TriOrb BASE 球駆動式全方向移動機構

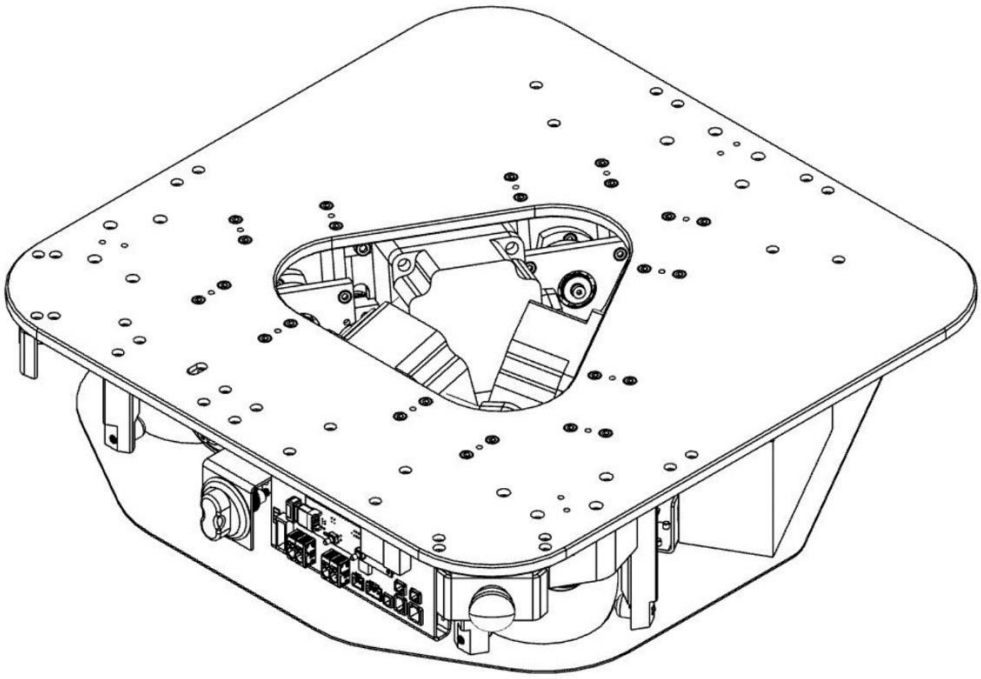
BASE500P100

取扱説明書

この度はトライオーブベースをお求めいただきまして、誠にありがとうございます。

よく読んで安全に正しくお使いください

- ご使用の前に、この取扱説明書をよくお読みの上、安全に正しくご使用ください。
- お読みになったあとは、大切に保管してください。



TriOrb

Introduction

This manual explains how to operate TriOrb BASE -BASE500P100- and precautions. Please be sure to read this before use and fully understand how to use it. In addition, please use the product after sufficiently reducing risks such as the surrounding environment. Customers are responsible for confirming compliance with laws and standards required for their usage environment.

table of contents

Introduction 1
table of contents..... 2
1. Requests before use ....Four
1.1. Safety precautions..... ....Four
2. System configuration..... ....Five
2.1. Drive unit ....Five
2.2. Auxiliary equipment 6
3. Installation method 7
3.1. Setup overview 7
3.2. Unpacking and repacking..... 7
3.3. Transport and storage..... 7
4. How to use 8
4.1. How to start..... 8
4.2. How to stop..... 8
4.3. Charging 8
4.4. Robot coordinate system..... 8
4.5. Remote control control..... 9
4.6. Program control..... 11
4.6.1. Connecting the USB cable..... 11
4.6.2. Control function list..... 11
4.6.3. UART control..... 11
Communication specifications 11
Communication data structure 12
Communication code list 12
Type definition..... 13
4.6.4. Python control..... 15
Installing the latest release..... 15
Connecting, starting and stopping the robot..... 15
Obtaining robot status..... 15
Robot posture control..... 16
Robot speed control..... 17
Changing robot settings..... 18
Example of sending and receiving communication data by specifying the communication code and value 19
5. Maintenance..... ....twenty one

5.1. Precautions during maintenance..... .....twenty one
5.2. Maintenance method ....twenty one
6. Service..... ....twenty one
6.1. Warnings..... ....twenty one
6.2. Errors..... ....twenty one
7. Specifications ....twenty three
7.1. Dimensions and weight..... ....twenty three
7.2. Basic specifications..... ....twenty three
Translational speed output characteristics ....twenty four
7.3. Accessories (optional)..... .....twenty five

1. Request before use

1.1. Safety precautions

Carefully read and understand the robot operating manual. Proper operation according to the manual will minimize the risk of accidents and injuries.

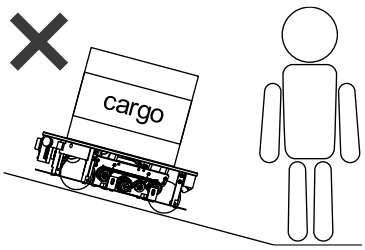
Ensure a safe working environment when using the robot. Make sure that there are no obstacles or dangerous objects in the surrounding area, and remove any elements that may interfere with robot operation. Also, maintain sufficient distance from surrounding people and workers.

Be sure to understand the limitations regarding the robot's performance and scope, and follow proper usage methods. Do not modify the robot or connect it to any electronic equipment that may damage the auxiliary equipment.

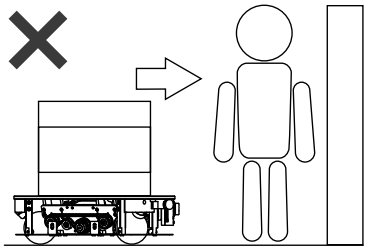
Before using the robot, check that the emergency stop function is working properly. In the unlikely event of abnormal operation, the robot can be stopped by pressing the emergency stop button.

If you notice any unusual movements, sounds, or odors from the robot, stop work immediately and report it to your administrator or manufacturer. If the problem is ignored, more serious accidents and damage may occur.

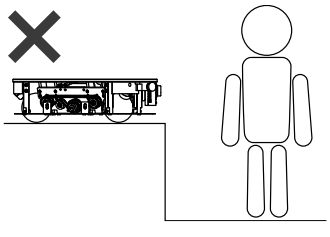
Do not use the robot in the following situations as this may increase the risk of accidents or injuries.



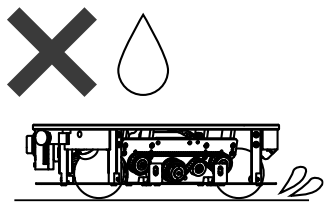
slope



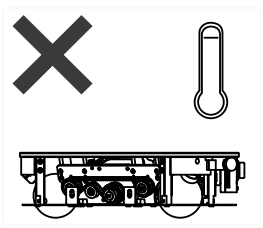
Contact with or getting caught in a robot environment of fear



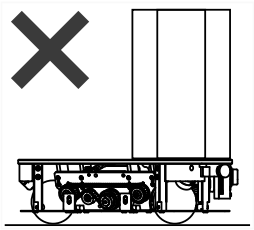
Fall danger area



Environment where there is water or oil on the running surface



high temperature environment

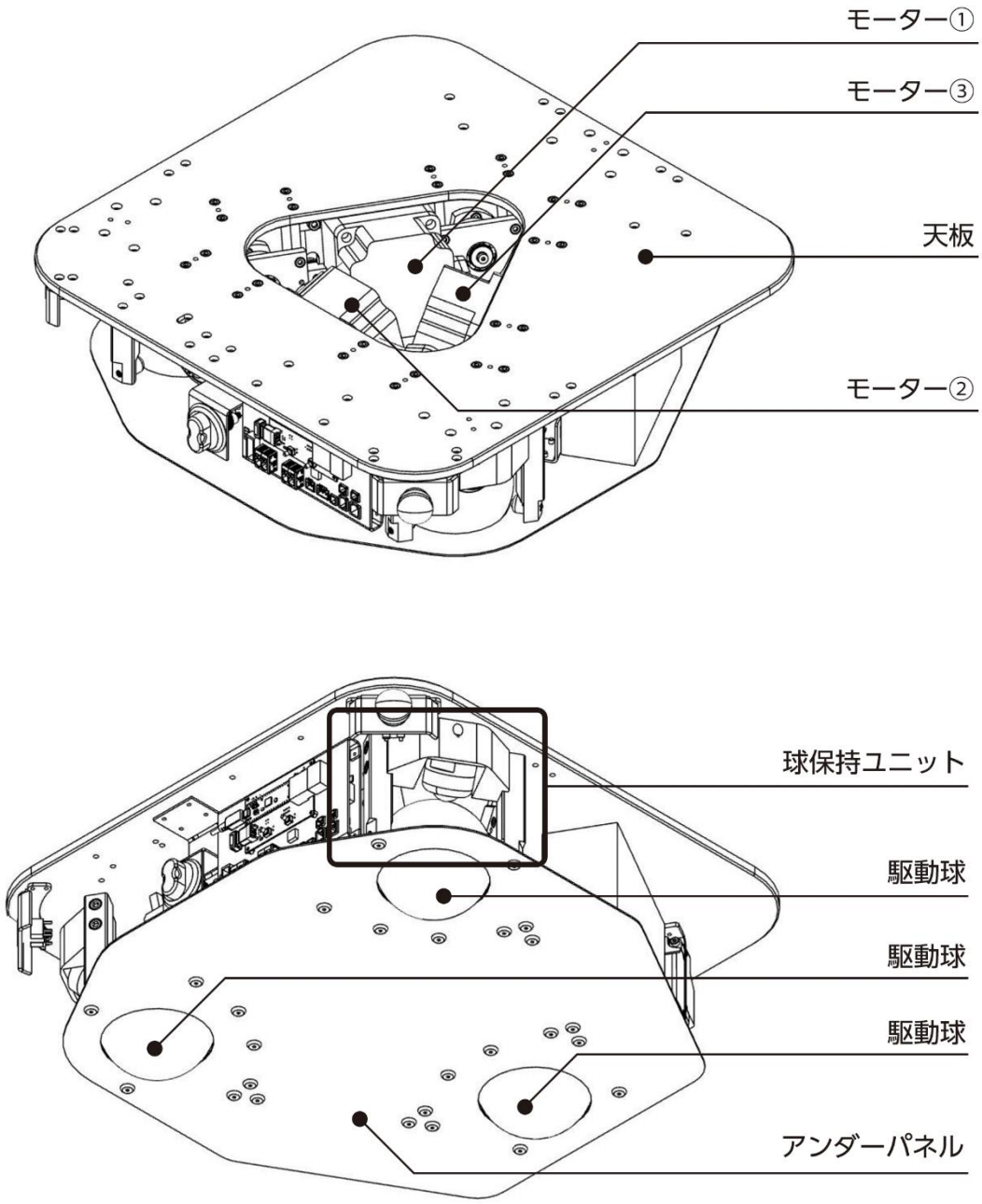


Dangerous loading methods

2. System configuration

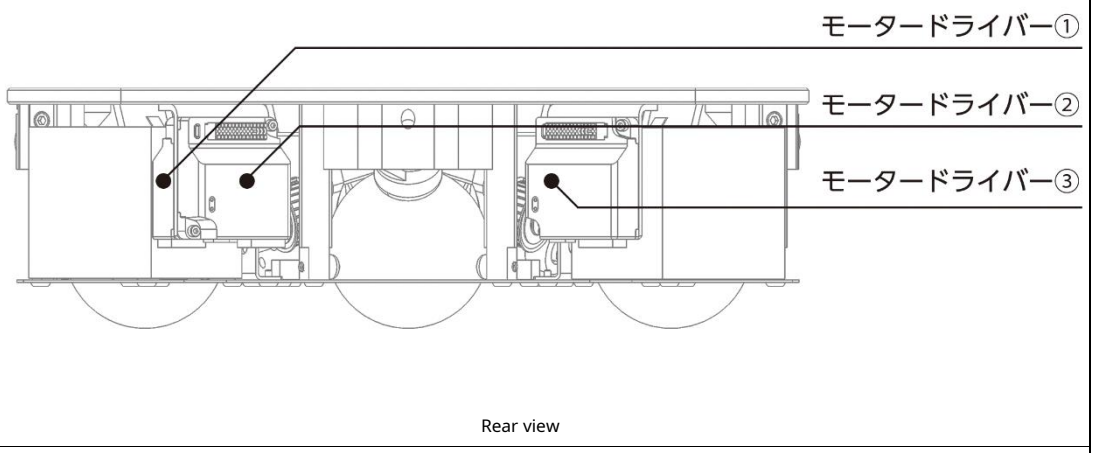
2.1. Drive section

The names of the main parts of the robot drive section are shown in the figure below. The shape of each part may change in the future.

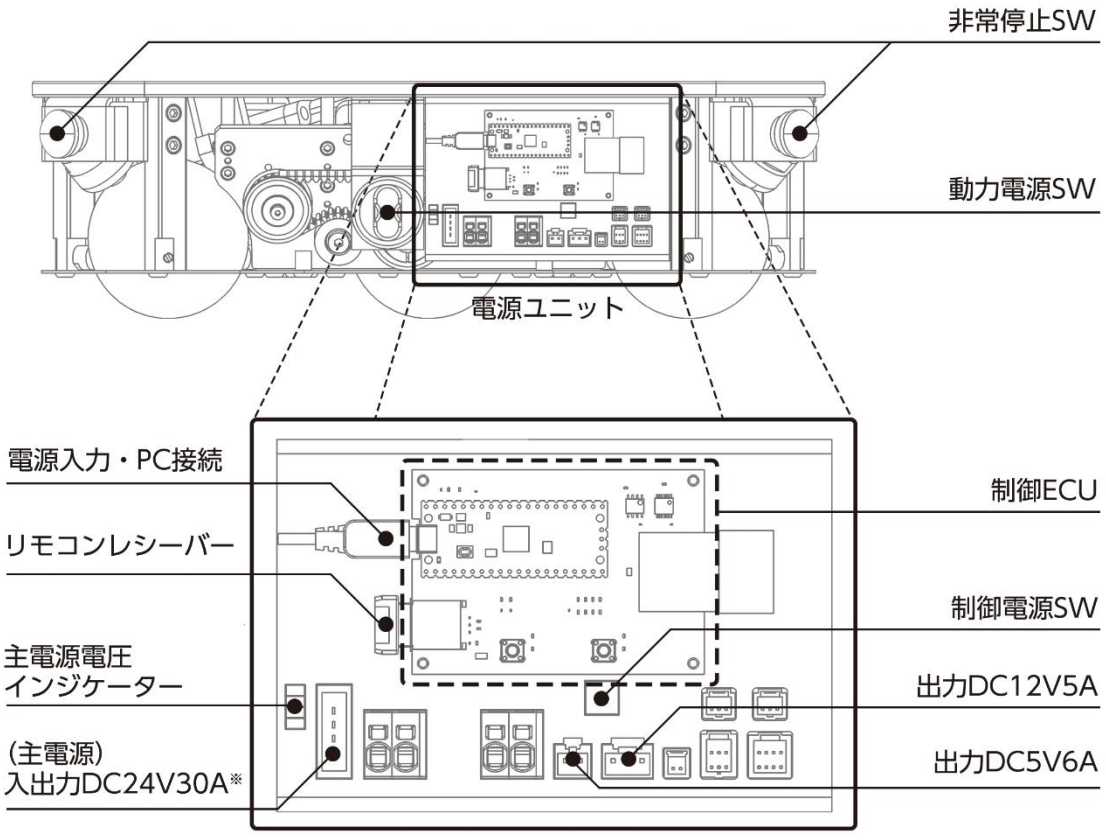


2.2. Auxiliary equipment

The part names of robot auxiliary equipment are shown in the figure below. The shape of each part may change in the future.



Rear view



Front view (power supply unit, control ECU, external device connections)

3.1. Setup overview

- Installing the battery

  Install the 2 batteries (optional) inside the robot (in the designated positions on the under panel). Connect the red and black battery connection wires in place during installation to the corresponding terminals on the battery. Be sure to connect the connection wires and terminals in the order of "black wire → red wire".(Red wire → red terminal, black wire → black terminal)

  Be very careful not to confuse the connection wires and terminals.

- Installing the remote control receiver

  Insert the receiver included with the remote control (optional) into the USB Type-A socket on the side of the control ECU. When inserting, be careful not to apply excessive force to avoid damaging the device or case.

3.2. Unpacking and repacking

Please note the following points when unpacking and repacking the robot.

- ➢ After unpacking, be sure to check that the robot is not damaged. Turning on power to a malfunctioning robot may result in unpredictable operation.
- ➢ When transporting the robot, secure a transport route and use a trolley etc. to reach the destination.
- ➢ If you need to lift the robot by hand to remove it from the packaging box, be sure to grasp only the part that can be lifted (top plate) and lift it. Electrical/electronic equipment, cables, etc. may be damaged if lifted by the wrong part.
- ➢ The robot is a heavy item, so please be careful when putting it in and taking it out of the packaging box.
- ➢ When repacking, be sure to put the robot in the packaging box and secure it in the same way as when it was delivered.
- ➢ Always remove the battery when repacking.

3.3. Transportation and storage

Please note the following points when transporting and storing the robot.

- ➢ Always remove the battery before transporting or storing it.
- ➢ If you need to lift the robot by hand during transportation or storage, be sure to grasp only the part that can be lifted (top plate) and lift it. Electrical/electronic equipment, cables, etc. may be damaged if lifted by the wrong part.
- ➢ When transporting items, secure a transport route and use a trolley etc. to reach the destination.
- ➢ After transportation, be sure to check that the robot is not damaged. Turning on power to a malfunctioning robot may result in unpredictable operation.

Do not store or install the product in the following environments.

- Places exposed to direct sunlight
- Locations where ambient temperature and humidity exceed storage conditions
- Places where the temperature changes rapidly and condensation occurs
- Location near corrosive or flammable gas
- Places with a lot of dirt, dust, or metal powder
- Locations where water, oil, chemicals, etc. may come into contact with
- Locations where vibrations and shocks are transmitted to the main unit

4. How to use

4.1. How to start

(1) Make sure the battery is connected or external power is being received.

(2) Check that the emergency stop SW is not pressed (lamp is lit).

(3) Turn on the power supply SW.

(4) Turn on the control power SW. (LED lit: green)

4.2. How to stop

(1) Turn off the control power SW. (LED off)

(2) Turn off the power supply SW.

4.3. Charging

When the battery level is low (below 18.0V), you can charge the battery using the following method.

When charging, be sure to use a charger suitable for the purpose. Using an incompatible charger may damage the battery or cause a fire.

- Charging with a charger

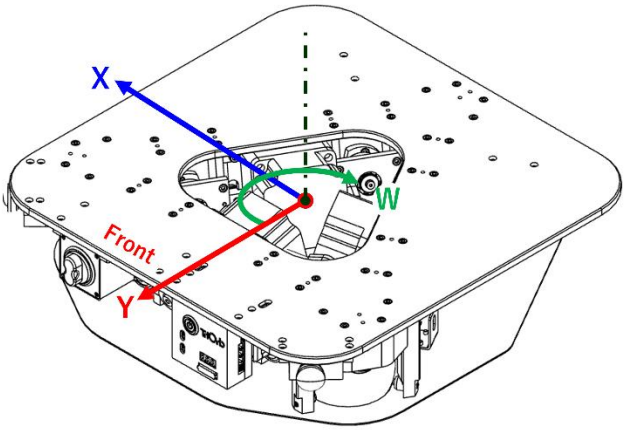
  You can remove the battery from the robot and connect the charger (optional) to charge it.

- Charging via external power supply

  With the battery connected to the robot, you can charge the battery by supplying power to the external power supply port installed on the power supply unit.

4.4. Robot coordinate system

The robot's coordinate system is shown below. Please note that the side where the power supply unit is installed is in the forward direction. Furthermore, he defined the origin of the coordinate system as the center of gravity of the equilateral triangle formed by connecting the three driving spheres.



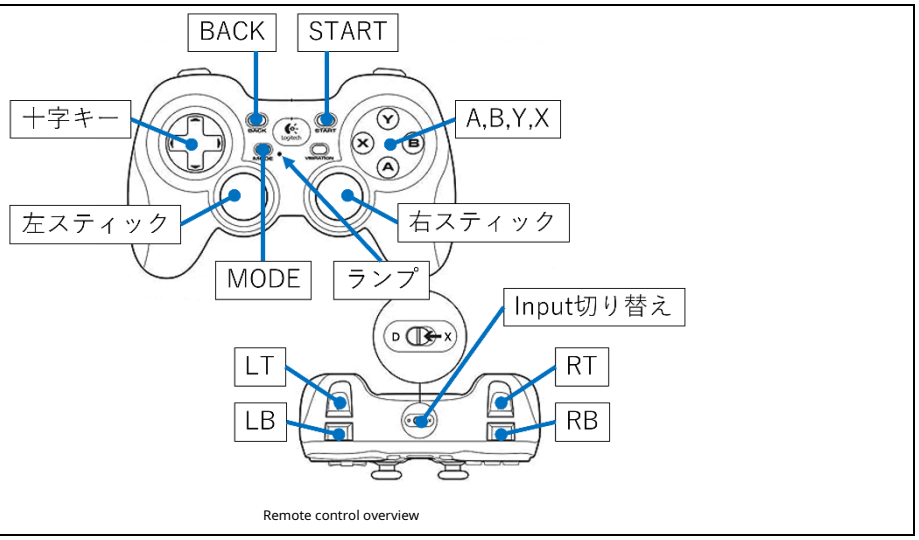
4.5. Remote control control

If a remote control receiver is connected to the control ECU, you can control the robot from the remote control. The robot can be operated if the lamp next to the MODE button (referred to below as lamp) lights up or goes off when you press a button such as START or BACK after starting the robot. If it is blinking, it means that the remote control is not recognized. If it is not recognized even after pressing the button several times, please restart it using the control power switch. Also, make sure that the Input switch on the top side of the remote control is set to D.

In order to instruct the robot to move, you must press the START button to excite the motor (current flows through the motor windings and generates torque). When in the energized state, a small energizing sound can be heard from the motor. Excitation can be canceled by pressing the BACK button.

The speed instructed to the robot changes depending on the speed setting value and the tilt of the stick that gives the speed instruction. There are 10 speed settings, and you can set values from 10 to 100% of the robot's maximum speed. The initial speed setting value at startup is 5 (50% of maximum speed). When the speed setting is set to the maximum, if the stick is tilted small, it will move at a small speed, and if it is tilted to the maximum, it will move at the speed set.

The functions of each button on the remote control are as follows.



Remote control overview

Input switching: By setting it to the D position, the remote control and control ECU can communicate.

START: Excite the motor and accept movement instructions.

BACK: De-energize the motor.

MODE: Switch the translation instruction button using the cross key and left stick. light next to button

When is lit, the cross key is a translation instruction button. Cross key: Translation instruction. The upward direction corresponds to forward movement. By inputting the translation speed setting value,

You can specify the value.

Left stick: Translation instruction. The upward direction corresponds to forward movement. Adjust the speed by tilting the stick.

When tilted to the maximum, the translation speed setting value can be specified. If you operate the stick too vigorously, instructions may not be sent properly. Right stick: Turn direction. Tilt it to the left to rotate it counterclockwise, and tilt it to the right to rotate it clockwise.

The turning speed can be adjusted by tilting the stick, and when the stick is tilted to the maximum right or left, the turning speed setting value can be specified. If you operate the stick too vigorously, instructions may not be sent properly.

X: Switches the speed/acceleration set for each remote control control and program control. LB: Decrease the speed setting value.

RB: Increase the speed setting value.

LT: Decrease the acceleration/deceleration time setting value.

RT: Increase the acceleration/deceleration time setting value.

4.6. Program control

You can send instructions to the robot using Python's USB serial communication. Although remote control operation is possible during program control, giving conflicting instructions or issuing instructions consecutively may cause unexpected operation. When operating the remote control, make sure that program control is stopped.

4.6.1. Connecting the USB cable

Connect the "Power input/PC connection" port in the control ECU diagram to the PC using a USB cable. Micro USB Type-B can be used on the control ECU side, and any USB standard can be used on the PC side.

When shipped, the power supply cable from "DC5V2A" in the power supply unit diagram is connected, so when controlling the program, remove the power supply cable and connect it to the PC.

4.6.2. Control function list

function	code	overview
Error information	0x0007	Obtain information on errors currently occurring in the motor
Operating status	0x0009	Obtain robot operating status
Power-supply voltage	0x0109	Get mains voltage
driving power	0x010B	Obtain the amount of electricity currently consumed by each motor
Error Reset	0x0201	Erase the error currently occurring in the motor
Start/pause	0x0301	Setting and referencing motor energization/deactivation
Standard acceleration/deceleration time	0x0305	Setting and referencing the robot's standard acceleration/deceleration time
Standard translation speed	0x0309	Set and refer to the standard translation speed of the robot
Standard turning speed	0x030B	Setting and referencing the standard rotation speed of the robot
Speed control (relative)	0x030F	Setting and referencing the robot's target speed (immediate operation)
Attitude control (relative)	0x0313	Target posture of the robot (x, y,) to set and refer to (immediate operation)
acceleration time	0x0315	Setting and referencing robot acceleration time
Deceleration time	0x0317	Setting and referencing robot deceleration time
Operating torque limit value	0x0319	Setting and referencing the torque limit deceleration time for all motors

4.6.3. UART control

Communication specifications

interface	UART
communication speed	115200bps
Flow control	invalid
Data length	8bit
stop bit	1bit

parity check	none
byte order	little endian

communication data structure

Start	code 1		value 1				code 2		value 2				End	
00													0d	0a

One communication data consists of 1 byte of Start to 2 bytes of End.

Both the data sent to the robot (control computer) and the data returned to the user's system (PC, etc.) have the data structure described in this section. The data length is variable up to 64 bytes and can be of any type.*1can contain communication codes for*2. Change the data length of the value according to the type in the communication code list. The above figure is an example of communicating two codes (4 bytes each).

In the communication code list, the R/W column indicates that R indicates read-only, W indicates write-only, and RW indicates read/write. When sending the R code, it is also necessary to append a byte string of appropriate length (the value does not affect it). However, since R* specifies and references each motor number, it is necessary to add the target number to the end of the byte string. When the W code is sent to the robot, the value at the time of sending will be echoed back. If the RW code is sent to the robot, the current value after writing the value will be returned. If an invalid value is sent to the robot using the RW code, the current value will be returned without writing the value.

The robot receives and returns data as a set. Therefore, regardless of the R/W code, he has to read the communication buffer every time he sends data. If a byte string different from the data structure described in this section is sent, unintended behavior may occur.

The value range of each code is based on the communication specifications and is not the range in which the product can be operated.

*1: We do not assume that redundant data will be sent, such as repeating the same code or including as many communication codes as possible from the list. Problems may occur, such as it taking a long time to return the product or the product not working properly.

*2: When conflicting instructions are sent, the one after the sequential processing results takes precedence. Example: Send start and stop codes at the same time.

Communication code list

code	Content	mold	unit	R/W
0x0007	Error information	TriOrbBaseError: 2 Part-Time Job	(See type definition)	R*
0x0009	Operating status	TriOrbBaseState: 2 Part-Time Job	(See type definition)	R*

0x0109	Power-supply voltage	float32	V	R*
0x010B	driving power	float32	W	R*
0x0201	Error Reset	uint8	0x01: Reset	W
0x0301	Start/pause	uint8	0x00: Echo back 0x01: Paused 0x02: Start 0x03 – 0xFF: Disabled	RW
0x0305	Standard acceleration/deceleration time	uint32	0: auto 1 - :ms Max: 1,000,000,000	RW
0x0309	Standard translation speed	float32	m/s Range: -3,600 to +3,600	RW
0x030B	Standard turning speed	float32	rad/s Range: -3,600 to +3,600	RW
0x030F	Speed control (relative)	TriOrbDrive3Pose	m/s, deg/s Range: -3,600 to +3,600	RW
0x0313	Attitude control (relative)	TriOrbDrive3Pose	m, deg Range: -3,600,000 to +3,600,000	RW
0x0315	acceleration time	TriOrbDrive3Vector	ms Range: 0.001 to 3,600	R/W
0x0317	Deceleration time	TriOrbDrive3Vector	ms Range: 0.001 to 3,600	R/W
0x0319	Operating torque limit value	uint16	0.1% Range: 0 - 2000	R/W

type definition

TriOrbBaseError: 2 bytes

Position	Type: Name	Content
1	uint8: alarm	motor alarm code
2	uint8: motor_id	Target motor ID

TriOrbBaseState: 2 bytes

Position		Type: Name	Content
1	1	-	-
	2	-	-
	3	-	-

	Four	bool:trq	Reaching the upper limit torque
	Five	bool: move	During the move
	6	bool: in_poss	Position control finished
	7	bool: s_on	Exciting
	8	bool: success	Success or failure of motor status acquisition
2		uint8: motor_id	Target motor ID

TriOrbDrive3Pose: 12 bytes

Position	Type: Name	Content
1-4	float: x	Robot X direction value
5-8	float: y	Robot Y direction value
9-12	float: w	Robot W direction value

TriOrbDrive3Vector: 12 bytes

Position	Type: Name	Content
1-4	float: v1	Motor ① value
5-8	float: v2	Motor ② value
9-12	float: v3	Motor ③ value

4.6.4. Python control

Installing the latest release

Install the Python control library using the following command. We have confirmed that it works with Python version 3.8.

```
pip install git+https://github.com/TriOrb-Inc/triorb-core.git
```

Connecting, starting and stopping the robot

```
fromtriorb_coreimportrobot

vehicle=robot("COM1")
vehicle.wakeup()#boot
vehicle.sleep()#pause
```

The first argument of the robot instance specifies the serial port connected to the robot (control ECU). For Windows systems, "COM1, COM2,,," etc. are common, and for Linux systems, "/dev/ttyUSB0, ttyUSB1,,," etc. are common. By using Robot instance functions, you can easily generate, send and receive the data strings shown in section 4.6.3. At this time, the return value of each function is the returned binary data converted into a type corresponding to the communication code.

By calling the wakeup function, the motor enters the excited state and can accept position control and speed control. By calling the sleep function, the motor enters a non-energized state and stops.

Get robot status

```
fromtriorb_coreimportrobot

vehicle=robot("COM1")

print(vehicle.get_motor_status(params=['error','state','voltage','power'], _id=[1,2,3]))
                                # specifiedIDGet various statuses of the motor driver.
```

You can get the motor status with the above function.

The get_motor_status function allows you to specify the information and ID you want to obtain as arguments. If no argument is given, all the status information that can be specified and the ID of the motor used for driving will be set. The return value is an array with len(params)*len(_id) elements. Also, if you execute this command when there is an error in the motor driver, an error reset will be executed and all motors will be de-energized. Also, obtaining the motor driver status may fail. The return value of a failed motor driver is given an exceptional value. robot state one

The list is as shown in the table below.

Key	Content	When acquisition fails
error	Obtain motor driver alarm information.	alarm variable: -1
state	TriOrbBaseState reference	0 except motor_id
voltage	Power supply voltage (unit: V)	0
power	Drive power (unit: W)	0

status code	value	Content
I0001	Float	Estimated operation completion time (unit: seconds)
I0002	None	Dormant
I0003	Float	Power supply voltage (unit: V)
E0001	None	[ERROR] The motor driver and motor cannot communicate. Occurs when drive power is lost or during an emergency stop.
E0002	None	[ERROR] Overcurrent error stop

Robot posture control

```
fromtriorb_coreimportrobot

vehicle=robot("COM1")
vehicle.wakeup()#boot

vehicle.set_pos_relative(x=1.0,y=-1.0,w=90.0)#Set the robot's posture using the local coordinate system. translational positionx=1.0[m], y=-1.0[m]and rotational positionw=90.0[deg]Take this posture.

vehicle.join()#Waiting for operation to complete

vehicle.set_pos_relative(x=1.0,y=0,w=0,acc=3000,dec=2000)#acceleration time3seconds, deceleration time2Set the robot to the local coordinate system position in seconds.x=1.0[m]Take this posture.

vehicle.join()

print(vehicle.set_pos_relative(x=360001, y=0, w=0))#If you enter an invalid value, you can retrieve the previous reading.
```

An example of a program that changes the robot's posture (translational position, rotational angle) is shown below. The robot accelerates to standard speed and decelerates to stop at the target pose. Acceleration/deceleration times are given as arguments in the attitude control function, and can also be set using the write_config function described later.

Attitude control functions do not interrupt the program. By using the join function, you can make the program wait until the operation completes. Although it is possible to perform other processing without using the join function, unexpected behavior may occur if you issue an instruction to drastically change the direction of travel before stopping. Please note that this directive is a theoretical value based on kinematics, so it may differ from the actual posture depending on road conditions, etc.

set_pos_relative:

- The robot's posture at the time the instruction is given is set as the origin.The amount of movement
- becomes constant when the same instruction is given.

Please note that if you give instructions for translational position and rotation angle at the same time, the expected posture will not be obtained. for example,vehicle.set_pos_relative(x=1.0,y=0.0,w=180.0)Even if you execute , the robot's posture will not be (1,0,180) in most cases. The robot moves forward while turning, so it moves in an arc. The final attitude changes depending on acceleration/deceleration time and standard speed.

If a value outside the upper and lower limit values is given to any of x, y, or w, the system enters read mode and the Returns the x,y,w values obtained. This value is retained in each function.

robot speed control

```
importtime
fromtriorb_coreimportrobot

vehicle=robot("COM1")
vehicle.wakeup()#boot

vehicle.set_vel_relative(x=0.1,y=0.1,w=0.1)#Set robot speed in local coordinate system. translation x=0.1[m/s], y=0.1[m/s]and swivelw=0.1[rad/s]moving at a speed of time.sleep(5.0)# Fivewait seconds vehicle.brake()#decelerate and stop vehicle.join()#Waiting for movement to complete



vehicle.set_vel_relative(x=0.1,y=0,w=0,acc=3000,dec=2000)#acceleration time 3seconds, deceleration time2Set the robot's velocity to the local coordinate system translation in seconds.x=0.1[m/s]Set to. time.sleep(5.0)# Fivewait seconds vehicle.brake()#decelerate and stop vehicle.join()#Waiting for movement to complete



print(vehicle.set_vel_relative(x=360001, y=0, w=0))#Get current speed
```

An example of a program that changes the robot's speed is shown below. After the robot accelerates to the commanded speed, it continues to move at a constant speed until new speed or attitude control is performed. You can also use the brake function to stop it. Acceleration time and deceleration time can be given as arguments in the speed control function, and can also be set using the write_config function described later.

Speed control functions do not interrupt the program. When using the join function, be sure to use brake immediately before

Please run the function.

set_vel_relative:

- The robot's posture at the time of instruction is the origin
- When given a forward command, the robot moves in the direction it is facing at the time of the command.

If a value outside the upper and lower limit values is given to x, y, or w, the mode will be set to read, and the current load will be

Returns the x,y,w values of the bot. This is calculated from the rotation speed detected by the motor driver. If the calculation fails, each value is returned as 0.

Change robot settings

```
fromtriorb_coreimportrobot

vehicle=robot("COM1")
print(vehicle.read_config())#Load current settings vehicle
.write_config({#Write settings
    "acc":1500,#Standard acceleration/deceleration time1.5[s]set
    to "std-vel":0.25,#standard translation speed0.25[m/s]set to
    "std-rot":0.5,#Standard turning speed0.5[rad/s]set to
    "torque":500,#Torque limit value50[%]set to
})
print(vehicle.read_config(["acc","std-vel","torque"]))#Load current settings
```

An example program for reading and writing robot parameters is shown below. The function for writing is write_config, and the function for reading is read_config. The written value will be retained until the robot control ECU is restarted.

The values that can be specified with the read_config and write_config functions are standard acceleration/deceleration time, standard translation speed, standard turning speed, and torque limit value. These values are primarily used for velocity control and attitude control; standard velocity applies only to attitude control. read_config specifies the value to be read in an array type, and write_config allows you to set the target to be written and its value in a dictionary type as shown in the table below.

Key	Content	Factory setting
acc	Standard acceleration time (unit: ms)If you set 0 or less, automatic control will be applied. Become the Lord.	500

std-vel	Standard translational speed (unit: meters per second)	0.2
std-rot	Standard rotation speed (unit: radian per second)	0.2
torque	Torque limit value (unit: 0.1%)	1000

Example of sending and receiving communication data by specifying communication code and value

```
fromtriorb_coreimport* import
time
import numpy as np vehicle
=robot("COM1")

query = []
query.append([RobotCodes(0x0301), b'\x02'])                    # excitation
query.append([RobotCodes(0x0313), TriOrbDrive3Pose(1,-1,0)]) control      # relative posture

query.append([b'\x05\x03', np.uint32(1000)])                   # Standard acceleration/deceleration time setting

vehicle.tx(code_array=query)#send print(
vehicle.rx())#reception

query = []
query.append([RobotCodes(0x0009), b'\x00\x01']) Get the       # motor driver stator
space (ID1)
val = TriOrbBaseState(motor_id=2)
query.append([b'\x09\x00', val])#Get motor driver status (ID2)

vehicle.tx(code_array=query)#send print(
vehicle.rx())#reception

time.sleep(5.0)
vehicle.tx(code_array=[[RobotCodes(0x0301), b'\x01']])#De-energization
vehicle.rx()

# Submitting incorrect data
vehicle.tx(code_array=[[RobotCodes(0x0301), b'\x02'],#Excitation (positive)
                       [RobotCodes(0x0305), b'\x03']])#Acceleration setting (incorrect)
# Return (STARTandENDonly)
```

```
print(vehicle.rx())# -> b'\x00\r\n'
```

The following is an example of a program that communicates by specifying a communication code and value. You can use the tx function for writing and the rx function for reading. After executing the tx function, use the rx function to receive data. If you execute the tx function one after another, most of the subsequent rx functions will not be processed correctly and the return value will be an array with 0 elements. Also, if you execute the rx function before executing the tx function, the program will stop as it continues to wait for received data. This is true even if the rx function is executed continuously.

The argument of the tx function is a double array whose elements are an array of communication codes and their values. Please convert the communication code to RobotCodes type and the value to the corresponding type. Some types use numpy. Both can be passed as bytes type, but please note that they are Little-Endian. If you send data that differs from the communication data structure, the robot will only return Start+End. The same thing happens when you send data that contains both normal and incorrect values, as shown in the program example "Sending incorrect data."

The rx function converts the returned data values to the appropriate type and returns an array containing them as elements. If you want to receive data as bytes, you can use the rx_bytes function.

5. Maintenance

5.1. Precautions during maintenance

Always power off the robot and remove the battery before starting work. Improper maintenance work may result in serious injury or property damage.

- ➢ Make sure you have enough working space before you start.
- ➢ Be sure to wear safety equipment such as work gloves when working.
- ➢ Check if there are any obstacles around the robot.
- ➢ Check the robot's body for damage.

5.2. Maintenance method

Perform regular maintenance on the drive ball, cables, timing belt, and ball casters before and after use. Each part can be maintained by removing the under panel at the bottom of the robot. If you need to lift the robot by grasping it, be sure to grasp only the part that can be lifted (top plate) and lift it. Electrical/electronic equipment, cables, etc., may be damaged if lifted by the wrong part.

- ➢ Clean each part of the robot to remove dirt and debris.
- ➢ Inspect each part of the robot and make sure there are no abnormalities.
- ➢ Adjust each part of the robot and make sure it is working properly.
- ➢ If water or oil gets on the product, wipe it off with a clean cloth.

6. Service

6.1. Warnings

- ● If the weight is insufficient, the driving ball may slip, so if slipping occurs, consider installing ballast or other equipment to ensure driving friction.
- ● If the power supply voltage is less than 18.0V, the robot may not operate properly, so please stop operation immediately and charge the battery.

6.2. Errors

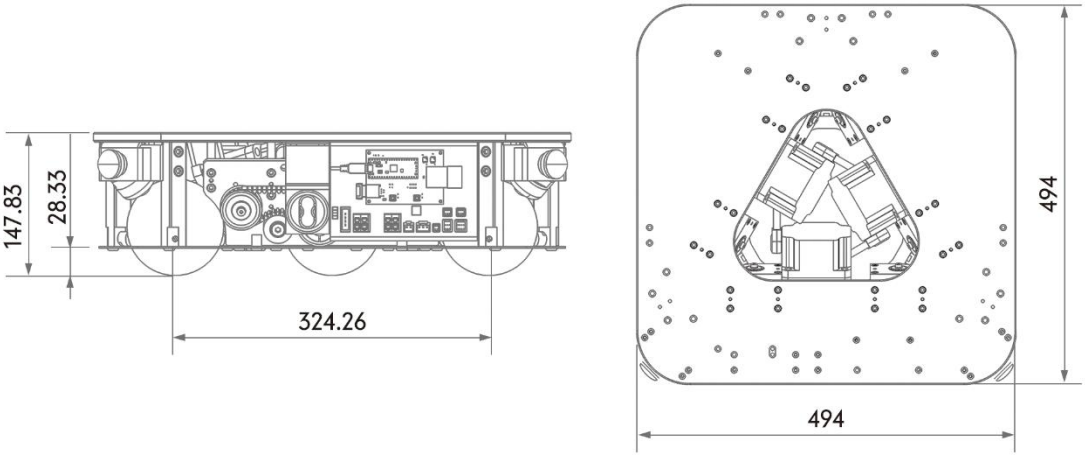
- ● If the robot does not move properly using remote control, check that the remote control lamp does not flash when you press the START button. If it blinks, there may be a problem with the control ECU or remote control communication, so please restart the control ECU by turning the control power switch off and then on again. If the problem does not improve, please contact the manufacturer for repair.
- ● If the emergency stop SW lamp does not light up correctly, there may be a problem with the emergency stop SW connection cable, or the emergency stop SW may be malfunctioning. Check that there are no problems with the connector connection, and if the problem persists, please contact the manufacturer for repair.

- If the control power switch lamp does not light up correctly, the main power may not be being supplied correctly, or the power supply unit may be malfunctioning. Check if the main power supply and cables are normal, and if the problem persists, remove the battery and contact the manufacturer.
- If all motors do not start (there is no excitation sound), there may be a problem with the motor control signal line coming from the control ECU. Check that there are no problems with the connector connection, and if the problem persists, please contact the manufacturer for repair.
- If a specific motor does not start (no excitation sound), use the "communication power supply" "RUN signal"“Mo There may be a problem with the controller control signal. Check that there are no problems with the connector connection, and if the problem persists, contact the manufacturer.
- Motor warning information can be obtained during program control. Please refer to the alarm list in "Oriental Motor Co., Ltd. BLV Series R Type Function Edition (HP-5141-4)" for details.

7. Specifications

7.1. Dimensions and weight

BASE500P100 (28kg)



7.2. Basic specifications

This data is not the limit value of the main unit, but the experimental value in our verification environment. The running performance against external disturbances varies depending on speed, approach angle, road surface conditions, etc., so please use with due consideration for safety.

		BASE500P100
environment	Place of use	indoor
	ambient temperature	0～40℃
	humidity	No condensation during confirmation
	others	Do not get wet with water or oil
Exercise specifications	motion axis	Forward/backward/left/right/swivel
	driving ball	φ100mm
	motor	100W (×3 axes)
	gear ratio	20
	Translational speed ( )*1	0.67m/s
	Turning speed ( )	6.0rad/s
	Climbing (loading 0kg)	1/6
	Climbing (loading 300kg)	1/6
	Passing through the groove (loading 0kg)	60mm
	Passing through the groove (loading 300kg)	45mm
	Level difference (loading 0kg)	15mm

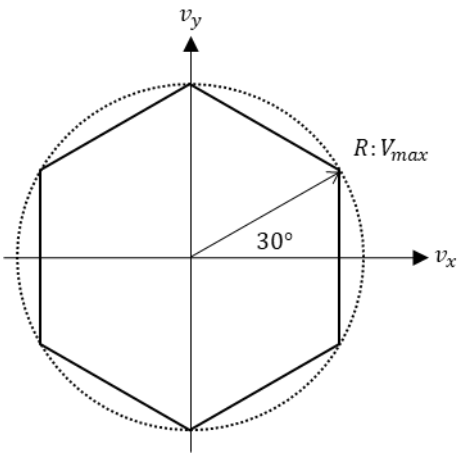
	Level difference (loading 300kg)	7mm
Operating time	battery	WP1236W (×2 pieces)
	capacity	9Ah
	Continuous operation time*2	3.5h
	battery weight	5.4kg
	Charging method	・12V charging after removal (100Vac, 120W) ・ 24V charging after connector (100Vac, 450W)

* 1The maximum translation speed varies depending on the direction of travel.

* 2Verification: Calculated using data from continuous operation at a speed of 0.3m/s on a smooth road surface

Translational speed output characteristics

TriOrb BASE has the following characteristics in speed output for control purposes. The figure below is a schematic top view of the driving sphere, coordinate system, and maximum translation speed of TriOrb BASE. Drive ball forward on two sides The direction and maximum speed are shown in a separate table. It becomes.



7.3. Accessories (optional)

- ➢ Battery (LONG shield battery WP1236W) ···2 pieces
- ➢ Remote control (Logitech gamepad controller F750r) ...1 piece
- ➢ Charger (Ohashi Sangyo 12V ACE CHARGER 10A) ···2 pieces

TriOrb

株式会社TriOrb

福岡県北九州市小倉北区浅野3-8-1 AIMビル6F
TEL：093-513-1023
https://www.triorb.co.jp