 0 Turn off 91 packet output
- SAVECONFIG Save settings
- REBOOT Restart takes effect

Example (synchronous trigger output):

- UNLOGALL Cancel all data output.
- LOG IMU91 ONMARK 1 Set the current COM port 91 data packet to synchronous trigger mode.
- SAVECONFIG Save settings
- REBOOT Restart to take effect

> When the output frame rate is set to a relatively high value (such as 500Hz), the preset baud rate of 115200 does not meet the output bandwidth requirements. At this time, the module baud rate needs to be set to a high value (such as 921600) before the module can Correct output data.
> The baud rate parameters are set and stored after power failure, and the reset module takes effect. The baud rate of the host computer or other hosts must also be modified accordingly.

2.2.6. UNLOGALL

Set the output frequency of all scheduled output messages to 0 (no output)

2.2.7. FRESET

Restore factory settings

2.3. More advanced configuration

In addition to the above common configurations, the module can also be configured with some advanced performance parameters. Generally, these parameters do not need to be modified and have been adjusted to the optimal value before leaving the factory.

2.3.1. Configure acceleration and gyroscope bandwidth

CONFIG IMU ABW <VAL> Adjust accelerometer bandwidth VAL = 2(20Hz), 3(40Hz), 4(80Hz), 5(125Hz), 6(230Hz, default)

CONFIG IMU GBW <VAL> Adjust gyroscope bandwidth VAL = 0 (12Hz), 3(47Hz), 4(80Hz), 5(116Hz, default), 6(230Hz)

Example: Adjust the gyroscope 3db cutoff bandwidth to 47Hz: CONFIG IMU GBW 3

- Do not configure values that are not marked in the manual, otherwise the output data may be abnormal. Changing the bandwidth of the accelerometer and gyroscope will not improve the accuracy. If you want to improve the accuracy, modification is not recommended.
- In order to reduce the impact of static noise or vibration, the accelerometer bandwidth can be appropriately reduced.
- The cost of lowering the bandwidth is greater phase delay of the output data. In some applications that require real-time high dynamics, the bandwidth should not be adjusted too low.

2.3.2. Adjust attitude Kalman filter (KF) parameters

Adjusting the KF process noise Q: Q is an important parameter in KF: the larger the Q, the more KF believes that the gyroscope is noisy and has low accuracy, and the more KF believes in measurement information such as gravity, which is manifested in attitude output versus vibration, instantaneous acceleration and deceleration, etc. More sensitive, however pitch/roll errors can also be corrected more quickly. On the contrary: the smaller Q is, the more KF believes that the gyroscope has low noise and high accuracy. The smaller the weight of the measurement information correction, the overall attitude output will be smoother and will not be affected by vibration, acceleration and deceleration, but the cumulative error correction will also be slower. Note that Q that is too large or too small may cause the attitude angle output accuracy to decrease or even diverge.

- CONFIG IMU ATT_Q <VAL> VAL range is 0.1 - 5, default is 1.0

Example: Adjust the Q value of KF to 2: CONFIG IMU ATT_Q 2

Bandwidth and KF parameters can be viewed through LOG USRCONFIG to view the current configuration

3. USB&RS232 data output protocol (custom binary protocol)

This protocol is a binary protocol defined by Super Core and can output all sensor information. Our UART (RS-232/TTL), USB, and RS-485 interface products support this protocol. The default serial port format is N-8-N-1 (8 data bits, 1 stop bit, 0 parity bit)

3.1. Sequence Data Protocol

3.1.1. Data frame format

After the module is powered on, it outputs frame data according to the preset frame rate (100Hz). The frame format is as follows:

field name	value	length (bytes)	description
Frame header	0x5A	1	0x5A
Frame type	0xA5	1	0xA5
Data field length	1-512	2	The length of the data field in the frame, LSB (low byte first) The length indicates the length of the data field (excluding frame header, frame type, length, CRC)
CRC check	-	2	16-bit CRC checksum of all fields (frame header, frame type, length, data field) except the CRC byte. LSB (low byte first)
Data field	-	1-512	The data carried in one frame is composed of several sub-data packets. The data packet contains two parts: data packet label and data. The tag determines the type and length of the data.

3.1.2. Factory default output

Factory default output: floating point IMU data frame (0x91)

3.1.3. Data field content

Floating point IMU data frame (0x91)

The data field has a total of 76 bytes. Contains module ID, temperature, IMU raw data, geomagnetism, air pressure, fused attitude data, etc.

Example of opening data frame: LOG IMU91 ONTIME 1, or LOG HI91 ONTIME 1 (requires firmware version >=1.5.0 support), for details, please refer to the configuration command chapter.

Byte offset	Name	Data type	Size (Byte)	Unit	Scale factor	Description
0	tag	uint8_t	1	-	-	Packet tag: 0x91
1	pps_sync_stamp	uint16_t	2	ms	1	PPS pulse synchronization timestamp, used for precise time synchronization. This value is defined as: the time elapsed from the last detected PPS synchronization pin pulse edge to the data sampling moment of this frame , range 0-8192, when it exceeds the maximum value, it will automatically roll back to 0. For example: if the user inputs a standard second pulse on the PPS pin, the value will be between 0-1000.
3	temperature	int8_t	1	°C	1	Average module temperature
4	air_pressure	float	4	Pa	1	Air pressure
8	system_time	uint32_t	4	ms	1	Node local timestamp information, accumulated starting from system startup, increasing by 1 every millisecond
12	acc_b	float	4*3	G	1	Acceleration after factory calibration, the order is: XYZ axis. 1G=1x local gravity acceleration, which can be approximated as 9.8 m/s^2
24	gyr_b	float	4*3	deg/s(dps)	1	Angular velocity after factory calibration, the order is: XYZ axis
36	mag_b	float	4*3	uT	1	Magnetic strength, the order is: XYZ axis
48	roll	float	4	deg	1	Roll angle
52	pitch	float	4	deg	1	Pitch angle
56	yaw	float	4	deg	1	Yaw angle
60	quat	float	4*4	-	-	Node quaternion set, order is WXYZ

Integer IMU data frame (0x92)

A total of 48 bytes, which is smaller than the floating-point data frame. Example of opening data frame: LOG HI92 ONTIME 1, for details, please refer to the configuration command chapter (requires firmware version >=1.5.0 support)

Byte offset	Name	Data type	Size (Byte)	Unit	Scale factor	Description
0	tag	uint8_t	1	-	-	Packet tag: 0x92
1	status	uint16_t	2	-	-	Status word, reserved
3	temperature	int8_t	1	°C	1	Average system temperature
4	pps_sync_stamp	uint16_t	2	ms	1	PPS pulse synchronization timestamp, used for precise time synchronization. This value is defined as: the time elapsed from the last detected PPS synchronization pin pulse edge to the data sampling moment of this frame , range 0-8192, when it exceeds the maximum value, it will automatically roll back to 0. For example: if the user inputs a standard second pulse on the PPS pin, the value will be between 0-1000.
6	air_pressure	int16_t	2	Pa	1	Atmospheric pressure +100000Pa: such as 2000, 102000Pa
8	reserved	-	2	-	-	Reserved
10	acc_b	int16_t*3	6	m/s^(2)	0.0048828	Acceleration: X, Y, Z axis (after factory calibration)
16	gyr_b	int16_t*3	6	rad/s	0.001	Angular velocity: X, Y, Z axis (after factory calibration)
22	mag_b	int16_t*3	6	uT	0.030517	Magnetic strength: X, Y, Z axis (after factory calibration)
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 angle
40	quat	int16_t*4	8	-	0.00003	Node quaternion set, the order is WXYZ

3.1.4. CRC

16-bit CRC implementation routine:

```
/*
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;
}
```

3.1.5. Data frame structure example

Use the serial port assistant to sample a frame of data, a total of 82 bytes, the first 6 bytes are the frame header, length and CRC check value. The remaining 76 bytes are the data field. Assume that the data is received into the C language array `buf`. As follows:

5A A5 4C 00 6C 51 91 00 A0 3B 01 A8 02 97 BD BB 04 00 9C A0 65 3E A2 26 45 3F 5C E7 30 3F E2 D4 5A C2 E5 9D A0 C1 EB 23 EE C2 78 77 99 41 AB AA D1 C1 AB 2A 0A C2 8D E1 42 42 8F 1D A8 C1 1E 0C 36 C2 E6 E5 5A 3F C1 94 9E 3E B8 C0 9E BE BE DF 8D BE

- 1. Determine the frame header and obtain the data field length and CRC check value:

Frame header: 5A A5

Frame data field length: 4C 00: (0x00<<8) + 0x4C = 76

CRC check value: 6C 51: (0x51<<8) + 0x6C = 0x516C

- 2. Check CRC

```
    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 obtained CRC value is 0x516C, which is the same as the CRC value carried in the frame, and the CRC check passes.

- 3. Receive data

Starting from `0x91`, the data structure and common conversion macros are defined for the data field of the data packet:

```
#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;
```

Receive data, starting from buf[6]=0x91 is the payload part:

```
     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);
```

Print received data:

```
     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);
```

Printed analysis results:

```
tag: 0x91
acc(G): 0.2242 0.7701 0.6910
gyr(deg/s): -54.708 -20.077 -119.070
mag(uT): 19.183 -26.208 -34.542
eul(deg): 48.720 -21.014 -45.512
Quat: 0.855 0.310 -0.310 -0.277
presure(pa): -0.000
timestamp(ms): 310205
```

4. RS-485 data protocol and instructions (Modbus protocol)

The RS485 communication protocol follows the Modbus RTU protocol specification. Data is sent and received in units of temporary registers. Each temporary register occupies 2 bytes. It adopts big-endian mode (high byte first). The module's preset The configuration and instructions are as follows:

- Modbus command:

  Write: 0x06 (Write Single Register): Write a single register (each Modbus register is 2 bytes)

  Read: 0x03 (Read Holding Registers): Read single or multiple register data

  Custom function code: 0x50, used for automatic allocation of Modbus ID, convenient for mass production deployment, firmware upgrade, etc.
- Modbus device address can be modified, factory default: 80 (0x50)
- Baud rate: 9600/115200, factory default: 115200, format: 8 data bits, 1 stop bit, no parity bit (N8N1)

> RS-485 also supports serial protocol. The protocol content is consistent with Chapter 2. Please contact us for specific information.

4.1. Data frame format

4.1.1. Read scratchpad (0x03)

Host sends:

field name	value	description
ID	1-0xFF	Modbus device address
FUN_CODE	0x03	Command code
ADDR_H	-	The higher 8 bits of the scratchpad address to be read
ADDR_L	-	The lower 8 bits of the scratchpad address to be read
LEN_H	-	To read the higher 8 bits of the register length (in units of the number of registers)
LEN_L	-	To read the lower 8 bits of the register length (in the number of registers)
CRC_L	-	CRC lower 8 bits
CRC_H	-	CRC higher 8 bits

Return from the machine (module):

field name	value	description
ID	1-0xFF	Modbus device address
FUN_CODE	0x03	Command code
LEN	-	Returns the length of the register data (not counting ID, FUN_CODE, LEN, CRC fields) in bytes
DATAH	-	Return the higher 8 bits of data
DATAL	-	Return the lower 8 bits of data
----	-	Return the higher 8 bits of data
----	-	Return the lower 8 bits of data
CRC_L	-	CRC lower 8 bits
CRC_H	-	CRC higher 8 bits

4.1.2. Write scratchpad (0x06)

field name	value	description
ID	1-0xFF	Modbus device address
FUN_CODE	0x06	Command code
ADDR_H	-	The Higher 8 bits of temporary register address
ADDR_L	-	The lower 8 bits of the temporary register address
DATA_H	-	Write the higher 8 bits of data
DATA_L	-	Write the lower 8 bits of data
CRC_L	-	CRC lower 8 bits
CRC_H	-	CRC higher 8 bits

Return from the machine:

field name	value	description
ID	1-0xFF	Modbus device address
FUN_CODE	0x06	Command code
ADDR_H	-	The Higher 8 bits of temporary register address
ADDR_L	-	The lower 8 bits of the temporary register address
DATA_H	-	Write the higher 8 bits of data
DATA_L	-	Write the lower 8 bits of data
CRC_L	-	CRC lower 8 bits
CRC_H	-	CRC higher 8 bits

4.1.3. CRC check

- Calculate CRC online: https://www.23bei.com/tool/59.html

*C code:

```
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	Function	R/W	Description
0x00	0	CTL	Control	W	See Modbus setting module chapter
0x04	4	BAUD	baud rate	R	baud rate
0x05	5	ID	ID	R	Modbus ID
0x1F	31	BW	Bandwidth	R/W	Cutoff frequency: 0:12Hz, 1:23Hz 2:32Hz 3:47Hz (default) 4:64Hz, 5:116Hz
0x34	52	ACCX	Acceleration X	R	Unit G (1G=1 gravitational acceleration), scale factor: 0.00048828
0x35	53	ACCY	Acceleration Y	R	Unit G (1G=1 gravity acceleration), scale factor: 0.00048828
0x36	54	ACCZ	Acceleration Z	R	Unit G (1G=1 gravity acceleration), scale factor: 0.00048828
0x37	55	GYRX	Angular velocity X	R	Unit deg/s, scale factor: 0.061035
0x38	56	GYRY	Angular velocity Y	R	Unit deg/s, scale factor: 0.061035
0x39	57	GYRZ	Angular velocity Z	R	Unit deg/s, scale factor: 0.061035
0x3A	58	MAGX	Magnetic strength X	R	Unit uT, scale factor: 0.030517
0x3B	59	MAGY	Magnetic strength Y	R	Unit uT, scale factor: 0.030517
0x3C	60	MAGZ	Magnetic strength Z	R	Unit uT, scale factor: 0.030517
0x3D	61	ROLL_H	Roll angle higher 16 bits	R	Unit deg, scale factor: 0.001
0x3E	62	ROLL_L	Lower 16 bits of roll angle	R	Unit deg, scale factor: 0.001
0x3F	63	PITCH_H	Pitch angle higher16 bits	R	Unit deg, scale factor: 0.001
0x40	64	PITCH_L	Low 16 bits of pitch angle	R	Unit deg, scale factor: 0.001
0x41	65	YAW_H	Heading angle higher16 bits	R	Unit deg, scale factor: 0.001
0x42	66	YAW_L	Lower 16 bits of heading angle	R	Unit deg, scale factor: 0.001
0x43	67	TEMP	Temperature	R	Unit ℃, scale factor: 0.01
0x44	68	PRS_H	Air pressure 16 bits higher	R	Unit Pa, scale factor: 0.01
0x45	69	PRS_L	Air pressure 16 bits lower	R	Unit Pa, scale factor: 0.01
0x46	70	Q0	Quaternion QW	R	Quaternion, scale factor: 0.00003
0x47	71	Q1	Quaternion QX	R	Quaternion, scale factor: 0.00003
0x48	72	Q2	Quaternion QY	R	Quaternion, scale factor: 0.00003
0x49	73	Q3	Quaternion QZ	R	Quaternion, scale factor: 0.00003
0x4A	74	SINGLE_X	Inclinometer X-axis angle	R	Inclinometer X-axis angle, 0-360, unit deg, scale factor: 0.005493
0x4B	74	SINGLE_Y	Inclinometer Y-axis angle	R	Inclinometer Y-axis angle, 0-360, unit deg, scale factor: 0.005493
0x66	102	KF_ACC_R	Accelerometer attitude feedback coefficient	R/W	Acceleration attitude correction coefficient adjustment, default: 10, range: 1-20, the smaller the value, the greater the acceleration correction.
0x70-0x77	112-119	PNAME	Device name	R	Device name string, ASCII code, occupies 8 registers in total
0x78	120	SW_VERSION	Software version	R	Software version

Address(Hex)	Address(Dec)	Name	Function	R/W	Description
0x79	121	BL_VERSION	BL version	R	BL version
0x7F-0x82	127-130	SN	The unique serial number of the product	R	The unique serial number of the product, occupies 4 temporary registers
0x01A0-0x01AF	160-175	ACC_CAL	Accelerometer calibration parameters	R	Accelerometer factory calibration parameters, scale factor: 0.001
0x01B0-0x01BF	176-191	GYR_CAL	Gyro calibration parameters	R	Gyro factory calibration parameters, scale factor: 0.001
0x01C0-0x01CF	192-207	MAG_CAL	Magnetometer calibration parameters	R	Magnetic sensor factory calibration parameters, scale factor: 0.001

4.3. Configuration instructions

> The default Modbus address for all the following configuration examples is 0x50 (factory default). If the Modbus ID has been modified by the user, the ID field and CRC field need to be changed.

Command	Value to Write to CTL Register	Command (Hex) ID=0X50
Save all configuration parameters to Flash	0x0000	50 06 00 00 00 00 84 4B
Restore factory settings	0x0001	50 06 00 00 00 01 45 8B
Set operating mode to 6-axis mode	0x0003	50 06 00 00 00 03 C4 4A
Set operating mode to 9-axis mode (AHRS)	0x0004	50 06 00 00 00 04 85 88
Set operating mode to pure gyroscope integration mode	0x0005	50 06 00 00 00 05 44 48
Set initial attitude offset (leveling) Pitch = Roll = Yaw = 0, effective immediately and saved even if powered off	0x0010	50 06 00 00 00 10 85 87
Set initial attitude (leveling) Pitch = Roll = 0, Yaw remains unchanged, effective immediately and saved even if powered off	0x0011	50 06 00 00 00 11 44 47
Set initial attitude (leveling) Pitch and Roll remain unchanged, Yaw = 0, effective immediately and saved even if powered off	0x0012	50 06 00 00 00 12 04 46
Clear all initial attitude settings, effective immediately and saved even if powered off	0x0013	50 06 00 00 00 13 C5 86
Set mounting orientation: set to horizontal installation (normal mode)	0x0020	50 06 00 00 00 20 85 93
Set mounting orientation: vertical installation with the positive Y-axis facing downward	0x0021	50 06 00 00 00 21 44 53
Set mounting orientation: vertical installation with the positive Y-axis facing upward	0x0022	50 06 00 00 00 22 04 52
Set mounting orientation: vertical installation with the positive X-axis facing upward	0x0023	50 06 00 00 00 23 C5 92
Set mounting orientation: vertical installation with the positive X-axis facing downward	0x0024	50 06 00 00 00 24 84 50
Reset	0x00FF	50 06 00 00 00 FF C4 0B
Configure baud rate to 4800 (takes effect after reset)	0x0100	50 06 00 00 01 00 85 DB
Configure baud rate to 9600 (takes effect after reset)	0x0101	50 06 00 00 01 01 44 1B
Configure baud rate to 19200 (takes effect after reset)	0x0102	50 06 00 00 01 02 04 1A
Configure baud rate to 38400 (takes effect after reset)	0x0103	50 06 00 00 01 03 C5 DA
Configure baud rate to 57600 (takes effect after reset)	0x0104	50 06 00 00 01 04 84 18
Configure baud rate to 115200 (takes effect after reset)	0x0105	50 06 00 00 01 05 45 D8
Configure baud rate to 230400 (takes effect after reset)	0x0106	50 06 00 00 01 06 05 D9
Configure baud rate to 460800 (takes effect after reset)	0x0107	50 06 00 00 01 07 C4 19
Configure baud rate to 921600 (takes effect after reset)	0x0108	50 06 00 00 01 08 84 1D
Configure Modbus ID (takes effect after reset)	0x0200 + ID	Configure Modbus ID to 0x03 Send 50 06 00 00 02 03 C5 2A

4.4. Read data example

1. Read the module product name, software version and SN number

TX (sent by host): 50 03 00 70 00 13 08 5D

ID=0x50, CMD=0x03, read starting address 0x70, read length: 0x13, CRC: 0x085D

RX (slave response): 50 03 26 00 43 00 48 00 31 00 30 00 58 00 28 00 4D 00 29 00 73 00 00 00 00 00 00 00 00 00 00 00 00 06 DD C2 9C 6D 0 6 97 0F 7F 5F

50 03 26: Slave address 0x50, command code: 0x03, data part total 0x26= 38 bytes, 00 43 00 48 00 31 00 30 00 58 00 28 00 4D 00 29 00 73 00 00 00 00 00 00 00 00 00 00 00 00 06 DD C2 9C 6D 06 97 0F , Data segment: Product name: CH10x(M), Software version: 0x73(115), SN:06DDC29C6D06970F, 7F 5F: CRC check

2. Read IMU attitude data

TX (sent by host): 50 03 00 34 00 18 09 8F

ID=0x50, CMD=0x03, read starting address 0x34, read length: 0x18, CRC: 0x098F

RX (slave response): 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

0x30 = The slave returns 48 bytes, the first register value is 0xFF01=-255, the second register value is 0x03B0=944...and so on.

- Acceleration: X=-255, Y=944, Z=1616 ->The result after multiplying the scale factor: 2))
- Angular velocity: X=-823, Y=-132, Z=145 -> The result after multiplying the scaling factor:
- Magnetic field: X=469, Y=-549, Z=-729 -> The result after multiplying the scaling factor:
- Euler angles: Roll (Roll) =8703, Pitch (Pitch) =32758, Yaw (Yaw) =-166937 -> The result after multiplying by the scaling factor: Roll= 8.703deg, Pitch=32.758deg, Yaw=- 166.937deg

4.5. Modbus ID automatic assignment

The automatic ID address allocation mechanism is used when multiple modules are connected to the same 485 bus during mass production deployment. The modules can cooperate with the host computer to complete automatic ID address allocation. This function is only open to mass production customers. Specific information Please contact us.

0x50 The custom command format is: [ADDR] 0x50 [SUB_CMD] [DATA_LEN] [DATA]

Currently supported list of custom commands:

1. Set the ID address through the SN number (0x0031): Format: 00 50 00 31 00 0A [SN] [NEW_ADDR] CRC

   SN: Device unique serial number, 8 bytes

   New ID address: 1-255, 2 bytes

2. ID address randomization (0x0030): This command will force all modules on the bus to discard the original ID and generate a new ID between MIN_ADDR and MAX_ADDR. Format: 00 50 00 30 00 06 [MIN] [MAX] FF FF CRC

   MIN: Minimum value to generate random ID, 2 bytes.

   MAX: The maximum value for generating a random ID is 2 bytes.

4.6. Firmware upgrade

The Modbus firmware upgrade protocol is only open to mass production customers. Please contact us for specific information.

5. CAN data protocol and instructions (CANopen protocol)

The CAN interface complies with the CANopen protocol. All communications use standard data frames and use TPDO1-7 to transmit data. Remote frames and extended data frames are not received/sent, and all TPDOs adopt asynchronous timing trigger mode.

5.1. CANopen default settings

Default configuration	Value
CAN baud rate	500KHz
CANopen node ID	8
Initialization state	Operational
Heartbeat package	None
TPDO output rate	1Hz - 200Hz (each TPDO)

5.2. CANopenTPDO

Channel	Frame ID	Data length (DLC)	Transmission method	Output frequency (Hz)	Data	Description
TPDO1	0x180+ID	6	Asynchronous timing (0xFE)	100	Acceleration	Type: int16 low byte first, 2 bytes for each axis, 6 bytes in total respectively X, Y, Z axis acceleration, unit is mG (0.001G)
TPDO2	0x280+ID	6	Asynchronous timing (0xFE)	100	Angular velocity	Type: int16 low byte first, 2 bytes for each axis, 6 bytes in total respectively X, Y, Z axis angular velocity, unit is 0.1dps (°/s)
TPDO3	0x380+ID	6	Asynchronous timing (0xFE)	100	Euler angle	Type: int16 Low byte first, 2 bytes for each axis, 6 bytes in total Sequence respectively They are roll angle: Roll, pitch angle: Pitch, and heading angle: Yaw. The unit is 0.01°
TPDO4	0x480+ID	8	Asynchronous timing (0xFE)	100	Quaternion	Type: int16 low byte first, 2 bytes for each element, 8 bytes in total respectively q_w q_x q_y q_z. The result after unit quaternion is expanded 10000 times. For example, when the quaternion is 1,0,0,0, the output is 10000,0,0,0.
TPDO6	0x680+ID	4	Asynchronous timing (0xFE)	20	Air pressure	Type: int32 4 bytes in total. Unit Pa
TPDO7	0x780+ID	8	Asynchronous timing (0xFE)	100	Inclinometer angle	Type: int32 Low byte first, 4 bytes for each axis, 8 bytes in total Sequence respectively are the X-axis and Y-axis. The unit is 0.01°

Use acceleration and angular velocity as examples to analyze data

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

- ID=0x188: Acceleration data frame sent by device with ID 8
- Acceleration X axis = 0x004A = 74 =74mG
- Acceleration Y axis = 0x001F = 731 = 31mG
- Acceleration Z axis = 0x03C8 = 968 = 968mG

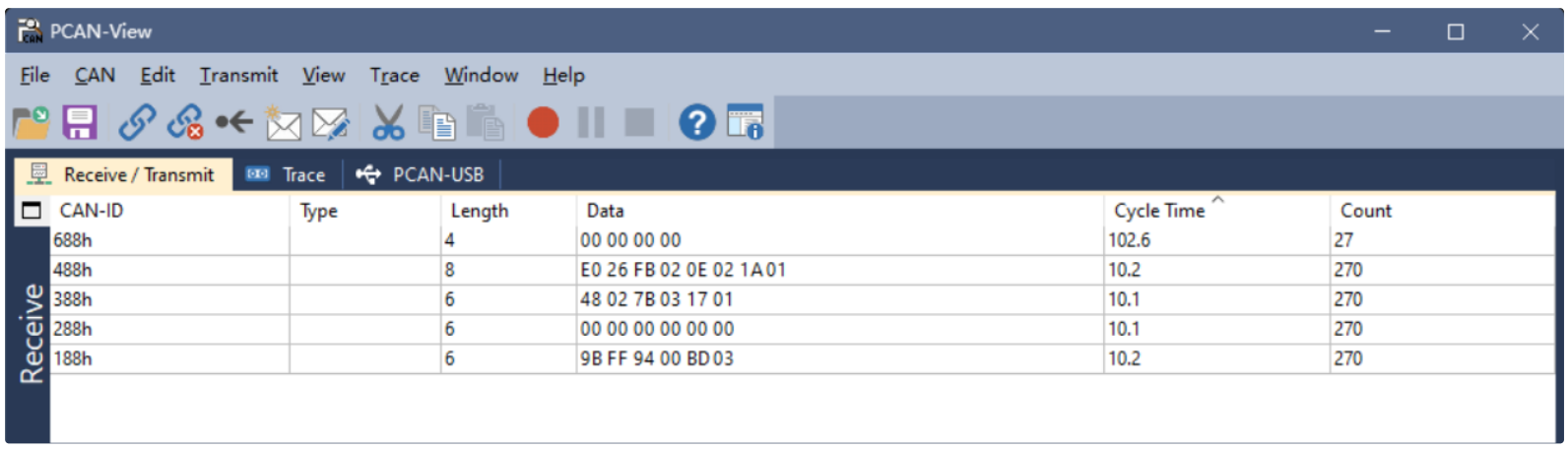
Angular velocity CAN frame: ID=0x288, DATA = 15 00 14 01 34 00

- ID=0x288: Angular velocity data frame sent by device with ID 8
- Angular velocity X axis = 0x0015 = 21 = 2.1dps
- Angular velocity Y axis = 0x0114 = 276 = 27.6dps

- Angular velocity Z axis = 0x0034 = 52= 5.2dps

5.3. Use the host computer to connect to the CAN device

Using the PCAN-View tool, combined with PCAN, the received CAN message and frame rate can be displayed in the receiving box (Rx Message), as shown in the following figure:



5.4. Configuration instructions

After all configuration changes, you need to send a save configuration command to save to Flash.

5.4.1. LLS (Layer Setting Services) protocol

- The master sends LLS command to the slave: `ID = 0x07E5, DATA = CS,00,00,00,00,00,00,00` where CS is the command field and DATA is the data field
- The slave returns the LLS command to the host: `ID = 0x07E4, DATA = CS, ECODE, 00, 00, 00, 00, 00, 00` where ECODE is the error code and 0 means no error.

Modify baud rate (valid after power cycle)

- CAN baud rate modified to 1000 kbit/s: `ID=0x07E5, DATA=13,00,00,00,00,00,00,00`
- CAN baud rate modified to 800 kbit/s: `ID=0x07E5, DATA=13,00,01,00,00,00,00,00`
- CAN baud rate modified to 500 kbit/s: `ID=0x07E5, DATA=13,00,02,00,00,00,00,00`
- CAN baud rate modified to 250 kbit/s: `ID=0x07E5, DATA=13,00,03,00,00,00,00,00`
- CAN baud rate modified to 125 kbit/s: `ID=0x07E5, DATA=13,00,04,00,00,00,00,00`
- CAN baud rate modified to 100 kbit/s: `ID=0x07E5, DATA=13,00,05,00,00,00,00,00`
- CAN baud rate modified to 50 kbit/s: `ID=0x07E5, DATA=13,00,06,00,00,00,00,00`
- CAN baud rate modified to 20 kbit/s: `ID=0x07E5, DATA=13,00,07,00,00,00,00,00`
- CAN baud rate modified to 10 kbit/s: `ID=0x07E5, DATA=13,00,08,00,00,00,00,00`

Modify node ID

`ID=0x07E5 · DATA=11,ID,00,00,00,00,00,00`

> ID modification range: 1-64, after taking effect, send the start node command (for example, the node start command data changes to 01 09) and SDO command (when sending the CAN frame ID changes to 0x609, pay attention to the new address)

Save configuration

`ID=0x07E5 · DATA=17,00,00,00,00,00,00,00`

> After all configuration changes, you need to send a save configuration command to save to Flash

5.4.2. NMT (Network Management) protocol

Globally enable/disable data output (enable asynchronous triggering)

Using CANopen NMT protocol frames:

- Enable global data output `ID=0x00, DATA=01 08`
- Close global data output `ID=0x00, DATA=02 08`

Reset

- Reset node `ID=0x00, DATA=81 08`

5.4.3. SDO (Service Data Object) protocol

SDO format:

- The master sends SDO command to the slave:

CAN_ID	CS command symbol (1B)	Data dictionary index (2B)	Sub-index (1B)	Data (4B)
0x600+ID	0x23 (write 4B)	Low byte first	Sub-index	Data, low byte first

- The slave replies SDO command to the host:

CAN_ID	SDO command (1B)	Data dictionary index (2B)	Subindex (1B)	Data (4B)
0x580+ID	0x60 (write success response)	low-order bit first	sub-index	data, low-end first

The following configuration operations all use fast SDO to write the data dictionary, where the TPDO channel and its corresponding parameter index are:

Channel	Frame ID	Parameter index address	Description
TPDO1	0x180+ID	0x1800	Acceleration
TPDO2	0x280+ID	0x1801	Angular velocity
TPDO3	0x380+ID	0x1802	Euler angle
TPDO4	0x480+ID	0x1803	Quaternion
TPDO6	0x680+ID	0x1804	Air pressure
TPDO7	0x780+ID	0x1805	Inclinometer output

Modify/close/enable data output rate

This configuration takes effect immediately

- `ID=0x608, DATA=23,00,18,05,00,00,00,00` Turn off acceleration output
- `ID=0x608, DATA=23,00,18,05,05,00,00,00` Acceleration 200Hz output
- `ID=0x608, DATA=23,00,18,05,0A,00,00,00` Acceleration 100Hz output
- `ID=0x608, DATA=23,00,18,05,14,00,00,00` Acceleration 50Hz output
- `ID=0x608, DATA=23,00,18,05,32,00,00,00` Acceleration 20Hz output
- `ID=0x608, DATA=23,00,18,05,64,00,00,00` Acceleration 10Hz output (minimum 10Hz)
- `ID=0x608, DATA=23,01,18,05,00,00,00,00` Turn off angular velocity output
- `ID=0x608, DATA=23,01,18,05,05,00,00,00` Angular velocity 200Hz output
- `ID=0x608, DATA=23,01,18,05,0A,00,00,00` Angular velocity 100Hz output
- `ID=0x608, DATA=23,01,18,05,14,00,00,00` Angular velocity 50Hz output
- `ID=0x608, DATA=23,01,18,05,32,00,00,00` Angular velocity 20Hz output
- `ID=0x608, DATA=23,01,18,05,64,00,00,00` Angular velocity 10Hz output (minimum 10Hz)
- `ID=0x608, DATA=23,02,18,05,00,00,00,00` Turn off Euler angle output
- `ID=0x608, DATA=23,02,18,05,05,00,00,00` Euler angle 200Hz output
- `ID=0x608, DATA=23,02,18,05,0A,00,00,00` Euler angle 100Hz output
- `ID=0x608, DATA=23,02,18,05,14,00,00,00` Euler angle 5Hz output
- `ID=0x608, DATA=23,02,18,05,32,00,00,00` Euler angle 20Hz output
- `ID=0x608, DATA=23,02,18,05,64,00,00,00` Euler angle 10Hz output (minimum 10Hz)
- `ID=0x608, DATA=23,03,18,05,00,00,00,00` Turn off quaternion output
- `ID=0x608, DATA=23,03,18,05,05,00,00,00` Quaternion 200Hz output
- `ID=0x608, DATA=23,03,18,05,0A,00,00,00` Quaternion 100Hz output
- `ID=0x608, DATA=23,03,18,05,14,00,00,00` Quaternion 50Hz output
- `ID=0x608, DATA=23,03,18,05,32,00,00,00` Quaternion 20Hz output
- `ID=0x608, DATA=23,03,18,05,64,00,00,00` Quaternion 10Hz output (minimum 10Hz)

- ID=0x608, DATA=23,04,18,05,00,00,00,00 Close air pressure output
- ID=0x608, DATA=23,04,18,05,05,00,00,00 Air pressure 200Hz output
- ID=0x608, DATA=23,04,18,05,0A,00,00,00 Air pressure 100Hz output
- ID=0x608, DATA=23,04,18,05,14,00,00,00 Air pressure 50Hz output
- ID=0x608, DATA=23,04,18,05,32,00,00,00 Air pressure 20Hz output
- ID=0x608, DATA=23,04,18,05,64,00,00,00 Air pressure 10Hz output (minimum 10Hz)

Taking the TPDO1 (acceleration) output rate as 100Hz (output once every 10ms) as an example: 0x23 writes four byte instructions for SDO. 0x00, 0x18 is to write 0x1800 index. 0x05 is the sub-index. 0x00, 0x0A= (0x00<<8) + 0x0A = 10 (unit is ms), any missing 0 will be added.

Set the positive and negative sign of the inclinometer output

- ID=0x608, DATA=23,9E,20,00,00,00,00,00 The positive and negative signs of the X-axis are the factory preset directions
- ID=0x608, DATA=23,9E,20,00,01,00,00,00 X-axis positive and negative signs are reversed
- ID=0x608, DATA=23,9F,20,00,00,00,00,00 The positive and negative signs of the Y axis are the factory preset directions
- ID=0x608, DATA=23,9F,20,00,01,00,00,00 Y-axis positive and negative signs are reversed

Set the zero point of the inclinometer (relative position output)

- ID=0x608, DATA=23,00,20,00,12,00,00,00 After writing, the current position is set as the output zero point (X=0, Y=0)
- ID=0x608, DATA=23,00,20,00,15,00,00,00 Cancel the zero point configuration after writing and output the real X, Y angle (equivalent to X, Y offset=0)

Set the tilt output resolution (customized function)

- ID=0x608, DATA=23,A2,20,00,00,00,00,00 The tilt data output resolution is 0.01° (default)
- ID=0x608, DATA=23,A2,20,00,01,00,00,00 The tilt data output resolution is 0.02°
- ID=0x608, DATA=23,A2,20,00,02,00,00,00 The tilt data output resolution is 0.1°

Redundant node configuration (customized function)

- ID=0x608, DATA=23,9D,20,00,01,00,00,00 Enable tilt output redundant node
- ID=0x608, DATA=23,9D,20,00,00,00,00,00 Disable inclination output redundant nodes

5.4.4. Synchronization protocol

Configure TPDO in synchronous mode

- Turn off all TPDOs (set the TPDO output rate to 0),
- Send CANopen synchronization frame, CANopen synchronization frame: ID:80, DATA: empty

6. CAN data protocol and instructions (SAE-J1939 protocol)

The default output protocol of the module is CANOpen. If you need SAE J1939 protocol, please contact our company.

PGN	Description
Communication mode	Broadcast communication
Default transmission time interval	100ms
Data length	8 bytes per PGN
PF (PDU format)	0xFF
PS (PDU specific)	When PF > 0xF0, it is the extended PGN address (GE), otherwise it is the destination address (DA)
Priority	3
Default J1939 address	0x08
Data format	The data format in all frames adopts LSB (low-order bit first), and is a signed integer unless otherwise specified

6.1. PGN message list

6.1.1. PGN65332(FF34) acceleration

CANID=0x0CFF3408

Name	Position (byte)	Description
Acceleration X	0-1	Unit G (1G=1 gravity acceleration), scale factor: 0.00048828
Acceleration Y	2-3	Unit G (1G=1 gravity acceleration), scale factor: 0.00048828
Acceleration Z	4-5	Unit G (1G=1 gravity acceleration), scale factor: 0.00048828
Reserved	6-7	-

6.1.2. PGN65335(FF37) Angular velocity

CANID=0x0CFF3708

Name	Position (byte)	Description
Angular velocity X	0-1	Unit deg/s, scale factor: 0.061035
Angular velocity Y	2-3	Unit deg/s, scale factor: 0.061035
Angular velocity Z	4-5	Unit deg/s, scale factor: 0.061035
Reserved	6-7	

6.1.3. PGN65345(FF41) heading angle

CANID=0x0CFF4108

SPN name	SPN location (byte)	Description
Yaw angle (Yaw)	0-3	0-360, unit °, scale factor: 0.001, clockwise is positive
Reserved	4-7	

6.1.4. PGN65354(FF4A) Inclinometer output

CANID=0x0CFF4A08 (only applicable to inclinometer products that output J1939 protocol)

Name	Position (byte)	Description
X tilt angle	0-3	Range 0-360 or ±180, unit: deg, scale factor: 0.001
Y inclination angle	4-7	Range 0-360 or ±180, unit: deg, scale factor: 0.001

6.2. Configuration instructions

6.2.1. Configuration format

The host sends: `ADDR+ CMD + STATUS + VAL`, the slave responds: `ADDR+ CMD + STATUS + VAL`

Field	Size (Byte)	Description
ADDR	2	Scratchpad address
CMD	1	0x06: write, 0x03: read
STATUS	1	Reserved
VAL	4	Value written or read

6.2.2. Collection of instructions

29'b extended frame address	data	description	description
0x0CEF08xx	37 01 06 00 [VAL]	VAL: 4 bytes	PGN: FF37 (acceleration) sending interval, unit ms, range: 5 -1000
0x0CEF08xx	34 01 06 00 [VAL]	VAL: 4 bytes	PGN: FF34 (angular velocity) sending interval, unit ms, range: 5 -1000
0x0CEF08xx	4A 01 06 00 [VAL]	VAL: 4 bytes	PGN: FF4A (heading angle) sending interval, unit ms, range: 5 -1000
0x0CEF08xx	41 01 60 00 [VAL]	VAL: 4 bytes	PGN: FF41 (inclinometer output) sending interval, unit ms, range: 5 -1000
0x0CEF08xx	00 00 06 00 00 00 00 00	-	Store all configuration parameters to Flash
0x0CEF08xx	00 00 06 00 01 00 00 00	-	Restore factory settings
0x0CEF08xx	00 00 06 00 FF 00 00 00	-	Reset
0x0CEF08xx	9A 00 06 00 [VAL]	VAL: 4 bytes	Configure baud rate (save settings, take effect after reset): 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	00 00 06 00 [VAL]	VAL: 4 bytes	Set the zero position, 0x12: Set the current position to the zero position, 0x15: Cancel the zero position setting, output the absolute physical angle
0x0CEF08xx	9E 00 06 00 [VAL]	VAL: 4 bytes	Set the positive and negative directions of the X-axis, 0: Default 1: Reverse
0x0CEF08xx	9F 00 06 00 [VAL]	VAL: 4 bytes	Set the positive and negative directions of the Y axis, 0: Default 1: Reverse
0x0CEF08xx	A3 00 06 00 [VAL]	VAL: 4 bytes	(Customized function) Set XY tilt output resolution 0: 0.001° (default), 1: 0.01°, 2: 0.1°

xx in the address field: the source address in the J1939 protocol, which can be any byte.
xx in data field: any byte

Example: ID=0x0CEF0855, DATA = 37 01 06 00 64 00 00 00: Set PGN:FF37 to 100ms period (10Hz)

7. Geomagnetic calibration

7.1. Geomagnetic Calibration Algorithm (Only used in 9-axis mode)

7.1.1. Calibration Steps

Geomagnetic calibration system operates seamlessly without user operation. Here's how it works:

1. **Automatic Data Collection**:
    - In the background, the system continuously collects geomagnetic field data over time.
    - It analyzes and compares this data, eliminating any abnormal readings.
    - Once sufficient data is acquired, the system attempts geomagnetic calibration.
2. **User Intervention Not Required**:
    - When using the **geomagnetic assist (9-axis) mode**, calibration occurs automatically.
    - However, the module provides an interface for users to check the current calibration status.
3. **Calibration Premise**:
    - The module must undergo sufficient attitude changes and maintain these positions for a specific duration.
    - This allows the internal calibration system to collect geomagnetic field information across different postures.
4. **Avoid Static State**:
    - Geomagnetic calibration cannot be performed when the module remains static.

7.1.2. Initial Calibration Operations:

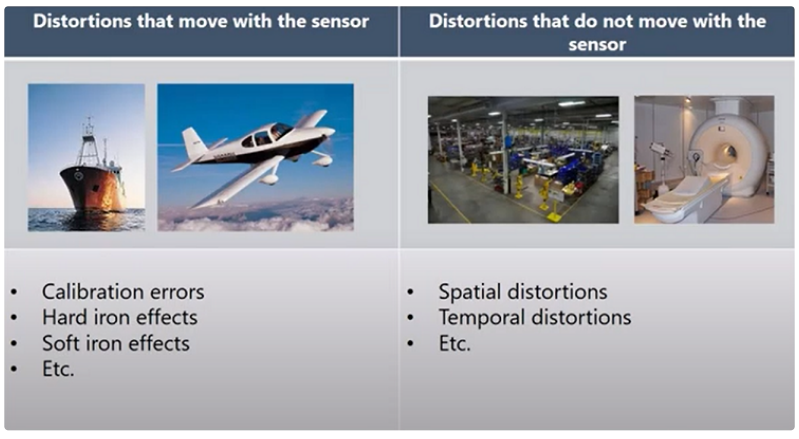
1. **Check for Interference**:
    - Assess the surroundings for magnetic field interference.
    - Common sources include iron-containing tables, computers, motors, and mobile phones.
    - Ideally, take the module outdoors. If not possible, keep it at least **0.5 meters away** from interference sources.
2. **Rotation Procedure**:
    - Rotate the module within a small range (no specific position, just rotation).
    - Ensure each axis completes at least **360° of rotation**.
    - This process typically completes the calibration.
    - If calibration fails, it indicates significant geomagnetic interference in the vicinity.
3. **Installation Location Change**:
    - If the customer relocates the module after the initial calibration (for instance, if it was calibrated separately but is now installed on the target device), recalibration is necessary.
    - Ensure that the target device accompanies you during recalibration to account for the new installation context.
4. **Viewing Geomagnetic Calibration Parameters**:
    - Utilize the `LOG MAGCONFIG` command to access and review the geomagnetic calibration parameters. This command provides essential information related to the calibration process.

```
MAG_MIS=
     1.000 0.000 0.000
     0.000 1.000 0.000
     0.000 0.000 1.000
MAG_BIS=
    15.723
    -0.320
   -14.014
OK
```

If MAG_BIAS shows three values other than 0,0,0. It means the calibration is successful.

7.1.3. Common Geomagnetic Interference Classification

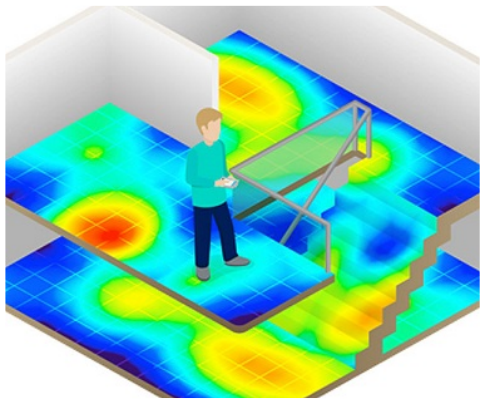
Geomagnetic interference can be divided into space magnetic field interference and magnetic field interference in the sensor coordinate system, as shown in the following figure:



Space Magnetic Field Interference: Understanding Fixed Geomagnetic Disturbances

Space magnetic field interference remains constant in the world coordinate system and does not vary with the sensor's posture. Let's delve into the details:

1. **Interference Sources**:
    - These sources are fixed and do not move alongside the sensor.
    - Examples include:
        - Fixed geomagnetic interference sources.
        - Furniture.
        - Household appliances.
        - Connecting wires.
        - Steel structures within buildings.
2. **Impact**:
    - Space magnetic field interference affects the sensor's accuracy.
    - These disturbances persist regardless of the sensor's orientation.
3. **Indoor Scenario**:
    - In typical indoor environments, various fixed interference sources contribute to the overall magnetic field distribution.
    - The accuracy of geomagnetic measurements depends on the degree of distortion caused by these fixed sources.



7.1.4. Geomagnetic Interference and Countermeasures

1. **Space Magnetic Field Interference (Uneven Environmental Fields)**:
    - Impact
        - Regardless of sensor calibration, spatial magnetic fields (including uneven environmental fields) cause distortion in the geomagnetic field.
        - Incorrect geomagnetic compensation results in inaccurate heading angles.
        - This challenge is particularly pronounced in indoor environments.
    - Countermasure
        - **Avoidance**: Efforts should focus on minimizing exposure to such interference sources.
2. **Interference in Sensor Coordinate System (Posture-Dependent Interference)**:
    - Definition
        - Interference sources move in tandem with the sensor's motion.
        - These sources (e.g., PCB boards, instruments) are fixed alongside the module and behave as part of the same rigid body as the magnetic sensor.
    - Impact

        - Such interference introduces hard magnetic or soft magnetic effects to the sensor.
        - These effects can disrupt accurate geomagnetic measurements.
    - Countermeasure
        - Geomagnetic Calibration
            - Perform calibration specifically to address this type of interference.
            - Calibration algorithms effectively eliminate unwanted effects caused by fixed interference sources.

7.1.5. Precautions for Using Geomagnetism

When operating in indoor environments, magnetic interference becomes particularly pronounced. Unfortunately, space magnetic interference cannot be entirely eliminated through calibration. Although the module incorporates a built-in homogeneous magnetic field detection and shielding mechanism for indoor use, the accuracy of the **geomagnetic assist (9-axis) mode** heading angle heavily relies on the degree of indoor magnetic field distortion.

Here are some key considerations:

1. **Indoor Magnetic Field Environment**:
    - If the indoor magnetic field environment is unfavorable (e.g., near computers, laboratories, workshops, or underground garages), even after calibration, the heading angle accuracy may not match that of the 6-axis mode. Large angle errors could occur.
2. **Fixed Magnetic Field Interference**:
    - The module's automatic geomagnetic calibration system effectively handles fixed magnetic field interference that is present during installation.
    - However, if magnetic field interference exists in the installation environment, it must be addressed:
        - Fix the interference source.
        - Ensure that the distance between the interference magnetic field and the module remains constant post-installation.
        - Example: If the module is installed on a metallic robot, both the robot and the module should be rotated and calibrated together.
        - Avoid separating the module from the robot during use to prevent relative displacement. If separation occurs, recalibration is necessary.