Force Torque Sensor User Manual

Aletschhorn generation
Doc: DOC-CM1-MAN
Revision: C
Last updated: June 12, 2025

This manual describes the selection, configuration, and operation of Bota Systems' Force Torque (FT) sensors of the Aletschhorn (Gen A) generation. It covers system commissioning, parameter configuration, communication interfaces, troubleshooting, and maintenance. For specific technical details—such as measurement ranges, resolutions, and mechanical dimensions—refer to the corresponding sensor datasheet.

Change log

Revision	Change
C	Added limitation for output rate of MiniONE products. Added support for firmware version 1.1.0. Added more communication parameters for ethernet and some serial sensors.
B	Document changes for firmware version 1.0.14 Added maximum allowed output rate for each baudrate selected
A	Document changes for firmware version 1.0.0

Table of Contents

1. Introduction

This chapter provides an overview of Force Torque (FT) sensors, including the general principles of operation, guidelines for selecting a suitable model, and definitions of key terms used throughout this manual.

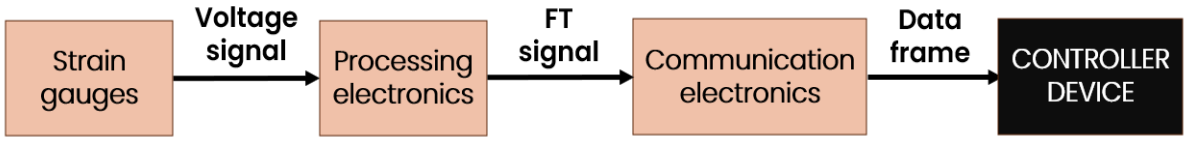
1.1. Force Torque sensor overview

A fully integrated six-axis FT sensor consists of two main components:

1. **Sensing element or 6-axis load cell**: An internal load cell is constructed from isotropic materials (e.g., aluminum, steel, or titanium). Specific regions of the load cell are designed to deform proportionally to the applied forces and torques, while the rest remains rigid. Strain gages are bonded to these deformable areas and convert mechanical stress into analog voltage signals.
2. **Data acquisition system**: The data acquisition system amplifies, filters, and digitizes the strain-gage signals at a fixed frequency. The resulting raw measurements are checked for validity, then processed into a *calculated wrench* (i.e., calibrated measurements), which is made available through one or more communication interfaces. The selected interface determines how sensor data are serialized and transmitted to external devices such as computers, industrial controllers, or programmable logic controllers (PLCs).

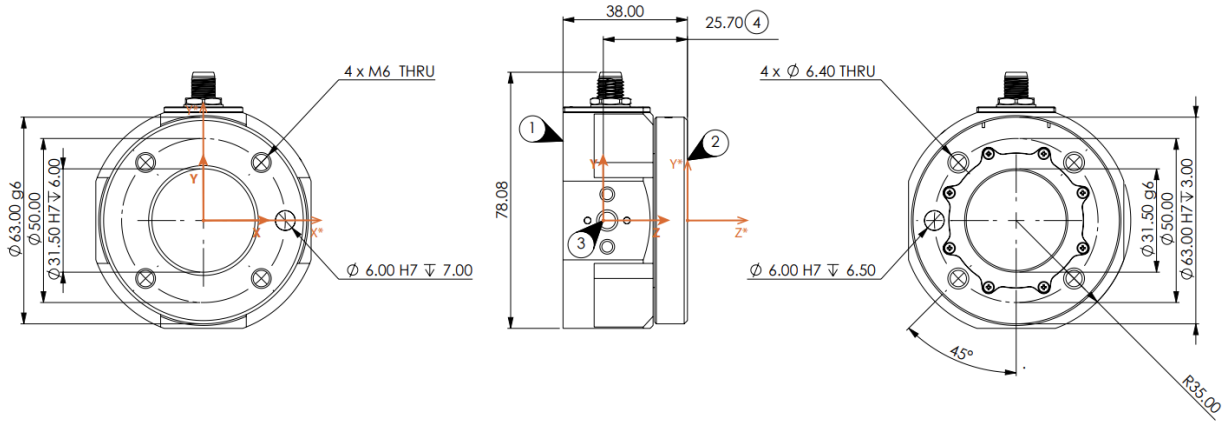
This combination of sensing element and data acquisition electronics determines the sensor's range, resolution, and accuracy. Larger measurement ranges generally reduce resolution and accuracy, while smaller ranges offer finer resolution. Bota Systems FT sensors support various ranges and share a unified interface design for streamlined integration. In addition to measuring forces and torques, they also measure acceleration and angular velocity.

FIGURE 1



All six force/torque components (Fx, Fy, Fz, Tx, Ty, Tz) are measured at a defined point called the **wrench frame** (2), typically positioned at the center of the measuring flange. Acceleration and angular velocity (IMU data) are measured in a separate **IMU frame** (3), as indicated in each sensor's datasheet. The location of these frames may vary depending on the specific sensor model.

FIGURE 2



1.2. How to choose a force torque sensor

Choosing the correct FT sensor involves estimating forces and torques during normal operation and checking that the selected sensor can withstand any potential overload situations (e.g., sudden impacts or emergency stops). The following points are recommended:

Range

Forces and torques expected during normal operation, including any lever-arm effects, should be identified. The chosen sensor is validated against these predicted loads to ensure safe operation within the sensor's rated capacity. When potential overloads (e.g., crashes or emergency stops) are anticipated, it is important to verify that the sensor's overload rating is not exceeded.

Combined loading

Many applications involve forces and torques acting simultaneously along multiple axes. The rated load typically assumes uniaxial loading. When multiple axes are excited at once, the effective maximum load per axis is reduced.

IMPORTANT

Once the maximum forces and torques have been calculated for the sensor is important to evaluate that the force and torques combined are within the combined loading graphs of the sensor model selected.

Noise-Free Resolution

Sensors with higher load capacities generally provide lower resolution, which may affect precision in measurement. It is recommended to evaluate noise-free resolution alongside the required measurement range to confirm that performance standards are met.

Sampling Rate & latency

Dynamic control applications often require higher sampling rates, but higher sampling may slightly increase the measurement's noise. For processes with slower dynamics, a lower sampling rate may suffice and can reduce noise.These considerations, in conjunction with the sensor datasheet specifications, typically guide the selection process.

1.3. Glossary

This section defines key terms used throughout the manual. Entries may be updated or expanded in later revisions if additional terms are introduced.

Wrench: It is a 6D vector, with the first 3 elements being the forces in the X, Y, Z directions and the later 3 the Torques in the same order.

Combined loading: The condition in which force and/or torque are simultaneously applied across multiple axes. If force is applied at a lever arm, an induced torque is also generated. Combined loading graphs in each sensor's datasheet illustrate allowable multi-axis loads.

Communication interface: A protocol that operates on a specific hardware interface (e.g., Modbus TCP over Ethernet, Bota ASCII protocol over USB). Both the physical connection and the data transmission format determine the communication interface.

Communication protocol: The rules governing how data are formatted and transferred across a network (e.g., TCP/IP, EtherCAT, Modbus). Protocols typically define message structures, timing, and error handling.

Configuration syntax A set of instructions used to read and write sensor parameters. Typically accessed in the sensor's Config state.

Flange: The mechanical interface for attaching the sensor to a system or end effector. Each sensor usually has two flanges:

- **Mounting flange:** Joins the sensor to the robot or system.
- **Measuring flange:** Supports the tool or end effector and is aligned with the wrench frame (unless otherwise specified).

Force Torque (FT) sensor: A precision device that measures forces and torques along multiple axes, typically six (Fx, Fy, Fz, Tx, Ty, Tz).

Hardware interface: The physical and data-link layers of the sensor's connection (e.g., USB, RS422, Ethernet). When combined with a communication protocol, the result is a communication interface.

Live data: The measurements available during normal operation, including force/torque (wrench), acceleration, angular velocity (IMU data), temperature, timestamp, and status.

Live data syntax: A set of instructions and/or data formats used while the sensor is in Run state to transmit measurement information (e.g., wrench, IMU data) to the user application.

Mounting frame: A coordinate frame that represents the interface where the sensor attaches to the robot or other system component. Used for referencing the sensor position and all other frames of the sensor.

Onboard memory: Non-volatile memory that retains configuration parameters and calibration data after power is removed.

Output rate The frequency at which live data are sent through the Bota protocol family communication interfaces. This may differ from the update rate if data throttling is applied.

Power cycle: Disconnect the power from the device, wait some seconds, and power it again.

Runtime syntax: Set of instructions used at runtime to modify the operation of the sensor.

State Change syntax: A collection of commands used to transition the sensor's internal state machine among Init, Config, and Run states.

Strain gages: Components that convert mechanical deformation (strain) into changes in electrical resistance, enabling force and torque measurements.

Update rate: The frequency at which the sensor acquires and processes raw data into final measurements before they are sent.

VCP (Virtual com port): A USB-based interface that presents the sensor as a serial port for data communication, easing integration with serial-based software tools.

Live data: refers to the data that the sensor provides in normal operation. These are the Wrench (force and torques), IMU data, Temperature, Timestamp and status of the measurements

Wrench: A six-component vector representing forces (Fx, Fy, Fz) and torques (Tx, Ty, Tz).

Wrench frame: The coordinate frame that the net force and torque are captured. The axes of this frame typically align with the flange or sensor's physical features. Specific details are provided in each sensor's datasheet.

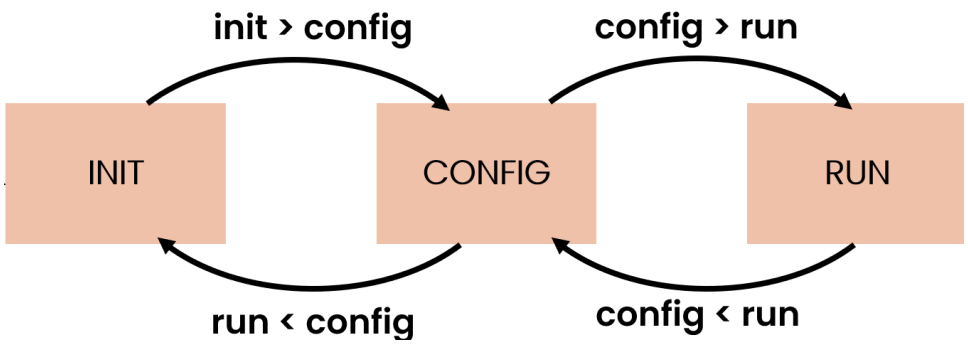
2. Sensor Operation & Configuration

Bota Systems' FT sensors measure force, torque, acceleration and angular velocity in all three axes, as well as the current room temperature. The data frequency, filtering and mode of operation can be configured by the user. Additionally, a series of different communication interfaces are provided by the sensors that can be selected and modified by the user.

2.1. State machine

All Aletschhorn (Gen A) sensors implement a three-state machine that governs their operation. The three states—Init, Config, and Run—are traversed in sequence on power-up or reset, as shown below.

FIGURE 3



Init State

- The sensor loads on-board memory parameters and prepares the primary communication interface.
- User communication is possible, but only transitions to other states can be requested (e.g., requesting a change to Config).

Init → Config

- A self-test is performed, along with initial data acquisition setup.

Config State

- All parameters can be modified and saved.
- Communication is carried out through the selected protocol's configuration syntax (e.g., Bota Binary, Modbus, EtherCAT).
- Various actions (e.g., loading defaults, saving parameters) are accessible see chapter 2.2.1.6.

Config → Run

- The operation parameters are applied.

Run State

- The sensor updates data through the configured communication interfaces in accordance with the live data syntax (see chapter 4.x for each protocol).

State and State transition errors

If an error occurred while in state, the error code is set, If a transition fails, the error code is set. The sensor may remain in the current state or proceed to the new state with an error condition. The error code can be retrieved from the Error Code parameter.

Error Code	Transition/State	Result state	Description
0x0018	Init to Config	Config with Error	Sensor cannot provide force and torque data, hardware malfunction
0x0100	Run state	Run with Error	Live data status has a flag set. It clears if status flags clear

2.2. Parameters

User-accessible parameters define how the sensor operates and communicates. These are categorized in Table 1. Each category differs in the access level and in when changes are applied. Parameters can be read or written using the configuration syntax of the chosen protocol (see chapter 3 for each interface's syntax).

TABLE 1

Parameter category	Access level	Applied on
State machine	In All states	directly
Manufacturer	in Config state	Config to Run transition
Communication	in Config state	After power on. They need to be modified then saved to onboard memory and then power cycle the sensor to apply
Operation	in Config state	Config to Run transition

IMPORTANT

Manufacturer parameters are protected. They can be modified only by the manufacturer, and they do not require changes under normal conditions. The user should not modify these parameters, doing so it will void the sensor's warranty.

TABLE 2, PARAMETER LIST

Bota Protocols parameters id: subid	EtherCAT parameter Object id:subid	Modbus parameter register map	Parameter category	Access in State	Parameter	Type/ representation	Description
1:1	Use EtherCAT State machine AL_Status see Chapter 3.3.1	101	**State machine**	R All states	Current state	uint8	It represents the current state of the sensor, with following values. 0: Init 1: Config 2: Run
1:2		107		W All states	Requested state	uint8	It represents the requested state of the sensor, with following values. 0: Init 1: Config 2: Run
1:3		108		R/W All states	Error code	uint16	Refer to the communication interface chapters for the specific error codes.
2:[1-6]	0x8000: [0x01-0x06]	400-411	**Operation** see chapter 2.2.1	R/W Config	Wrench offset	float[6]	Vector of 6 Wrench offset values added to Wrench in SI units [Fx, Fy, Fz, Tx, Ty, Tz] see chapter 2.2.1.1
3:1	0x8001:0x01	103		R/W Config	Application mode	uint8	See Table 3, Application Mode
4:1	0x8001:0x02	104		R/W Config	Application submode	uint8	See Table 4, Application Submodes
4:2	0x8001:0x02	110		R Config	Update rate	float	Live data Update rate in Hz. To be used by master application to determine the read rate.
5:[1-6]	0x8002: [0x01-0x06]	500-511		R/W Config	Temperature coefficients	float[6]	Vector of 6 multiplied by temperature and added to Wrench. See chapter 2.2.1.2
7:1	0x8003:0x01	105		W Config	Action request	uint8	The request number of the available actions. See chapter 2.2.1.6
8:1	0x8003:0x02	106		R Config	Action error code	uint8	The result of the last action run. To clear request action Idle. See chapter 2.2.1.6 for more details
9:[1-6]	0x6000 See chapter 3.3.3	0-26		R	Single read Wrench. Result of the Single read wrench action.	float[6]	Vector of 6 Wrench of the measurements in SI units [Fx, Fy, Fz, Tx, Ty, Tz] This parameter can be used to check measurements validity before starting the sensor or Taring the sensor. See chapter 2.2.1.7.

Bota Protocols parameters id: subid	EtherCAT parameter Object id:subid	Modbus parameter register map	Parameter category	Access in State	Parameter	Type/ representation	Description
6:1	Not available	Not available	**Communication** see chapter 2.2.2	R/W Config	Bota protocols, throttled output rate	uint16	Live data Output rate in Hz (int) for Bota protocol family. If set to 0 matches the update rate (async operation).
10:[1-4]	Not available	200-203		R/W Config	Ethernet IP address	uint8[4]	A vector of 4 unsigned integers Example: 192 168 0 1
11:[1-4]	Not available	204-207		R/W Config	Ethernet subnet Mask	uint8[4]	A vector of 4 unsigned integers Example: 192 168 0 1
12:[1-4]	Not available	208-211		R/W Config	Ethernet Gateway	uint8[4]	A vector of 4 unsigned integers Example: 192 168 0 1
13:[1-4]	Not available	212-215		R/W Config	Ethernet DNS	uint8[4]	A vector of 4 unsigned integers Example: 192 168 0 1
30:[1-4]	Not available	Not available		R/W Config	Ethernet UDP Broadcast IP	uint8[4]	A vector of 4 unsigned integers Default: 255.255.255.255 to be used when used within a subdomains
31:1	Not available	Not available		R/W Config	Ethernet UDP Broadcast port	uint8	Port number. Default: 30302. To be changed when multiple sensors are used.
14:1	Not available	109		R/W Config	Baud rate index	uint8	Baudrate selection for serial communication interface Default. 4: 460800 see chapter 2.2.2.4
32:1	Not available	111		R/W Config	Serial Standard	uint8	It selects the different serial standards supported, with following values. 0: RS422 full-duplex 1: RS485 half-duplex muti-drop (Only for Modbus RTU) 2:RS232 full-duplex Default: 0: RS422, see chapter 2.2.2.7
32:2	Not available	112		R/W Config	RS422 and RS485 termination	uint8	It enables the termination for RS422 or RS485, with the following values. 0: No termination 1: 120Ohm termination Default: 1, see chapter 2.2.2.8
15:1	Not available	Not available		R/W Config	primary protocol index	uint8	Primary communication interface selection. See Table 11, Communication interfaces for choosing it.
16:1	Not available	Not available		R/W Config	USB protocol index	uint8	USB communication interface selection. See Table 11, Communication interfaces for choosing it.
17:1	Not available	100		R/W Config	Modbus RTU slave ID	uint8	The slave ID used in the Modbus RTU and TCP communication interfaces Default: 1

2.2.1. Operation parameters

These parameters control data processing and are intended to be modified by the user so the sensor's operation fits best to the application. See chapter 2.2 for the list of all parameters in Table 2 and refer to the sub-chapters below for details.

2.2.1.1. Wrench offset

Force torque sensor measurements are not absolute measurements as it always includes biases cause of the following reasons. This parameter is intended to be used by the application to adjust the measurements to the expected value or tare the sensor.

- **internal stresses** coming from mounting the sensor and the mounting body.
- **gravity**, depending on the payload of the sensor or even its own mass the sensor measurements always include gravity and acceleration component.
- **Temperature,** heat gradients can cause measurement drift.
- **Extended time,** the sensor measurements will slightly change over days.

The wrench offset is a vector of 6 [Fx,Fy,Fz,Tx,Ty,Tz] and is added to the measurements

$$output\ wrench = sensor\ wrench + wrench\ offset$$

To adjust the output wrench to the required values or tare the sensor (all values 0) use the following formula. The user can obtain the current Output Wrench while in Config State using the Single read Wrench parameter after triggering a Single read action (see chapter 2.2.1.6)

$$required\ Wrench\ offset = desired\ Output\ wrench - current\ output\ wrench + current\ wrench\ offset$$

2.2.1.2. Temperature coefficients

A vector of six coefficients multiplied by the sensor's temperature reading and added to each wrench component. When set to zero, no temperature compensation is applied.

$$sensor\ wrench = sensor\ wrench + dot\ product(Temperature\ coefficient\ , temperature)$$

2.2.1.3. Application modes

6-axis FT sensors have a wide range of applications. Most of the applications of these types lie in one of the following categories:

- **Measuring applications with data post-processing** in which latency and data rate are not crucial but resolution and noise levels have to be as low as possible. For instance, a testbench application.
- **Control applications with real-time data processing** in which the sensor is used together with an actuated system like an articulated robot or other moving system. Their latency and data rate are of the highest priority. For instance, closed-loop force control.
- **Dynamic Control Application with real time data processing** where the sensor is used together with an actuated system and not only the force but also the motion data like acceleration and angular rates are used together as control signals. For instance, in a humanoid.
- **Communication specific** where the mode of operation is tailored to a platform, for example Fanuc robots, where the App mode will be set automatically when the corresponding communication interface will be selected.

Bota Systems have developed different modes of operation called **APPLICATION MODES** tailored to different use cases. There is also custom app modes designed for specific applications, such as FANUC integration.

TABLE 3, APPLICATION MODES

App mode	Description	Best for	Notes
1	Wrench only	Measuring and control applications	Biggest range of output rate and cutoff frequency for Wrench. From 10hz to 3840hz
2	Wrench and IMU		Allowing to measure Wrench and IMU data with different filtering
6	FANUC mode	For FANUC platforms	It allows a seamless integration of the sensor in the FANUC environment. It will be set automatically by choosing the corresponding communication interface.

2.2.1.4. Application submodes

Each of these APPLICATION MODES has a set of possible configurations called **APPLICATION SUBMODES,** allowing the user to select the best signal filtering for the application. The different submodes apply a different low pass filtering, selecting the update rate, the cut-frequency of the signal and the latency.

IMPORTANT

For use in control applications is recommended to match the sensors update rate to the update rate of the control loop. The web configurator app.botasys.com can be used to select the corresponding application mode and submode for a given update rate.

The Table 4 includes all the possible APPLICATION SUBMODES configurations. The update rate refers to the frequency of the sensor generating data. Latency is the time delay from the physical application of the force to the generation of the corresponding response, it is introduced by the filter and is not including the communication interface latency.

TABLE 4, APPLICATION SUBMODES

App submode	Update rate(Hz)	Notes	Filter type	Cutoff frequency (Hz)	Latency (ms) not including the interface
0	10		Sinc3	2.623	300.2
1	20		Sinc3	5.237	150.2
2	25		Sinc3	6.549	120.2
3	50		Sinc3	13.09	60.15
4	100		Sinc3	26.2	30.15
5	200		Sinc3	52.49	15.15
6	250		Sinc3	65.49	12.19
7	270		Sinc3	71.05	11.25
8	400		Sinc3	105.4	7.655
9	500		Sinc3	133.4	6.092
10	800		Sinc3	212.2	3.905
11	1000	Not supported by MiniONE	Sinc3	268.9	3.123
12	1600	Not supported by MiniONE	Sinc3	430.0	2.03
13	2133	Not supported by MiniONE	Sinc3	578.5	1.561
14	3200	Not supported by MiniONE	Sinc3	882.8	1.092
15	3840	Not supported by MiniONE	Sinc3	1070	0.9359

16	10		Sinc4	2.277	400.2
17	20		Sinc4	4.544	200.2
18	25		Sinc4	5.688	160.2
19	50		Sinc4	11.37	80.15
20	100		Sinc4	22.75	40.15
21	200		Sinc4	45.55	20.15
22	250		Sinc4	56.82	16.2
23	270		Sinc4	61.64	14.95
24	400		Sinc4	91.33	10.15
25	500		Sinc4	115.6	8.071
26	800		Sinc4	183.7	5.155
27	1000	Not supported by MiniONE	Sinc4	232.6	4.113
28	1600	Not supported by MiniONE	Sinc4	370.9	2.65
29	2133	Not supported by MiniONE	Sinc4	497.8	2.03
30	3200	Not supported by MiniONE	Sinc4	756.3	1.405
31	3840	Not supported by MiniONE	Sinc4	914.2	1.196

2.2.1.5. Update rate

This parameter is read only. It is intended to be used by the driver or the interface software to extract the update rate of the sensor based on the current configuration. Upon correct configuration it will return the values of the update rate from the Table 4.

2.2.1.6. Action request

Except for sensor parameters a set of actions can be performed using the parameter action request and action Error code. The available actions are summarized in the Table 5. The listed actions are only available in Config State.

Writing one of the following numbers to the action request parameter will trigger an action that can last up to 10 sec and the minimum time between commands should be greater than 10ms. Only after the action is completed will the device transmit the response command. If the response command has the error "action failed" the user can use the parameter "action error code" for detailed reasoning.

The "action error code" parameter is clear upon triggering a new action. To clear the error code the idle action can be requested.

TABLE 5, ACTION REQUESTS

Request #	Action description	Action error code
0	Idle (always succeeds used to reset error)	0: Success
1	Save operation parameters	0: Success 1: EEPROM Error
2	Save communication parameters	0: Success 1: EEPROM Error
3	Request a single wrench (it will update the single read Wrench) It is used to tare the sensor and test that the sensor measurements are valid before starting.	0: Success 1: Fail, check error code will be in the form of Status from live data syntax
4	Load default communication parameters (but not save)	0: Success 1: Load Failed
5	Load default operation parameters (but not save)	0: Success 1: Load Failed
6	Load Saved Communication parameters	0: Success 1: EEPROM Error
7	Load Saved Operation parameters	0: Success 1: EEPROM Error

2.2.1.7. Single read Wrench

When one single measurement is required while in Config state (before starting), the user can trigger the Request single Wrench action to get a single measurement. The wrench (vector of 6) can be read from this parameter after the action is completed successfully. The generated Wrench will be sampled with Application mode 1 and submode 16 to get the highest resolution see Table 3 and Table 4. For each protocol reading this parameter is slightly different. Refer to the specific protocol chapter for it.

The result of this parameter can be used by the user to tare the sensor before entering Run state or test that the sensor measurements are valid before transitioning to Run state. Refer to chapter Wrench offset for more details.

2.2.2. Communication parameters

These parameters modify the different communication interfaces supported to ensure system compatibility.

2.2.2.1. Primary and USB protocol index

These parameters are used to change the primary or USB communication interface accordingly. The user can modify this parameter while in Config state and to apply them the parameters must be saved by requesting the "save communication parameters" action (see chapter 2.2.1.6) and the device need to be restarted either by power cycling or performing and CONFIG to INIT transition with the state machine.

Some sensors come with more than one cable connector, the correct connection should be used for the selected Primary communication interface. If USB is used as primary any connector can be used.

The list of available interfaces and their corresponding index can be found in Table 11**.**

2.2.2.2. Ethernet IP, Subnet, Gateway, DNS

These parameters are specific to the communication interfaces that are using Ethernet as hardware interface, specifically: Modbus TCP, Bota Socket, etc. They are all formatted the same way as IP addresses and each of these parameters has 4 sub-ids corresponding to the 4 parts of the IP addresses. Each part is expressed as uint8 and has a range of 0-255.

Example, changing DNS to 12.23.34.45:

- Modify parameter 13:1 to 12
- Modify parameter 13:2 to 23
- Modify parameter 13:3 to 34
- Modify parameter 13:4 to 45

The user can modify these parameters while in Config state and to apply them the parameters have be saved by requesting the "save communication parameters" action (see chapter 2.2.1.6) and the device need to be restarted either by power cycling or performing CONFIG to INIT transition with the state machine.

2.2.2.3. Ethernet UDP broadcast port and IP

These parameters are specific to the communication interface Bota Socket. Intended to be used when several sensors are used in the same network or the UDP port is used already by another application. The Broadcast IP can be used when the sensor is connected on a network behind a gateway. The IP of the

device that is going to receive the data (where the driver software is running) should be used so the data can be forwarded to it. The format of the broadcast IP is as with the above parameters. Same as for the chapter above, the parameters needs to e saved and the sensor has to cycle through init or restart to apply them (see chapter 2.2.2.2.)

2.2.2.4. Bota protocols output rate

All Bota protocols provide measurements in selected output rate. The user can change this rate to match the requirements of the application by modifying the Application mode and submode of the device, this way the best performance of the sensor can be achieved. Each submode, has a specific refresh rate and latency if the application requirements don't fit this combination of refresh rate and latency the user can override the refresh rate by modifying this parameter. By default, this parameter is set to 0 that means that the output rate is the same as the refresh rate.

Example,
Requirement to have latency of 30ms and output rate of 30Hz the user will have to set

- Submode: 4
- Output rate: 30

2.2.2.5. Serial Baudrate

This parameter is used to modify the Baudrate of all communication interfaces that are using RS422/RS485/RS232 as hardware interface, specifically: Modbus RTU, Bota protocols family, etc. The user can modify this parameter while in Config state and to apply it the parameter has be saved by requesting the "save communication parameters" action (see chapter 2.2.1.6) and the device need to be restarted either by power cycling or performing and CONFIG to INIT transition with the state machine. In the Table 6 the available options can be found. The baudrate should be selected appropriately to provide enough bandwidth for the data transmission. If data are throttled by the baudrate the sensors will provide a warning on the status register. See Table 8

TABLE 6, BAUDRATES

Baud Rate Index	Baud Rate (bps)	Max output rate (Hz)
0	9600	25
1	57600	150
2	115200	300
3	230400	600
4 (default)	460800	1250
5	921600	2500
6	250000	600
7	500000	1250
8	1000000	2500
9	2000000	5000

2.2.2.6. Modbus slave id

This parameter is used to modify the slave id used by Modbus RTU and Modbus TCP communication interface. The slave id can be set from 1 to 255 and the default value is 1. The user can modify this parameter while in Config state and to apply it the parameter has be saved by requesting the "save communication parameters" action (see chapter 2.2.1.6) and the device need to be restarted either by power cycling or performing and CONFIG to INIT transition with the state machine.

2.2.2.7. Serial Standard

The sensor supports RS422 and RS232 for all Bota Protocols (Bota binary, Bota ASCII, Bota Quiet) and Modbus RTU. Additionally, when multidrop connectivity is required RS485 can be selected as field bus. RS485 is only supported by Modbus RTU. RS485 is a 2-wire protocol (RS485 4-wire is not supported)

2.2.2.8. RS422/RS485 Termination

Enables a terminating resistor of 120Ohm on the sensor receiver side between RX+ and RX- when in RS422 standard. Alternatively, between A and B when in RS485 standard. The standard can be selected.

2.2.3. Manufacturer Parameters

Manufacturer parameters are protected the user is not allowed to modify them or will void the sensor's warranty. They will not need to be modified if the sensor is operated under normal conditions. In case the sensor is overloaded, the device must be returned to the manufacturer for recalibration and the manufacturer parameters will be changed.

2.3. Live data

The sensor will transmit live data while in RUN state, depending on the Application mode selected (see chapter 2.2.1.3) the data consists of the Status, Wrench, timestamp, IMU data and temperature. The syntax and the representation of the Live data are described in the following chapters as they are different for each communication interface.

TABLE 7, LIVE DATA

data	Description	Units
Status	Refer to Status Table 8 in chapter Live data syntax	Bit Flags
Force X	X component of force	N
Force Y	Y component of force	N
Force Z	Z component of force	N
Torque X	X component of Torque	Nm
Torque Y	Y component of Torque	Nm
Torque Z	Z component of Torque	Nm
Acceleration X	X component of Acceleration in m/s^2	m/s^2
Acceleration Y	Y component of Acceleration in m/s^2	m/s^2
Acceleration Z	Z component of Acceleration in m/s^2	m/s^2
Angular rate X	X component of Rotation in rad/s	rad/s
Angular rate Y	Y component of Rotation in rad/s	rad/s
Angular rate Z	Z component of Rotation in rad/s	rad/s
Timestamp	Timestamp from powerup	microseconds
Temperature	Temperature of the sensor	degrees C

2.3.1. Wrench

It consists of the 3 components of the force and torque acting on the sensor measured at the wrench frame.

2.3.2. Acceleration

It consists of the 3 components of the acceleration acting on the sensor measured at the IMU frame.

2.3.3. Angular rate

It consists of the 3 components of the angular velocity of the sensor measured at the IMU frame.

2.3.4. Timestamp

Is the time stamp of the wrench measurements measured in microseconds from powering up the sensor. It can be used to ensure no packages are dropped and evaluate the Update rate selected by the parameters.

2.3.5. Status

When the sensor is used in a closed loop control application the status of the live data must be constantly monitored to ensure safe operation of the system. The status consists of status flags represented by bits. Each flag has a severity level, and a safety stop should be performed in case the severity is Error. The list with all flags is on the Table 8.

TABLE 8, STATUS FLAGS

Bit#	Description	Status 0=normal	Severity	Reason
0	Measurements are throttled	1: the bandwidth of the communication protocol is lower than the sensors output rate	Warning	Baudrate too low
1	Overrange	1: one or more axis has exceeded the rated load. This check is ignoring the Wrench offsets set.	Warning	Application exceeds forces in Fx
2	Invalid measurement	1: The Wrench (force and torques) are not valid should not be used	Error	Overload of the sensor or permanent damage
3	Raw measurements	1: live data is raw measurements used for debug	Warning	Application mode is set wrongly
4-15	reserved			

2.4. LED Indicators

Each sensor is equipped with LEDs to indicate its state and communication activity. The LEDs can be on the sensor body or the connector of the device.

2.4.1. State machine indicators

Each Sensor is communicating the state machine state through a green LED and an error State through a red LED. The blink pattern is according to the EtherCAT standard, and it is defined below.

The Green LED is marked on the sensors as “RUN” and the red LED is marked as “ERR”.

TABLE 9, LED STATES

State	Green LED	Red LED
Init	Off	On if any state error
Config	Blinking slowly	On if any state error
Run	Continuous on	On if any state error or Measurements error

2.4.2. Connection indicators

For each of the supported hardware interfaces the sensor provides connection information through a yellow and a green LED. The Table 10 describes the functionality of the LEDs for each hardware interface used.

TABLE 10, LED COMMUNICATION

	EtherCAT	Ethernet	RS422/RS485	USB
Yellow LED	**L/A** Link and activity	Activity	Sensor is transmitting	-
Green LED	-	Link	Sensor is receiving	-

3. Communication Interfaces

Each sensor supports one Primary communication interface that can be selected by the user, and a USB communication interface that is always present and can be used for configuring the sensor and reading the measurements. The USB interface can be used as a primary interface and is recommended for the easiest possible installation.

The user can choose from Table 11 the hardware interfaces and the specific protocol, each hardware interface is available on a specific sensor connector except of USB that is available in every connector. Each sensor model can support all or a subset of the Hardware interfaces described in the Table 11. See the product datasheet for the available options and the Table 2 for all options.

TABLE 11, COMMUNICATION INTERFACES

Hardware interface	Connector Marking	#: Primary Communication Interface options	#. USB Interface options
RS422/RS485 /RS232* **According to Product model. See table 2 to select*	*SERIAL*	**0**: Bota Binary (RS422/232 selectable) **1**: Bota ASCII (RS422/232 selectable) **3**: Modbus RTU 2 wire (RS485 only) **4**: Fanuc FT sensor (on request)	**0**: Bota Binary over VCP[1] **1**: Bota ASCII over VCP **2**: Bota Quiet over VCP
Ethernet	*ENET*	**5**: Bota Socket **6**: Modbus TCP **7**: Ethernet IP [COMING SOON]	
EtherCAT in or in/out	ECAT IN ECAT OUT	**8**: CANopen over EtherCAT (CoE)	
USB	All connectors support USB	**2**: USB as Primary Interface. (disables the Primary interface) The USB can be used, see next column	

All sensors are available in multiple configurations, supporting different hardware interfaces. The Table 12 summarizes the communication interfaces supported by each sensor configuration. For ordering details contact sales.

TABLE 12, PRODUCT SUPPORT

Sensor	USB	RS422/485/232	UART TTL	Ethernet	EtherCAT In	EtherCAT Out
MiniONE	USB &UART	UART	UART			
MiniONE PRO	Industrial & EtherCAT	Industrial		Industrial	EtherCAT	
Rokubi /Medusa	Industrial & EtherCAT	Industrial		Industrial	EtherCAT	EtherCAT
SensONE	Industrial & EtherCAT	Industrial		Industrial	EtherCAT	EtherCAT
LaxONE	Industrial & EtherCAT	Industrial		Industrial	EtherCAT	EtherCAT
MegaONE	Industrial & EtherCAT	Industrial		Industrial	EtherCAT	EtherCAT
PixONE T25	Industrial & EtherCAT	Industrial	Industrial & EtherCAT	Industrial	EtherCAT	EtherCAT
PixONE T60	Multiprotocol	Multiprotocol	Multiprotocol	Multiprotocol	Multiprotocol	Multiprotocol

In the chapters below all the **Communication interfaces** are described. Specifically, how the data are being transmitted (timing and structure), how to modify the sensor parameters and access the sensor's state machine over a communication interface. Both the **configuration syntax** and the **live data syntax** are implemented for all communication interfaces.

[1] Virtual com port

3.1. Bota Protocol family

Bota protocols is a proprietary family of protocols developed by Bota Systems. It can operate over USB or RS422/232, and there is also a variant for Ethernet (Bota Socket). The all share the same **configuration syntax** but have different **live data syntax.**

In all Bota protocols (except for Bota Quiet) the sensor is transmitting **live data** in packets with the option to throttle the data rate. For this family the data output rate can be set by the user or set 0 zero and the sensor will automatically provide the latest measurements as soon as possible meaning the **Output rate = Update rate**. For more details see chapter 3.1.2**.**

Bota Binary, Bota ASCII, Bota quiet are supported only by USB and RS422/RS232. The difference between them is the **live data** encoding. **Binary** is using a data packet for live data as compact as possible; **ASCII** is useful for human readable output and **Quiet** does not transmit **live data** in RUN state. It is intended to be used as a configuration channel mainly by USB for commissioning.

Bota Socket is supported only over Ethernet. This protocol uses two different connections, a TCP connection for the configuration syntax, with the same syntax as all other Bota protocols. A UDP connection is used for the transmission of the **live data syntax** in binary encoded packets.

3.1.1. Configuration syntax

The user can modify or read parameters in two ways. One is human readable values in ASCII where the value is represented as a real number or in HEX format where the value will be parsed based on the type/representation of the parameter that is shown in the corresponding column of the Table 2.

Below is described the syntax of requesting to modify or read a parameter and the corresponding response to these requests.

TABLE 13, BOTA PROTOCOLS CONFIGURATION SYNTAX

Description	Direction	Syntax	Example
Modify parameter value in hex format	Request Transmitting to the device	<request id>,<param_id>,<param_subid>,<value>\n request_id wh: modify value in hex format rh: read value in hex format	Set temperature coefficient of Fz to 2.132 Request: wh,5,3,400872B0\n Successful response Response: wh,0,400872B0\n
read parameter value in hex format		wa: modify value in ASCII format ra: read value in ASCII format param_id, Param_subid refer to Table 2	read the temperature coefficient of Fz Request: rh,5,3,0\n The successful response is 2.132 Response: rh,0,400872B0\n
Modify parameter value request in ASCII format		value if Hex format number of bytes should be extracted from type. If ASCII a string representation of the number. If read command this value is ignored.	Set temperature coefficient of Fz to 2.132 Request: wa,5,3,2.132\n Successful response Response: wa,0,2.132\n
read parameter value request in ASCII format			read the temperature coefficient of Fz Request: ra,5,3,0\n The successful response is 2.132 Response: ra,0,2.132\n

Response of all requests except read parameter	Response Receiving from the device	<request_id>,<Status>,<value>\n request_id wh: modify value in hex format rh: read value in hex format wa: modify value in ASCII format ra: read value in ASCII format Status 0: success 1: Wrong State 2: syntax error (timeout) value will be 0 zero 3: read only 4: Write only 16: invalid value (out of bounds) 17: action failed (check action error code) 18: Invalid ID 19: Invalid SubID value if Hex format number of bytes should be extracted from type. If ASCII a string representation of the number. If read command this value is ignored. - if Status 1 to 4 value is zero - if Status 16 to 19 value is the actual value of the param in sensor's memory. If modify request failed. - if is a response to "read parameter" is the current value of the requested parameter	read the temperature coefficient of Fz Request: ra,5,3,0\n The successful response is 2.132 Response: ra,0,2.132\n or an Error example read the temperature coefficient of unknown subid Request: ra,5,8,0\n The error response is invalid subid Response: ra,19,8\n

3.1.2. Live data syntax

3.1.2.1. Bota Binary Protocol

It is using a binary data packet structure for live data to be as compact as possible for real-time application and maximum bandwidth. The Table 14 and Table 15 describe the **live data** packet structure for the different application modes.

Wrench only application mode

TABLE 14

Section	data	Data Type	Size (Bytes)
Header	Identifier (0xAA)	Uint8	1
Data	Status	Uint16	2
Data	Force X	Float (LSB)	4
Data	Force Y	Float (LSB)	4
Data	Force Z	Float (LSB)	4
Data	Torque X	Float (LSB)	4
Data	Torque Y	Float (LSB)	4
Data	Torque Z	Float (LSB)	4
Data	Timestamp	Uint32 (LSB)	4
Data	Temperature	Float (LSB)	4
Checksum	CRC	Uint16 (LSB)	2

Wrench and IMU application modes

TABLE 15

Section	data	Data Type	Size (Bytes)
Header	Identifier (0xAB)	Uint8	1
Data	Status	Uint16	2
Data	Force X	Float (LSB)	4
Data	Force Y	Float (LSB)	4
Data	Force Z	Float (LSB)	4
Data	Torque X	Float (LSB)	4
Data	Torque Y	Float (LSB)	4
Data	Torque Z	Float (LSB)	4
Data	Timestamp	Uint32 (LSB)	4
Data	Temperature	Float (LSB)	4
Data	Acceleration X	Float (LSB)	4
Data	Acceleration Y	Float (LSB)	4
Data	Acceleration Z	Float (LSB)	4
Data	Angular rate X	Float (LSB)	4
Data	Angular rate Y	Float (LSB)	4
Data	Angular rate Z	Float (LSB)	4
Checksum	CRC	Uint16 (LSB)	2

CRC is calculated for the data section only (excluding header) and using the CRC-16-CCITT (Kermit) Implementation.

Characteristics of CRC

- **Polynomial**: 0x1021 (reversed as 0x8408)
- **Initial Value**: 0xFFFF
- **Final XOR**: 0xFFFF
- **Bit-wise processing**: Yes (LSB-first)

This is a widely used CRC-16 variation known as **CRC-16/KERMIT** or **CRC-16/CCITT (reverse)**.

3.1.2.2. Bota ASCII Protocol

It is using a human readable **live data** packet for easy way to log and observe data; the bandwidth of this protocol is limited so it is only recommended for measuring applications and low output rates. The Table 16 describes the **live data** packet structure for the different application modes the data are tab ("\t") separated and arrive in lines ("\n").

Wrench only application mode.

TABLE 16

Section	data	Representation	Encoding
Data	Status	integer	decimal
Data	Force X	Real	variable decimal points
Data	Force Y	Real	variable decimal points
Data	Force Z	Real	variable decimal points
Data	Torque X	Real	variable decimal points
Data	Torque Y	Real	variable decimal points
Data	Torque Z	Real	variable decimal points
Data	Timestamp	Integer	variable decimal points
Data	Temperature	real	variable decimal points
Header	Delimiter		\n

The packet is not going be always the same length because the integer and real representations are not always the same length. Each packet ends with a new line character "\n".

Example packet

6 0.5 12.5 200.3 0.012 0.025 1.32 12223554 25.5

Status: 6, Overrange and invalid measurements flags set
Fx: 0.5N
Fy: 12.5N
Fz: 200.3N
Tx: 0.012Nm
Ty: 0.025Nm
Tz: 1.32Nm
Timestamp: 12223554 us after powerup
Temperature: 25.5 degrees Celsius

Wrench and IMU application modes

TABLE 17

Section	data	Representation	Encoding
Data	Status	integer	decimal
Data	Force X	real	variable decimal points
Data	Force Y	real	variable decimal points
Data	Force Z	real	variable decimal points
Data	Torque X	real	variable decimal points
Data	Torque Y	real	variable decimal points
Data	Torque Z	real	variable decimal points
Data	Timestamp	Integer	variable decimal points
Data	Temperature	real	variable decimal points
Data	Acceleration X	real	variable decimal points
Data	Acceleration Y	real	variable decimal points
Data	Acceleration Z	real	variable decimal points
Data	Angular rate X	real	variable decimal points
Data	Angular rate Y	real	variable decimal points
Data	Angular rate Z	real	variable decimal points
Header	Delimiter		\n

3.1.2.3. Bota Quiet Protocol

Bota Quiet is suitable when the USB interface is used only for configuration purpose. It is not transmitting any **live data** while in RUN state.

3.1.2.4. Bota Socket Protocol

This protocol is the Ethernet Implementation of Bota Binary. In this protocol a TCP connection is used for the **configuration syntax** on port 23 (telnet). When the sensor is in RUN state the live data are being broadcasted on UDP port 30302 in binary packets according to the Table 14 and Table 15.

3.2. Modbus

Modbus communication protocol is available over RS485 or Ethernet hardware interface.

3.2.1. Configuration syntax

The device parameters are mapped to Modbus registers. The register address can be found in 3rd column of the Table 2. Function 3 can be used to read and 6 or 16 to modify the parameters.

3.2.2. Live data Syntax

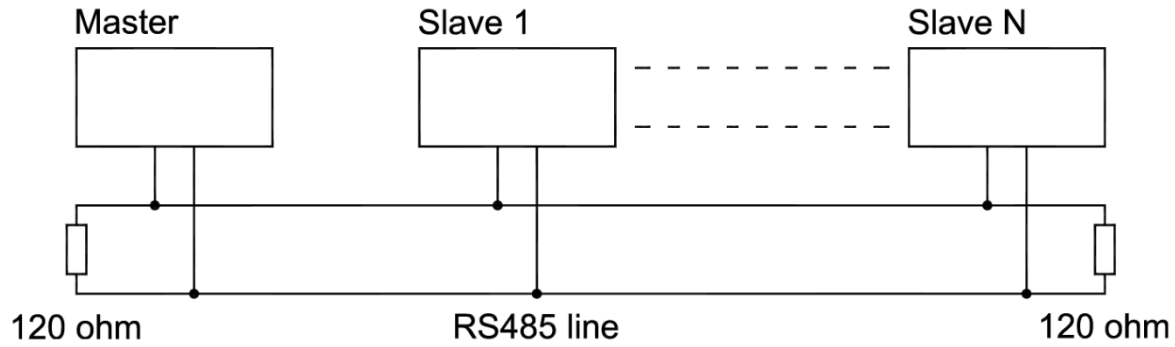
The device Live data are mapped to Modbus registers, the register address can be found on the Table 18. If application mode 1 is selected (Wrench only) the user should read the addresses 1 to 14 using function 3. If application mode 2 is selected (Wrench and IMU) the user should read the addresses 1 to 26 using function 3. Live data can be read from the sensor in Run state. Alternatively in config state after single read Wrench request see chapter 2.2.1.7.

TABLE 18

Register address	data	Data Type	Access
0	Status	16 bits	Read Only
1	Force X	Float	Read Only
3	Force Y	Float	Read Only
5	Force Z	Float	Read Only
7	Torque X	Float	Read Only
9	Torque Y	Float	Read Only
11	Torque Z	Float	Read Only
13	Timestamp	Uint32	Read Only
15	Temperature	Float	Read Only
17	Acceleration X	Float	Read Only
19	Acceleration Y	Float	Read Only
21	Acceleration Z	Float	Read Only
23	Angular rate X	Float	Read Only
25	Angular rate Y	Float	Read Only
27	Angular rate Z	Float	Read Only

3.2.3. Modbus RTU

Modbus RTU is implemented over the RS422/RS485 hardware interface. When this protocol is selected the hardware interface switches to **RS485** mode using only two data wires (Y and Z) to enable BUS style connectivity between slaves. Terminating resistance should be used at both ends of the bus.



3.2.4. Modbus TCP

Modbus TCP uses ethernet communication and it is possible to operate in any network configuration. Master and slave device must be in the same subnet or configured correctly. The network configuration is very critical as it affects the latency of the systems so it is recommended to check the latency of the systems before being used in a control application.

Modbus TCP is using port 502 so communication through this port should be allowed by networking devices (firewalls, routers).

3.3. EtherCAT CoE

3.3.1. State machine access

CanOpen over EtherCAT protocol is a fully defined protocol up to the Application layer. EtherCAT SDO is used for **configuration syntax** and PDO for **live data syntax**. The state machine is being controlled by the native EtherCAT state machine, and it is synchronized based on the Table 19.

TABLE 19

EtherCAT States	Sensor States
Init	Init
Preop	Config
SafeOP	Run
OP	Run

3.3.2. Configuration syntax

The device parameters are mapped to EtherCAT Object dictionary. The objects ID and SID can be found in 2nd column of the Table 2 and the Service Data Object (SDO) can be used to modify them.

3.3.3. Live data Syntax (PDO mapping)

The device Live data are mapped to EtherCAT Object dictionary that is mapped to the process data. The process data objects (PDO) mapping is fixed to the object 0x6000. The Object can be found on the Table 20. If application mode 1 is selected (Wrench only) the IMU acceleration and Rotation data will not be updated. If application mode 2 is selected all data is updated. Live data can be read from the sensor using PDO in Run/OP. Alternatively in config/PreOP state using SDO after single read Wrench request see chapter 2.2.1.7.

TABLE 20

Object address	Data Type	Access	data
0x6000:0x01	Uint16	Read Only	Status
0x6000:0x02	Real	Read Only	Force X
0x6000:0x03	Real	Read Only	Force Y
0x6000:0x04	Real	Read Only	Force Z
0x6000:0x05	Real	Read Only	Torque X
0x6000:0x06	Real	Read Only	Torque Y
0x6000:0x07	Real	Read Only	Torque Z
0x6000:0x08	Uint32	Read Only	Timestamp

0x6000:0x09	Real	Read Only	Temperature
0x6000:0x0A	Real	Read Only	Acceleration X
0x6000:0x0B	Real	Read Only	Acceleration Y
0x6000:0x0C	Real	Read Only	Acceleration Z
0x6000:0x0D	Real	Read Only	Angular rate X
0x6000:0x0E	Real	Read Only	Angular rate Y
0x6000:0x0F	Real	Read Only	Angular rate Z

4. Getting Started (Commissioning)

4.1. Safety & prerequisites

The user should verify that the force torque sensor is rated for maximum loads and torques expected from the operation. The user should be aware of the dynamic loads caused by the robot during acceleration or deceleration of the mounted masses.

CAUTION

Proper sensor mounting is critical to ensure optimal performance. The sensor must be installed using all specified fasteners and tightened to the recommended torque. The mounting surface must be a rigid material (e.g., steel or aluminum) and must be free from dust or debris before assembly. **See sensor's datasheet**

4.2. Mechanical & electrical installation

4.2.1. Sensor and cabling mounting

WARNING

Proper sensor mounting is critical to ensure the stated sensor performance. Failure to comply with the recommendation and good practices described in this chapter can lead to abnormal sensor operation or damage

The sensor must be installed using all specified fasteners and tightened to the recommended torque. The mounting surface must be a rigid material (e.g., steel or aluminum) and must be free from dust or debris before assembly. **See products drawing**

Mechanical installation of the sensor is crucial for the performance of the sensor. Bota systems recommend purchasing sensors as part of the **robot kit** offering to ensure proper mounting and shorter time-to-market. The kits include all necessary mechanical adapters, fasteners, cabling, cabling management and software for integration the sensor to a specific robotic system.

Cabling for the end effector like vacuum hoses or heavy cables must be mounted on both sides of the sensor rigidly to avoid inducing forces by relative movement of them.

The sensor cable should be routed such that there is no stress applied to it. Stress can be induced from pulling, bending, kinking. The cable jacket material should be protected from sharp edges and hot surfaces. The cable near the sensor connector should be restricted as stressing the connector mounting can affect the measurements. To connect the M8 end of the cable to the sensor, align the slot with the key to the connector and apply gentle force. Do not rotate the connector while connecting to the sensors as it could bend the pins of the sensor's connector. The static bending radius of the cable should not be less than 10 times the sensors cable diameter, for non-static mounting (for example energy chain) the turning radius should be more than 25 times the diameter.

When designing a custom adapter for mounting the sensor to a robot/system the following points need to be considered:

- The adapter should feature all fasteners and alignment features of the sensor, to ensure the sensor's measuring and positional accuracy.

- The adapter should use the whole mounting surface of the sensor to ensure sufficient friction between the two surfaces.
- Sufficient thread length should be provided to be properly mounted on the robot/system. The adapter should be designed and calculated to bear the overload forces/torques of the sensor at least.
- Cable management should be considered when designing custom adapters as tagging on the sensor cable can cause parasitic measurements

4.2.2. Power Supply and grounding

The sensor can be supplied from a wide range of voltage, the rated voltage supply is 5 to 60 volts for the dedicated power supply and 6 to 60V for the PoE power supply, with a power consumption of 1.5Watts. The sensors support Depending on the hardware interface used the power can be supplied separately or on the data lines.

The sensor is recommended to be supplied from an isolated power source for minimum noise. In the case of an isolated power source the shielding of the cable must be connected to GND at the source. Alternatively, if the sensor is supplied by the robot's or the system's power supply the user has to make sure no ground loops are created through the sensor as the sensors body is connected to the GND through 1 Mohm resistance and 100nF capacitance, in case of high ground loop current the sensor electronics can be damaged.

TABLE 21

Description	Min value	Typical range	Max value
Power supply	5V Volts	6 – 60 Volts	75 Volts
Power consumption	0.2 Watt	1.5 Watts	2 Watts
Grounding	1 Mohm 100nF to ground		

4.2.3. Communication Interface selection

Each sensor supports one or more hardware interfaces, the connection for each of them is summarized on the Table 22. The specific connector and pinout used for each sensor can be found in the sensor's datasheet. Some sensors have more than one connector, each of them dedicated to specific communication interfaces but all of the connectors support USB.

TABLE 22

Ethernet (ENET)	EtherCAT in/out (ECAT IN or OUT)	RS422 (SERIAL)	RS485 (SERIAL)	RS232 (SERIAL)	USB (all connectors)
TX+ (PoE 6-60V)	TX+ (PoE 6-60V)	TX+ (Y)	TX+/RX+ (Y)	TX+/RX+ (Y)	D+
TX- (PoE 6-60V)	TX- (PoE 6-60V)	TX- (Z)	TX-/RX- (Z)	TX-/RX- (Z)	D-
RX+ (PoE 6-60V)	RX+ (PoE 6-60V)	RX+ (A)			
RX- (PoE 6-60V)	RX- (PoE 6-60V)	RX- (B)			
Dedicated Power 5-60V	Dedicated Power 5-60V	Dedicated Power 5-60V	Dedicated Power 5-60V	Dedicated Power 5-60V	Dedicated Power 5-60V
GND	GND	GND	GND	GND	GND

4.3. Configuration for system compatibility

Primary communication interface selection

Before installing the sensor in its final system is important to make the sensor compatible with it. To ensure compatibility sensor's primary communication interface must be compatible with the system.

The first step is defining the physical layer of the communication interface also referred here as hardware interface, and also the suitable required protocol.

To configure the primary protocol, connect your sensor to a computer via USB and navigate to app.botasys.com using Chrome, Edge or other chrome-based browser. Click "Connect" and select the sensor you wish to configure. Then, click on "Configuration" and under the "Communication" tab, choose the desired interface from the "Primary Communication Interface" dropdown menu.

A set of communication parameters will appear related to the selected communication interface. Proceed with modifying the parameters according to the systems configuration.

Save the parameters and restart the sensor to apply the changes.

4.4. Configuration for Application

Using the same process with setting up the primary communication interface click on the tab "operation". There all the operation parameters can be set the user can either explicitly set the Application mode and sub mode directly or choose from a list of options like the update rate the filter type and output rate to obtain the corresponding Application mode and submode. After this the parameters must be written to the sensor using the write button and saved using the save button.

4.5. Application implementation

The sensor should be configured already to be compatible with the system through the primary communication interface.

Every time a control application starts the sensors offsets need to be adjusted. To perform this the user needs to bring the system/robot to a known state where the sensor measurements are to be expected. For example, in a sanding robot application the robot needs to go to an initial pose where the weight of the tool is aligned with Z axis and no external force is applied so the user can easily estimate the expected wrench of the sensor and calculate the initial offset.

CAUTION

When the sensor is used in a closed loop control application the status of the live data must be constantly monitored to ensure safe operation. If Invalid measurements are reported the system should safely stop or stop using the measurements of the sensor.

5. Maintenance

5.1. Regular inspection

Check regularly the sensor fasteners. Wipe of fine dust and maintain the sensor clean. Inspect for bending of the product as this can indicate that the sensor has been involved in a crash.

5.2. Re-Calibration

Regular calibration is not required for proper sensor operation. In case the sensor is used in quality certified system regular recalibration and evaluation is offered as a service.

In case the sensor has been involved in crash and the measurements are reported invalid through the status variable or the sensor has very high offsets, the sensor's accuracy might be compromised, and a recalibration or repair is required.

Contact the sales representative for more information.

5.3. Firmware update

All Bota systems' sensors are regularly maintained and can be updated to the latest version through a USB connection. The Web configuration can be used for that.

6. Troubleshooting

6.1. Force/Torque reading Issues

TABLE 23

Issue	Problem Identification
High Noise	Peaks in data readings, with the sensor untouched and at stable environmental conditions, greater than 0.2% of full-scale counts are abnormal. Noise can be caused by mechanical vibrations and electrical disturbances, and poor ground is most of the times an issue to robotic systems. Make sure that the DC supply voltage for the sensor is a regulated USB standard DC voltage. A component failure within the system can also be the issue.
Measurement drift	After a load is removed or applied, the raw gage reading does not stabilize but continues to increase or decrease. A shift in the raw gage reading is observed more easily in the resolved data mode using the bias command. Some drift from a change in temperature or mechanical coupling is normal. Mechanical coupling occurs when the sensor's degrees of freedom are restricted. The upper part is in contact with the bottom part. That can be a tape mounted between them, a wrong mounting in the system etc. See Installation section for the last one.
Hysteresis	When the sensor is loaded and then unloaded, gauge readings do not return quickly and completely to their original readings. Hysteresis is caused by mechanical coupling or internal failure.
Sensor not streaming data	The sensor should be correctly installed and powered. Refer to Installation section for more information. The sensor is not in RUN mode (see Sensor Operation Overview)

The sensor has an inherent low temperature drift and a temperature drift from external thermal sources. The two of them are combined and can be seen as an overall temperature drift. The first which is from its inside electronics power consumption can be calibrated if let the sensor work for an amount of time (15 minutes is adequate) in stabilized room temperature. The second has to do with external temperature changes. In each case, the sensor is subjected to offset phenomena and need to be calibrated (or else biased) systematically. To do this, the sensor need to be in a reference condition of temperature and force as soon as the user application demands it.

7. Tools & Software

A wide variety of software exists for all sensor devices. The compatibility of the products with the software provided is ensured per generation. Products and software in the same generation are fully compatible or can be updated to be, see chapter 5.3.

Bota systems offer different levels of integration for all kind platforms.

7.1. Discovery tool

It is a JavaScript utility that can explore a network and find connected sensors using Ethernet as primary hardware interface. The tool will provide the IP, subnet, DNS and gateway of the devices discovered. If the devices have as primary protocol the Bota socket protocol the tool can directly modify the IP, subnet, DNS and gateway of the discovered devices.

7.2. Web configurator

The Web configurator is the fastest way to configure and visualize data from the sensor. It is a web-based application that requires no installation and can directly connect to the sensor through USB.

It can visualize the data with reduced refresh rate. It can modify, read and save all parameters of the sensor. It provides a wizard to help the user select the correct application mode and submode. Finally, it provides access to the sensor documentation and can perform Firmware update for the devices.

7.3. ROS

A pre-built driver is available for ROS 2, leveraging the ROS 2 Control framework. The driver is distributed as a Debian package, allowing straightforward installation using the Ubuntu package manager.

The driver integrates seamlessly with ROS 2 Control, simplifying configuration and management of the hardware within the ROS 2 ecosystem. It follows best practices and remains aligned with ongoing ROS 2 updates. To install follow this link

7.4. Bota FT Stack

Bota FT Stack is a modular, layered software architecture for force sensing and control. It unifies the essential components into a cohesive platform, simplifying hardware integration and accelerating deployment. The solutions in Bota FT Stack are supported by a wide range of platforms and ecosystems, providing robust drivers, extensive utilities and functions, as well as containerized applications for force control.

Whether you need low-level access or ready-to-use applications, the Bota FT Stack ensures smooth and efficient experience, enabling developers to focus on their application logic rather than complex system integration.

Learn more about Bota Stack here.