General Purpose Robot Arm for Industry Use

ZERØ

VERTICAL ARTICULATED ROBOT USER MANUAL

Support Version of R1.3.0 or higher

Document Number: M-0101-231214

December 2023

Thank you for your purchase of the industrial robot “ZERO”

- When using this product, knowledge and skills in “Occupational Safety and Health Training”, “Electrical Engineer Certificate” and “Python” programming language are required.
- Please read the user manual and other instructions carefully before using and use it correctly.
- To improve product performance, specifications may change without prior notice.
- Instructions such as the User manual,

· Must be safely stored by product users.

· May be modified without notice due to changes in product specifications.

· Unauthorized copying of part or all of the content is prohibited.

This user manual supports the following models:

Robot	Controller (version)	JOG stick	Teaching Pendant
ZERO ZRA Series	ZC1*** (Over R1.3.0)	ZJ1000	ZP1000

Trademarks and patents:

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany. Other names used in this publication may be trademarks utilized by third parties for their own purposes and may infringe the rights of the owners.

EtherCAT is an open real-time telecommunications network developed by Beckhoff Automation GmbH, Germany, and its rights are protected by Beckhoff.

This product includes the following instructions.

Installation Guide

This is a simple guide, from unpacking to installation.

Before Unpacking

Safety Precautions
Mounting the Manipulator and Controller Installation
JOG Stick
Connecting Teaching Pendant and PC
Wiring and Power Supply
JOG Operation and ABS Homing
Troubleshooting Guide

Safety Precautions

The following safety contents must be complied with.

Compliance Terms

Safety Instructions

Risk Assessment

Industrial Health and Safety Education
Maintenance/Inspection
Safety Measures
Warranty and Disclaimer

User Manual (this document)

This is a guide for the product and programs.

A Overview

B Hardware

C Instructions

D Software

Z Documentation

A OVERVIEW

1. System Introduction	Product information, international specifications
2. System Installation	Start-up procedure, unpacking, attachments/accessories, transportation

B HARDWARE

1. System configuration	Model name, system configuration
2. Manipulator	Overview, part names, installation, dimension drawings, specifications, connectors, movement range, end-effector design
3. Controller	Model name and label, part names, installation, external drawings, specifications, connectors, and controller status display
4. JOG stick	Product label, part names, installation, external drawings, specifications, functions
5. Teaching pendant	Product label, part names, installation, external drawings, specifications, functions
6. Wiring and power supply	Wiring system, power supply

C TEACHING

1. JOG operation	JOG operating mode
2. PC connection	Connecting Teaching Pendant and PC
3. Restoring ABS	Notes, order, confirmation
4. Teaching	Basic operations, instruction sequence, instruction data transmission
5. Coordinate system and posture	Coordinate system, general coordinate system, world coordinate system, base coordinate system, tool coordinate system, user coordinate system, posture

D SOFTWARE

1. Programming Instructions	PC and operating environment, programming instructions
2. Robot Library	Data types, modules, method summary, robot library
3. Memory Map	Overview, Shared memory, Memory I/O
4. Program implementation steps	Overview, how to proceed

Z DOCUMENTATION

1. Block diagram	System’s block diagram, hardware’s block diagram
2. Maintenance	Inspection, maintenance
3. Glossary	Glossary
4. Troubleshooting	Error log, troubleshooting

NOTE

A OVERVIEW

NOTE

1 SYSTEM OVERVIEW

1. PRODUCT INFORMATION

1. ROBOT "ZERO" STRUCTURE

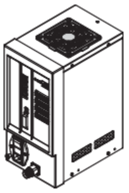
This product contains the following components:

"ZERO" (= Robot)

"ZERO" includes manipulator and robot controller

Manipulator

The manipulator is a 6-axis vertical articulated 6-axis robot actuated by servo motors. Attaching different end-effectors to the manipulator-tip allows the robot to be adapted for various tasks.

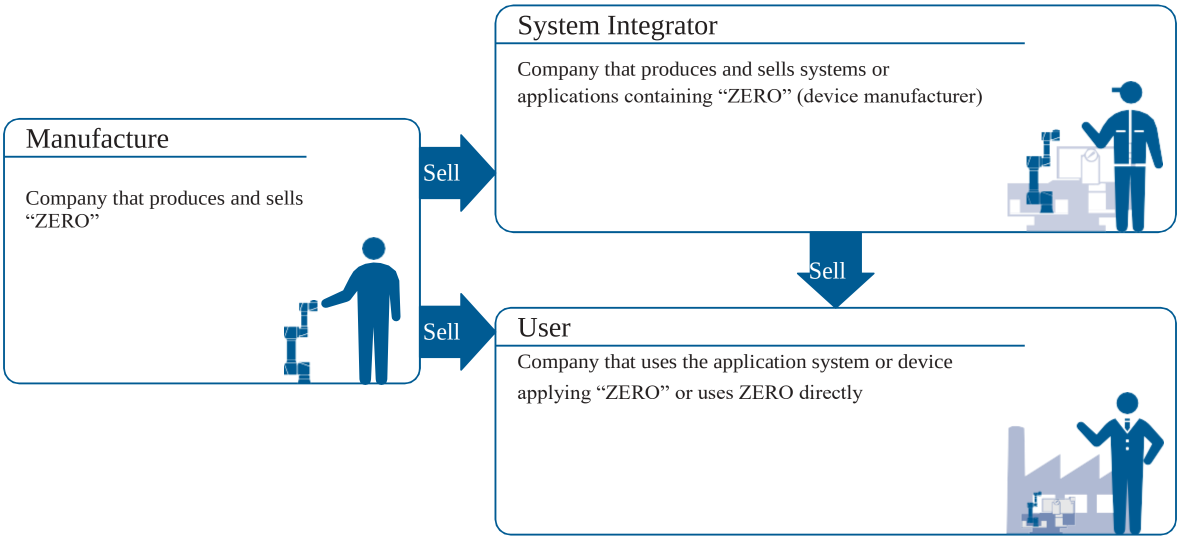


Controller

The controller is a control board that includes control circuitry and a power supply board. The controller handles communications with the host controller through I/O interfaces, and comprehensively controls motions of the manipulator.

2. Manufacturer, system integrator, and user

Manufacturers, system integrators, and users are defined in this document as follows:



Manufacture

Company that produces and sells “ZERO”

System Integrator

Company that produces and sells systems or applications containing “ZERO” (device manufacturer)

User

Company that uses the application system or device applying “ZERO” or uses ZERO directly

3. Notes in the instructions

- The specification values (ratings, performance) are those obtained under the conditions of an individual test and cannot guarantee the values obtained under combined conditions.
- Specifications may change or production may be discontinued without notice due to product improvements or company circumstances.

4. PURPOSE OF USE

This product is an industrial robot. It is designed and manufactured for typical industrial products at factories. It is not suitable for home use or for the following purposes. The company does not guarantee the following:

Usage related to military or weapons

All military-related applications where the end user and the ultimate purpose of use are military or weaponry.

Applications requiring high safety and reliability

Nuclear energy control devices, combustion devices, aerospace equipment, transportation, railway equipment, shipbuilding equipment, lifting devices, amusement vehicles, medical equipment, nursing machinery, safety equipment, automotive production loading equipment

Other devices that pose a risk to life.

Use in harsh conditions or environments

Outdoor works, chemically polluted works, electronically disturbed works, works subject to vibration or impact, dusty locations, and mining operations in mines

Use in conditions or environments not described in the User Manual

Failure to follow the warnings and precautions stated in the Terms of use, etc., may lead to injury (death or serious injury), accidents, or malfunctions. The company is not responsible for this.

The company cannot predict all possible risk and problem situations. The warnings, cautions, and other information stated in the Terms of use, etc., are within the scope of what our company can predict.

1. SUITABLE STANDARDS

Machine type instructions

. . . . 2006/42/EC

Machine safety – Electrical equipment of machine – Part 1: General requirements

. . . . EN60204-1:2018

Robots and robot devices – Safety requirements for industrial robots – Part 1: Robot

. . . . EN/ISO 10218-1：2011

EMC

. . . . EN61000-6-2:2005

. . . . EN55011：2009+A1:2010

KCs

. . . . . S2-W-5-2017

2. Environment specifications

Specifications	Manipulator	Controller
IP	IP40	
Vibration – impact	—	JIS B3502

2 SYSTEM INSTALLATION

1. START-UP PROCEDURE

UNPACKING ☞ A OVERVIEW

- Check for any signs of damage during transportation.
- Please lift the device up to avoid putting pressure on the plastic lid.
- Check if the contents inside are damaged or deformed.
- Ensure that the product received matches the ordered product and is complete.
- Check if it includes a manipulator and controller.
- Ensure there is enough space on a flat surface.
- Wear protective gear such as helmets, goggles, safety shoes, gloves, and remove anything that may get caught, such as straps, before taking the device out.
- Please preserve the opened box and fixtures in case it is needed when moving.

TRANSPORTATION Moving to installation space ☞ A OVERVIEW

- The installation must be performed by at least two people.
- Use a crane or trolley to move as close to the installation space as possible.
- Be careful not to cause excessive impact or vibration when moving to the installation space or when transporting onto the trolley.
- Be careful not to apply too much force to protruding parts such as switches, terminals, connectors, and cooling fans.
- Be careful not to place any heavy objects on the plastic casing during transportation.
- When temporarily fixing, use at least one bolt to tighten.
- Proceed while maintaining the transportation posture until it is fixed to the installation space.

NSTALLATION ☞ B HARDWARE

- Secure the manipulator to the horizontal mounting surface within the safety zone (inside the safety fence) using fixed hex bolts (M8) and prevent it from skewing or falling.
- Ensure the manipulator is rigid enough and fixed in the appropriate position. The recommended roughness of the mounting surface is Rz25 or above. The mounting surface must have sufficient strength and rigidity to withstand the reaction force and load during operation.
- Ensure a space that does not affect the manipulators or other peripheral devices.
- Fix the controller horizontally using screws outside the safety zone (outside the safety fence) where the controller can be easily seen. Please do not remove the rubber feet.
- Installation should consider the location and method of work necessary for future inspection, repair, and maintenance.
- Ensure fixation even when temporarily installed for operational testing and trial runs.

WIRING AND POWER SUPPLY ☞ B HARDWARE

- Be careful not to incorrectly plug in connectors or cable them incorrectly.
- Plug in the connector firmly until you hear a “click”.
- Securely install and fix the screw connector.
- Please connect the power supply after completing the entire electrical wiring system.
- When installing cables and pipes, be careful not to trip or fall.
- Keep the cables between devices or external input/output cables separate from the power lines or ground lines of other devices.
- Ensure the use of protective cables for external input/output cables.
- Do not apply a voltage different from the voltage specified in the User Manual for each end-effector.
- Be careful to ensure that the terminal connections and poles (+ and -) are not incorrect.
- Ensure the wiring space and fixation so that the cable mass, reaction force, load, etc., do not affect the connectors. Protect the connections when necessary.
- Ensure prevention of electric shock, static electricity, improved noise performance, and elimination of unnecessary electromagnetic wave radiation.
- Use cables of the specified size for grounding and keep the distance to the grounding point as short as possible.
- Use a dedicated grounding method and separate grounding when installing other large devices.

HOMING, TEACHING

C TEACHING

- Connect the controller to the teaching PC.
- Teaching by using the JOG stick outside the safety zone (outside the safety fence).
- Only TURN ON the activation switch when you want to operate the manipulator.
- If it is unavoidable to conduct teaching within the safety zone (inside the safety fence), ensure safety.
- Check the priority level of robot control and use the lock key, emergency stop switch, and interlock key to indicate during the teaching operation.
- Ensure an emergency exit in case of an emergency.
- After working, return the safety protection equipment to its original condition.
- Confirm that there are no obstacles such as peripheral devices within the range of operation of the manipulator before operating.

PROGRAMMING

D SOFTWARE

- Basic knowledge of Python is required.
- Programming can be easily done with this product's "Robot Library".
- Sample programs for basic operations are available.
- A memory map is provided.

OPERATION CONFIRMATION AND CHECKING

D SOFTWARE

- Ensure that there are no people within the safety zone (inside the safety fence) and no obstacles within the operating range of the manipulator.
- Operate outside the safety zone (outside the safety fence). When working within the safety zone (inside the safety fence), ensure safety.
- Check operation after confirming that all emergency stop switches are active.
- Check if the manipulator, including peripheral devices, can be stopped by the emergency stop switch.
- Perform a full operation check and ensure that the device can be operated safely.

BEFORE AUTOPILOT

D SOFTWARE

- Ensure that there are no people in the safety zone and no obstacles within the operator's range of motion.
- Check if everything in the system, including related peripheral devices, is capable of automatic operation.
- When starting control, first drive at a low speed and check if the vehicle is operating normally.

Complete the start-up procedure

Testing methods, troubleshooting, and other technical information are listed in the documentation **Z** section.

Z DOCUMENTATION

2. UNPACKING

RECEIVING

· Check for any signs of damage during transportation.

If there are any signs of damage, please unpacking in the presence of the carrier.

Preserve all packaging materials. They may be necessary when filing a claim for damages.

· Check if the controller and manipulator are concluded.

· Please check if the received product matches the order details.

UNPACKING PREPARATION

· Ensure there is enough space on a flat surface. The product may fall from unstable positions.

· Wear protective gear such as helmets, safety shoes, gloves, and remove anything that may get caught, such as straps.

UNPACKING

· Two or more people must work together.

· Use a crane or trolley to move as close to the installation space as possible.

When being carried by one person, ensure there is no significant difference in strength between the two people.

Also, ensure stability is not lost due to opening or closing doors, etc.

· Do not hold anything except the fixed fixtures. In particular, plastic parts may become the cause of damage.

AFTER UNPACKING

· When temporarily installing the manipulator, secure it by tightening it with at least one bolt.

· Do not remove the fixtures until the installation process is complete.

· Please preserve the packaging materials or the original fixing accessories, they can be used for repackaging in the same condition as when delivered if relocation or transportation is needed.

포인트

Preparation before moving

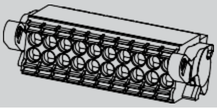
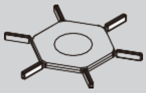
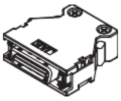
Please forward all User manuals related to this product to the end user.

The system integrator must prepare a User Manual for the entire system, including this product, and deliver it to the user.

Please preserve all packaging materials and packages in the same condition as when delivered.

When reusing wooden pallets for international transportation, please check International Standard No. 15 “Regulations on wooden packaging material in international trade”.

3. ATTACHMENTS/ACCESSORIES

Attachments/Accessories	Model (Manufacturer)	Quantity	Note
Manipulator	ZRA-05***	1 pcs	—
Controller	ZC100*	1 pcs	—
User Manual	—	1 copy	PDF File
Installation Manual	—	1 copy	Manual / PDF File
Safety Manual	—	1 copy	Manual / PDF File
I/O Connector	DFMC 1,5/10-ST-3,5-LR (1790564) (PHOENIX CONTACT COM)	3 pcs	(20 pin) 
Safety Connector	As above	1 pcs	As above
Coding profile keys	CP-DMC 1,5 NAT (1790647) (PHOENIX CONTACT COM)	1 set (6 pcs)	— 
Jumper Connector	E2010101 (ZEUS CO., LTD.)	1 pcs	— 
Manipulator Cable	E2021701 (ZEUS CO., LTD)	1 pcs	･ Length 3m ･ The supplied ferrite cores cannot be removed
Ferrite Core	—	2 pcs	･ For power cable ･ Diameter 4.5 - 8.5 mm

4. TRANSPORTATION

1. MOVING TO INSTALLATION SPACE

❗	At least two people must work together. Use a crane or trolley to move as close to the installation space as possible. When transporting by manpower, ensure there is no significant difference in strength between the two people. Do not lift with one hand when opening or closing door, etc.	⚠ ⚠ ⚠

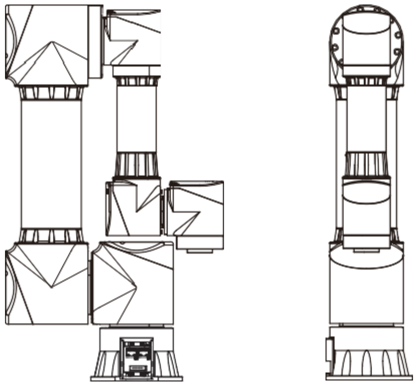
· Controller

Lift and move the lower part to avoid bumps to the front and intake/exhaust vents.

· Manipulator

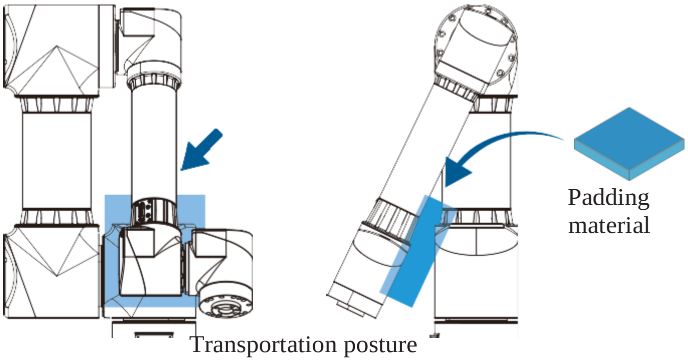
Do not hold the articulation (joint part) or bottom flange.

ZRA-05**P



Transportation posture

ZRA-05**N



Transportation posture

Padding material is used to avoid damage to the manipulator.

Keep the padding material along with the product box as it will be reused when it needs to be transported.

2. TRANSPORTING WHEN CHANGING LOCATIONS

❗	At least two people must work together. Damage caused by improper packaging is not covered by the warranty.	⚠ ⚠ ⚠
	Please pack safely with the packaging materials provided at the time of purchase. Transport the device as a "machinery requiring precision" transaction. There is a risk of damaging the manipulator if it is subjected to impact or load on the joints during transportation.	⚠ ⚠

Controller

Be careful not to damage the front vents and the intake/exhaust vents, and place them in a dedicated cardboard box in the same condition as when delivered.

Manipulator

Please pack the product in the same condition as when delivered, in the transport position, using the packaging materials available at the time of delivery

Transporting the product under conditions different from those specified may cause accidents or malfunctions.

When reusing wooden pallets for overseas transport, please check International Standard No. 15 "Regulations on wooden packaging material in international trade".

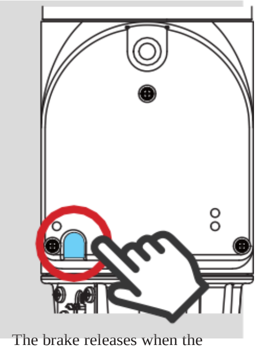
PACK THE MANIPULATOR

1. Manipulator transportation posture

To change the position, connect the controller and the manipulator with a cable and then TURN ON the power of the controller. When the power is plugged in and the brake release button is pressed on each joint, the brake will be released. (The brakes are only released when the button is pressed.)

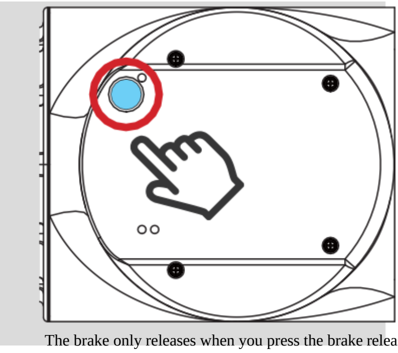
Brake Release Button

The brake will not release unless power is connected to the robot



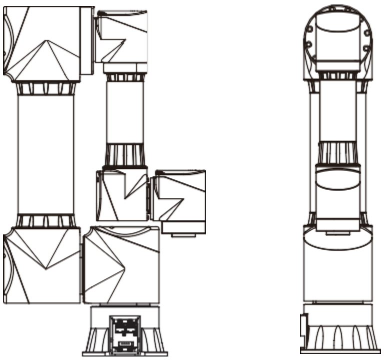
The brake releases when the brake release button is pressed

1st, 2nd, 3rd joints



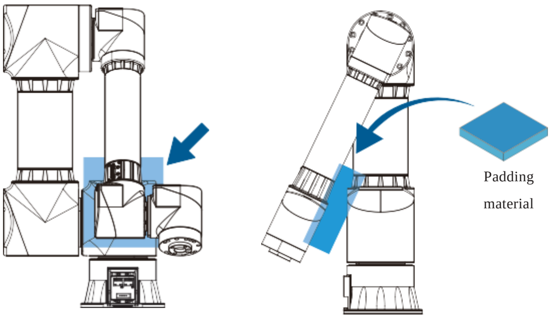
The brake only releases when you press the brake release button (about 0.5 seconds)

ZRA-05**P



Transportation posture

ZRA-05**N



Ensure to insert padding material to avoid damaging the manipulator

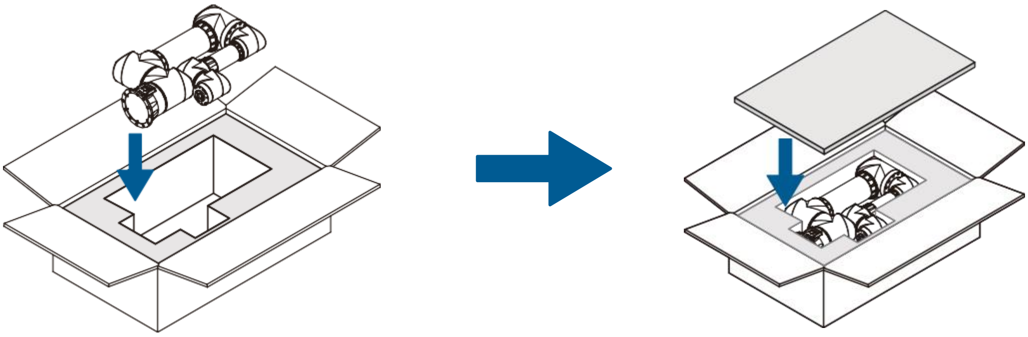
Be careful not to let the padding material fall off.

Transportation posture

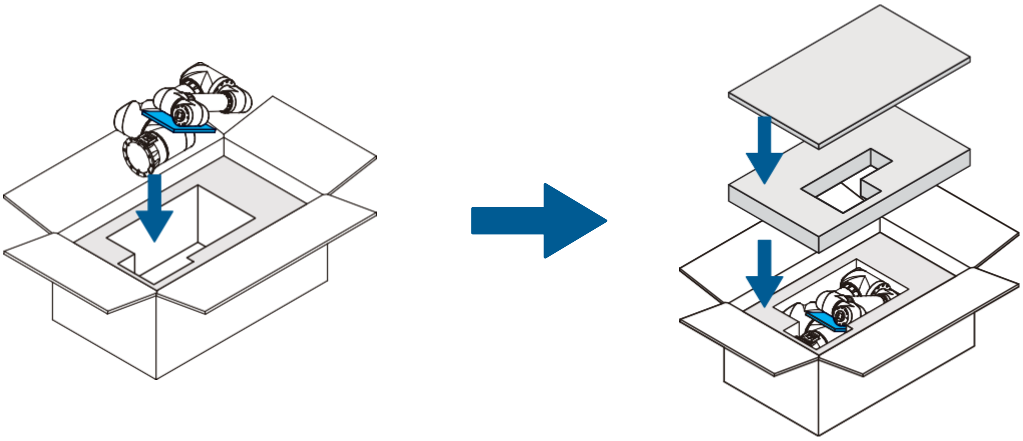
2. Use dedicated packing box

Place the manipulator into the packing box.
Use padding material to ensure proper protection.

ZRA-05**P



ZRA-05**N

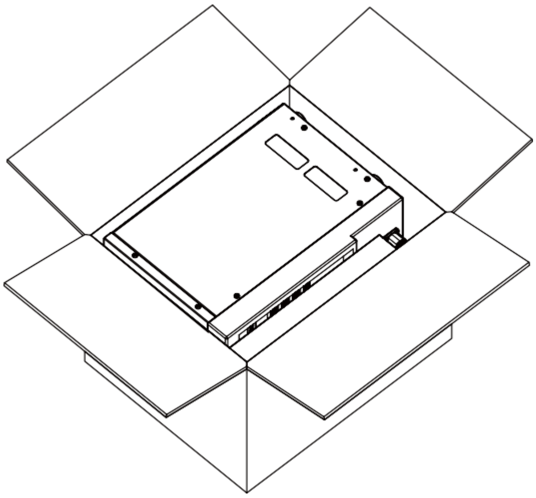


TRANSPORTATION

Pack the Controller

1. Use dedicated packing box

Place the controller on its side in a dedicated packing box, and when placing the controller, disconnect all connectors.



B HARDWARE

NOTE

B HARDWARE

1 SYSTEM CONFIGURATION

1. MODEL NAME

This product is supplied with manipulator and controller.

Manipulator model name: ZRA - 05 15 P

Product code

Maximum load

Classification number

Arm type

ZEUS Robot Type A (First)

Maximum load

Symbol	Dimension
XX	XX kg

Arm type

Symbol	Dimension
N	Turn Around Motion Type
P	Pass Through Type

Models with a difference of 120mm or more between 1st Arm and 2nd Arm are Pass Through Type

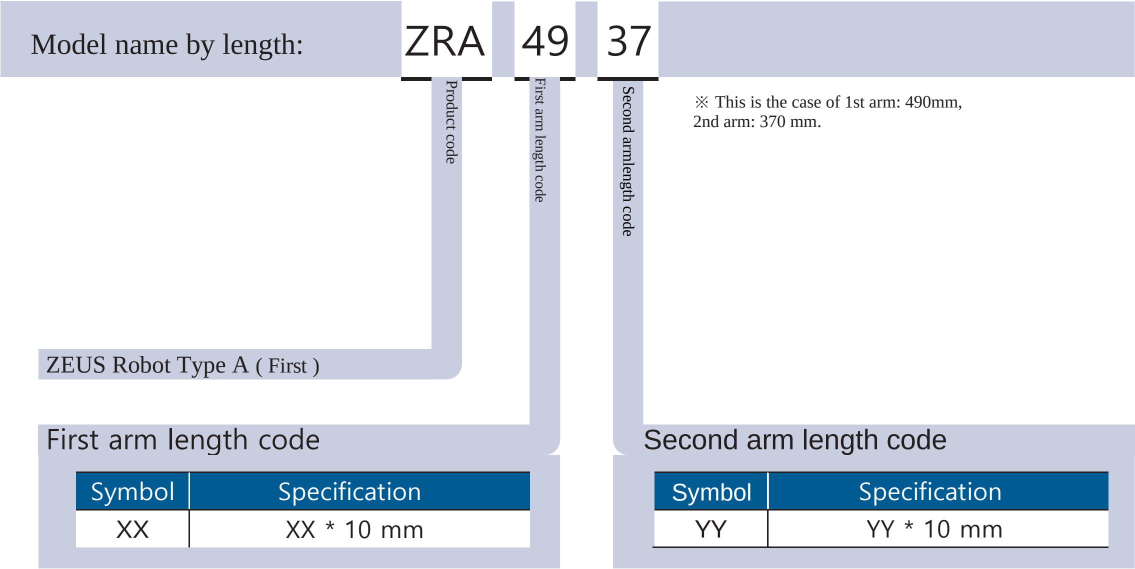
Classification number

No.	Total length (mm)	Arm length 1 (mm)	Arm length 2 (mm)	Model name
1	590	320	270	ZRA-0501N
2	660	320	340	ZRA-0502N(*)
3	660	390	270	ZRA-0503P(*)
4	690	320	370	ZRA-0504N
5	690	420	270	ZRA-0505P
6	730	390	340	ZRA-0506N
7	760	320	440	ZRA-0507N
8	760	390	370	ZRA-0508N
9	760	420	340	ZRA-0509N
10	760	490	270	ZRA-0510P
11	790	420	370	ZRA-0511N
12	830	390	440	ZRA-0512N
13	830	490	340	ZRA-0513P
14	860	420	440	ZRA-0514N(*)
15	860	490	370	ZRA-0515P(*)

*) 4 models are typical models.

Only use 590mm ~ 860mm models.

1 MODEL NAME



※ This is the case of 1st arm: 490mm,
2nd arm: 370 mm.

First arm length code

Symbol	Specification
XX	XX * 10 mm

Second arm length code

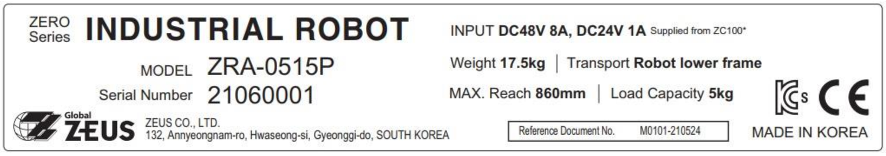
Symbol	Specification
YY	YY * 10 mm

※ XX and YY of the symbol contain digits.

It is possible to match the length of the arm in accordance with the customer's layout because the total length of the first and second arms is less than 860mm. For more details, please contact.

Use both model names interchangeably when necessary.

Example: Model name on label of Manipulator



ZERO Series **INDUSTRIAL ROBOT**

MODEL ZRA-0515P

Serial Number 21060001

INPUT **DC48V 8A, DC24V 1A** Supplied from ZC100*

Weight **17.5kg** | Transport **Robot lower frame**

MAX. Reach **860mm** | Load Capacity **5kg**

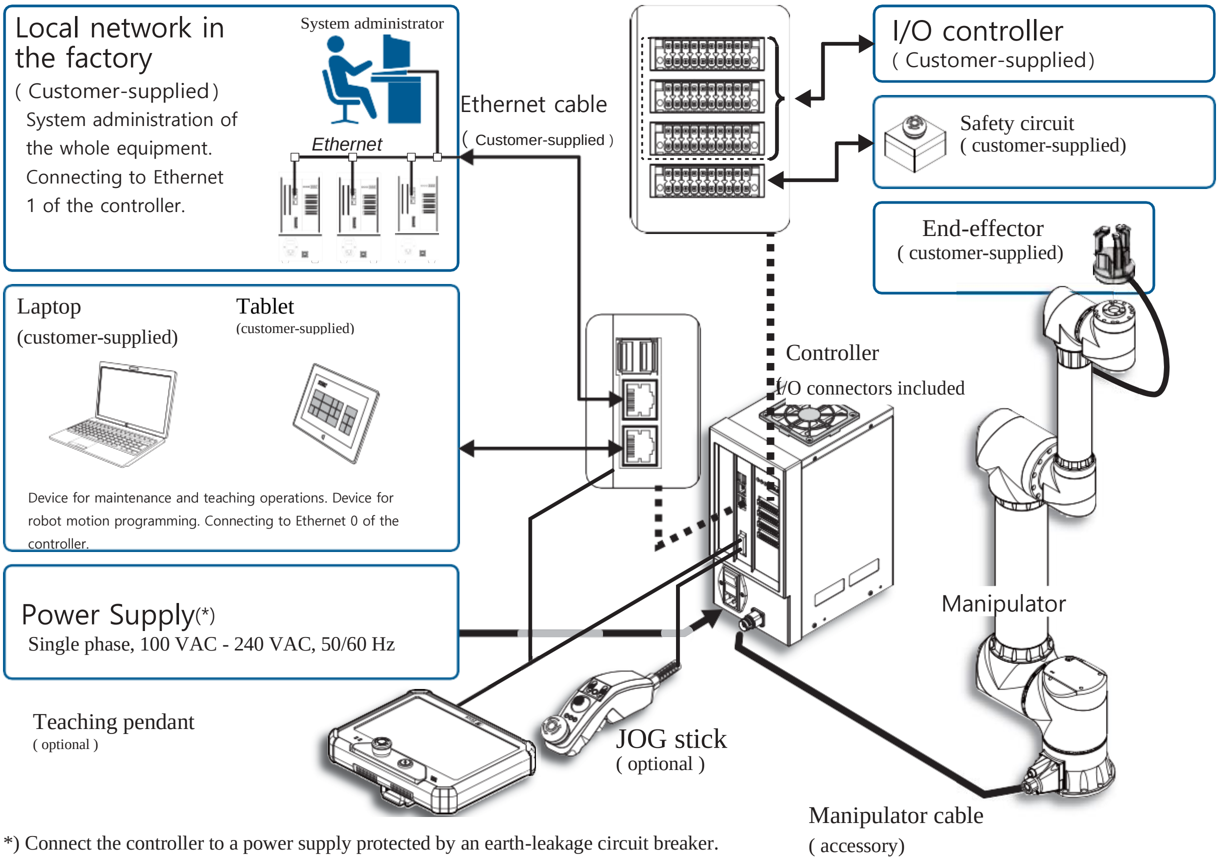
Global ZEUS

ZEUS CO., LTD.
132, Annyeongnam-ro, Hwaseong-si, Gyeonggi-do, SOUTH KOREA

Reference Document No. M0101-210524

MADE IN KOREA

2. System configuration



System administration of the whole equipment. Connecting to Ethernet 1 of the controller.

Device for maintenance and teaching operations. Device for robot motion programming. Connecting to Ethernet 0 of the controller.

Single phase, 100 VAC - 240 VAC, 50/60 Hz

*) Connect the controller to a power supply protected by an earth-leakage circuit breaker.

B Hardware

2 MANIPULATOR

1. Characteristics

It is possible to match the length of the arm in accordance with the customer's layout because the total length of the first and second arms is less than 860mm.

It can be divided into “pass-through type” and “turn around motion type” based on the difference in the length of the first arm and the second arm. For more details, please contact.

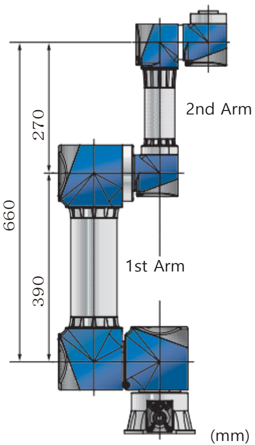
Example of Arm length

Pass-through type

First arm > Second arm

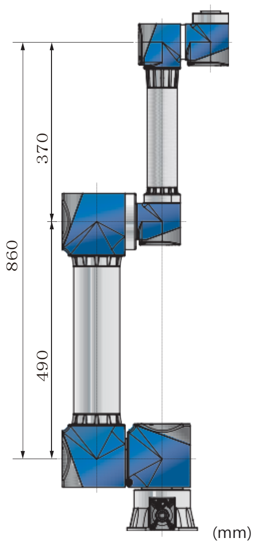
- There are many points that cannot be reached in the perpendicular coordinate system operation of the height near the manipulator mounting surface.
- Pass-through operation is possible

ZRA-0503P



Arm length	660
First arm	270
Second arm	390

ZRA-0515P



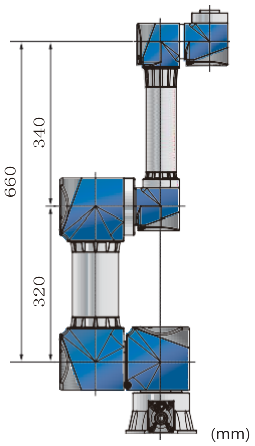
Arm length	860
Second arm	370
First arm	490

Turn around motion type

First arm < Second arm

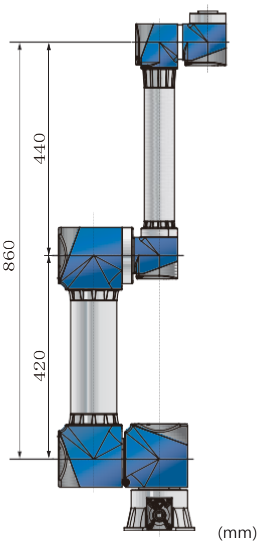
- There are few points that cannot be reached in the perpendicular coordinate system operation of the height near the manipulator mounting surface.
- Pass-through operation is not possible.

ZRA-0502N



Arm length	660
Second arm	340
First arm	320

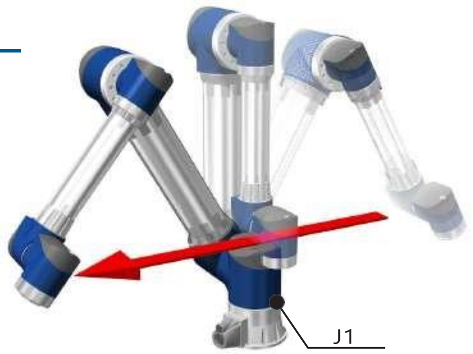
ZRA-0514N



Arm length	860
Second armi	440
First arm	420

｢What is pass-through motion?

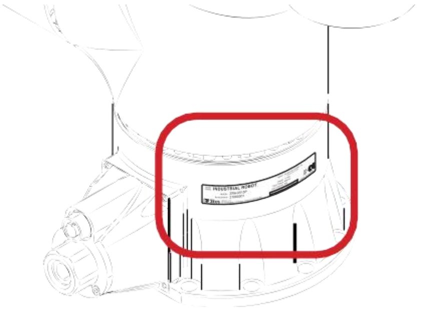
As the characteristic operation of the manipulator has the 1st arm longer than the 2nd arm. As shown on the right, do not rotate the J1 but can move to the completely opposite side of the manipulator.



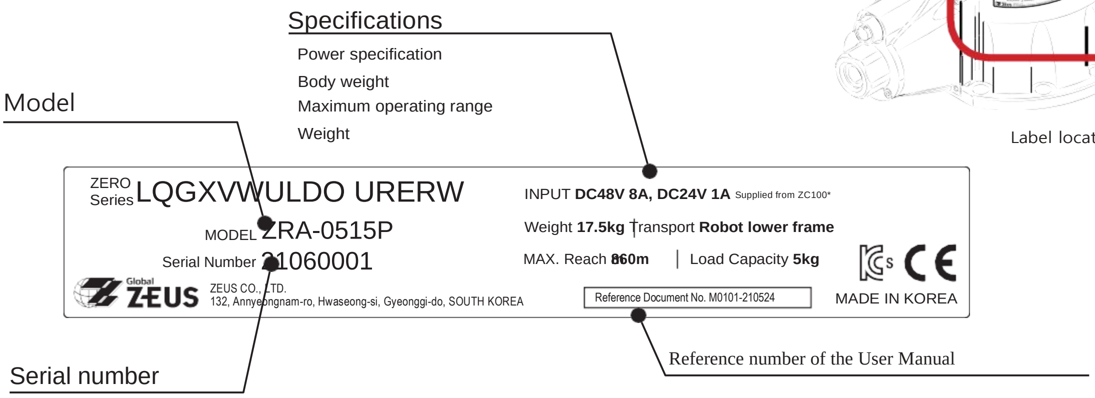
2. Label

The product label and C.CODE label are pasted on the manipulator.

Product label



Label location



ZERO Series LQGXVWULDO URERW

MODEL ZRA-0515P

Serial Number 21060001

Global ZEUS

ZEUS CO., LTD.
132, Annyeongnam-ro, Hwaseong-si, Gyeonggi-do, SOUTH KOREA

INPUT **DC48V 8A, DC24V 1A** Supplied from ZC100*

Weight **17.5kg** †ransport **Robot lower frame**

MAX. Reach **860m** | Load Capacity **5kg**

Reference Document No. M0101-210524

MADE IN KOREA

Serial number description

2 1	0 6	0 0 0 1
Year of manufacture	Month of manufactu-re	Manufactu-re number

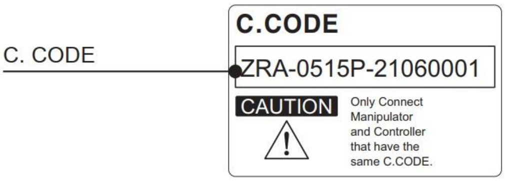
Year of manufacture: "21"= 2021 (last 2 digits)
Month of manufacture: "01"= January ~ "12"= December

Manufacture number: "0001"~ "9999"

For example, this product label has model name of manipulator is "ZRA-0515P", serial number is "21060001".

C. CODE LABEL

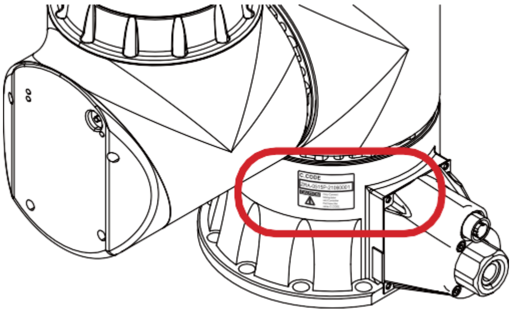
❗	Only connect manipulator and controller that have the same C.CODE (Connector code)	⚠



C.CODE

ZRA-0515P-21060001

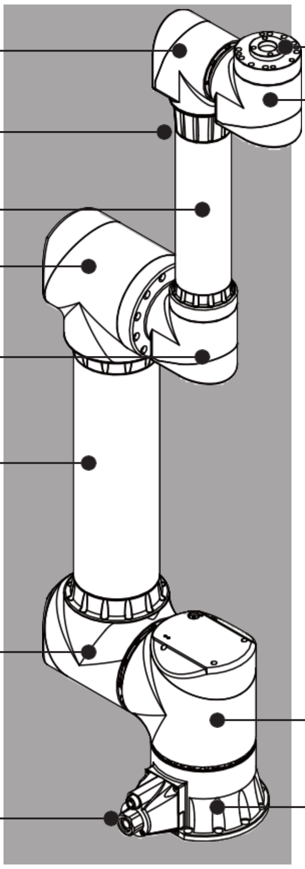
CAUTION ⚠ Only Connect Manipulator and Controller that have the same C.CODE.



부착 위치

For example, this C.CODE label has model name of manipulator is "ZRA-0515P", serial number is "19030006" .

2. Part names



Joint 5 (J5)
Small joint

Arm I/O connector
Located in the back

2nd Arm

Joint 3 (J3)
Middle joint

Joint 4 (J4)
Small joint

1st Arm

Joint 2 (J2)
Middle joint

Junction Box
This is a cover of the manipulator cable connecting to the controller. Junction Box connected to Manipulator

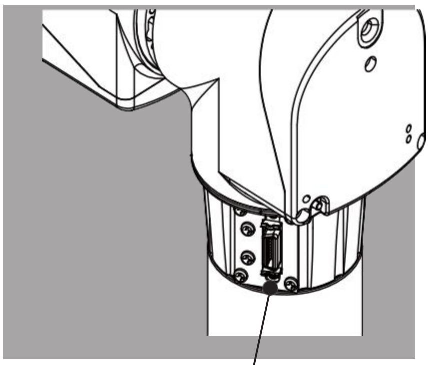
Top flange

Joint 6 (J6)
Small joint

Joint 1 (J1)
Middle joint

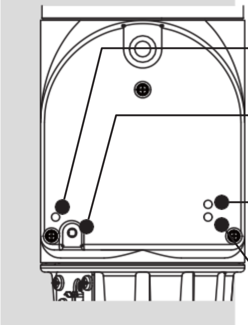
Bottom flange

Details of Arm I/O

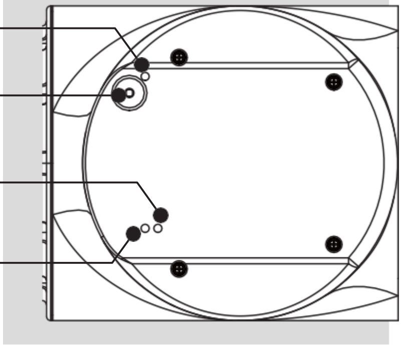


Arm I/O connector

Details of 4th, 5th and 6th Joints



Details of 1st, 2nd, and 3rd Joints



State display LED

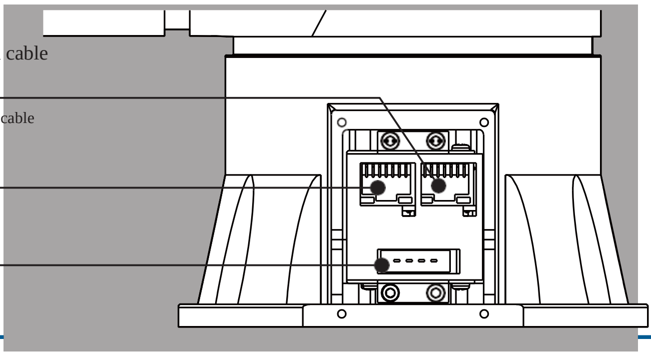
Brake Release Button

EtherCAT Status LED
LOUT

EtherCAT Status LED
LIN

Details of Junction Box

Remove the cover to see the following connectors



Connector for the EtherCAT communication cable
LIN
Connect the EtherCat connector end of the manipulator cable

Connector for the EtherCAT communication cable
LOUT
Use this for connecting Slave Device

Power supply connector

3. INSTALLATION

⚠ Caution

Install correctly according to the installation conditions.

Installation conditions

Item	Specifications
Operating temperature	0 ℃ - 40℃
Operating humidity	30 %RH - 85 %RH (non-condensing)
Operating atmosphere	Indoor use only (no direct sunlight). Free from corrosive gases, flammable gases, oil mist, any liquids including water, dust, flammables, and grinding agent. Good ventilation.
Pollution degree	2 (Comply with IEC60664-1)
Vibration and impact	Comply with IEC61131-2 (controller only) Vibration during operation 0.5G or less (No excessive vibration or shock)
Degree of protection	IP40 (manipulator, controller)
Power Supply	Supplied from compatible controller
Grounding	Class D (Ground resistance of 100 Ω or less)
Noise	No strong electromagnetic field in proximity (*)

*) The robot may operate incorrectly due to strong electromagnetic field.

The specifications mentioned in this document are general specifications. For detailed content, please refer to the delivery specification sheet.

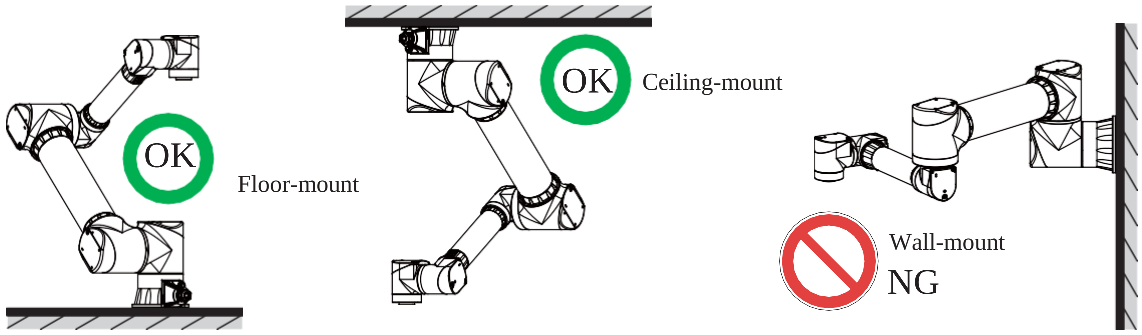
⚠ Caution

We recommend independently testing all operating programs and features outside the operational space of other devices in case the robot is used together or in combination with devices that have the potential to damage the robot.

Mounting styles

Caution

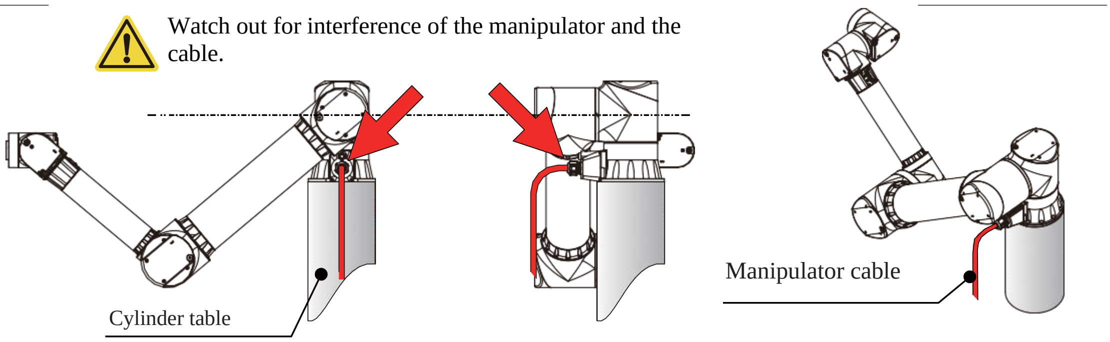
Observe the mounting styles specified and mount the manipulator properly



Caution

Do not allow cables or connectors to be impacted within the operating range of the manipulator in the case of installing the manipulator on a cylinder table.

Watch out for interference of the manipulator and the cable.



Caution

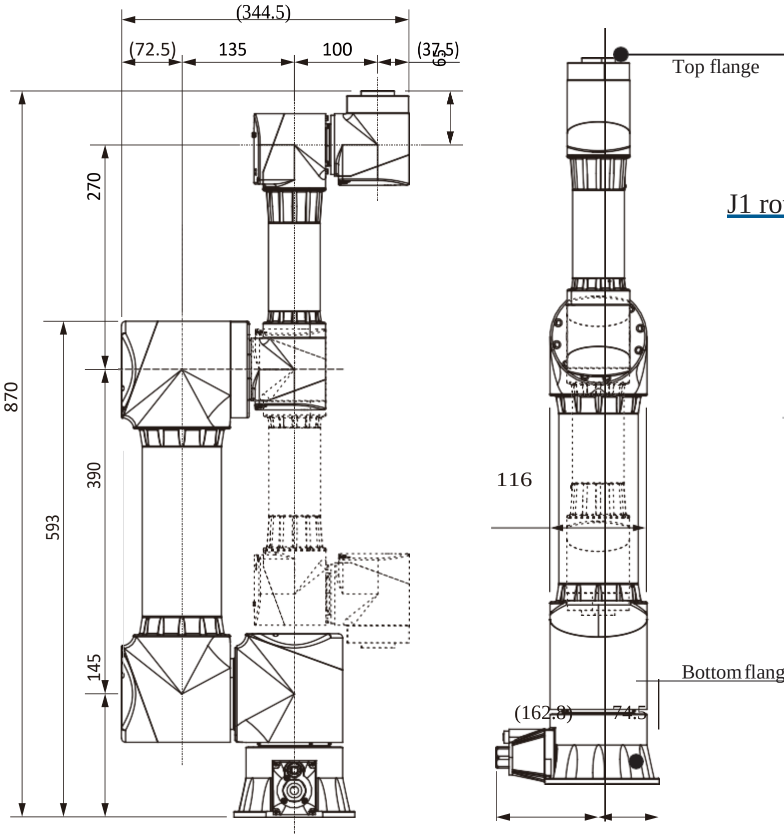
Please refer to the dimensional diagram for top flange installation in relation to the end-effector installation, and the dimensional diagram for the fixed installation of the bottom flange when installing the manipulator. It is recommended to attach all 7 screws for the bottom flange.

4. DIMENSIONAL DIAGRAM

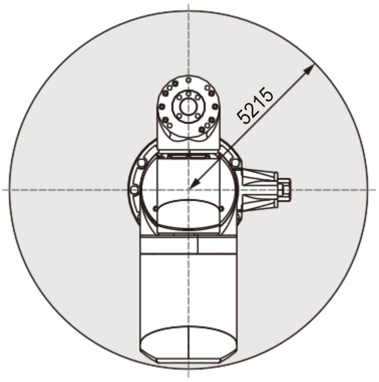
ZRA-0503P Arm length: 660 mm Pass Through Type

Not to scale

(mm)

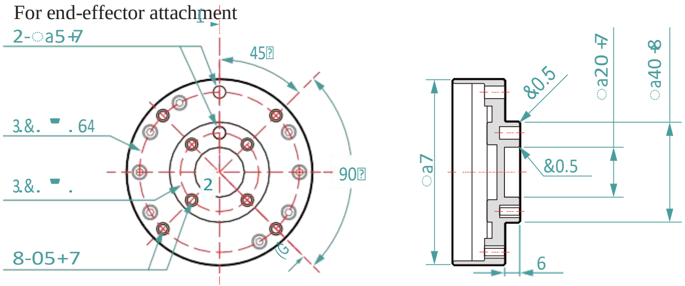


J1 rotational range



Top flange

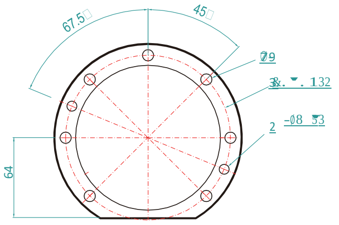
For end-effector attachment



Cross section E-O-É

Bottom flange

For mounting



BOTTOM VIEW

Mounting bottom flange

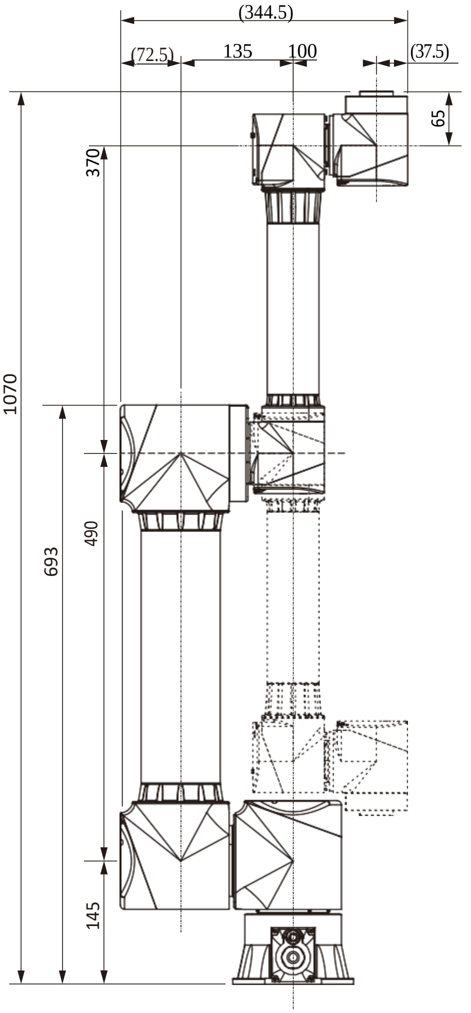
To mount bottom flange, using M8 hex sockethead cap screws of at least 30 mm long is recommended.

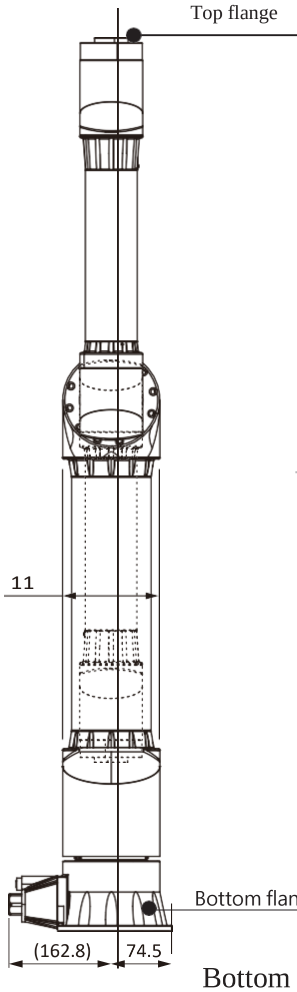
The recommended tightening torque is 22 Nm.

ZRA-0515P   Arm length: 860 mm   Pass Through Type

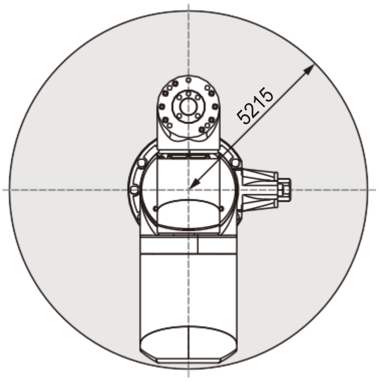
Not to scale

(mm)



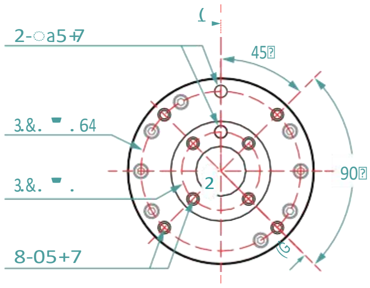


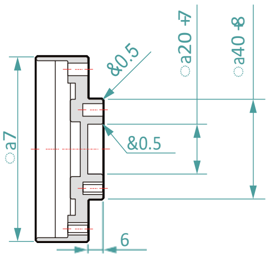
J1 rotational range



Top flange

For end-effector attachment

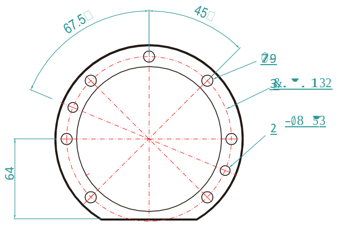




Cross section E-O-É

Bottom flange

For mounting



BOTTOM VIEW

Mounting bottom flange

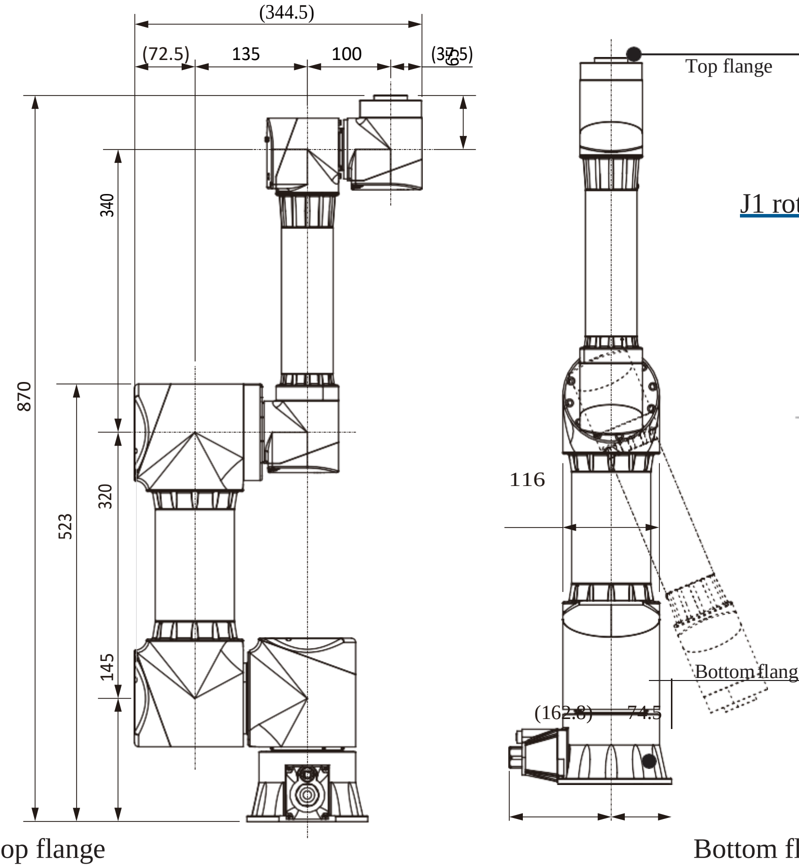
To mount bottom flange, using M8 hex sockethead cap screws of at least 30 mm long is recommended.

The recommended tightening torque is 22 Nm.

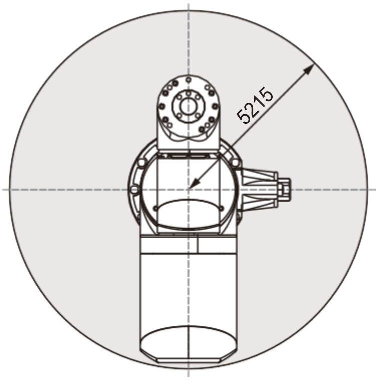
ZRA-0502N Arm length: 660 mm Turn Around Motion Type

Not to scale

(mm)

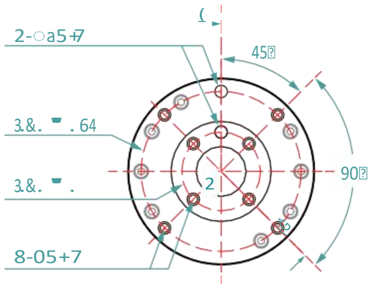


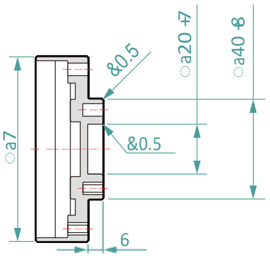
J1 rotational range



Top flange

For end-effector attachment

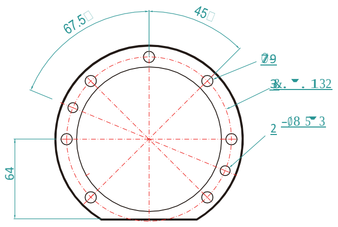




Cross section E-O-É

Bottom flange

For mounting



BOTTOM VIEW

Mounting bottom flange

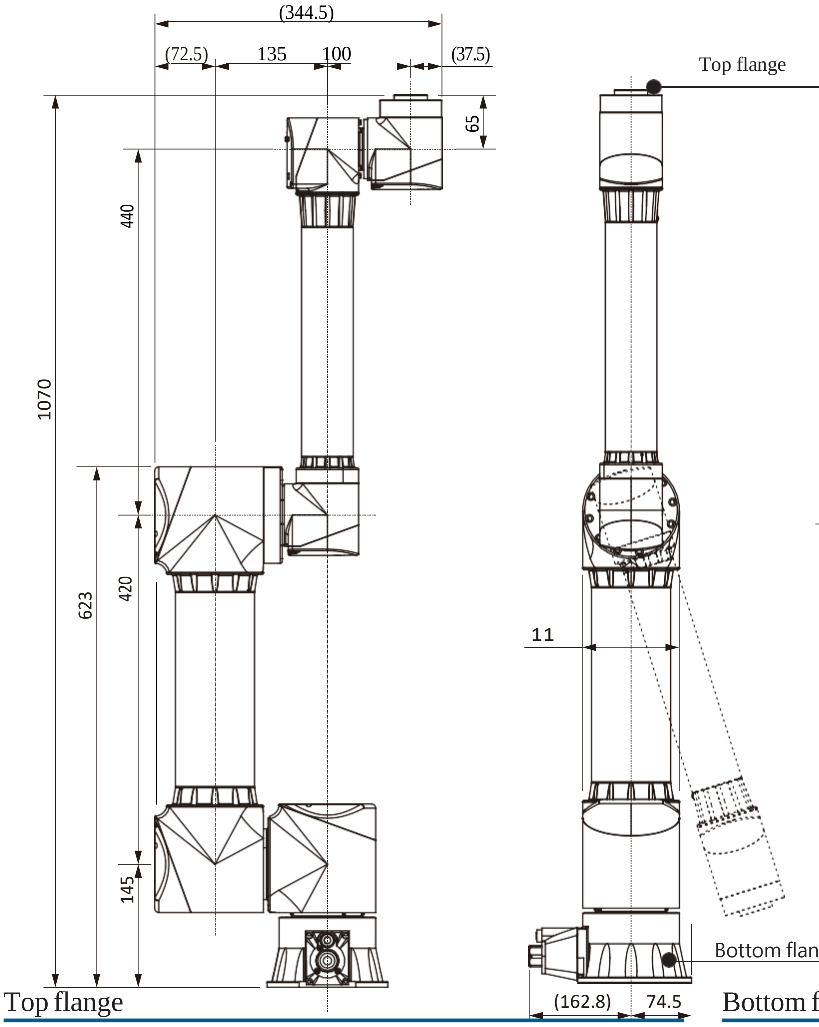
To mount bottom flange, using M8 hex sockethead cap screws of at least 30 mm long is recommended.

The recommended tightening torque is 22 Nm

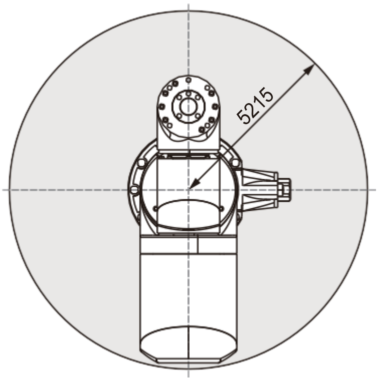
ZRA-0514N Arm length: 860 mm Turn Around Motion Type

Not to Scale

(mm)

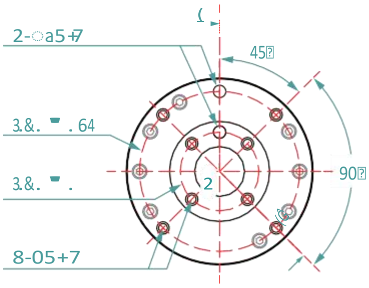


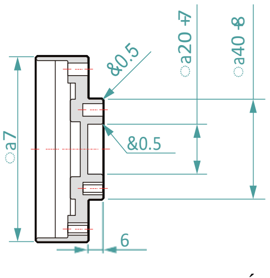
J1 rotational range



Top flange

Top flange

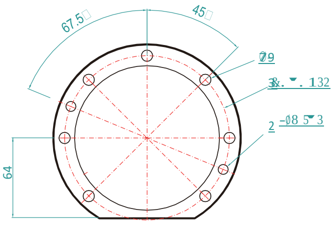




Cross section E-O-É

Bottom flange

For mounting



BOTTOM VIEW

Mounting bottom flange

To mount bottom flange, using M8 hex sockethead cap screws of at least 30 mm long is recommended.
The recommended tightening torque is 22 Nm.

5. SPECIFICATIONS

Item		Unit	ZRA-0503P	ZRA-0515P	ZRA-0502N	ZRA-0514N
Structure		—	Vertical Articulated robot			
Degrees of motion freedom (DOF)		—	6			
Mount direction		—	Floor, Ceiling			
Drive system		—	BLDC motor			
Position detection method		—	Multi-turn Absolute Encoder (Battery Backup)			
Position control method		—	Servo control			
Break		—	J1, J2, J3: Holding brake (Disc brake) J4, J5, J6: Holding brake (Mechanical stopper)			
Payload (*1)	Standard	kg	5			
	Maximum		7	5	7	5
Arm Length (1st Arm + 2nd Arm)		mm	660 (390 + 270)	860 (490 + 370)	660 (320 + 340)	860 (420 + 440)
Work area		mm	1320	1720	1320	1720
Motion range (*2)	J1	deg	480 (± 240)	480 (± 240)	480 (± 240)	480 (± 240)
	J2		480 (± 240)	480 (± 240)	480 (± 240)	480 (± 240)
	J3		480 (± 240)	480 (± 240)	300 (± 150)	300 (± 150)
	J4		480 (± 240)	480 (± 240)	480 (± 240)	480 (± 240)
	J5		480 (± 240)	480 (± 240)	480 (± 240)	480 (± 240)
	J6		720 (± 360)	720 (± 360)	720 (± 360)	720 (± 360)
Resultant Velocity (*3)		mm/sec	4420	5540	4570	5700
Repeatablity		mm	±0.02			
Permissible load inertia (*5)	J4	$x10^{-4}$ $kg \cdot m^2$	0.15	0.15	0.15	0.15
	J5		0.27	0.27	0.27	0.27
	J6		0.33	0.33	0.33	0.33
Outer Dimensions		—	149 x 331 x 873	149 x 331 x 1073	149 x 331 x 873	149 x 331 x 1073
Body Weight		kg	17.2	17.5	17.2	17.5
Compatible controller		—	ZC1***			
Arm I/O (for Tool)		—	8 input ports, 4 output ports / Asynchronous communication RS-422 1 port / DC 24 power output			
Manipulator cable length		m	3			
Manipulator mount		—	M8 screws at 7 spots (refer the dimension drawing) (*6)			
End-effector mount		—	M5 screws at 4 spots (refer the dimension drawing)			
Noise		dB	Under 70 (Based on our test)			

*1) The payload includes loads of tasks, weight of tools, and so. Allowable torque exceeding error c13 and overload error c14 may occur even within specification depending on the posture, speed, acceleration/deceleration time, direction of operation, etc. Adjust motion factors and variables then.

*2) Refer to “5. Coordinate Systems and Posture” for information on definitions of axes. Depending upon arm postures, unreachable points exist even within the work envelop.

*3) Value is for a reference.

*4) In case of the maximum load at the maximum speed.

*5) Depends on operating conditions, such as acceleration and deceleration.

*6) Using screws of at least 30 mm long is recommended.

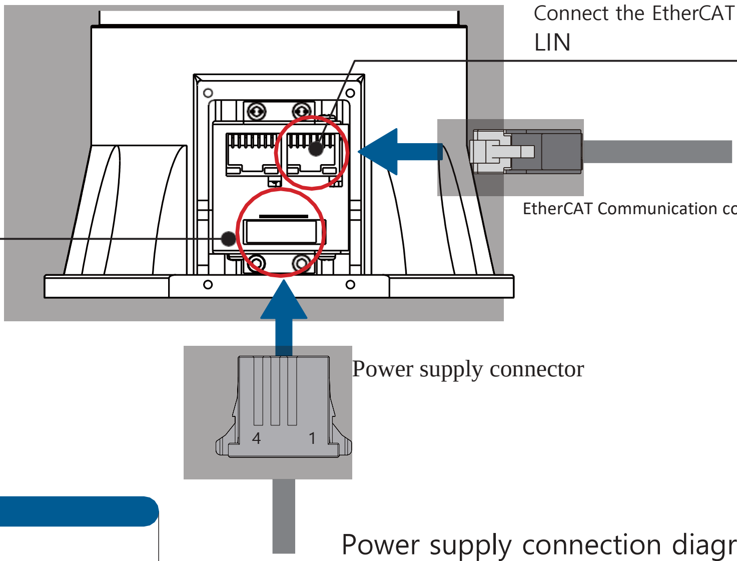
Replenishment) This product is a stop category "0". Corresponds to PL = d.

The compressed air pipe cannot pass through inside of the manipulator.

6. Connector

1. Connecting the manipulator cable

Remove the Junction Box, refer to the drawing below, then connect the connector of the manipulator cable.
(Manipulator cable is an accessory)



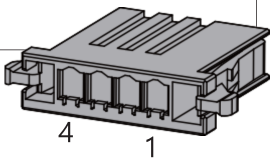
Connect the EtherCAT cable
LIN

EtherCAT Communication connector

Power supply connector
Connector model (manipulator side)
S04B-F31SK-GGXR
(JST)

Power supply connector

Cable
Connector model
F31FSS-04V-KX
(JST)



Power supply connection diagram

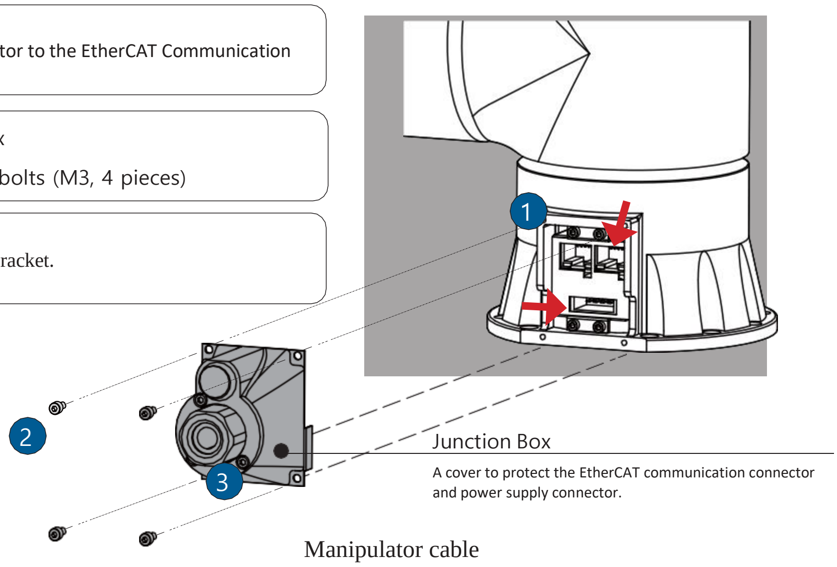
Connector	Signal	Description
1	24V	24V power supply
2	48V	48V power supply
3	GND	24V_GND, 48V_GND
4	FG	Grounding

Connecting method

1 Connect the power supply connector to the EtherCAT Communication connector.

2 Tighten bolts of Junction Box
Bolts: Hexagon countersunk bolts (M3, 4 pieces)

3 Fasten the connector cover panel bracket.



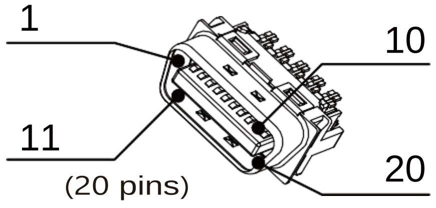
Junction Box
A cover to protect the EtherCAT communication connector and power supply connector.

Manipulator cable
(The manipulator cable is integrated with the junction box.)

2. Arm I/O connector

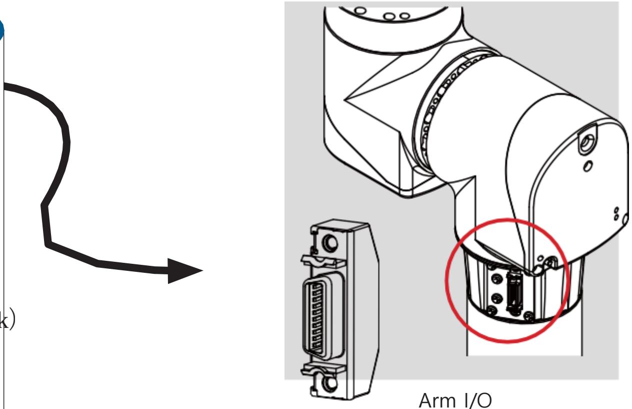
Arm I/O is the I/O port used for the tool attached to the end of the manipulator.

케이블 측



Connector Model
DF02P020F22A1
(JAE)

Header（type of screw lock）
DF02D020B22
(JAE)



Connector Model (Manipulator side)
DF02R020NA3
(JAE)

Connector and cable are not included.
Customer should provide when needed.

Connector pinout

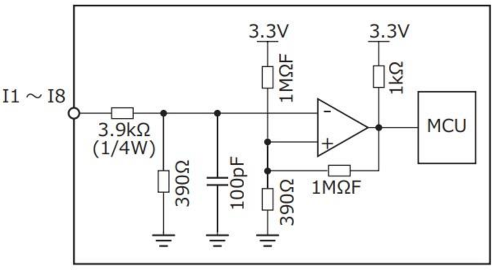
Connector	Signal Name	Description	Connector	Signal Name	Description
1	24V_OUT	24V power supply output	11	24V_OUT	24V power supply output
2	I1	Common input	12	I2	Common input
3	I3	Common input	13	I4	Common input
4	I5	Common input	14	I6	Common input
5	I7	Common input	15	I8	Common input
6	O1	Common output	16	O2	Common output
7	O3	Common output	17	O4	Common output
8	D+	RS422_TXD+/RS485_D+	18	D-	RS422_TXD-/RS485_D-
9	RD+	RS422_RXD+	19	RD-	RS422_RXD-
10	G24	GND power supply	20	G24	GND power supply

3. Output/input circuit of Arm I/O

Common input circuit

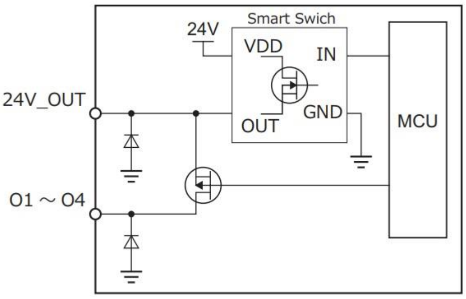
Item	Specifications
Method	Comparator circuit input（non-isolated）
Rated voltage	DC24 V
Input ON voltage	9 V typ.
Input impedance	4.3 k Ω typ.

n)



Common output circuit

Item	Specifications
Method	High side switch (non-isolated)
Rated voltage	DC24 V
Rated current	0.5 A (output current limit 0.7 A - 2.1



Example of common output/input connection

Notes

・The total current consumption of the device connected to the Arm I/O should be kept below 100mA (maximum 200mA)

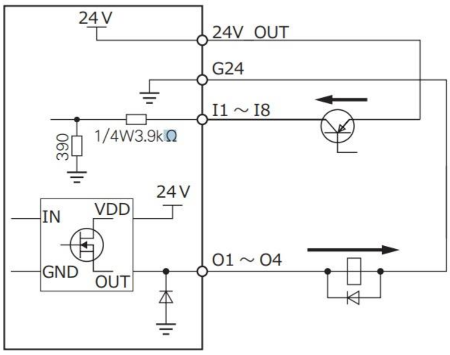
・ It is mandatory to prepare countermeasures for any increase in case of using inductive loads such as relays, magnetic induction at the output.

・Arm I/O wiring,

①Ensure sufficient reduction from motor transmission cables or high voltage cables.

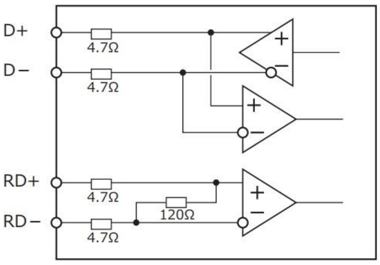
②Prepare noise countermeasures such as using shielded cables.

③Use a length of less than 1m.



Asynchronous transmission circuit

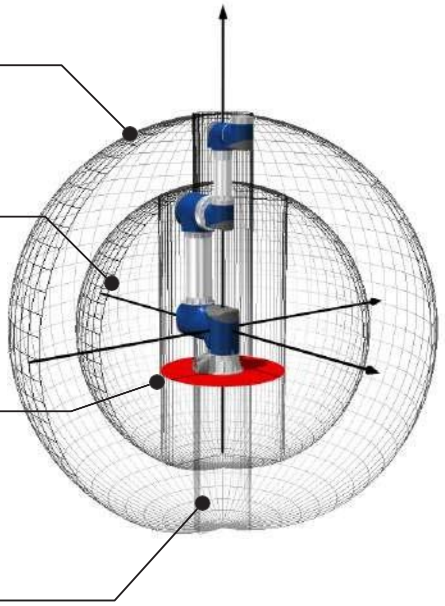
Is the interface of asynchronous transmission RS485 or RS422



🚫	It is strictly forbidden to connect with asynchronous transmission circuits other than the devices specified by our company.	⚠

7. MOVEMENT RANGE

ZRA-0503P Arm length: 660 mm Pass Through Type



Maximum reachable range of the top flange:

R725 sphere（around J2 rotantional axis)

1st Arm movable range:

（where contact with the manipulator or pinch point risk exists)

R500 sphere（around J2 rotantional axis)

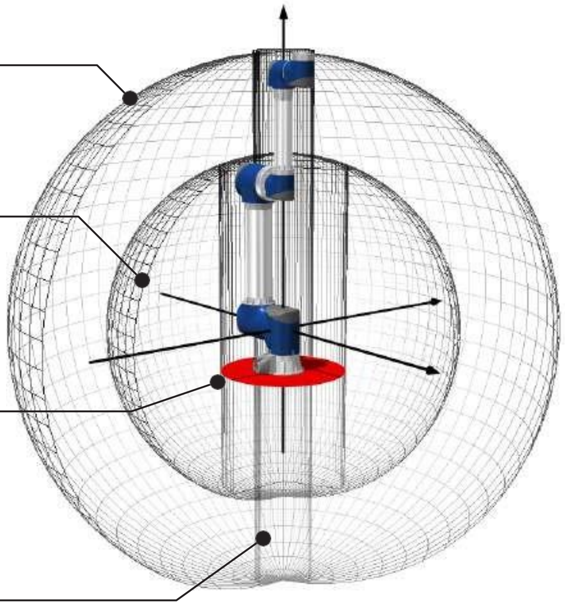
J1 minimum rotational range

(where contact with the manipulator or pinch point risk exists at J2=0° J3=180°)

Unreachable points of the top flange when top flange faces up or down

R215 cylinder (around J1 rotantional axis)

ZRA-0515P Arm length: 860 mm Pass Through Type



Maximum reachable range of the top flange:

R925 sphere (around J2 rotantional axis)

1st Arm movable range

(where contact with the manipulator or pinch point risk exists)

R600 sphere (around J2 rotantional axis)

J1 minimum rotational range:

Where contact with the manipulator or pinch point risk exists at J2=0° J3=180°)

Unreachable points of the top flange when top flange faces up or down

R100 cylinder (around J1 rotantional axis)

Replenishment

Depending upon arm postures, unreachable points exist even within the work envelop.

7 MOVEMENT RANGE

ZRA-0502N Arm length: 660 mm Turn Around Motion Type

Maximum reachable range of the top flange

R725 sphere (around J2 rotantional axis)

1st Arm movable range

(Where contact with the manipulator or pinch point risk exists)

R435 sphere (around J2 rotantional axis)

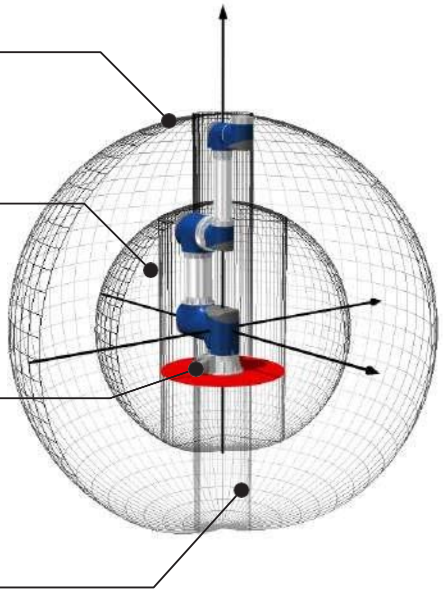
J1 minimum rotational range:

(Where contact with the manipulator or pinch point risk exists at J2=0° J3=180°)

R215 cylinder (around J1 rotantional axis)

Unreachable points of the top flange when top flange faces up or down

R100 cylinder (around J1 rotantional axis)



ZRA-0514N Arm length: 860 mm Turn Around Motion Type

Maximum reachable range of the top flange

R925 sphere (around J2 rotantional axis)

1st Arm movable range

(Where contact with the manipulator or pinch point risk exists)

R520 sphere (around J2 rotantional axis)

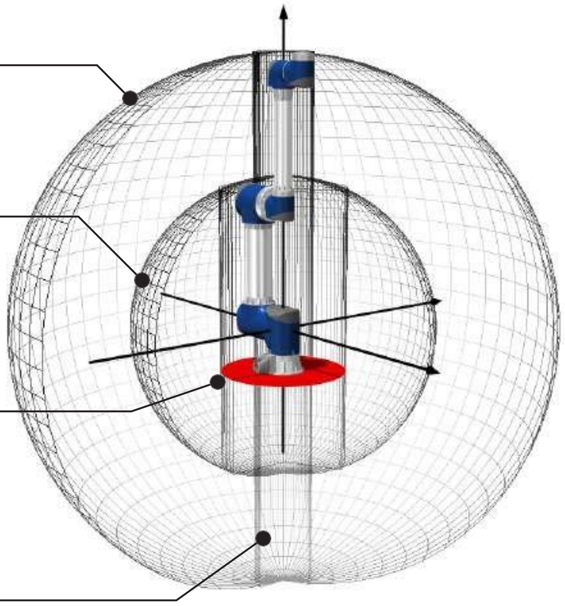
J1 minimum rotational range

(Where contact with the manipulator or pinch point risk exists at J2=0° J3=180°)

R215 cylinder (around J1 rotantional axis)

Unreachable points of the top flange when top flange faces up or down

R100 cylinder (around J1 rotantional axis)



Replenishment

Depending upon arm postures, unreachable points exist even within the work envelop.

8. END-EFFECTOR DESIGN

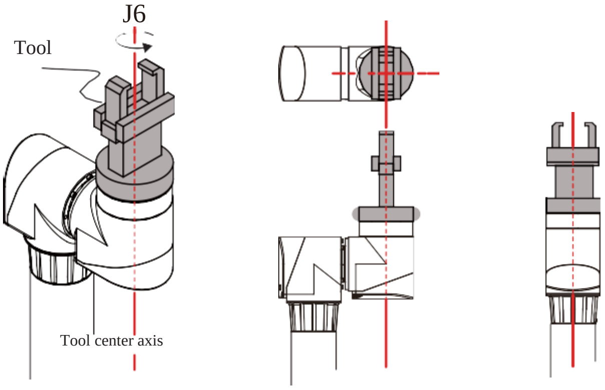
⚠ Caution

When designing a tool which will be attached to the top flange, thoroughly validate manipulator postures and motion ranges. See below for examples.

Example 1

Recommended

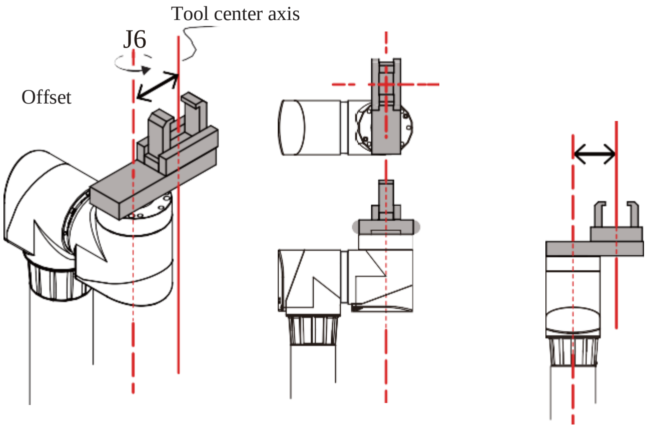
The tool center point coincides with the J6 rotational axis.
If the distance from the top flange to the outermost part of the tool is large, the load on the manipulator will be significant, leading to vibration or a low operating speed.



Example 2

Not recommended

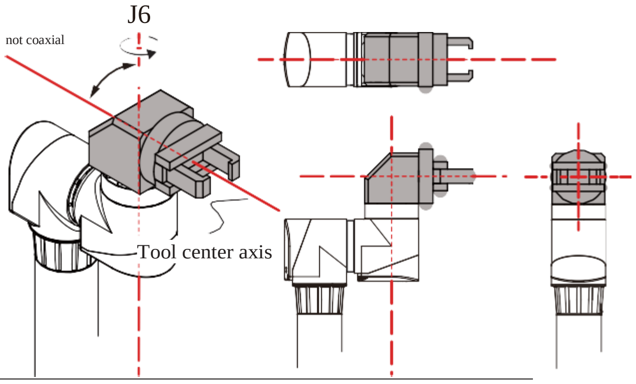
There is an offset between the central axis of the tool and the J6 rotational axis, the robot may become unable to handle a work piece.



Example 3

Not recommended

Because the central axis of the tool and the J6 rotational axis are not coaxial, the robot may become unable to handle a work piece.



NOTE

B Hardware

3 CONTROLLER

1 . MODEL NAME AND LABEL

1. Model name

Model name of controller ZC 1001

ZC: Product code

1001: Model

Product code

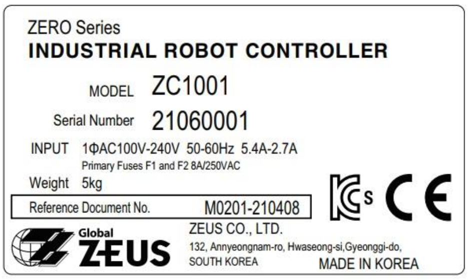
Model

Symbol	Robot
1000	ZERO series
1001	ZERO series + ZP1000

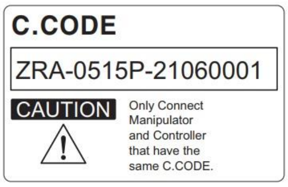
2. Label

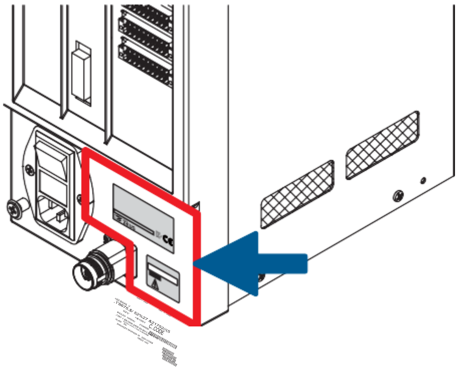
Labels | Label location

Product label


ZERO Series
INDUSTRIAL ROBOT CONTROLLER
MODEL ZC1001
Serial Number 21060001
INPUT 1ΦAC100V-240V 50-60Hz 5.4A-2.7A
Primary Fuses F1 and F2 8A/250VAC
Weight 5kg
Reference Document No. M0201-210408
ZEUS CO., LTD.
132, Annyeongnam-ro, Hwaseong-si,Gyeonggi-do,
SOUTH KOREA
MADE IN KOREA
Global ZEUS

C. Code Label


C.CODE
ZRA-0515P-21060001
CAUTION
Only Connect Manipulator and Controller that have the same C.CODE.



For example, this label has the manipulator’s serial number as “21060001”, C. CODE “0515P-21060001”. This representation varies with each product. The serial number system is the same as that of the manipulator.

Please connect the controller to the manipulator according to the same C.CODE combination.

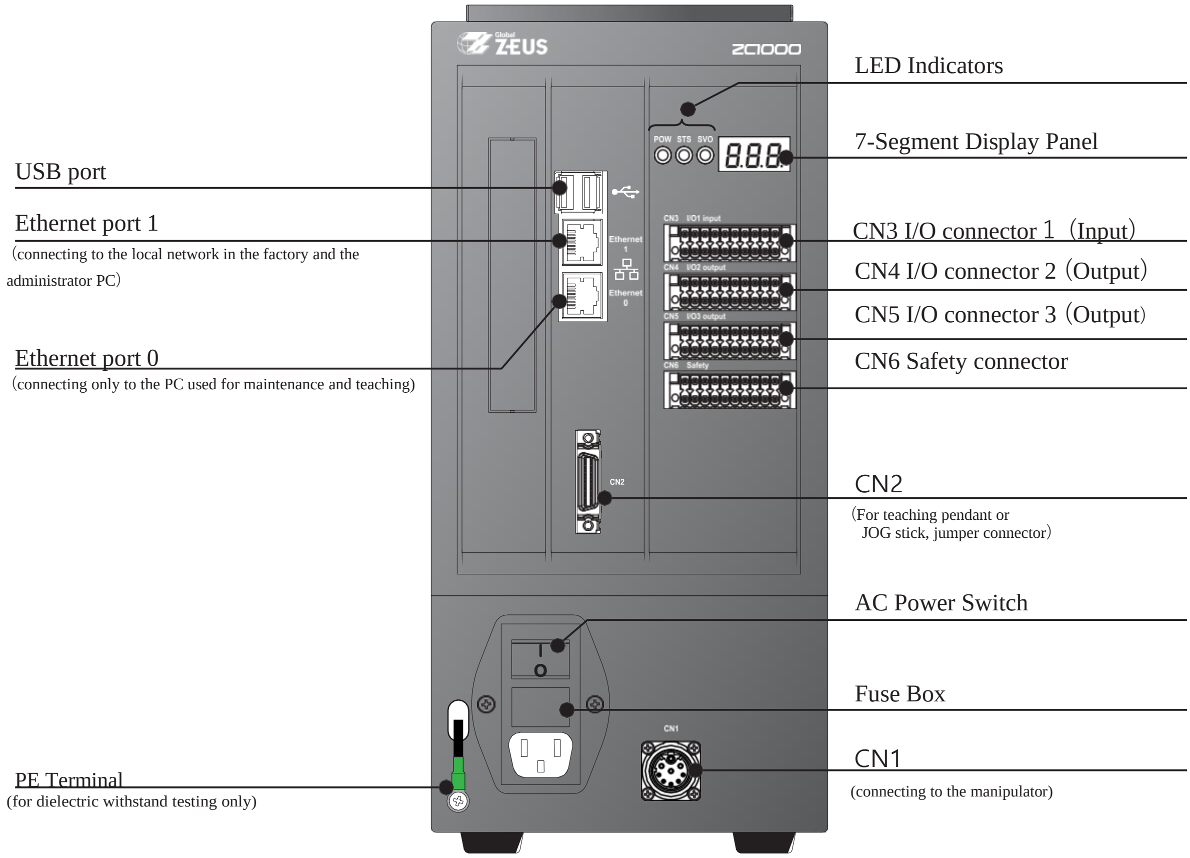
C.CODE represents the combination of the controller and the manipulator.
Please verify the C.CODE label and then connect the C.CODE accordingly.

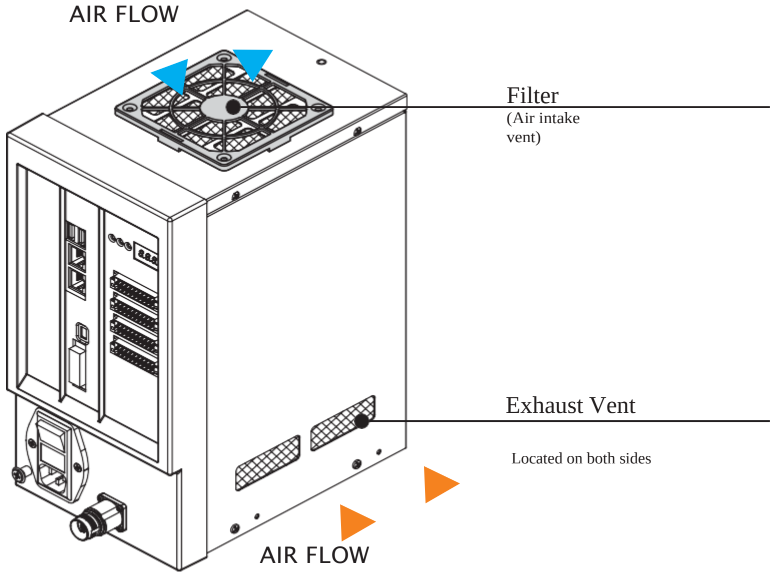
2. PART NAME

Caution

The PE end-effector is dedicated for internal voltage testing.

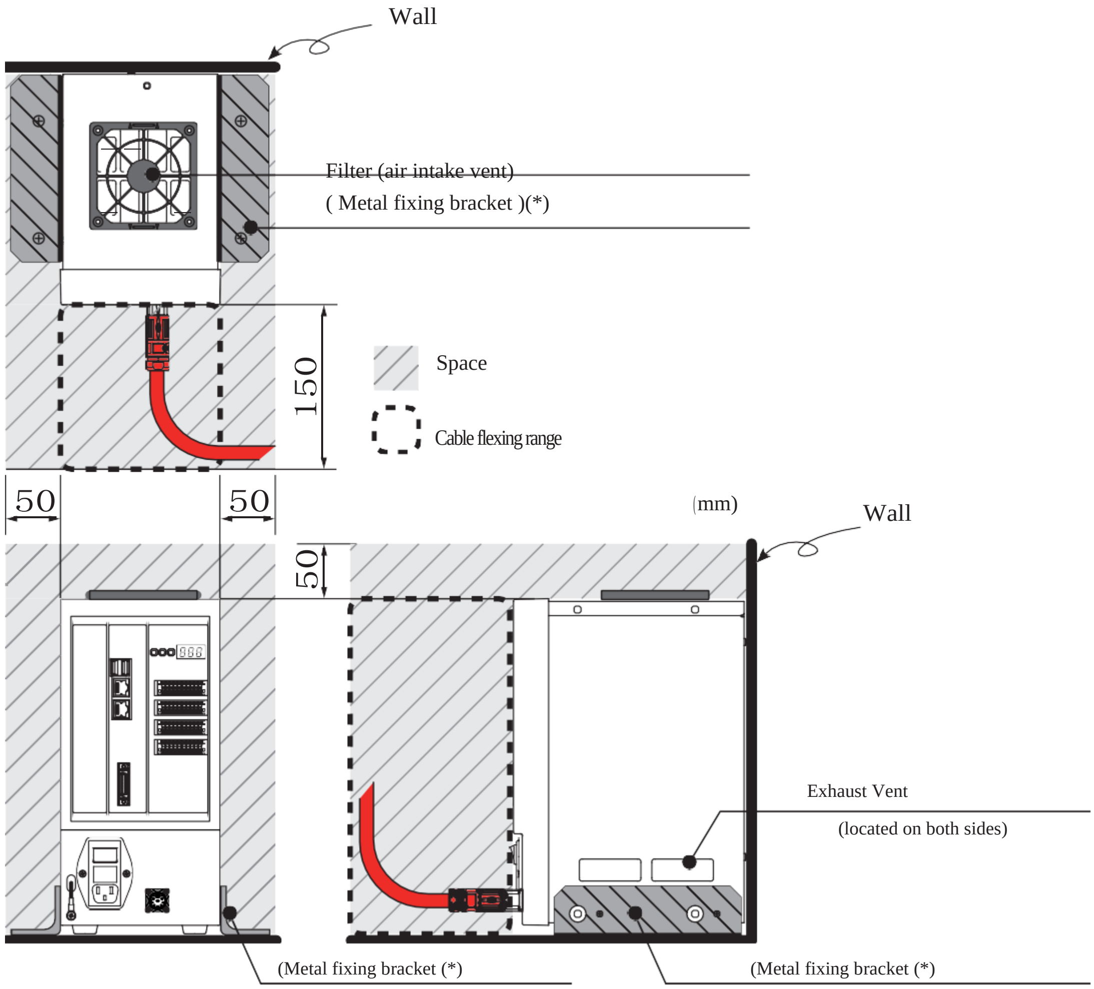
- Do not remove the screws.
- Do not connect anything.


Global ZEUS
ZC1000
LED Indicators
POW STS SVO
8.8.8.
7-Segment Display Panel
USB port
Ethernet port 1
（connecting to the local network in the factory and the administrator PC）
Ethernet 1
Ethernet 0
Ethernet port 0
（connecting only to the PC used for maintenance and teaching)
CN3 I/O1 input
CN4 I/O2 output
CN5 I/O3 output
CN6 Safety
CN3 I/O connector 1（Input）
CN4 I/O connector 2（Output）
CN5 I/O connector 3（Output）
CN6 Safety connector
CN2
CN2
（For teaching pendant or JOG stick, jumper connector）
AC Power Switch
Fuse Box
CN1
CN1
(connecting to the manipulator)
PE Terminal
(for dielectric withstand testing only)



⚠ Caution

🚫	Do not install in enclosed spaces. Do not block the air intake and exhaust vents	⚠ ⚠
❗	Please install in an environment with an ambient temperature below 40 ℃. Refer to the drawing below and install the controller in a flat position with sufficient space. It is recommended to use dedicated screws (M3 x 4 pieces) on the side of the controller, with anti-reversal treatment. Replace the designated filter periodically.	⚠ ⚠

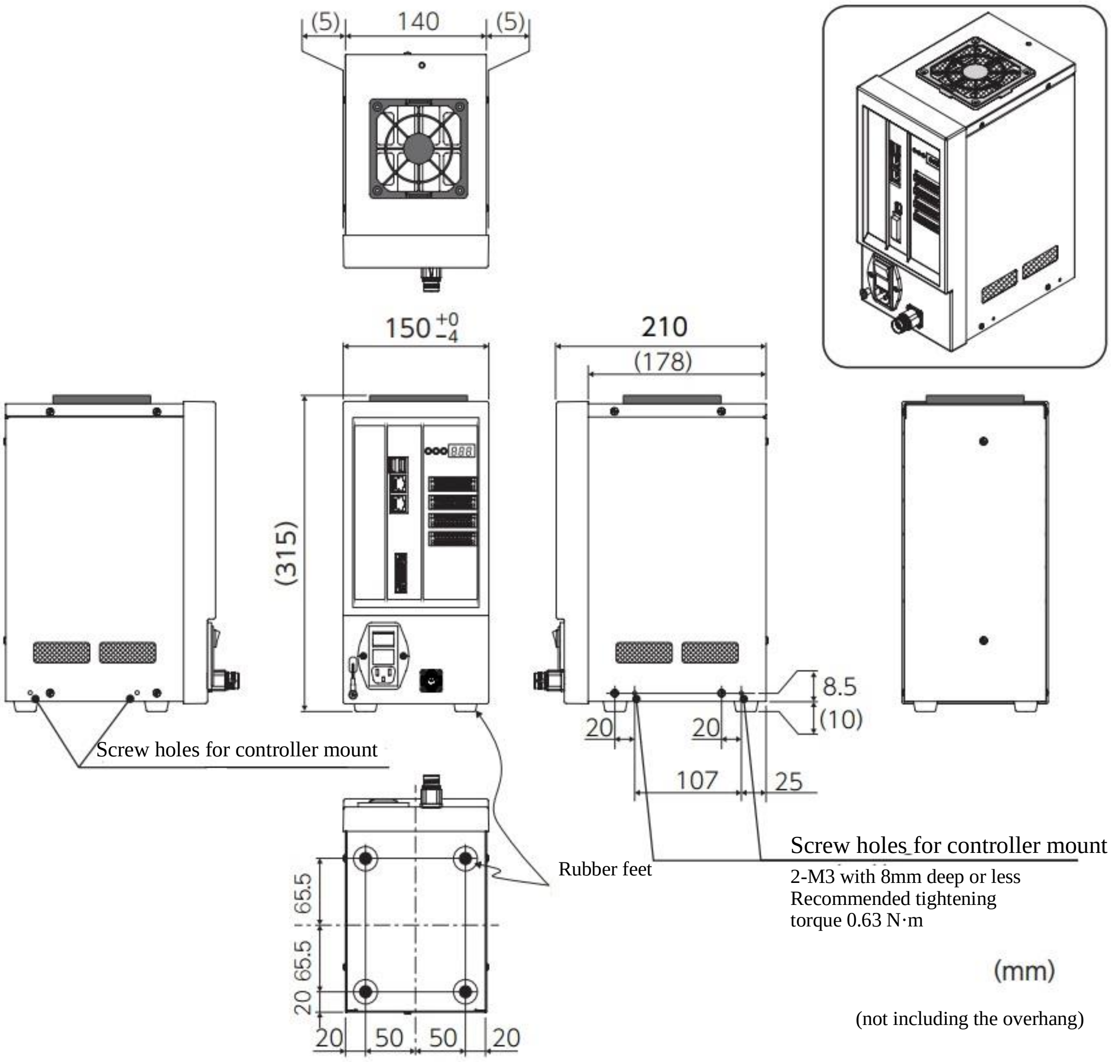


*) User-supplied

4. Dimension drawing

Caution

When designing a metal fitting, make it so that the cover fixing screws are 20 mm away from the controller mount holes and also the air inlets will not be blocked.



Screw holes for controller mount

Rubber feet

Screw holes for controller mount
2-M3 with 8mm deep or less
Recommended tightening
torque 0.63 N·m

(mm)

(not including the overhang)

5 . SPECIFICATIONS

1. General specifications

Item		ZC1000	ZC1001	Note
Compatible Manipulator		ZERO series		In case using Teaching Pendant(ZP1000) ZC1001 is necessary
General Specifications	Dimensions	(See a dimension drawing)		The overhang is not included
	Weight	5 kg		—
	Number of Control Axes	6 axes		—
	ProgrammingMethod	Off-line programming with a PC		Application programs are transferred with TFP and executed.
	Programminglanguage	Python		Use the special libraries for the robot operation
	Storage Memory	eMMC		—
	Teaching method	PC Jog Stick	PC Jog Stick Teaching Pendant	Monitoring, storing and controlling data via http with a web browser
Display function	7-segment display panel	3 digits		—
	Status LED indicators	3 lamps		—
Interface (Controller)	Manipulator Connector	1 Port		—
	Input	16 Bit		Isolated; selection of high-side or low-side
	Output	16 Bit		Isolated; selection of high-side or low-side
	Safety	1 Port		EMS x 2; Mode; Servo-On input Servo power monitor
	Ethernet	2 Port		—
	USB	2 Port		—
	JOG Stick	1 Port		A special input device I/F for teaching
Interface (ArmI/O)	Digital input	8 Bit		Not isolated; comparator input
	Digital output	4 Bit		Non-isolated; high-side switch
	Asynchronous communication	1 Ch		RS422/RS485
	Power output	24 V		0.2 A max
Power supply specifications (*)	Voltage	Single-phase 100 VAC - 240 VAC		—
	Frequency	50 Hz - 60 Hz		—
	Current	2.7 A, 230 VAC / 5.4 A, 115 VAC		—
	In-rush current	75 A, 230 VAC		—
	leakage current	5.0 mA, 240 VAC		—
	Rated short circuit current	1,500 A		UL File No. E10480
Grounding		Type 3 grounding or above		Grounding resistance value of 100 Ω or below
Safety	Rating	ISO 10218-1		Certified
	Voltage-resistance	1,500 VAC		Primary-FG, 1 minute
	Insulation resistance	1 M Ω or above		I/P - FG 500VDC / 25°C / 70%RH
EMC		EN61000-6-2:2005 EN55011：2009+A1:2010		Heavy industrial level

*) Input voltage
Be no power outage for more than 20ms
Gain sufficient power including in-rush current
Use fuses with rated current: 8A, rated breaking capacity: AC250 V / 1,500 A

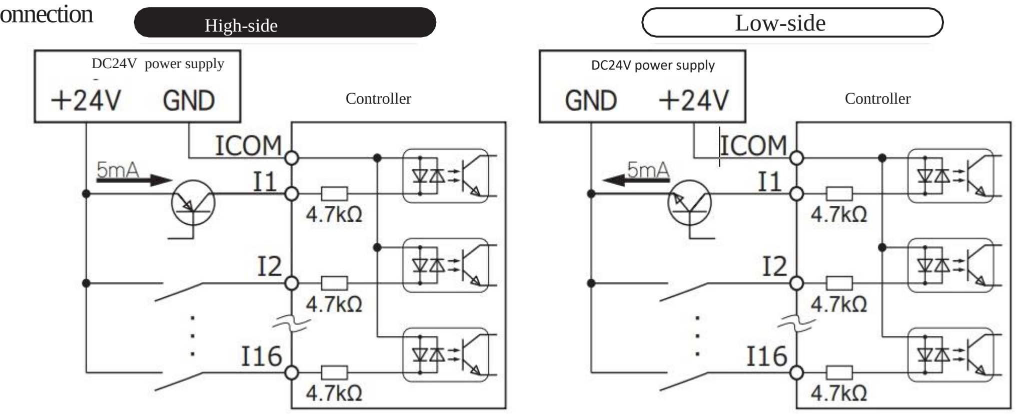
The specifications mentioned in this document are general specifications. For detailed content, please refer to the delivery specification.

2. Output/input circuit of I/O connector

Input circuit

Item	Specifications
Method	Port coupling unit (16-point common power supply)
Rated voltage	DC 24 V
Input ON current	5 mA typ. (Input OFF current less than 2.5mA)

Example of connection

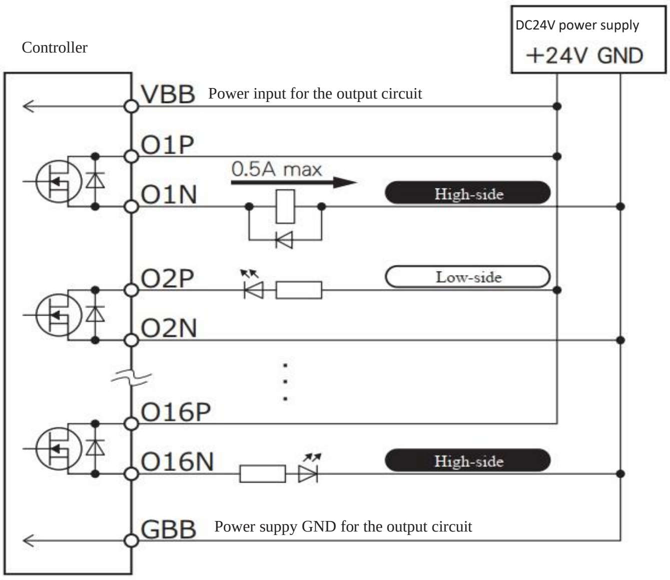


Output circuit

Item	Specifications
Method	Independent output drain, source (separate top and bottom correspondence)
Rated voltage	DC 24 V
Rated current	0.5 A (output current limit 0.6 A - 1.2 A)

- The output can be connected to various devices such as signal devices, relays, and buzzers.
- Please refer to the wiring example in the user manual of the connection device.
- Please connect a protective circuit (diode) when connecting devices with inductive components such as relays.

Example of connection



1. I/O connector

CN3: I/O connector 1 (input)

Connector	Signal Name	Description	Connector	Signal Name	Description
1A	P24	24V output of controller (*1)	1B	–	—
2A	IN1	Common input	2B	IN2	Common input
3A	IN3	Common input	3B	IN4	Common input
4A	IN5	Common input	4B	IN6	Common input
5A	IN7	Common input	5B	IN8	Common input
6A	IN9	Common input	6B	IN10	Common input
7A	IN11	Common input	7B	IN12	Common input
8A	IN13	Common input	8B	IN14	Common input
9A	IN15	Common input	9B	IN16	Common input
10A	G24	Common input	10B	ICOM	Common input

CN4: I/O connector 2 (output)

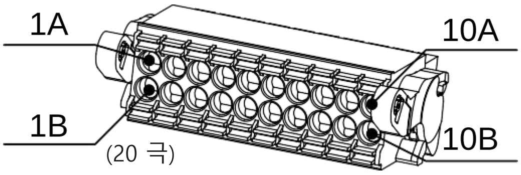
Connector	Signal Name	Description	Connector	Signal Name	Description
1A	P24	24V output of controller (*1	1B	VBB	24V power input for output circuit (*2)
2A	O1P	Common output drain	2B	O1N	Common output source
3A	O2P	Common output drain	3B	O2N	Common output source
4A	O3P	Common output drain	4B	O3N	Common output source
5A	O4P	Common output drain	5B	O4N	Common output source
6A	O5P	Common output drain	6B	O5N	Common output source
7A	O6P	Common output drain	7B	O6N	Common output source
8A	O7P	Common output drain	8B	O7N	Common output source
9A	O8P	Common output drain	9B	O8N	Common output source
10A	G24	Controller GND 24V (*1)	10B	GBB	GND 24V for output circuit (*2)

CN5: I/O connector 3 (output)

Connector	Signal Name	Description	Connector	Signal Name	Description
1A	P24	24V output of controller (*1)	1B	VBB	24V power input for output circuit(*2)
2A	O9P	Common output drain	2B	O9N	Common output source
3A	O10P	Common output drain	3B	O10N	Common output source
4A	O11P	Common output drain	4B	O11N	Common output source
5A	O12P	Common output drain	5B	O12N	Common output source
6A	O13P	Common output drain	6B	O13N	Common output source
7A	O14P	Common output drain	7B	O14N	Common output source
8A	O15P	Common output drain	8B	O15N	Common output source
9A	O16P	Common output drain	9B	O16N	Common output source
10A	G24	Controller GND 24V (*1)	10B	GBB	GND 24V for output circuit (*2)

Model name of I/O connector, safety connector: DFMC 1,5/10-ST-3,5-LR 1790564

(Phoenix Contact JSC.)



Feature division 🟦 of reference I/O Software

🟦 Memory map

*1)Dòng điện tiêu thụ sử dụng ở I/O thiết bị đầu cuối, đầu vào, đầu ra hãy để tổng dưới 100mA.
*2)Thiết bị cung cấp nguồn điện mạch điện đầu ra (VBB, GBB) I/O connector 2 và VBB, GBB của 3 được nối với nhau.
Không được nối nguồn điện khác vào thiết bị đầu cuối này.

2. Safety connector

Connect the safety connector correctly

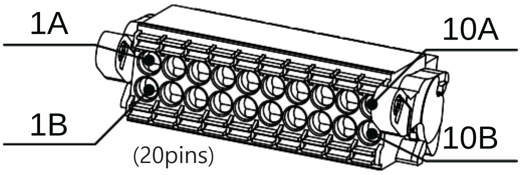
If not, the manipulator cannot be operated.

☞ 5 Refer to the wiring and power supply

N6: Safety connector

Connector	Signal Name	Description	Connector	Signal Name	Description
1A	EMS1_H+ (P24)	Emergency stop switch 1a, Controller 24V	1B	EMS1_L+(P24)	Emergency stop switch 1a, Controller 24V
2A	EMS1_H-	Emergency stop switch 1a (*3)	2B	EMS1_L-	Emergency stop switch 1a (*3)
3A	EMS2_H+	Emergency stop switch 2a (*3)	3B	EMS2_L+	Emergency stop switch 2a (*3)
4A	EMS2_H-	Emergency stop switch 2a (*3)	4B	EMS2_L-	Emergency stop switch 2a (*3)
5A	MODE_H+	Mode switch (*3)	5B	MODE_L+	Mode switch (*3)
6A	MODE_H-	Mode switch (*3)	6B	MODE_L-	Mode switch (*3)
7A	SVON_MON+	Servo-ON monitor output	7B	SVON_MON_-	Servo-ON monitor output
8A	READY_H	READY contact output	8B	READY_L	READY contact output
9A	SVON_H+	Servo-ON input	9B 10B	SVON_H- G24	Servo-ON input
10A	NC				

Safety connector(*)
model name: DFMC 1,5/10-ST-3,5-LR 1790564
(Phoenix Contact JSC.)
Same as I/O connectors 1, 2, and 3



Connecting I/O connector and safety connector

- Separate the motor drive cable from the high-voltage cable and then use a sheathed cable.
- Consider the interference to use under 15m.
- It is mandatory to use a protective circuit (diode) and increased coping in case of connecting devices with self-inductive components such as relays, electromagnets at the output.

*1), *2) : The internal circuit has polarity.

+ 24V is output at the (signal name) input circuit.

If short-circuited with GND including FG, the controller may be damaged.

Route the wiring appropriately so that the output signal does not contact the DC 24V line and the output is not turned ON.

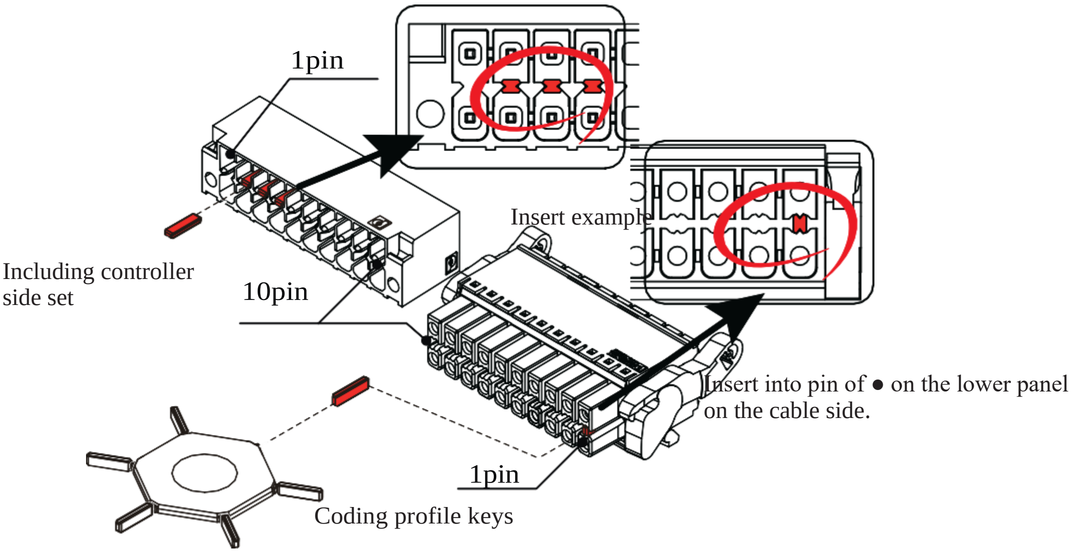
*3) : Please refer to the connector connection example and then connect.

3. Coding profile keys

Caution

Please insert each of the provided coding profile keys into the safety connector and I/O connectors 1, 2, 3 to prevent incorrect connector connections.

The controller is equipped with coding profile keys.


1pin
Insert example
Including controller side set
10pin
Insert into pin of ● on the lower panel on the cable side.
1pin
Coding profile keys

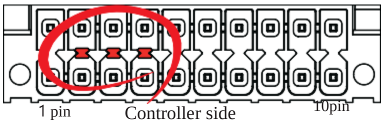
Connector		Pin No.									
		1	2	3	4	5	6	7	8	9	10
Controller side	CN3 I/O Connector 1		◎	◎	◎						
	CN4 I/O Connector 2	◎		◎	◎						
	CN5 I/O Connector 3	◎	◎		◎						
	CN6 Safety connector	◎	◎	◎							

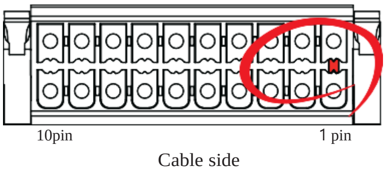
(◎Insert coding profile keys into place)

Connector		Pin No.									
		1	2	3	4	5	6	7	8	9	10
Cable side	CN3 I/O Connector 1	●									
	CN4 I/O Connector 2		●								
	CN5 I/O Connector 3			●							
	CN6 Safety connector				●						

(●Insert coding profile keys into place)

In case of CN3 I/O connector 1





7. Display the status on the controller

Display the robot’s status on the LED indicators and the 7-segment Display Panel of the controller

7-segment LED’s displays

Display the item below in the 7-segment display device.

The period blinking in the bottom right corner of the 7-segment display panel indicates that the controller system is in operation.

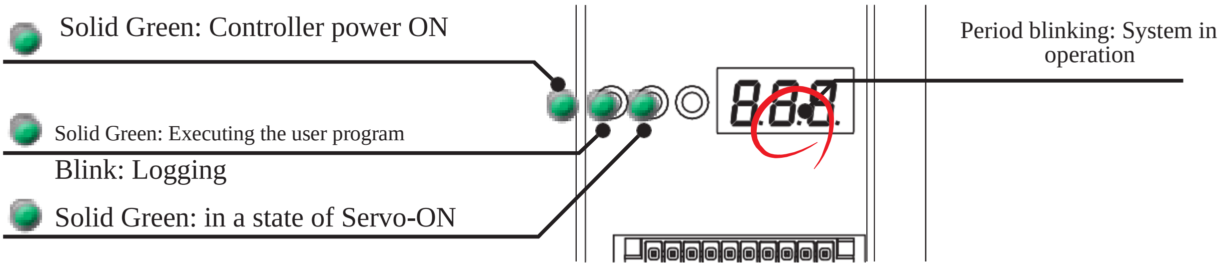
Display	Description
---	Starting the controller
ini	Initializing the controller
rdy	READY state (stand-by)
inc	ABS Lost state (*1)
tch	Teach Mode
JoG	JOG Operation Mode
run	Executing User program
PAu	Pausing User Program
PoF	Processing Power OFF
E**	System-Defined Error (*2, *4)
c**	System-Defined Error Fatal (*2, *5)
u**	User-Defined Error (*3, *4)
r**	User-Defined Error Fatal (*3, *5)

Z Document
D Software

*1) When started for the first time, the manipulator is in a state of the absolute position being lost.
*2) For more information on system-defined errors, refer to “Troubleshooting " .
*3) Any user-defined errors can be created using Python programming.
*4) For a non-fatal error, eliminate the cause, and then recover with “Error reset signal.
*5) For a fatal error, eliminate the cause, then power cycle.

LED Indicators

LED



Solid Green: Controller power ON

Solid Green: Executing the user program
Blink: Logging

Solid Green: in a state of Servo-ON

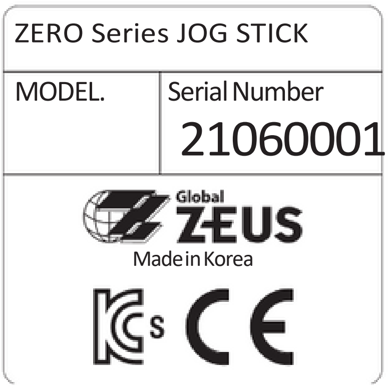
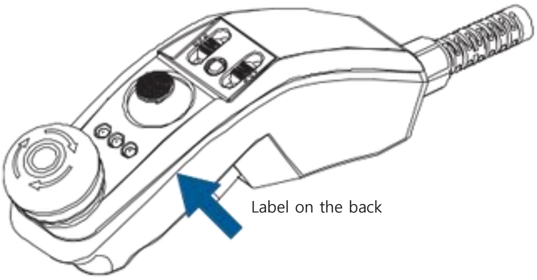
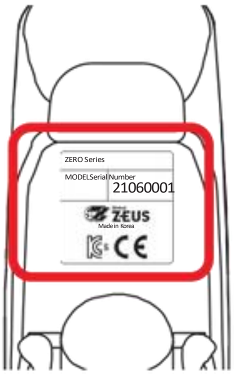
Period blinking: System in operation

NOTE

B. Hardware

4 JOG stick

1. Product label

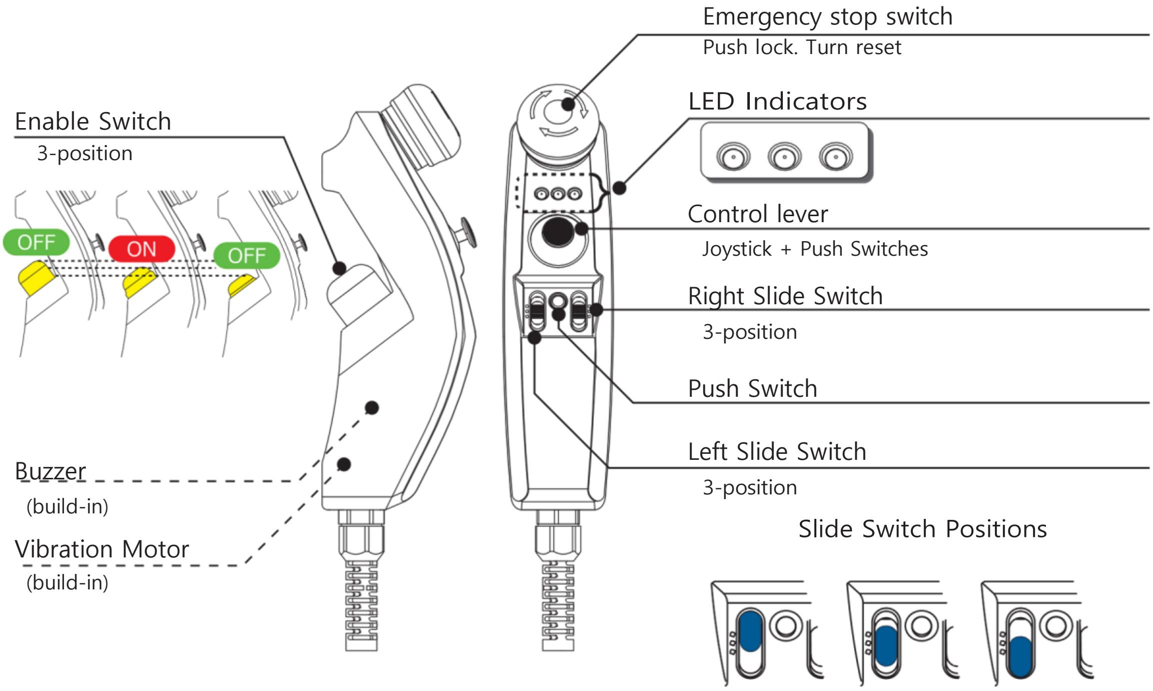
Label	Label location
Product label  ZERO Series JOG STICK MODEL. Serial Number 21060001 Global ZEUS Made in Korea	 Label on the back  On the back

For example, the serial number of the label above is ‘21060001’. Each product has a different serial number.

The serial number system is the same as that of manipulator.

2. Part name

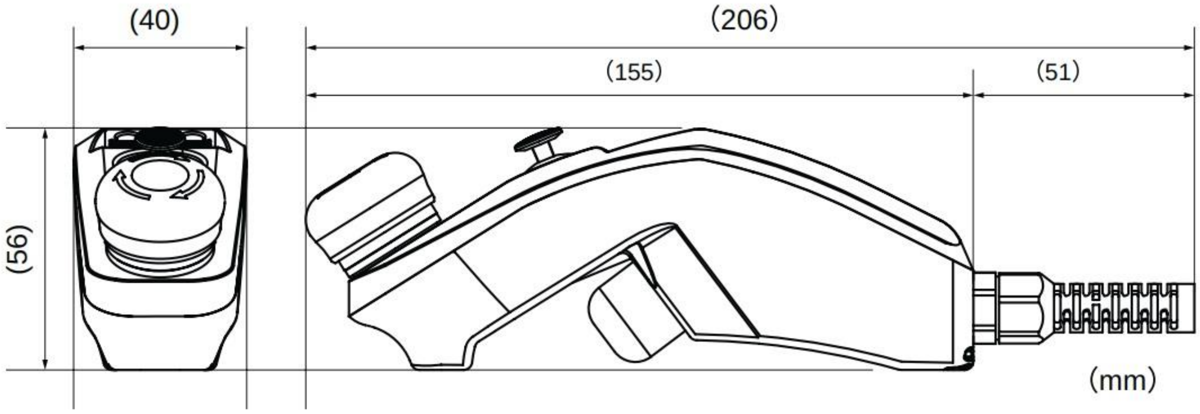
The JOG stick (optional) can be used to JOG the axes of the manipulator. JOG operation moves to the home position or is used in teaching operation.



3. Installation

❗	Please store the JOG Stick in the designated location when the it is not in use.

4. Dimension drawing



5. Specifications

Item		Specification	Remark
General Specifications	Model	ZJ1000	–
	Dimensions	H56 mm × D155 mm × W40 mm	Not including a cable
	Weight	600 g or less	–
	Frame material	ABS resin	Color: Yellow, Black
	Power supply volatage	DC24 V ± 10%	–
	Power consumption	5 W or less	–
	Cable length	5 m	–
Environmental Specifications	Operating temperature	0 ℃ – 40 ℃	–
	Operating humidity	30 % – 85 %	–
	Storage temperature	– 40 ℃ – 85 ℃	–

6. Features

<table>
<tr><th>Name</th><th>Features</th></tr>
<tr><td>Emergency Stop Switch</td><td>If pressed hard, it will return to emergency stop state.
To turn the servo back on, turn it clockwise, remove the emergency stop, and then press the Enable switch.</td></tr>
<tr><td>Enable Switch</td><td>Press and turn on the servo.
If you lift your hand or press deeper, the servo will turn off</td></tr>
<tr><td>L Slide Switch</td><td>Change the operating connection
<table>
<tr><th>Switch Positions</th><th>Top</th><th>Center</th><th>Bottom</th></tr>
<tr><td>Joint coordinate system</td><td>J1, J2</td><td>J3, J4</td><td>J5, J6</td></tr>
<tr><td>Orthogonal coordinate system</td><td>X axis, Y axis</td><td>Z axis, Rz axis</td><td>Ry axis, Rx axis</td></tr>
</table></td></tr>
<tr><td>R Slide Switch</td><td>Change “operating panel activity” on browser screen and “JOG stick operation”
<table>
<tr><th>Operating position</th><th>Top</th><th></th><th></th></tr>
<tr><td>Operating</td><td>JOG Stick</td><td colspan="2">Operating panel</td></tr>
</table></td></tr>
<tr><td>Circuit Breaker Switch</td><td>If you press and supply power to the controller at the same time, it will run in “JOG operation mode”</td></tr>
<tr><td>Control Lever</td><td>Is the control lever + fuse switch.
・Control Lever
Tilt up, down, left and right and use the supporting arm.
Select the connection to be operated with the L slide switch
・Fuse switch
(Not applicable)</td></tr>
<tr><td>LED1</td><td>Indicates the status of the robot with a green LED
Power on JOG stick (green) /power off (light off)</td></tr>
<tr><td>LED2</td><td>(Not applicable)</td></tr>
<tr><td>LED3</td><td>(Not applicable)</td></tr>
<tr><td>Buzzer</td><td>Indicates status with a buzzer sound.
・Sounds when teaching</td></tr>
<tr><td>Vibration Motor</td><td>Indicates status by vibrating alarm
・If the end of the manipulator is close to a point where it cannot be moved, it will vibrate.</td></tr>
</table>

Buzzer pattern

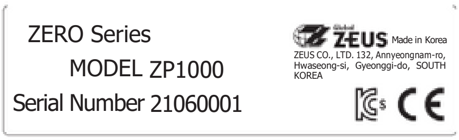
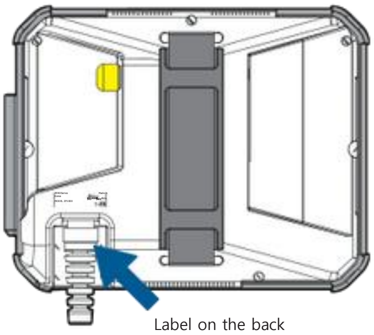
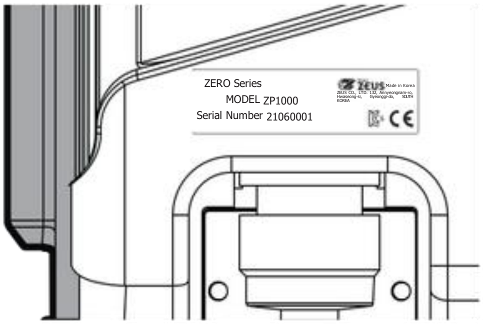
Alarm sound	Meaning
「Pi」	Sound once in the fillowing cases. ・When the controller operates. ・When starting the “Hand Homing” or “Direct Move” operating in the “Move To” of the operationg when teaching.

NOTE

B. Hardware

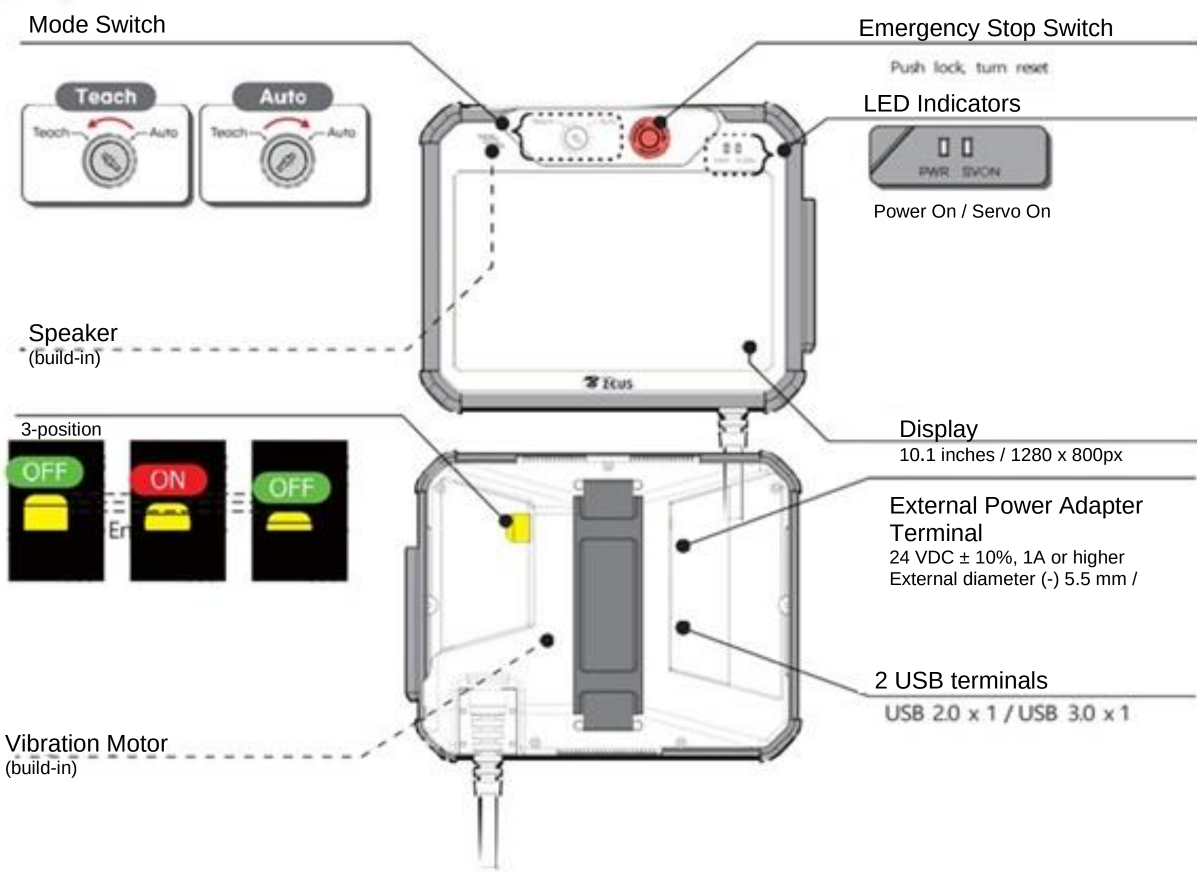
5 Teaching pendant

1.Product label

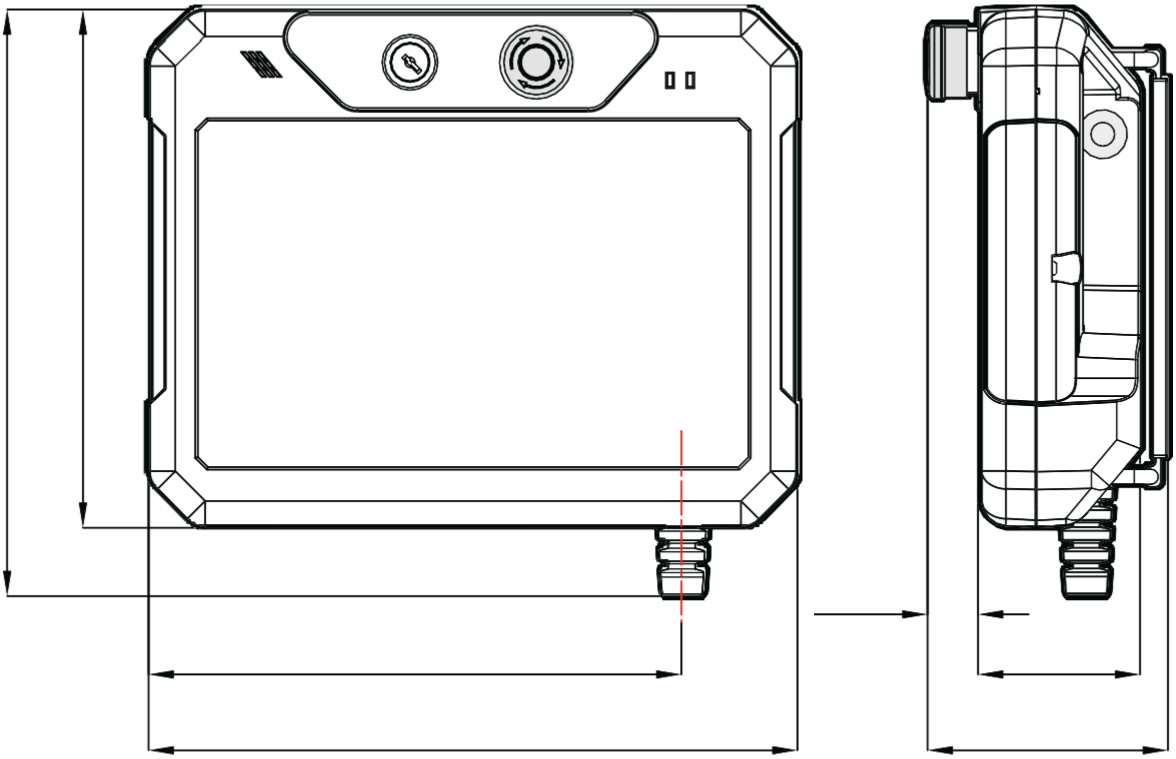
Label	Label location
Product label  ZERO Series MODEL ZP1000 Serial Number 21060001 ZEUS Made in Korea ZEUS CO., LTD. 132, Annyeongnam-ro, Hwaseong-si, Gyeonggi-do, SOUTH KOREA	 Label on the back  On the back

For example, the serial number of the label above is "21060001". Each product has a different serial number.
The serial number system is the same as that of manipulator.

2. Part name


Mode Switch
Teach
Auto
Teach
Auto
Teach
Auto
Emergency Stop Switch
Push lock, turn reset
LED Indicators
PWR SVON
Power On / Servo On
Speaker
(build-in)
3-position
OFF
ON
OFF
En
Display
10.1 inches / 1280 x 800px
External Power Adapter Terminal
24 VDC ± 10%, 1A or higher
External diameter (-) 5.5 mm /
2 USB terminals
USB 2.0 x 1 / USB 3.0 x 1
Vibration Motor
(build-in)

3. Dimension drawing



Dimension excluding rubber feet and cable)

4. Specifications

Item		Specification	Remark
General Specifications	Model	ZP1000	—
	Dimensions	H95.1 mm × D257 mm × W205 mm	Not including a cable
	Weight	1.2 kg or less	—
	Frame material	PC + ABS resin	Color: Black
	Power supply volatage	DC24 V ± 10%	—
	Power consumption	12 W or less	—
	Cable length	3 m	—
Environmental Specifications	Operating temperature	0℃ – 40 ℃	—
	Operating humidity	30 % – 85 %	—
	Storage temperature	– 40 ℃ – 85 ℃	—
	Storage humidity	10 % – 90 %	—
	Cooling	Natural cooling	—

5. Features

<table>
<tr><th>Name</th><th>Features</th></tr>
<tr><td>Emergency Stop Switch</td><td>If pressed hard, it will return to emergency stop state.
To turn the servo back on, turn it clockwise, remove the emergency stop, and then press the Enable switch.</td></tr>
<tr><td>Enable Switch</td><td>Press and turn on the servo.
If you lift your hand or press deeper, the servo will turn off</td></tr>
<tr><td>Mode Switch</td><td>Convert from operating mode to teachning mode and remote mode (automatic operating mode)<table><tr><th>Switch Positions</th><th>Left</th><th>Right</th></tr><tr><td>Mode</td><td>Teaching mode</td><td>Remote mode</td></tr></table></td></tr>
<tr><td>External Power Adapter Terminal</td><td>If the power adapter is connected, the teaching pendant’s power will turn on. It should not be used in normal circumstances.
24VDC ± 10%, 1A or higher
External diameter (-) 5.5 mm / internal diameter (+) 2.1 mm
※ Do not connect the adapter while power is being supplied.</td></tr>
<tr><td>USB Terminal</td><td>Open data such as error logs, teaching points saved in the teaching pendant. Upload the software update file.
USB 2.0 x 1/ USB 3.0 x 1</td></tr>
<tr><td>LED Indicators</td><td>Indicates the status of the robot and teaching pendant with LEDs.
・PWR: Power on teaching pendant (green) /OFF (light off)
・SVON: Power on servo of robot (green) /OFF (light off)</td></tr>
<tr><td>LCD</td><td>Displays the teaching screen of teaching pendant
Can confirm the status of robot and teaching pendant.</td></tr>
<tr><td>Speaker</td><td>Indicates status by sound
・Sounds when teaching</td></tr>
<tr><td>Vibration Motor</td><td>Indicates status by vibrating alarm
・If the end of the manipulator is close to a point where it cannot be moved, it will vibrate.</td></tr>
</table>

Speaker sound pattern

Alarm sound	Meaning
「Pi」	Sound once in the following case. ・When approaching the area near the joint angle limit, speed limit, special areas.

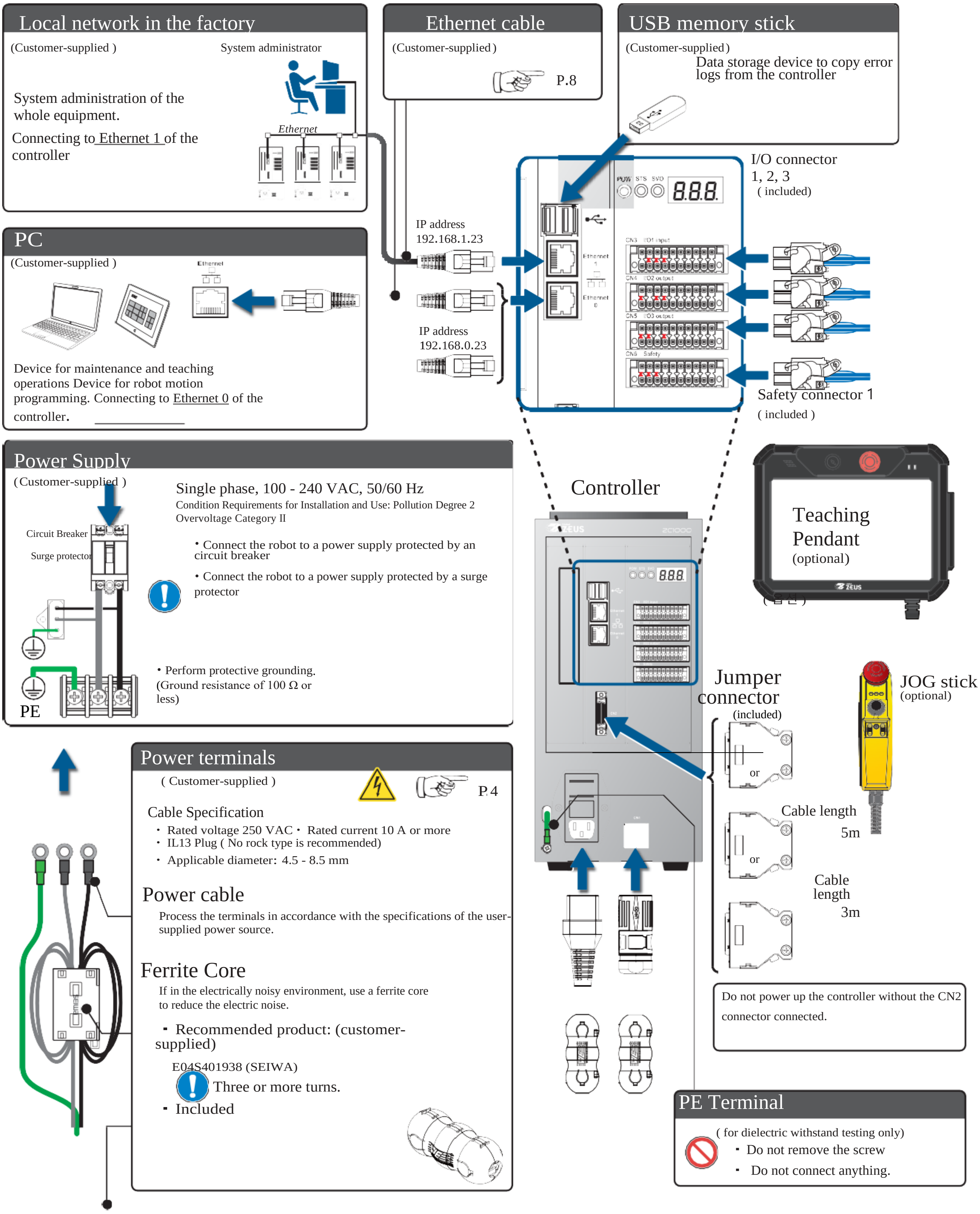
B. Hardware

6 Wiring and Power Supply

1. Wiring

1. Complete wiring diagram

Wiring is arranged exactly as shown below



Local network in the factory

(Customer-supplied)

System administrator

System administration of the whole equipment.

Connecting to Ethernet 1 of the controller

Ethernet cable

(Customer-supplied)

P.8

USB memory stick

(Customer-supplied)

Data storage device to copy error logs from the controller

I/O connector 1, 2, 3 (included)

IP address 192.168.1.23

IP address 192.168.0.23

Safety connector 1 (included)

PC

(Customer-supplied)

Device for maintenance and teaching operations Device for robot motion programming. Connecting to Ethernet 0 of the controller.

Power Supply

(Customer-supplied)

Single phase, 100 - 240 VAC, 50/60 Hz

Condition Requirements for Installation and Use: Pollution Degree 2 Overvoltage Category II

- Connect the robot to a power supply protected by an circuit breaker
- Connect the robot to a power supply protected by a surge protector
- Perform protective grounding. (Ground resistance of 100 Ω or less)

Controller

Teaching Pendant (optional)

Jumper connector (included)

JOG stick (optional)

Cable length 5m

Cable length 3m

Power terminals

(Customer-supplied)

P.4

Cable Specification

- Rated voltage 250 VAC
- Rated current 10 A or more
- IL13 Plug (No rock type is recommended)
- Applicable diameter: 4.5 - 8.5 mm

Power cable

Process the terminals in accordance with the specifications of the user-supplied power source.

Ferrite Core

If in the electrically noisy environment, use a ferrite core to reduce the electric noise.

- Recommended product: (customer-supplied)

E04S401938 (SEIWA)

Three or more turns.

- Included

Do not power up the controller without the CN2 connector connected.

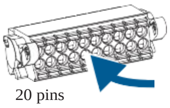
PE Terminal

(for dielectric withstand testing only)

- Do not remove the screw
- Do not connect anything.

I/O controller

(Customer-supplied)



20 pins

P.6

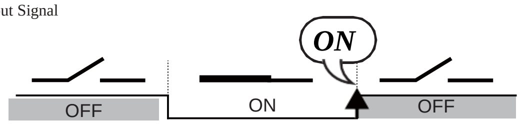
Connect to I/O connectors 1, 2, 3.
CN3 : I/O1 input
CN4 : I/O2 output
CN5 : I/O3 output

For wiring work, refer to the signal pinouts and the circuit diagram.

An input signal is detected at a rising edge of the input.

When an output becomes ON, its corresponding output port will be high.

Input Signal



I/O Cable

(Customer-supplied)

- Provide sufficient separation from high voltage lines and motor power lines.
- AWG16-24
- Take noise into account and keep the cable length within 15 meters.

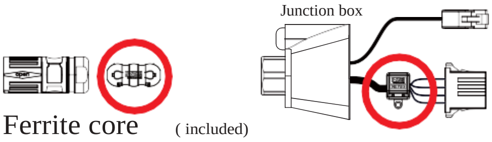
Manipulator cable

Included

Cable length 3m

P.5

The manipulator cable is integrated with the junction box.



Ferrite core

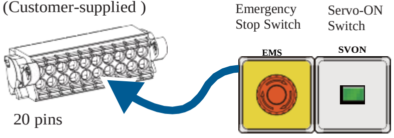
(included)

Do not remove any of the included ferrite cores.

The C.CODE is unique to each robot. Connect the con- troller with its C.CODE matching manipulator. Check the C.CODE labels on the manipulator and the controller. Only a C.CODE matching pair of them may be connected.

Safety Circuit

(Customer-supplied)



20 pins

P.7

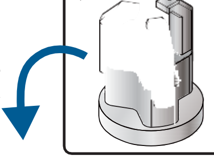
Connect to the safety connector.

Connect an emergency stop switch or a Ser- vo-ON switch.

Improper connections disable Servo-ON

End-Effector

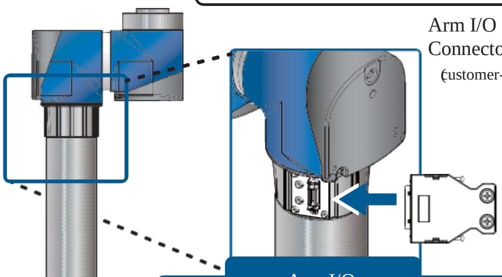
(Customer-supplied)

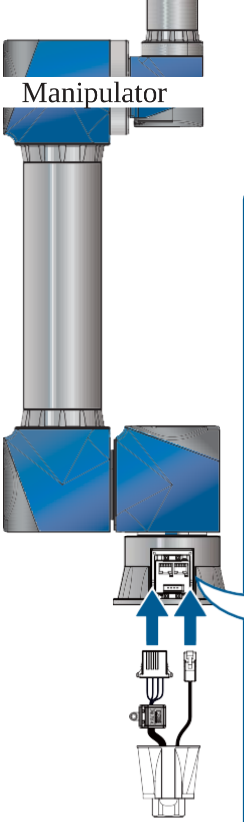


End-Effector for tooling attachment

「2 Manipulator 」

Connect to the arm I/O connector.



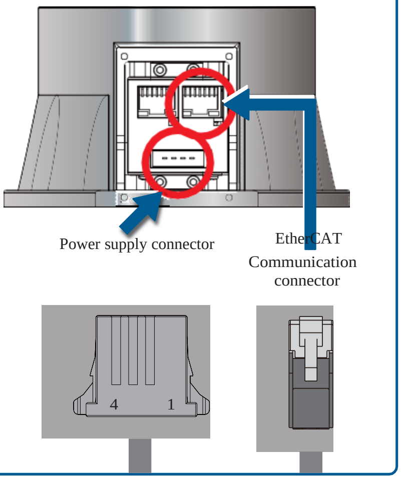


Keep the current consumption of a device connecting to the Arm I/O to be no higher than 100 mA.

Connecting the Manipulator Cable

「2 Manipulator 」

Connect the EtherCAT cable to the port “LIN” on the right



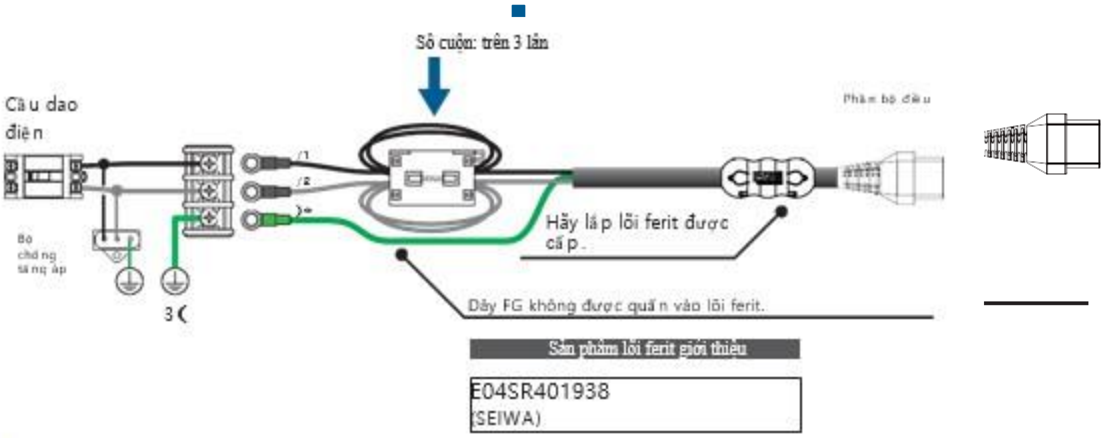
2. Power cable

전원 케이블

Please use cables with recommended specifications.
·Rated voltage: 250 VAC
·Rated current: above 10A
·IL13 socket (without locking device)
·Outer diameter: 4.5 - 8.5 mm

Method of connecting to the power supply

In case of use in an environment with a lot of electrical noise, wrap a ferrite core as shown in the figure below around the end of the cable connected to the power source, then install a round socket with an insulating cover. For the round socket installation, choose a shape or size that fits the power supply device being used.



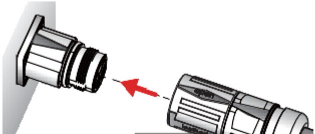
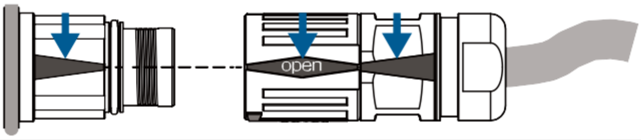
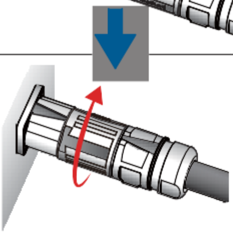
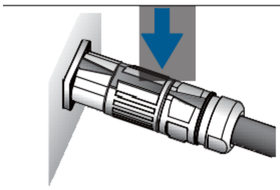
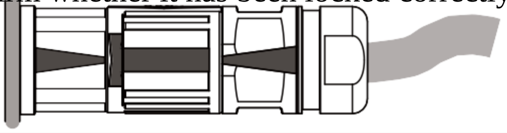
⚠ Caution

❗ You must use a power cable that matches the specifications for the voltage and current used. The electrical wiring work must be carried out by a person with professional certification. There is a risk of fire or electric shock. ⚠ ⚡

3. Manipulator cable (included)

The cable connecting the controller to the manipulator. Used to supply power and perform signal transmission.

Method to connect to the controller

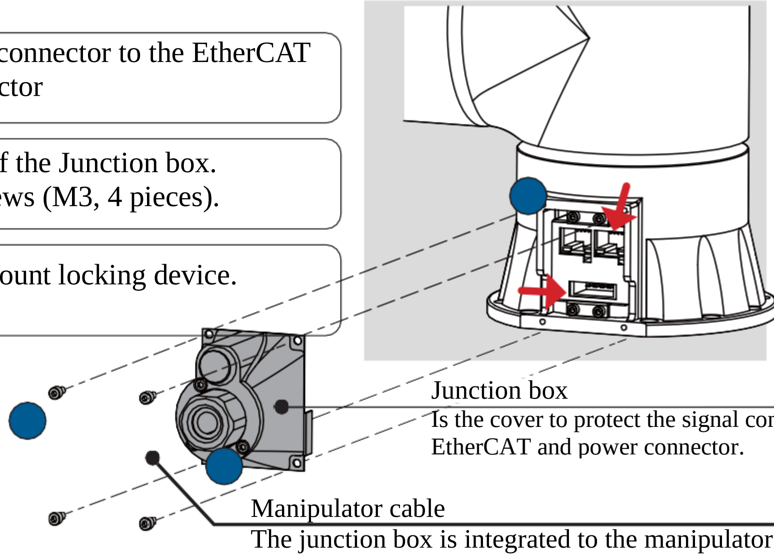
	Fully insert the connector matching the ‘open’ assembly symbol. 
	Rotate the connector housing approximately 90 degrees and lock it securely.
	Confirm whether it has been locked correctly. 

Method to connect to the manipulator

1. Connect the power connector to the EtherCAT communication connector

2. Tighten the screws of the Junction box. Screws: hexagonal screws (M3, 4 pieces).

3. Tighten the panel mount locking device.



Junction box
Is the cover to protect the signal connector EtherCAT and power connector.

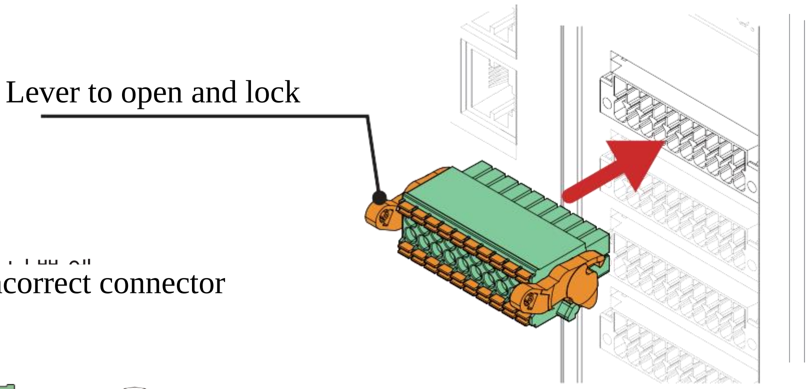
Manipulator cable
The junction box is integrated to the manipulator cable.

!	Please connect the controller and the manipulator to form the correct combination. Confirm the C.CODE label of the manipulator and the controller, then connect so that the C.CODE is consistent.	⚠ ⚠

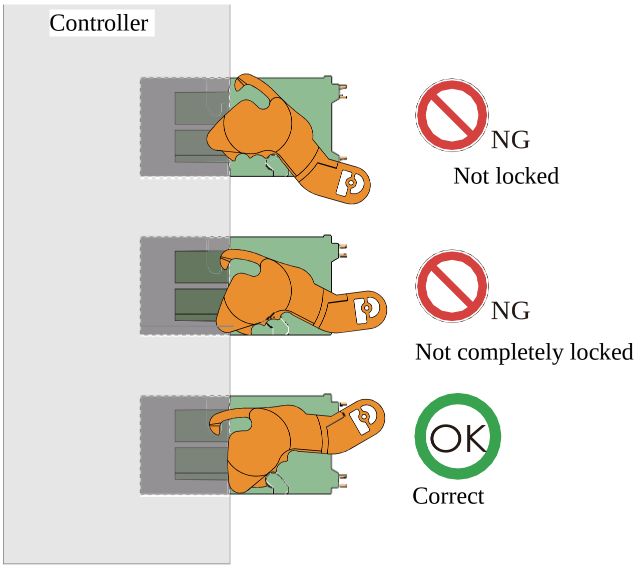
4. Connecting of I/O connector

Please firmly attach the connector to the controller until the lock engages. If the connector is correctly connected, the operation lock lever on both the left and right sides should automatically lock.

Method to connect to the controller

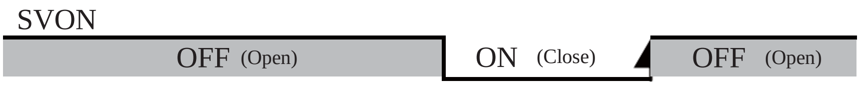


Examples of correct or incorrect connector



!	・Keep the wiring away from high-voltage lines or motor drive cables, and use a sheathed cable. ・Use AWG16-24. ・Consider the interference and use within 15m.	⚠

Regarding turning on the servo
Turn on the side servo.
Use a mechanical inspection operation switch (contact a).

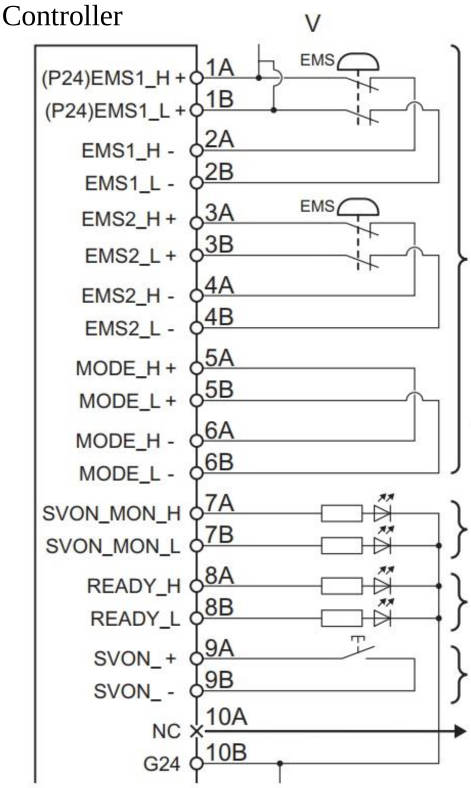


5. Wiring of safety connector

Caution

Let’s use the mechanical contact switch type.

Controller



The emergency stop switch must be installed in one position. Short-circuit the unused EMS input switch

If a dual electrical circuit like is not connected even on one side, it will result in an error.

Servo ON monitor (optional)

READY monitor (optional)

ON servo input (required)

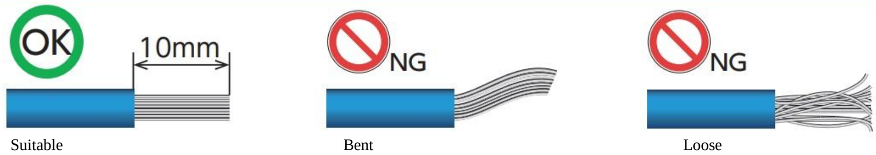
Do not connect any ends.

Caution

Refer to the following and then process the cable sheath.
For handling the sheath, use a cable stripper. If the electrical cable is not properly processed, it may result in contact errors or unintended operations.

포인트!

The method of stripping the cable sheath of the I/O connector and the safety



Suitable Bent Loose

6. Ethernet cable

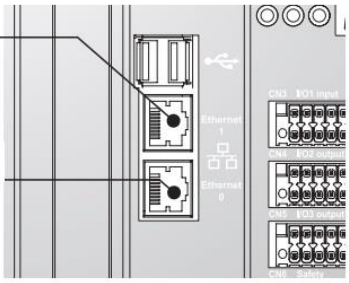
Use Ethernet 0 (below) when connecting to PC or Tablet for maintenance.
Use Ethernet 1 (above) when connecting to PC for monitoring the factory network.
The cable can be used for both straight and curved lines. Use a cable on Ethernet CAT5.

Ethernet 1 : Connecting to PC for monitoring the factory network

IP address: 192.168.1.23
Subnet mask: 255.255.255.0

Ethernet 0 : Connecting to PC for teaching or maintenance (dedicated)

IP address: 192.168.0.23
Subnet mask: 255.255.255.0



포인트 !!

Wiring connection

The cable must not be used in in contact with oil or water.

Do not allow the cable to be bent or heavily compressed
Ensure the bending radius of the cable is the largest possible.

In case of wiring that allows the cable moves, a flexible cable must be used. Place the cable in the cable tray to minimize bending or heavy compression.

Keep as far away as possible to prevent incorrect operation due to interference when arranging signal cables and high voltage cables, high-frequency current cables in the cable tray.

7. JOG stick and jumper connector

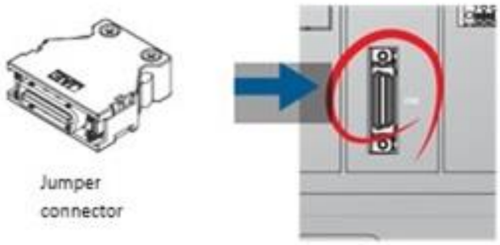
Switch the robot’s operating mode by connecting to CN 2 of the controller.

If the jumper connector is connected …> it is remote mode (automatic operation mode)
If the JOG stick is connected…> it is teaching mode.

Remote mode (automatic operation mode)

Is the robot’s automatic operation mode

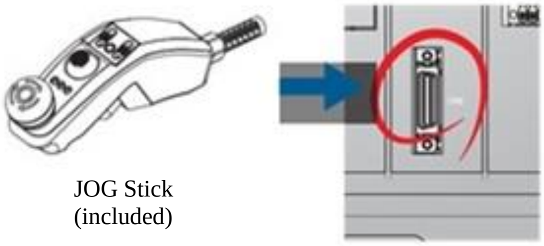
Please connect the jumper connector



Teaching mode

Is teaching mode or JOG operation

Please connect the JOG stick

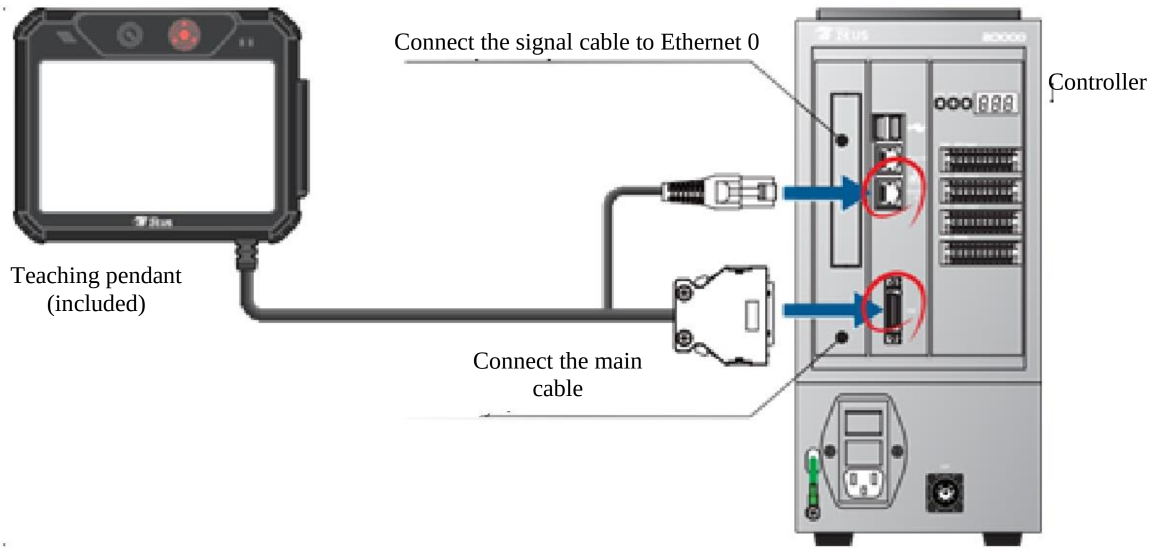


JOG Stick
(included)

❗	Except when using JOG stick, the controller’s CN2 must always be connected to the jumper connector (included).	⚠

8 Teaching Pendant cable

The teaching pendant cable consists of two cables. Connect the signal cable to Ethernet 0 (below) of the controller. Connect the main cable to CN2 of the controller.

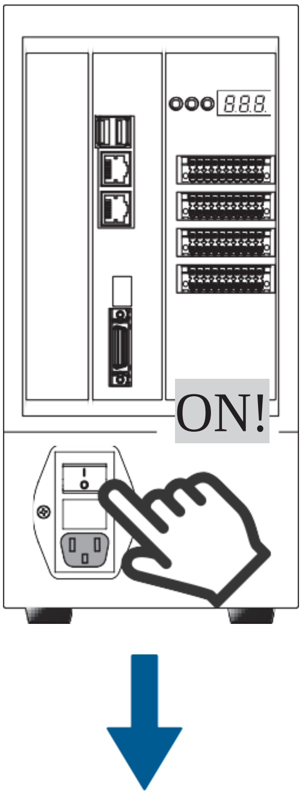

Connect the signal cable to Ethernet 0
Controller
Teaching pendant (included)
Connect the main cable

❗	Do not connect or disconnect the cable when power is supplied to the controller. Secure the teaching pendant accurately to the controller's Ethernet and CN2 ports. Additionally, be careful not to pull or bend the cable excessively. If teaching is completed, remove the teaching pendant from the controller and protect it from damage or malfunction due to dropping or breaking. Connect the jumper connector to CN2 of the controller unless teaching.	⚠

2. Power supply

⚠ Caution		
❗	Please confirm that all wiring connections have been completed before supplying power to the controller.	⚠ ⚠
🚫	Do not connect or disconnect any connectors while power is on.	⚠ ⚠ ⚠

1. Turning on the Power



If power is supplied, the status is displayed on the controller’s 7-segment display panel.

7-segment display

- - -	Start the controller

↓ (About 10s)

ini	Initialize the controller

↓ (About 10s)

Display either if initialization is complete

inc	Loss of ABS homing Display when ABS homing is lost when starting for the first time. Please recover the homing (*).
rdy	Ready (= standby state) ABS homing has been recovered. Standby state
c88 E88	Error Please confirm the error code and fix the error.

2. Homing, teaching

☞ Please refer to the user manual C .Teaching

*) Manipulator’s information is lost when starting for the first time. Export in the state of losing the ABS homing’s information.

NOTE

C

Teaching

1.JOG stick operation
2.PC access
3.ABS homing recovery
4.Teaching
5.Coordinate system and posture

NOTE

Teaching

1 JOG Stick Operation

1. JOG Operation Mode

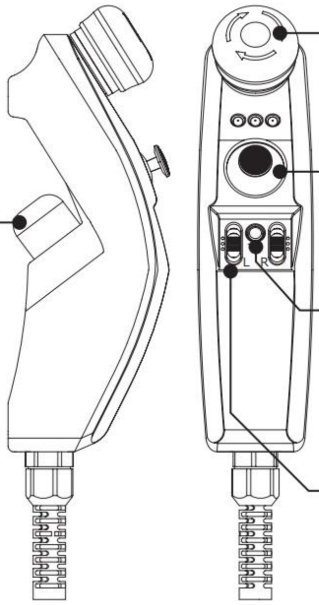
1. What is JOG operation mode

It is the mode used to operate the JOG Stick and control the manipulator. It is not used to access the PC but can control the robot.

- The operator can safely control the robot at a distance from the robot.
- The robot can be controlled even when ABS is lost.
- The robot operates using a joint coordinate system.
- The manipulator can be easily changed using the homing posture.
- It can control the robot by axes, but cannot control multiple axes.

Item	Specifications
Control speed	5% of maximum specification speed J1 ～ J4 : 8.9 deg/s J5, J6 : 13.4 deg/s
Control range	0.25deg increments (If the controller lever is pressed, change in 5deg increments)

Name and features of parts used in the JOG operation



Emergency stop switch

If the switch is pressed, it returns to emergency stop state. To turn the servo back on, turn the switch, remove the emergency stop state and press the Enable switch.

Controller lever

It is the control + push switch.
Move up, down, left and right and move the manipulator.

Enable switch

If the switch is pressed, the servo is turned on.
If you release your finger or press harder, the servo turns off.

Push switch

Move using JOG operation mode.
If the power supply is connected in the pressing state, move using JOG operation mode. Press continuously until the controller's 7-segment display appears.

L slide switch

Change the operation joint.
Top: J1, J2
Middle: J3, J4
Bottom: J5, J6

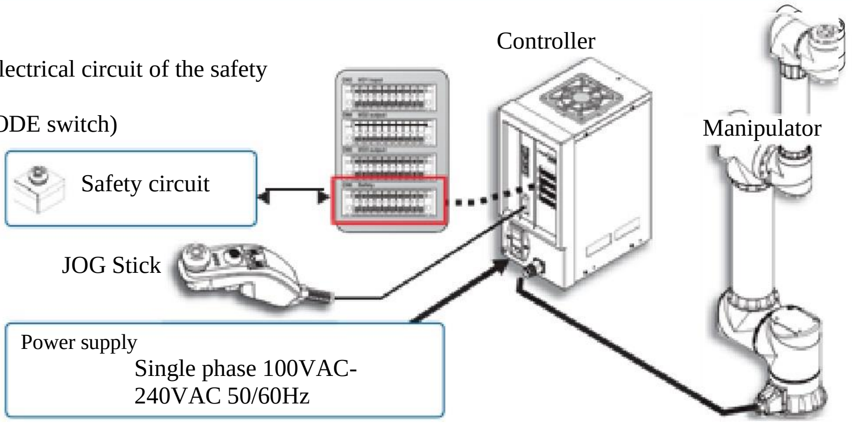
2. Move and finish

Move

Step 1. Access

Wiring as shown below

It is mandatory to connect the electrical circuit of the safety connector (*)
(emergency stop switch and MODE switch)

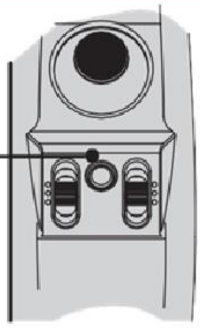


*) Refer to safety circuit 5 "5. Wiring and Power Supply" 5 Wiring of safety connector".

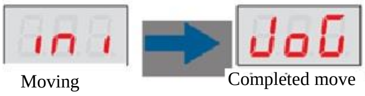
Step 2. Move

Press the push switch of the JOG stick and turn on the controller.

Press continuously until the controller's 7-segment display appears



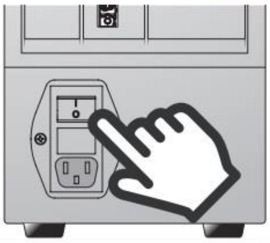
7-segment display



Moving Completed move

종료

Turn off the power of the controller.



In case an error occurs

JOG operation error

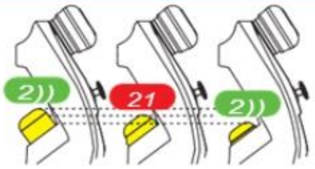
Turn off and turn on the controller again.

3. Operation

Step 1. Turn on servo

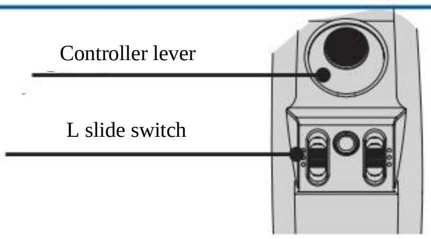
Press the Enable switch

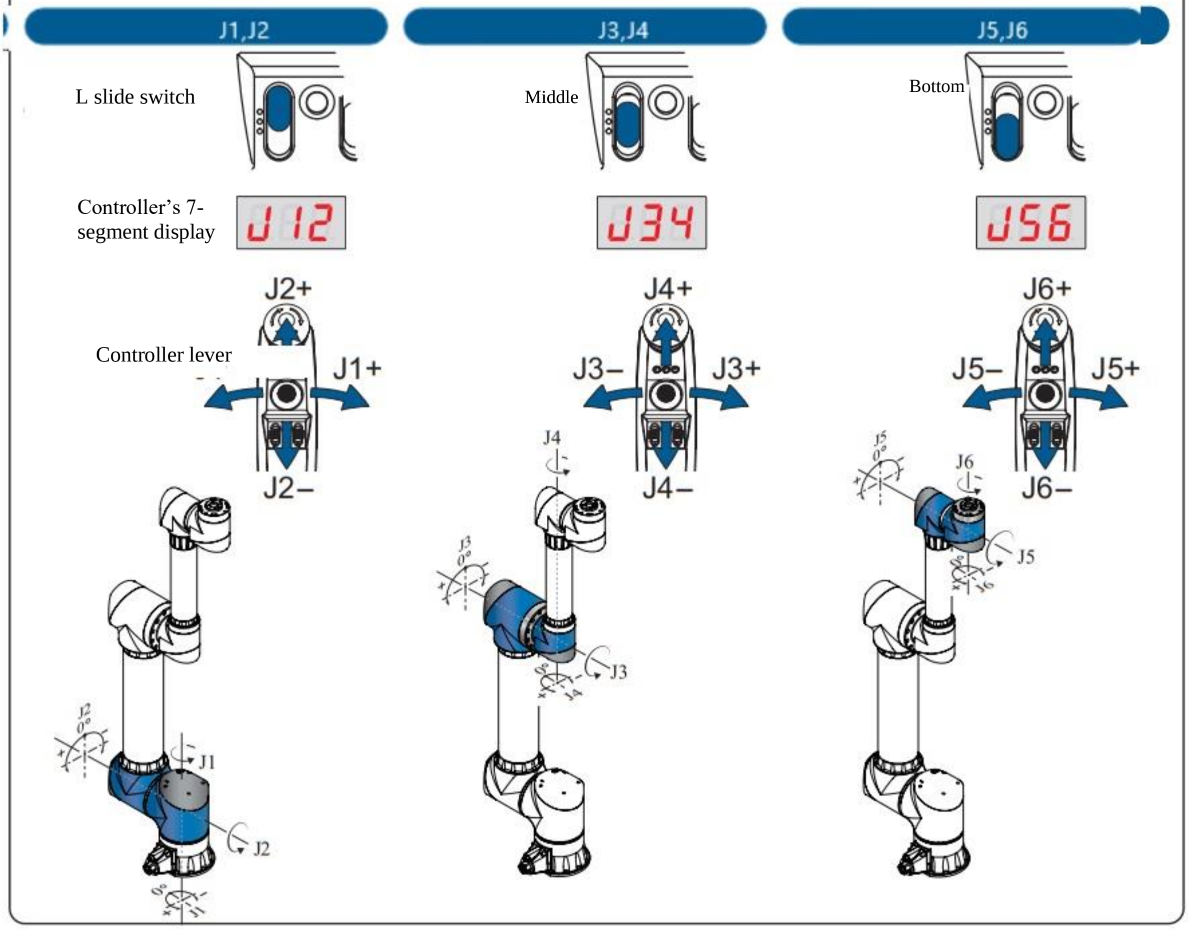
If you release or press harder the Enable switch, the servo is turned off.



Step 2. Operation

After changing the upper/middle/lower L slide switch, select the connection you want to move.
Tilt the controller lever and then move the robot.



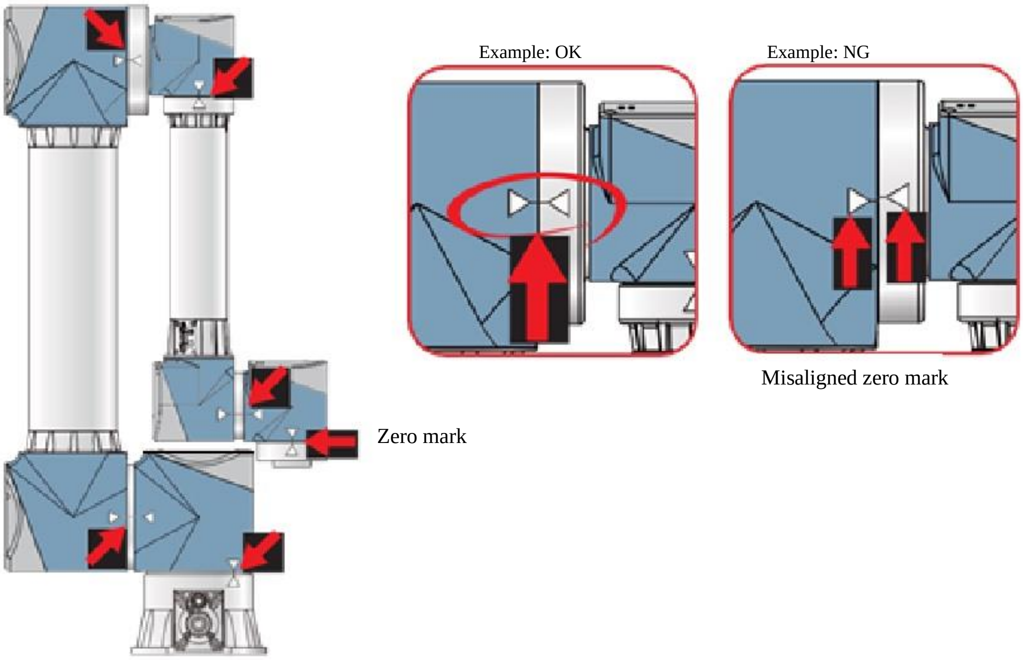


Confirm the working connection on the controller's 7-segment display panel and then operate the controller lever.

Step 3. Homing posture alignment

Control the JOG stick, align all joints exactly at the zero mark to create the home position posture. Home position posture (*)

(Example: ZRA4937)



*) The home position varies depending on each model. Please confirm the exact location of the zero mark. The zero mark is emphasized in this document. Please proceed according to the actual position standards.

Please align the zero mark within 1mm.

In case the JOG stick is released from the controller while the JOG stick is in operation

Stop the robot's operation.

Reconnect the JOG stick to the controller to run again
(Do not restart the controller)

NOTE

Teaching

2 PC Access

1. Connect the PC to the controller

1. Software preparation

Prepare the two following software

FFFTP.exe

「FFFTP」: FTP client software

Use FTP (File Transfer Protocol) to transmit files between the PC and the controller.

URL https://osdn.net/projects/ffftp/

Select 32-bit or 64-bit versions depending on the computer used. When installing, use the original settings.

(FFFTP is a product of FFFTP Project)

ttermpro.exe

「FFFTP」: FTP client software

Use FTP (File Transfer Protocol) to transmit files between the PC and the controller.

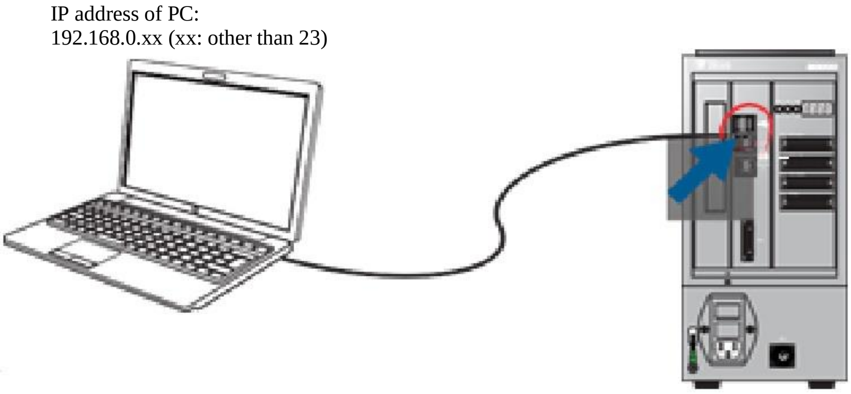
URL https://osdn.net/projects/ffftp/

Select 32-bit or 64-bit versions depending on the computer used. When installing, use the original settings.

(FFFTP is a product of FFFTP Project)

2. IP address setting

Set up the network of the access PC with the controller.



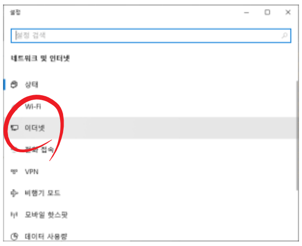
IP address of PC:
192.168.0.xx (xx: other than 23)

Controller

Connect to below (Ethernet 0)
IP address of Controller:
192.168.0.23

Step 1

Click on “Ethernet” of the “Network and Internet” on the control panel.



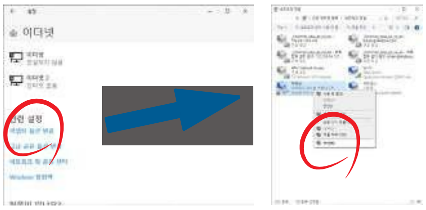
설정

설정 검색

네트워크 및 인터넷

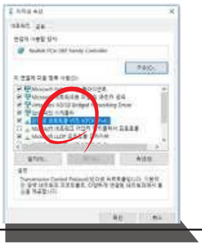
Step 2

Click on “Change adapter options” of “Ethernet”, right click on the ethernet icon and then click on Properties.



Step 3

Click on Properties of internet protocol 4 version (TCP/Ipv4).

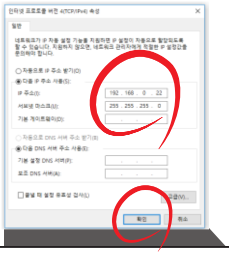


Step 4

Set up IP address and subnet mask as follows:

IP address (I)	192.168. 0. XX
Subnet mask (U)	255.255.255. 0

(XX: set with a number other than 23)



인터넷 프로토콜 버전 4(TCP/IPv4) 속성

일반

네트워크가 IP 자동 설정 기능을 지원하면 IP 설정이 자동으로 할당되도록 할 수 있습니다. 지원하지 않으면, 네트워크 관리자에게 적절한 IP 설정값을 문의해야 합니다.

○ 자동으로 IP 주소 받기(O)
● 다음 IP 주소 사용(S):
IP 주소(I): 192 . 168 . 0 . 22
서브넷 마스크(U): 255 . 255 . 255 . 0
기본 게이트웨이(D): . . .

○ 자동으로 DNS 서버 주소 받기(B)
● 다음 DNS 서버 주소 사용(E):
기본 설정 DNS 서버(P): . . .
보조 DNS 서버(A): . . .

□ 끝낼 때 설정 유효성 검사(L)

(Above example is for Windows 10)

1 Connect the PC to the controller

3. Access setting

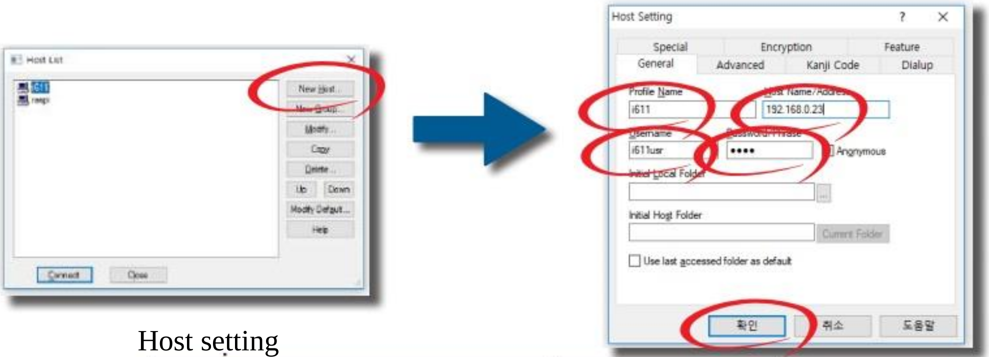
Set up the two following software to access the controller.

FFFTP.exe

「FFFTP」Run FFFTP

New Host...

Click on new host and set up host.



Host Setting

Special | Encryption | Feature | General | Advanced | Kanji Code | Dialup

Profile Name: i611

Host Name/Address: 192.168.0.23

Username: i611usr

Password/Phrase: ••••

□ Anonymous

Initial Local Folder:

Initial Host Folder:

Current Folder

□ Use last accessed folder as default

확인 | 취소 | 도움말

Host setting

Profile Name	i611
Host Name/Address	192.168.0.23
Username	i611usr
Password/Phrase	i611

Connect

Click to Connect

ttermpro.exe

「Tera Term」 Run 「Tera Term」

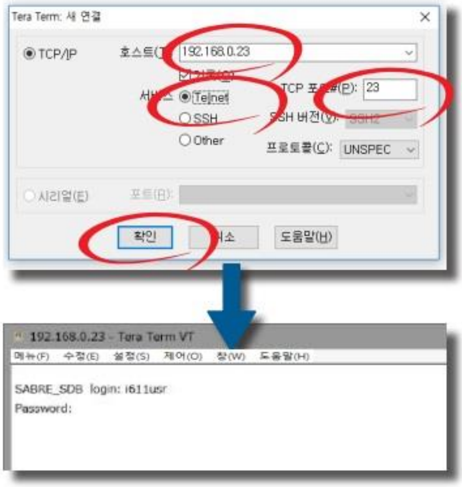
Host setting

Host (T)	192.168.0.23
Service	Telnet
TCP Port # (P)	23

(The controller's power must be turned on)

Controller confirmation

login	i611usr
Password	i611



Tera Term: 새 연결

● TCP/IP 호스트(T): 192.168.0.23

☑ 히스토리(O)

서비스: ● Telnet ○ SSH ○ Other

TCP 포트#(P): 23

SSH 버전(V): SSH2

프로토콜(C): UNSPEC

○ 시리얼(E) 포트(R):

확인 | 취소 | 도움말(H)

192.168.0.23 - Tera Term VT

메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

```
SABRE_SDB login: i611usr
Password:
```

(Above example is for Windows 10)

C. Teaching

3 ABS Homing

1. Notes

❗	It is essential to return to the ABS home position when connecting the power supply to the Robot for the first time. Ensure all joints are aligned to the zero mark (illustrated mark) using the JOG operation. Align the zero mark within a range of 1 mm. The servo must be turned on during the process of ABS homing. Connect the servo ON switch to the JOG stick first.	⚠ ⚠

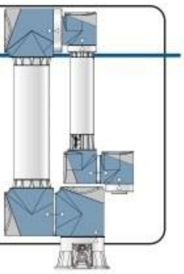
Homing should only be done once when unpacking. It is not necessary to do it frequently.

2. ORDER

Step 1 Adjust the home position of the manipulator

Operate the JOG stick to precisely align all axes to zero mark to create the homing posture.

1 Operate the JOG stick



Step 2 Activate the Robot software enc_reset.py

Connect to the controller by Telnet

Activate the software to ABS homing

In case all joints are performed consistently

```
$enc_reset.py
```

In case of indicating joint

Add joint number into blank space

```
$enc_reset.py 56
```

← In case of joints 5, 6

Confirmation message is displayed

```
Target Joint(s) : All
OK? [Y/n] Y
```

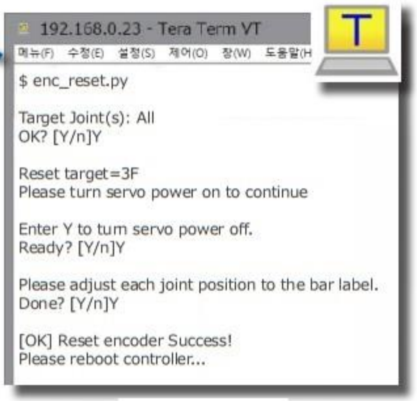
← In case all joints are performed consistently

← Enter “Y” (in uppercase)

Message displays

```
Reset target = 3F
Please turn servo power on to continue
```

← In case all joints are performed consistently



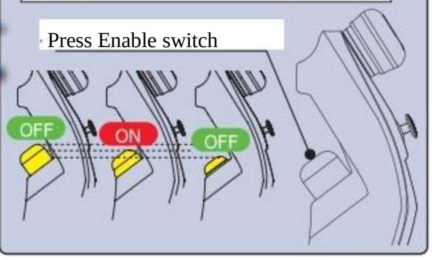
192.168.0.23 - Tera Term VT
메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H

```
$ enc_reset.py

Target Joint(s): All
OK? [Y/n]Y

Reset target=3F
Please turn servo power on to continue

Enter Y to turn servo power off.
Ready? [Y/n]Y

Please adjust each joint position to the bar label.
Done? [Y/n]Y

[OK] Reset encoder Success!
Please reboot controller...
```

Example: ABS homing consistently

Turn servo power ON

Press Enable switch



When the servo is turned ON, the confirmation message is displayed.

```
Enter Y to turn servo power off
Ready? [Y/n] Y
```

← Enter “Y” (in uppercase)

Message displays

```
Please adjust each joint position to the bar label.
Done? [Y/n] Y
```

← Enter “Y” (in uppercase)

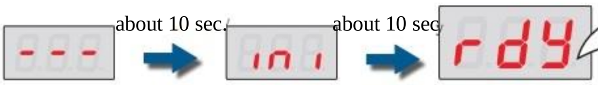
Message displays

```
[OK] Reset encoder success!
Please reboot controller...
```

Reboot the controller

Step 3 Reboot the controller

When the ABS homing is completed, the rdy display of the controller appears.



Complete !

3. Confirmation

After ABS homing is completed, please ensure to check if the ABS homing is performed correctly according to the method below?

Step 1 Run the confirmation program confirm_home.py

Connect to the controller by Telnet

Run the confirmation program

```
$confirm_home.py
```

While pressing the Enable switch on the JOG stick, perform confirm_home.py

The confirmation message will be displayed

```
Are you ready to move? (y/n) y
```

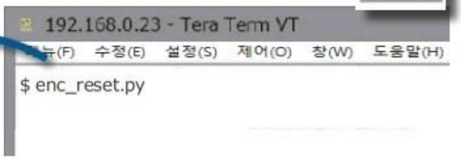
Enter "y" (in lowercase)

When you enter "y" and press Enter, all joints will return to the zero mark of each joint.
The manipulator's speed of confirm_home is 1%.
J1~J4 : 1.8deg/s
J5, J6 : 2.7deg/s

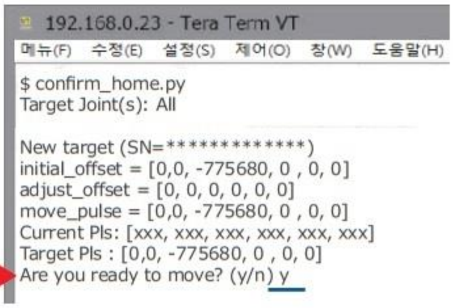
The confirmation message will be displayed when the operation is complete.

```
Position OK? (y/n) y
```

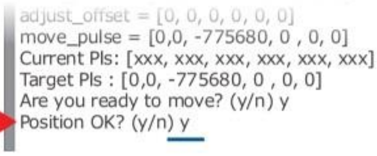
Enter "y" (in lowercase)


192.168.0.23 - Tera Term VT
메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

```
$ enc_reset.py
```


192.168.0.23 - Tera Term VT
메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

```
$ confirm_home.py
Target Joint(s): All

New target (SN=*************)
initial_offset = [0,0, -775680, 0 , 0, 0]
adjust_offset = [0, 0, 0, 0, 0, 0]
move_pulse = [0,0, -775680, 0 , 0, 0]
Current Pls: [xxx, xxx, xxx, xxx, xxx, xxx]
Target Pls : [0,0, -775680, 0 , 0, 0]
Are you ready to move? (y/n) y
```



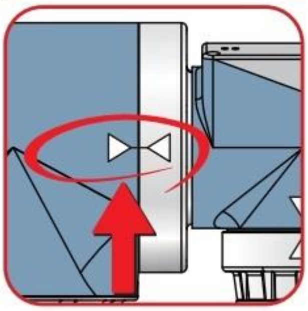
```
adjust_offset = [0, 0, 0, 0, 0, 0]
move_pulse = [0,0, -775680, 0 , 0, 0]
Current Pls: [xxx, xxx, xxx, xxx, xxx, xxx]
Target Pls : [0,0, -775680, 0 , 0, 0]
Are you ready to move? (y/n) y
Position OK? (y/n) y
```

Step 2 Confirm after performing confirm_home.py

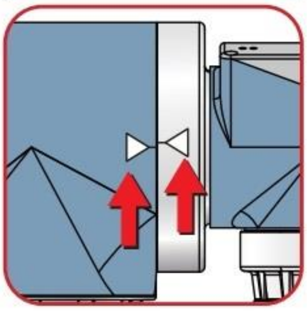
Please check if the position of zero mark of all joints is correct?

Example after performing confirm_home.py

Successful



Unsuccessful



Is the zero mark misaligned?

C Teaching

4 Teaching

1. Simple operations

⚠ ATTENTION

❗	When operating the power-assisted lifting arm for the first time, Always choose the common coordinate system.. After confirming that there are no obstacles within the operating range of the power-assisted lifting arm, proceed with the Jog operation. Keep your eyes on the power-assisted lifting arm during the Jog operation. In case of emergency, stop the power-assisted lifting arm by pressing the emergency stop switch on the JOG stick	⚠ ⚠
🚫	Do not turn off the control panel while the power-assisted lifting arm is operating.	⚠

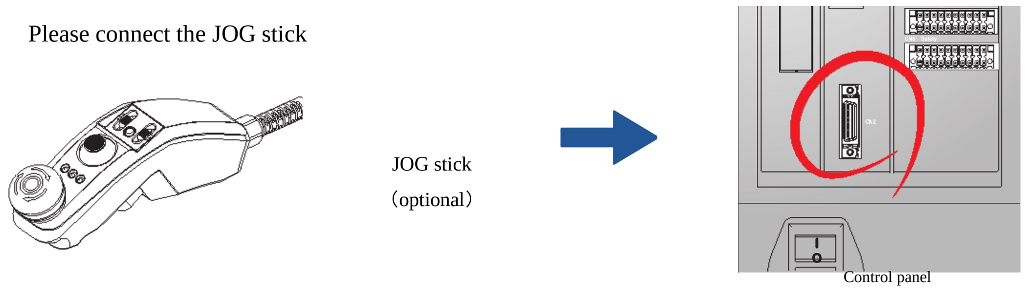
1.Simple operations

The operating mode of the robot is switched by connecting the CN2 connector of the controller

Teaching mode (JOG stick))

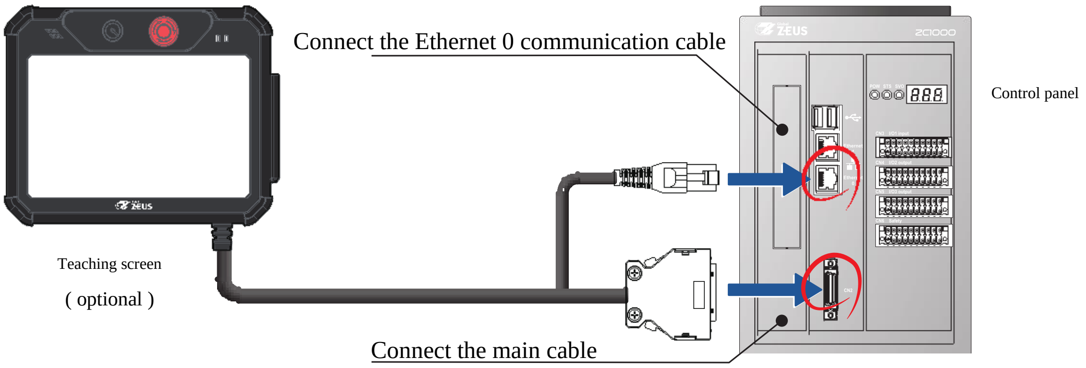
This is the mode to perform Jog operations or teaching

Please connect the JOG stick



Teaching mode (Teaching screen)

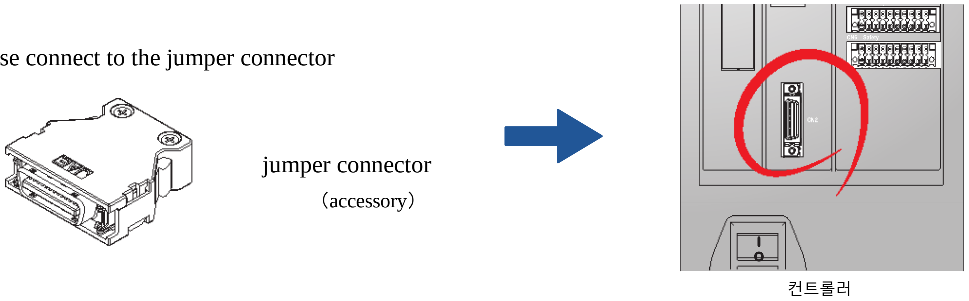
Connect to the teaching screen.



Remote mode (autonomous operating mode)

This is the mode to perform automatic operations of the robot.

Please connect to the jumper connector



Always connect to the jumper connector, unless when using the JOG stick or the teaching screen

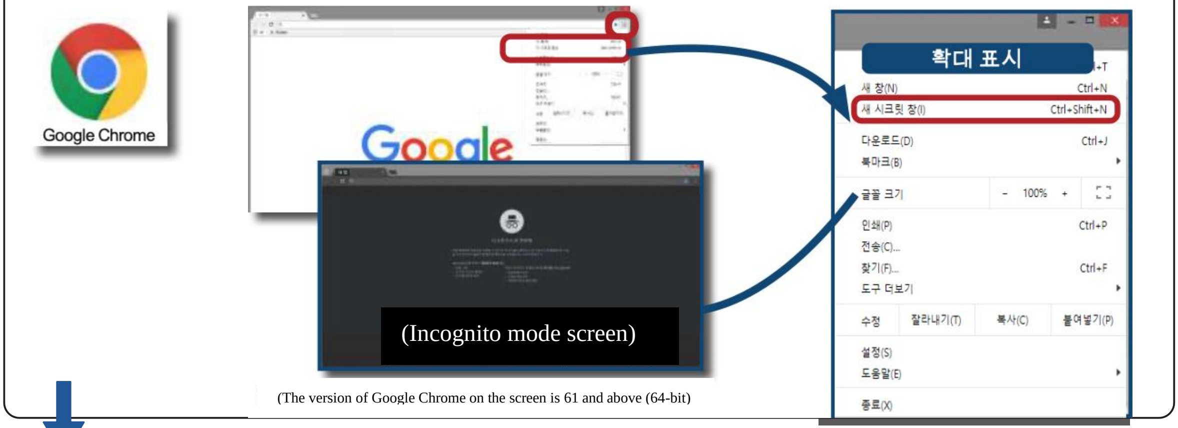
2. Preparations

Connect to the teaching computer

Start a web browser (Google Chrome) in Incognito mode.

Enter the connect address and run the teaching screen.

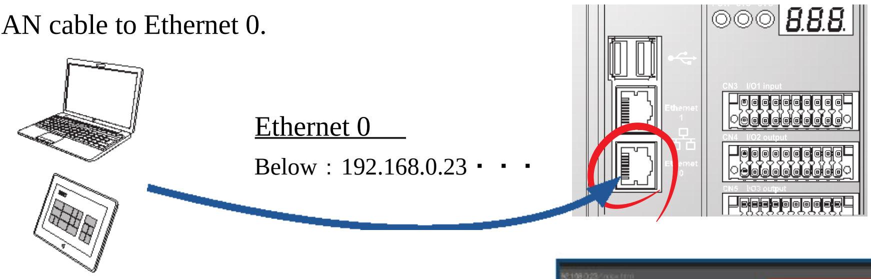
Step 1 Start a web browser in Incognito mode. .



(The version of Google Chrome on the screen is 61 and above (64-bit)

Step 2 Connect the control panel and PC..

Connect LAN cable to Ethernet 0.



Ethernet 0

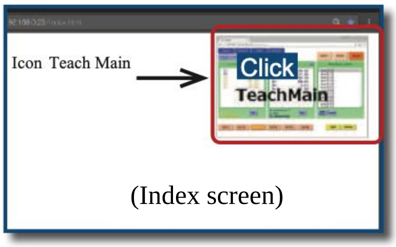
Below： 192.168.0.23・・・

Enter the follow address to the browser

Connect address to the control panel

http://192.168.0.23

When connect to the control panel, the index screen appear



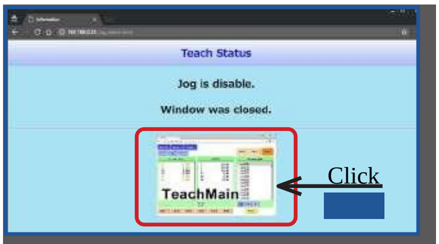
(Index screen)

포인트

When the JOG stick is unable to connect

Teaching is impossible.

Return to the index screen .
After connect to the JOG stick, please press the Teach Main symbol.

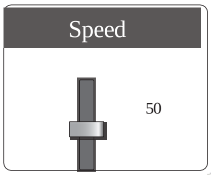

Teach Status

Jog is disable.

Window was closed.

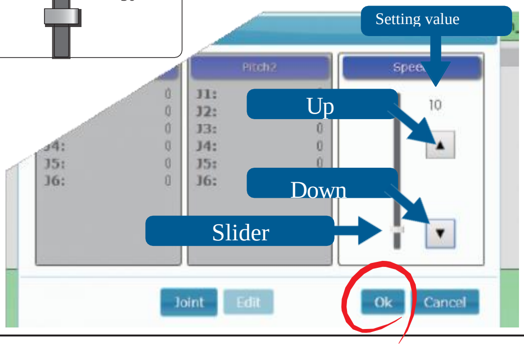
Pitch and movement speed settings

Pitch and movement speed settings of the power-assisted lifting arm when operating 1 time on the JOG stick



Continuous motion setting (constant speed)

Continuous operation when the JOG stick control is tilted.



Speed Press the Sped button to select..

▲▼ Adjust Jog operation speed(%) using the slider or the up down buttons

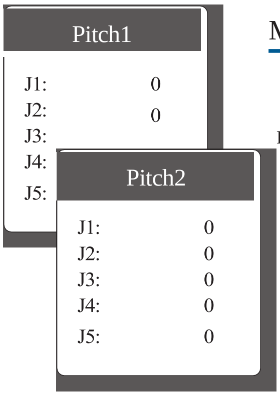
Speed The maximum speed (100%) is as follows .

World coordinate system XY : 80 mm/s

Common coordinate system Joint : J1~J4 5.3 deg/s

J5 ,J6 8.0 deg/s

OK Click OK to return to the Teach Main screen



Pitch1

J1: 0
J2: 0
J3:
J4:
J5:

Pitch2

J1: 0
J2: 0
J3: 0
J4: 0
J5: 0

Movement level of Pitch setting

Tilt the JOG stick control one time to create a determined amount of movement. .

Pitch1 Pitch2 Press Pitch1 or Pitch2 button to select

Pitch movement amount can be set separately in the common coordinate system and the world coordinate system

/ XY Coordinate system can be changed using Joint button or XY

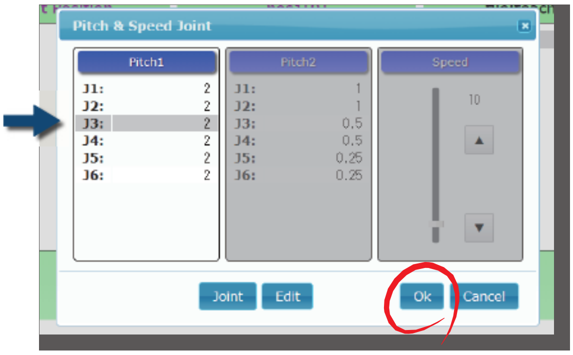
Two types of Pitch movement levels can be set for each joint Pitch1 and Pitch2

・Choose a joint you want to set and input it using PC keyboard or the 10-button panel. Setup the range

・Common coordinate system：0 ～ 2 [deg]

・World coordinate system：0 ～ 10 [mm]

OK Click OK to return to the Teach Main screen



Pitch1		Pitch2	
J1:	2	J1:	1
J2:	2	J2:	1
J3:	2	J3:	0.5
J4:	2	J4:	0.5
J5:	2	J5:	0.25
J6:	2	J6:	0.25

보충

Pitch 1 Pitch2

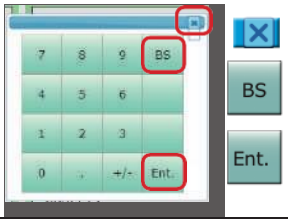
Speed Pitch movement speed according to option Pitch 1 or Pitch 2

Input using PC keyboard

Choose the joint you want to set and input a value directly by pressing Enter on the PC.

Input using 10-button panel

Choose the joint you want to set, press the Edit button and input a value using 10 buttons.



Close button

Backspace button

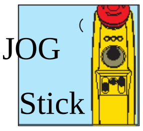
Enter

3. Jog operating method of the power-assisted lifting arm .

When the teaching screen is displayed and completes settings on operation amount and speed, you can perform Jog operations of the power-assisted lifting arm. Operations of the power-assisted lifting arm can be operated by using “JOG stick” or “control panel”.

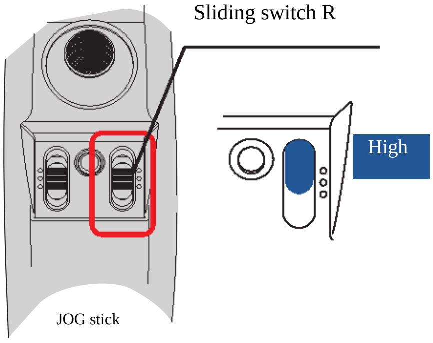
Method 1 Using JOG stick

☞ P.7

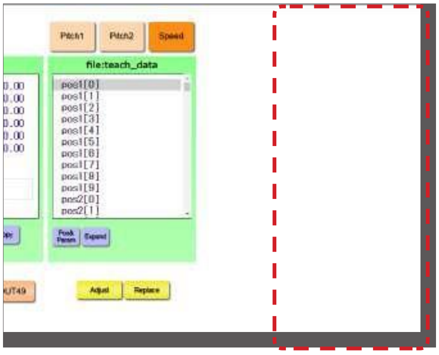


Pull the R stick of the JOG stick to “High”

(Control panel on the teaching screen is not displayed.))

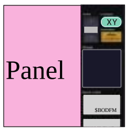


Control panel is not displayed



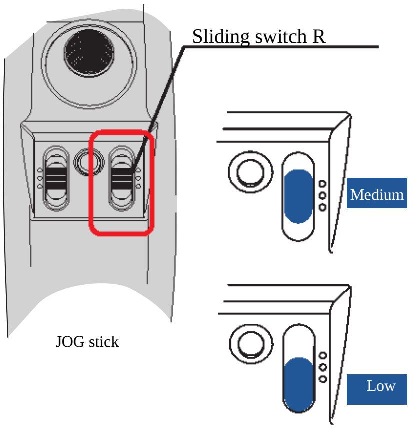
Method 2 Using the control panel

☞ P.8

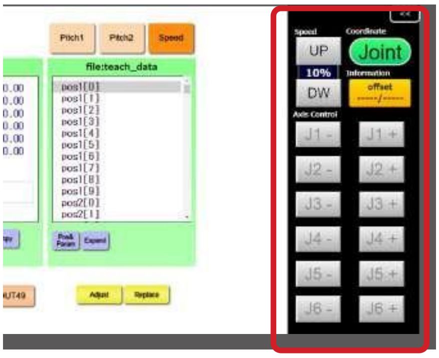


Pull the R stick of the JOG stick to “Medium” and "Low"

（The control panel of the teaching screen is displayed. Operation of the control handle of the JOG stick and the L stick is not shown.

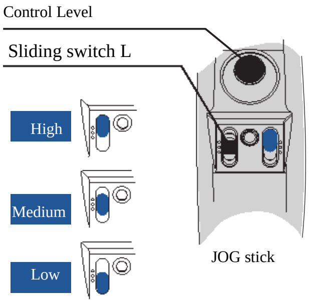


Control panel is shown



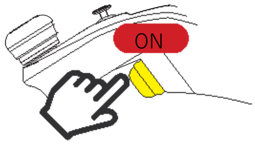
Method 1 Using the JOG stick

Use the JOG stick (optional) to control the operations of the joints of the power-assisted lifting arm. Move Jog back to the original position or use during teaching.



Step 1 Pull sliding switch R to “High”.

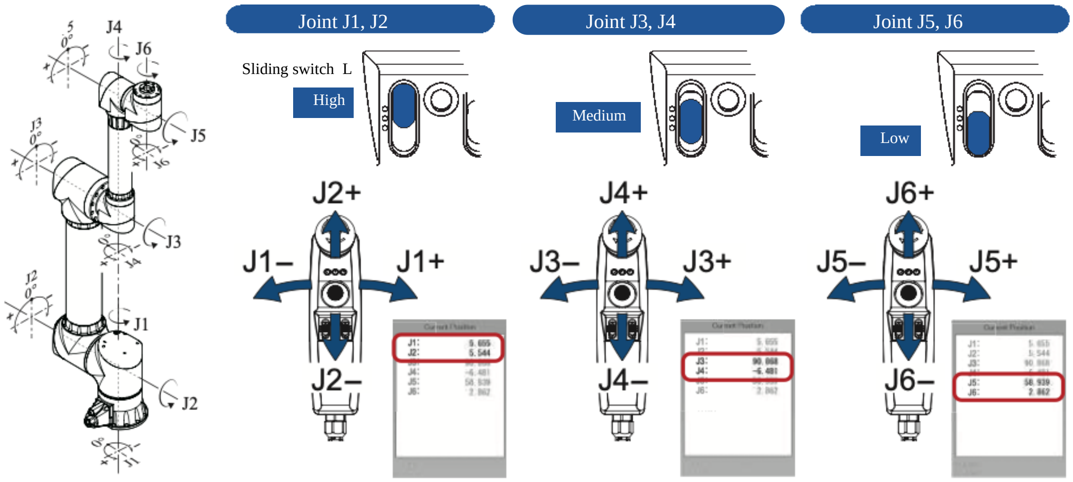
Step 2 Press Enable switch to turn on the servo.



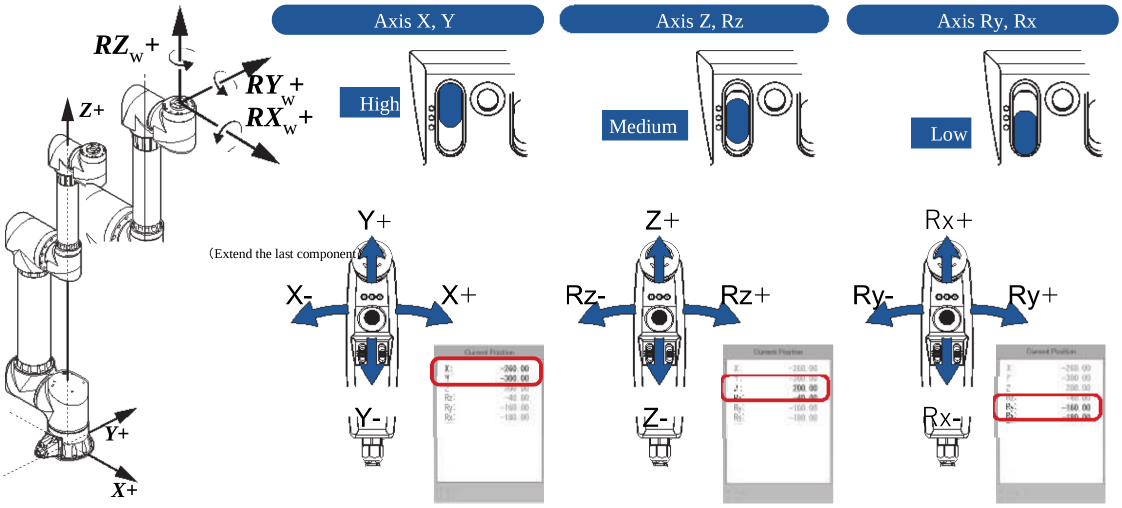
Step 3 Tilt the control handle to perform Jog operations.

Pull sliding switch L to choose the joint you want to operate.

Common coordinate system （Display “Joint” on the teaching screen



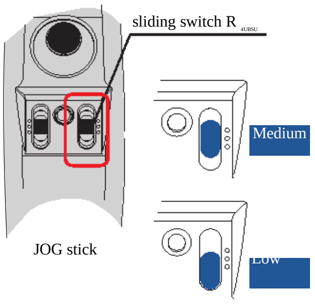
World coordinate system (Display “XY” on the teaching screen



Method 2 Using the control panel

Panel

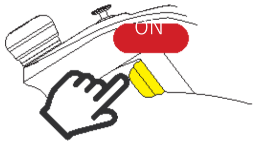
The control panel operates each joint by using the operating buttons placed on the teaching screen of the PC.



Step1 Pull sliding switch R to “Medium” or “Low”

Operating UI display on the teaching screen.

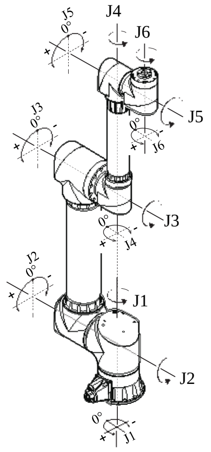
Step 2 Press the Enable switch to turn on the servo.

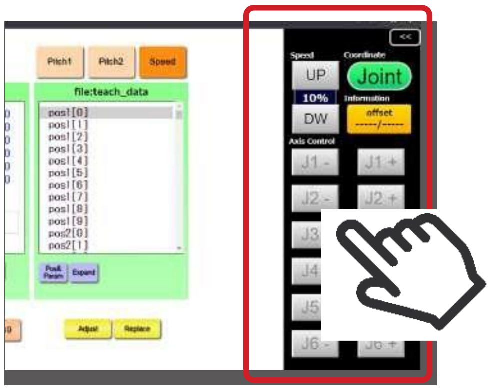


Step 3 Choose the coordinate system you want to operate on and press the stick button to perform Jog operations.

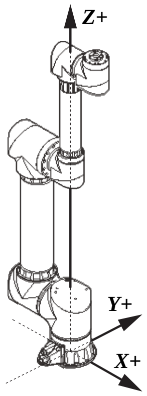
When the control panel is activated, the operations of JOG stick control handle and sliding switch L is not used

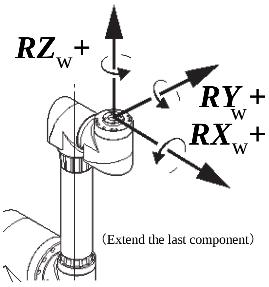
Common coordinate system



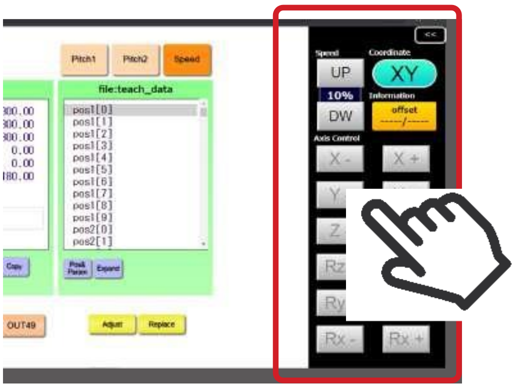


World coordinate system





(Extend the last component)

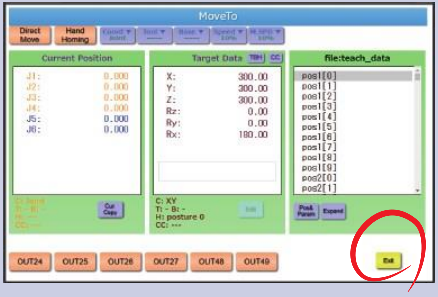


4. Teaching screen

Press the switch buttons on the Teach Main screen to switch screens

Move To screen

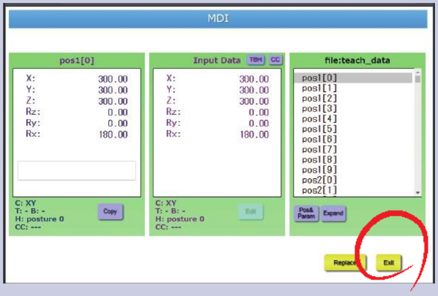
P.16



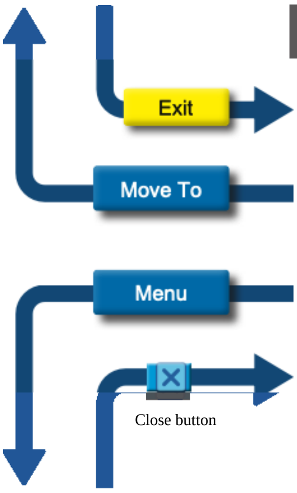
You can use the Direct Move to move directly to the teaching points and Hand Homing to move to the original posture.

MDI screen

P.22

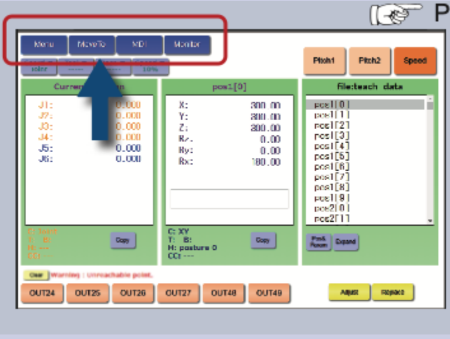


Input parameters directly into the teaching data

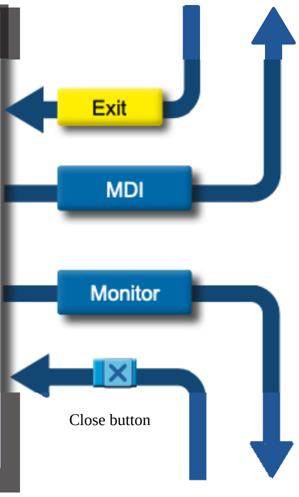


Teach Main screen

P.10

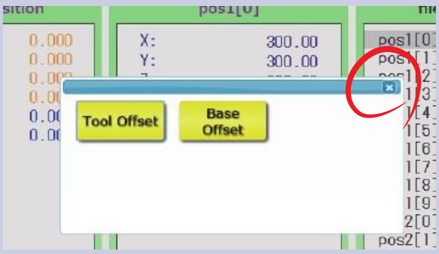


You can setup and change the teaching data, operate Jog, set the speed, set the coordinate system, etc.



Menu screen

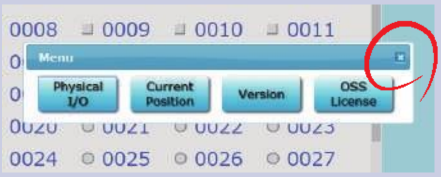
P.11



Tool Offset and Base Offset settings

Monitor screen

P.23

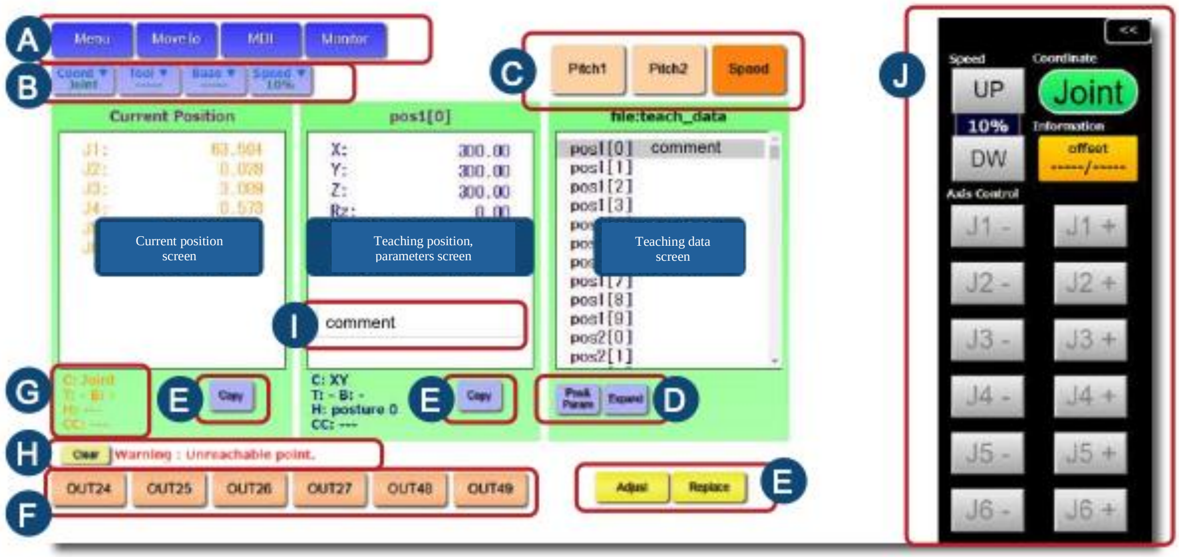


Monitor different data like the coordinate of the power-assisted lifting arm and I/O,…
You can perform Jog operations or thao tác đầu ra cổng trong khi theo dõi.

Move To screen | Control panel | MDI screen | Monitor screen

Teach Main screen

This is the main screen of the teaching activity. Setup and change the teaching data, setup the Jog operating speed, setup and change the coordinate system and adjust the offset.



A Change Menu and screens

Menu ☞ P.11 — Display the Menu screen

Move To ☞ P.16 — Change to Move To screen

MDI ☞ P.22 — Change to MDI screen

Monitor ☞ P.23 — Display Monitor screen

B Display and setup ☞ P.12

Coord ▼ Joint — Change coordinate system

Tool ▼ ---- — Select Tool offset

Base ▼ ---- — Select Base offset

Speed ▼ 50% — Setup Jog operation speed

C Change Jog operation Mode ☞ P.13

Pitch1 Pitch2 — Pitch mobile mode

Speed — continuous operation mode

D Filter teaching data ☞ P.13

Pos& Param — Change display category

Expand — Change expand display

E Interact to coordinate data, teaching data ☞ P.14

Copy — Copy the displayed coordinates to clipboard

Adjust — Display coordinate values right before TIRNING OFF servo

Replace — Save coordinate data

F Output port activity ▪ ☞ P.15

OUT24 ・・・ OUT49

Control the user I/O port

G Coordinate information

C : coordinate system T : Tool Offset setup value

B : Base Offset setup value

H : Posture CC : cross counter

H Error/warning information ☞ P.15

Display the content in the text inbox

Clear Clear text

I Display comment

Display the comment inputted to the data

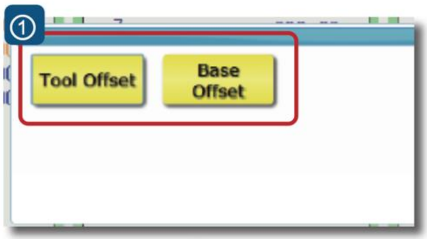
J Control panel ☞ P.19

Is displayed when change the sliding switch R on the JOG stick to “Medium” or “Low”. You can control Jog power-assisted lifting arm using the operating buttons displayed on the screen.. .

Teach Main screen

A Nút Menu

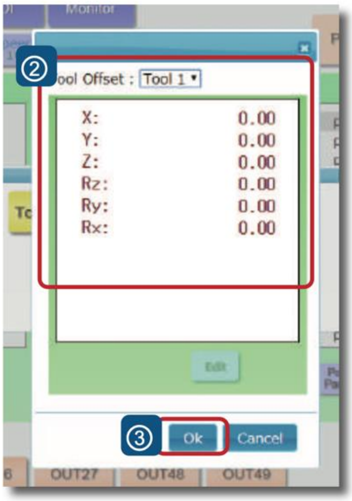
Offset settings



① Select Offset setup

Tool Offset : Tool Offset setup
Tool Offset is set according to the end-device, with Top Flange center set as the original point.

Base Offset : Base Offset setup
With Bottom Flange center as the original point, from the world coordinate system set the Base offset appropriately to the state of the power-assisted lifting arm



Tool Offset : Tool 1 ▾

X:	0.00
Y:	0.00
Z:	0.00
Rz:	0.00
Ry:	0.00
Rx:	0.00

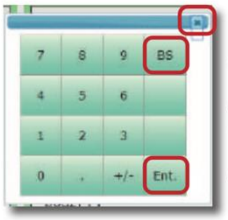
(Offset has set an option using Tool ▾ ---- , Base ▾ ----)

② Select and input into each setup field.

Select the axis you want to setup and input using PC keyboard or a 10-button panel.

Select the axis you want to setup and input a value directly by pressing Enter on the PC keyboard.

Select the axis you want to setup and input a value by pressing the Edit button and input a value using 10 buttons



☒ Close button

BS Backspace button

Ent. Enter button

③ OK Click OK to complete the setup.

Color of the buttons on the teaching screen

Identify colors according to function groups.

🟧 Orange: Action groups

🟦 Blue: Setup and menu groups

🟦 Light blue: Edit group (have only edit functions)

🟪 Purple: Select option group

🟨 Yellow: Important operations group

The symbol ▼ next to a button

Coord ▾ Joint

When you press a button with the symbol ▼ a popup will show. To close the window, press the button with the symbol ▼ again.

Purple button

Coord ▾ Joint

The second purple button displays the selected option.

Move To screen | Control panel | MDI screen | Monitor screen

Teach Main screen

B Coord ▾ Joint | Tool ▾ ---- | Base ▾ ---- | Speed ▾ 50% | Buttons

Coord ▾ Joint Switch coordinate

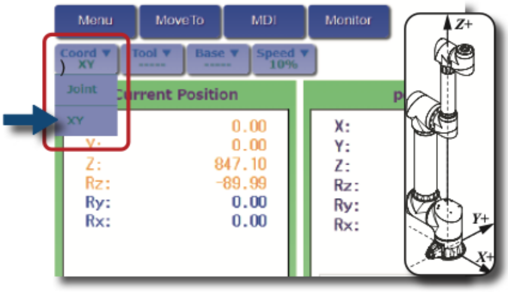
Select a coordinate system when moving the power-assisted lifting arm

The current position of the power-assisted lifting arm displayed in Current Position is also switched to the selected coordinate system screen. .

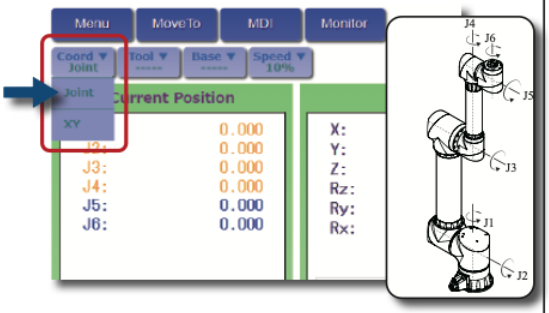
XY : World coordinate system (orthogonal coordinate system)

（(Include Base coordinate system, Tool coordinate system and user coordinate system)

Joint : Common coordinate system



X: 0.00
Y: 0.00
Z: 847.10
Rz: -89.99
Ry: 0.00
Rx: 0.00



J1: 0.000
J2: 0.000
J3: 0.000
J4: 0.000
J5: 0.000
J6: 0.000

Tool ▾ ---- Base ▾ ---- Select Offset

Change Offset setup

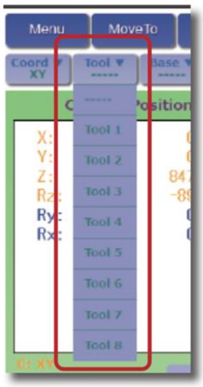
Tool ▾ ----: Select Tool Offset

With Top Flange center set as the original point, select according to The installed end-device. .

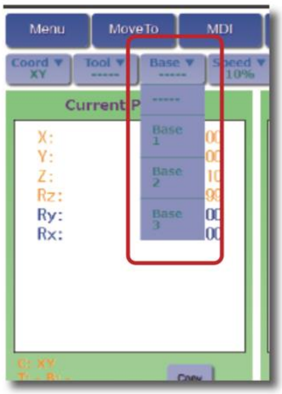
Base ▾ ----: Select Base Offset

With Top Flange center set as the original point, select according to the setup state of the power-assisted lifting arm from the world coordinate system.

Tool offset



Base offset



・Select Tool Offset or Base Offset will automatically switch to the Tool coordinate system or the Base coordinate system.

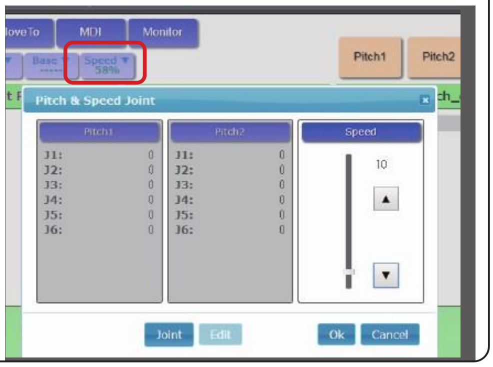
Please select「----」when deleting Offset.

・This button can be used in the Teach Main screen and the Move To screen

Speed ▾ 50% Speed setup

Set movement speed or movement amount of the power-assisted lifting arm when operating once on the Jog stick.

・For example: You can use independently, Pitch 1 for large movement and Pitch 2 for small movement (accurate positioning).



Pitch1: J1: 0, J2: 0, J3: 0, J4: 0, J5: 0, J6: 0

Pitch2: J1: 0, J2: 0, J3: 0, J4: 0, J5: 0, J6: 0

The maximum of Speed (100%) is as follows

World coordinate system XY : 80 mm/s

Common coordinate system Joint : J1~J4 5.3 deg/s

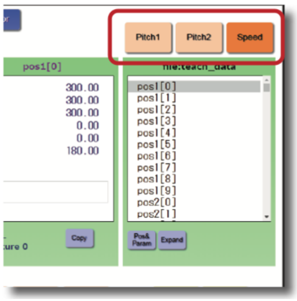
J5 ,J6 8.0 deg/s

Move To screen | Control panel | MDI screen | Monitor screen

Teach Main screen

C Pitch1 Pitch2 Speed Buttons

Change Jog operation mode.


Pitch1 Pitch2 Speed
pos1[0]
300.00
300.00
300.00
0.00
0.00
180.00
file:teach_data
pos1[0]
pos1[1]
pos1[2]
pos1[3]
pos1[4]
pos1[5]
pos1[6]
pos1[7]
pos1[8]
pos1[9]
pos2[0]
pos2[1]

Speed : continuous operation mode

[redacted]rate continuously while the handle of the JOG stick is tilted

Speed — Speed according to the setup.

[redacted]

Pitch1 Pitch2 : Pitch move mode

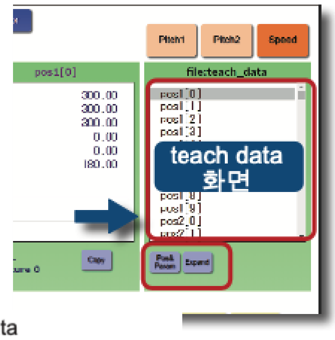
[redacted]f the JOG stick tilt once, a determined amount of mc

Speed is according to the setting values Pitch1 Pitch2

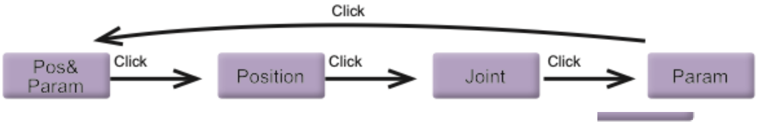
Reference P.5 to know the operate level and speed setup method..

D Pos& Param Expand Buttons

Teachdata screen Change display category of the Teach Data scree


Pitch1 Pitch2 Speed
pos1[0]
300.00
300.00
300.00
0.00
0.00
180.00
file:teach_data
pos1[0]
pos1[1]
pos1[2]
pos1[3]
pos1[8]
pos1[9]
pos2[0]
pos2[1]

Filter 1 : Pos& Param Change displayed data



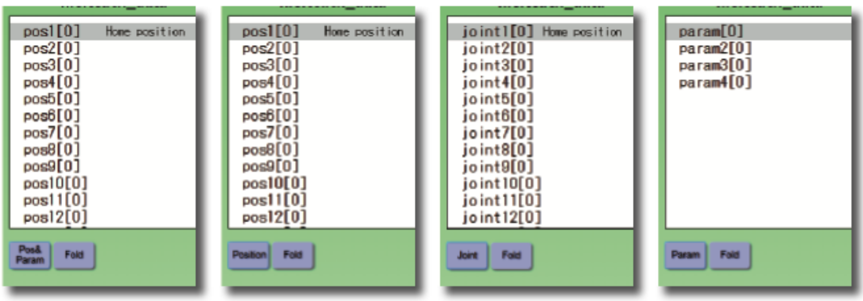
Displayed data

- Position Data
- Joint Data
- Param Data

- Position Data

- Joint Data

- Param Data


pos1[0] Home position
pos2[0]
pos3[0]
pos4[0]
pos5[0]
pos6[0]
pos7[0]
pos8[0]
pos9[0]
pos10[0]
pos11[0]
pos12[0]

pos1[0] Home position
pos2[0]
pos3[0]
pos4[0]
pos5[0]
pos6[0]
pos7[0]
pos8[0]
pos9[0]
pos10[0]
pos11[0]
pos12[0]

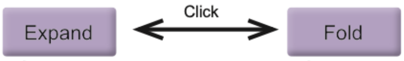
joint1[0] Home position
joint2[0]
joint3[0]
joint4[0]
joint5[0]
joint6[0]
joint7[0]
joint8[0]
joint9[0]
joint10[0]
joint11[0]
joint12[0]

param[0]
param2[0]
param3[0]
param4[0]

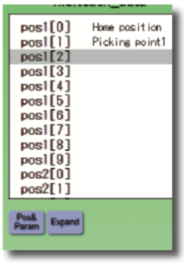
화면 예시

[redacted]

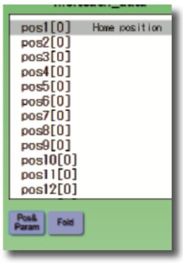
Filter 2: Expand Change layout display



Expanded layout


pos1[0] Home position
pos1[1] Picking point1
pos1[2]
pos1[3]
pos1[4]
pos1[5]
pos1[6]
pos1[7]
pos1[8]
pos1[9]
pos2[0]
pos2[1]

Fold layout


pos1[0] Home position
pos2[0]
pos3[0]
pos4[0]
pos5[0]
pos6[0]
pos7[0]
pos8[0]
pos9[0]
pos10[0]
pos11[0]
pos12[0]

Move To screen | Control panel | MDI screen | Monitor screen

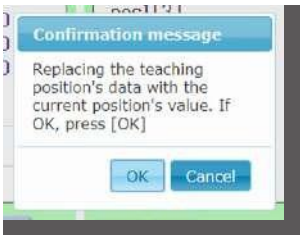
Teach Main screen

E Copy Adjust Replace Buttons

Copy — Copy a displayed coordinate value

Replace — Save positional data

Save the coordinate of the current position screen into a selected teaching data in the teaching data screen.

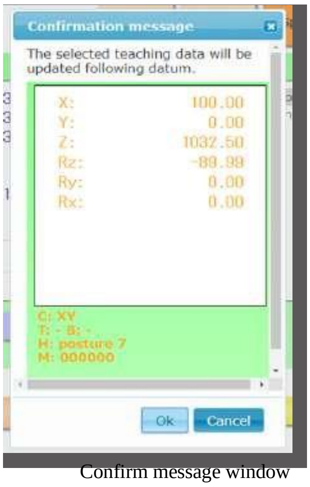

Confirmation message

Replacing the teaching position's data with the current position's value. If OK, press [OK]

Confirm message window

Adjust — Adjust and save positional data.

Use the edit current position function (*) to save a selected teaching data on the teaching data screen
*) the add coordinate value function right before TURNING OFF the Servo


Confirmation message

The selected teaching data will be updated following datum.

X:	100.00
Y:	0.00
Z:	1032.50
Rz:	-89.99
Ry:	0.00
Rx:	0.00

C: XY
T: - B: -
H: posture 7
M: 000000

Confirm message window

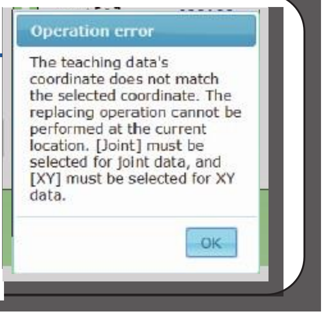
Do not turn off the power of the control set during saving the data.

포인트

The case of mismatch coordinate system

Current position screen

If the coordinate system displayed on the screen is different from the coordinate system of the saved training point, an error message will appear.
After aligning the coordinate systems, please save..


Operation error

The teaching data's coordinate does not match the selected coordinate. The replacing operation cannot be performed at the current location. [Joint] must be selected for Joint data, and [XY] must be selected for XY data.

Warning : Unreachable point.

Move To screen | Control panel | MDI screen | Monitor screen

Teach Main screen

F OUT24 … OUT49 Buttons

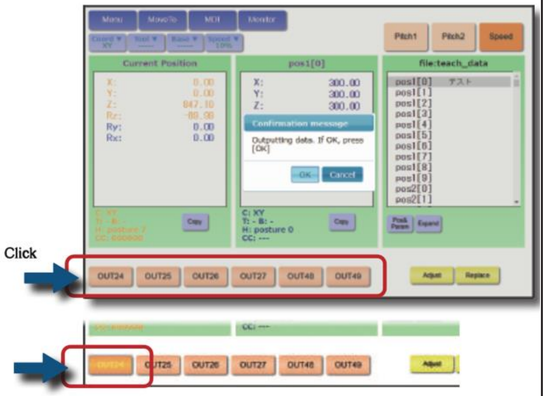
Operate output ports...

Control registered output ports.
（Port No.16 ～ 31）

Output state is represented using text colors.

Black text: OFF state for output port

Yellow text: ON state for output port



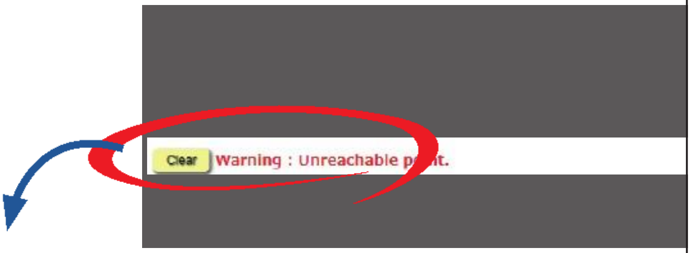
Confirmation message
Outputting data. If OK, press [OK]
OK Cancel

On state for output port

H Error/warning information

If a warning occurs during the teaching process, a notification will be shown.

Clear : Clear message



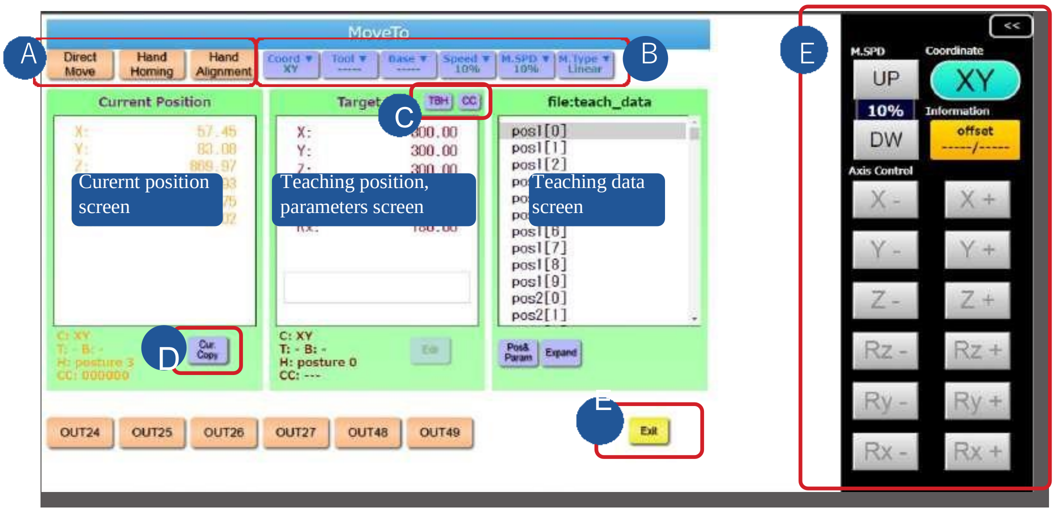
Warning : Unreachable point.

Display	Meaning
Warning - Angle limit over	The operate position of the target exceeds the operating range (±240°)
Warning - Unreachable point	Go through an unreachable point during Direct Move
Warning - Area over	Exceeds movement range (± 240°) during Direct Move
Warning - Speed over	Upper speed limit of the teaching mode has been exceeded.

Teach Main screen | Control panel | MDI screen | Monitor screen

Move To screen

Have the Operation "Direct Move” to move directly to a saved teaching point and the Opertaion "Hand Homing” to move to the original position..



A Select operation

☞ P.17

Direct Move

Operate displayed position on the screen.
Instructive position/parameter

Hand Homing

Return each axis to the original position.
Move to the 0° postion on each axis

Hand Alignment

Perform the operation Hand alignment..
Adjust the direction of the end-device according to the vertical direction or the horizontal direction.

B Display and setup

☞ P.17

Coord ▾ Joint

Change coordinate system.

Base ▾ ----

Select Base Offset .

M.SPD ▾ 50%

Set working speed of Direct Move.

Tool ▾ ----

Select Tool offset..

Speed ▾ 50%

Setup Jog operation.

M.Type ▾ Linear

Select operation method of Direct Move .

C Setup coordinate postion

☞ P.18

TBH

Setup Tool Offset, Base Offset and posture.

CC

Setup cross counter..

D Teaching data operation

☞ P.18

Cur. Copy

Copy current coordinates.

E Exit Move To screen

Exit

Return to Teach Main screen.

F Control panel

☞ P.19

Is display when the sliding switch R of the JOG stick is placed in the “Medium” position or the “Low” position. Can control the power-assisted lifting arm using the operation buttons on the screen.

The Operation panel on the Teach main screen is also similar

Move To screen

A Direct Move | Hand Homing | Hand Alignment Buttons

Direct Move Move to the coordinate displayed on the Target Data screen.

☞ P.26 Operate of the robot using「Direct Move」

Hand Homing Move the power-assisted lifting arm to the original position

☞ P.25 Move back to the original coordinate using「Hand Homing」

Hand Alignment Perform operations of hand adjust..

P.29

B M.SPD ▾ 50% | M.Type ▾ Linear Buttons

M.SPD ▾ 50% Direct Move | Hand Homing | Hand Alignment Use Direct Move, Hand Homing, Hand Alignment. to setup working speed

M.SPD ▾ 100% of MSPD is based on the speed of the move axis on the longest distance, as follows:

M.Type ▾ Linear : 80mm/s M.Type ▾ PTP J1~J4 : 5.3 deg/s
J5, J6 : 8.0 deg/s

M.Type ▾ Linear Setup automatic mode.

Coord ▾ XY Select movement mode to move between 2 points in the world coordinate system.

- M.Type ▾ Linear : Linear interpolation operation
- M.Type ▾ PTP : PTP operation
- M.Type ▾ PTP(C2) : PTP operation (C2= Use information of the cross counter set)

포인트 !

When the coordinate system of the target position does not match the selected coordinate system

Display a dialog. The coordinate system is inconsistent. Direct Move Even when you press Direct Move

the power-assisted lifting arm cannot operate for safety reasons. Coord ▾ Joint After selecting the coordinate system with

Coord Joint Direct Move Please press Direct Move again·

When move to the target position... ・ ・ ・

１）When reach the target position

An incoming inbox dialog is displayed..

（（The displayed notification is different in Direct Move and Hand Homing）

２）When you miss the control handle or the Enable switch of the JOG stick while operating

The operation is cancelled.

（(A notification dialog Abort appears.)

３）When go through an unmovable point

Stop the operation. (A notification dialog Unreach appears.)

Teach Main screen | Control panel | MDI screen | Monitor screen

Move To screen

C Setup positional coordinates

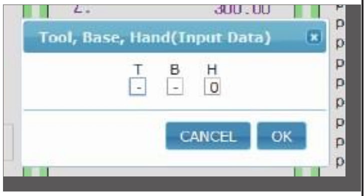
TBH **Setup Tool offset, Base offset and posture. .**

T : Tool offset

Select according to the end-device. Setup value: "-" (cancel setup), "1" ~ "7"

B : Base offset

Select according to the setup state of the power-assisted lifting arm. Setup value: "-" (cancel setup), "1" ~ "3"

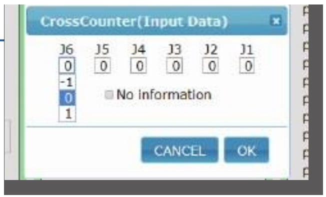


H : Position

Select according to the position of the power-assisted lifting arm. Setup value:� "0" ~ "7"

CC **Setup cross counter.**

Rotation information from J1 to J6 is recorded according to 0° of each axis, regardless whether the axis rotates in the direction – or the direction +..

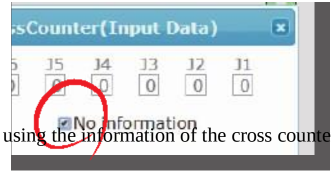


-1 : Rotate more than 180° in the direction -

0 : Do not rotate

1 : Rotate more than 180° in the direction +

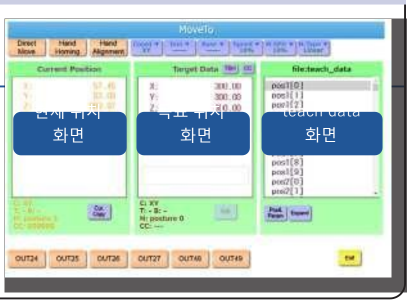
If you do not use the information of the cross counter, please select ☑ No Information



The case of not using the information of the cross counter

D Operations on teaching data

Cur. Copy Copy current coordinate



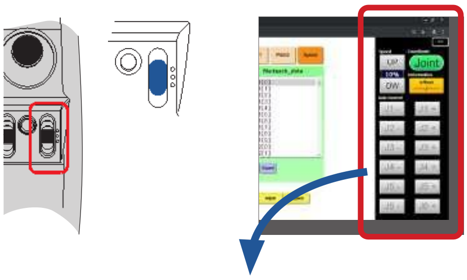
The coordinate of the current position screen is copied on the target position screen..

When save as teaching data, make edits on the MDI screen.

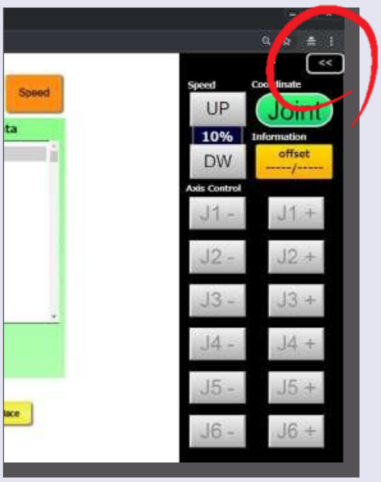
The target position modified in the Move To screen is for confirming movement and should not be reflected in the training data

Control panel

Can be used to control the power-assisted lifting arm using the operation panel instead of the JOG control lever. Can setup operation board, check error codes, verify “posture” of the power-assisted lifting arm and check the definition of the coordinate system and common axes.



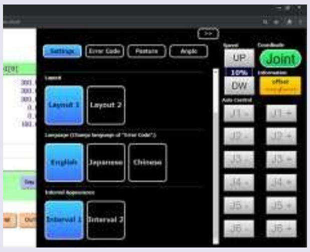
Control panel



<< Press the button << to extend the control panel..

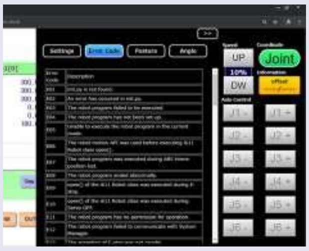
Settings

Setup the control panel.



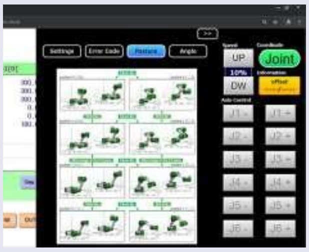
Error Code

Display list of error codes.



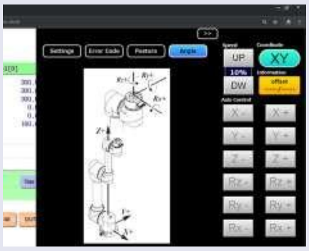
Posture

Display the “position” of the power-assisted lifting arm



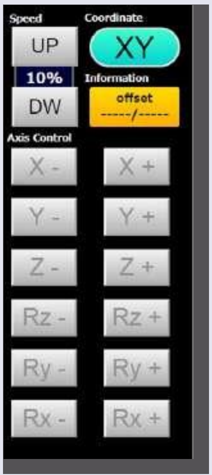
Angle

Display the definition of the coordinate system or common axes of the power-assisted arm

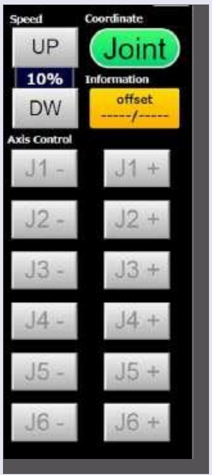


Control panel

Control board description



Orthogonal coordinate system



Common coordinate system

XY Joint Change coordinate system

Press and hold the button (about 1 second) to change.

Button X Y ⟷ Joint Button

Press and hold
(about 1 second)

Coord ▾ Joint link to the coordinate system set by the Coord Joint button.

UP DW Change movement speed

For each button press, Setting changes to "1%, 3%, 5%, 10%
15%, 20%, 30%, 50%"...

Operation speed when set into 100%

world coordinate system XY : 80 mm/s

common coordinate system : J1~J4 5.3 deg/s
. . . . . . J5 ,J6 8.0 deg/s

Button UP
10%
Button DW

Display current setup.

「Link to the setting "Pitch and movement speed settings".. :

Speed Continuous operation mode

Pitch1 Pitch2 : Pitch moving mode

X- … Rx+ / J1- … J6+ Axis control buttons

Display change buttons depending on select coordinate system..

Common coordinate system Nút J1- … J6+

Orthogonal coordinate system Nút X- … Rx+

Button operations

Quick press: perform 1 time Pitch movement
Hold press: continuous movement

offset ----/---- Display offset information

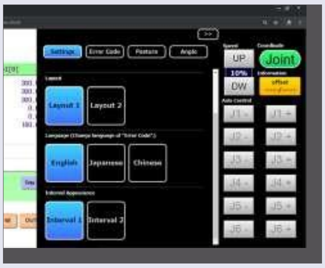
Display setting values of Tool offset, Base offset .

Offset is set using buttons Tool ▾ ---- , Base ▾ ----

Teach Main screen | Move To screen | MDI screen | Monitor screen

Control panel

Settings Setup operation control panel



Screen:

Switch between full screen display and normal screen display.

Layout:

Change button position..

Language:

Change display language of the error code list..

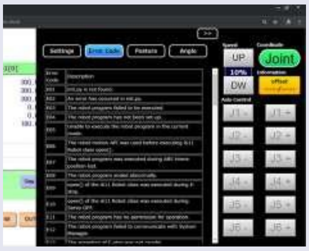
「English」 「Japanese」 「Chinese」

Interval Appearance:

When an operation occur, set the display mode of the current operation invalid before accepting the next operation. .

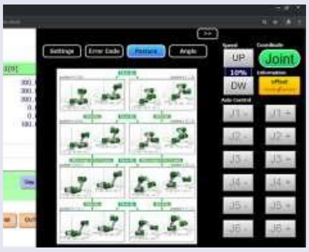
Interval 1: Display the window「Data in Process…」.
Interval 2: Display a dimmed screen.

Error Code Display error code list.

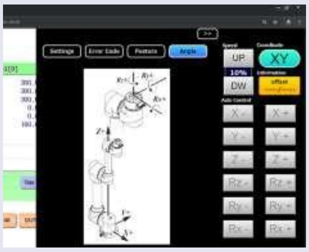


Deatiled information about the errors can be changed to English, Japanese and Chinese.

Posture Display “posture” of the power-assisted lifting arm.



Angle Display the axis angle definition or the common axis of the power-assisted lifting arm

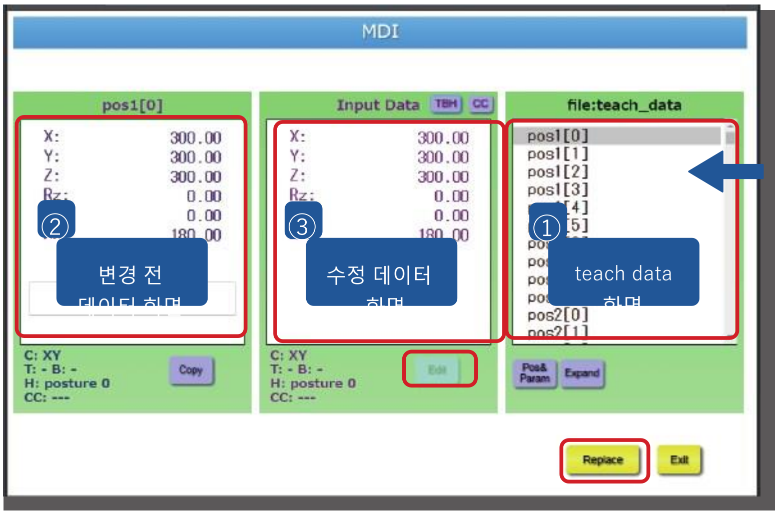


Teach Main screen | Move To screen | Control panel | Monitor screen

MDI screen

MDI = 「Manual Data Input」

Change selected teaching data.



Step 1 Teach data — Select the teaching data you want to make changes in the teach data screen.

Click : Select data

Double click to the background : Input description (32 characters max).

Step 2 Import the Position data.

The coordinate values of the teaching data selected in the previous data are displayed before the screen changes.

Input using PC keyboard

Select the axis you want to set and enter the value directly by pressing the Enter key on the PC keyboard.

Input using 10-button panel

Select the axis you want to change and input the value using the PC keyboard or a 10-button panel.

Step 3 Register position data

Replace Save position data

Replace the selected teaching data with position data from the Change Data screen.

포인트 /

Number of teaching points that can be set

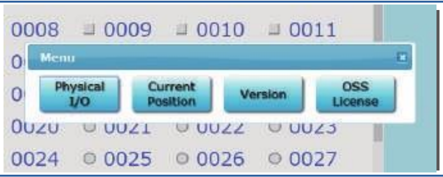
Data	Display	Points
Orthogonal coordinate system data	pos1[0] ～ [9] … pos20[0] ～ [9]	200
Common coordinate system data	joint1[0] ～ [9] … joint20[0] ～ [9]	200
Parameters data	param[0] ～ [9] … param4[0] ～ [9]	40

Monitor screen

Monitor different data such as coordinates of the power-assisted lifting arm and I/O. It is possible to perform Jog operations or output port operations during monitoring.

Display Change — Change the display items

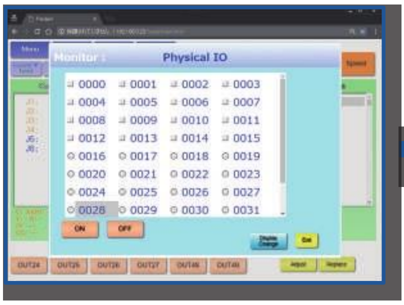
Select display items from physical I/O, current control set position, version information, and OSS license information.



Physical I/O — Physical I/O

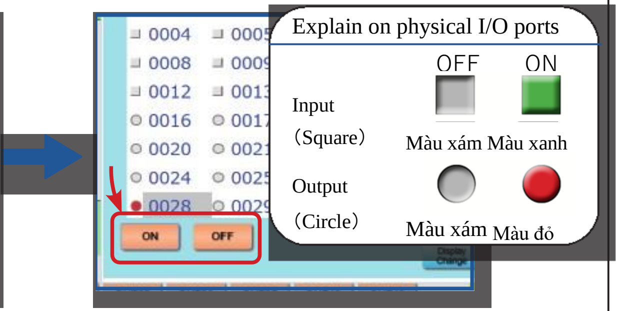
Monitor physical I/O ports.

Operate by using ON OFF buttons output state.



Monitor : Physical IO

□ 0000	□ 0001	□ 0002	□ 0003
□ 0004	□ 0005	□ 0006	□ 0007
□ 0008	□ 0009	□ 0010	□ 0011
□ 0012	□ 0013	□ 0014	□ 0015
○ 0016	○ 0017	○ 0018	○ 0019
○ 0020	○ 0021	○ 0022	○ 0023
○ 0024	○ 0025	○ 0026	○ 0027
○ 0028	○ 0029	○ 0030	○ 0031

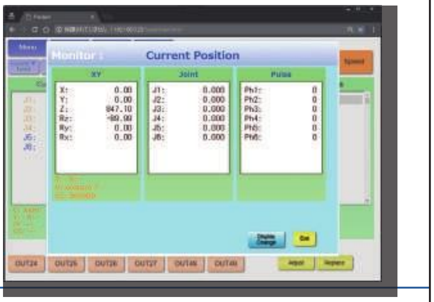


Explain on physical I/O ports

	OFF	ON
Input (Square)	▨ Màu xám	🟩 Màu xanh
Output (Circle)	○ Màu xám	🔴 Màu đỏ

Current Position — Display current position of the power-assisted lifting arm

Three types of information are displayed on the screen: world coordinates, common coordinates, and encoded pulse set.

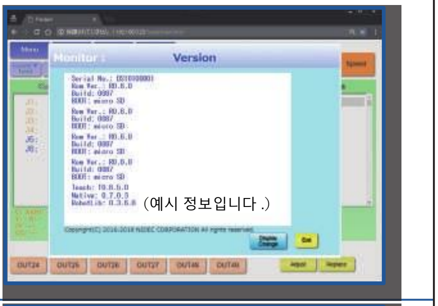


Monitor : Current Position

XY		Joint		Pulse	
X:	0.00	J1:	0.000	Ph1:	0
Y:	0.00	J2:	0.000	Ph2:	0
Z:	847.10	J3:	0.000	Ph3:	0
Rz:	-89.90	J4:	0.000	Ph4:	0
Ry:	0.00	J5:	0.000	Ph5:	0
Rx:	0.00	J6:	0.000	Ph6:	0

Version — Version information

Display version of each software on the system



Monitor : Version

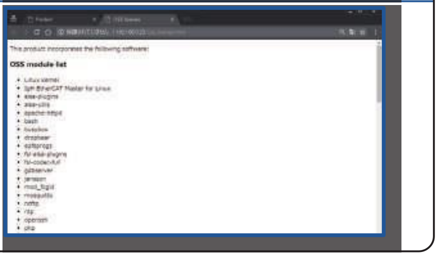
```
Serial No.: 0010000001
Rom Ver.: R0.0.0
Build: 0007
BOOT: micro SD
Rom Ver.: R0.0.0
Build: 0007
BOOT: micro SD
Rom Ver.: R0.0.0
Build: 0007
BOOT: micro SD
Rom Ver.: R0.0.0
Build: 0007
BOOT: micro SD
Teach: 10.0.5.0
Native: 0.7.0.0
RobotLib: 0.3.6.0
```

(예시 정보입니다 .)

Copyright(C) 2016-2018 NIDEC CORPORATION All rights reserved.

OSS License — OSS license information

Display license information for open-source software used in this product



This product incorporates the following software:

OSS module list

- Linux kernel
- IgH EtherCAT Master for Linux
- alsa-plugins
- alsa-utils
- apache-httpd
- bash
- busybox
- dropbear
- e2fsprogs
- fsl-alsa-plugins
- fsl-codec-lib
- gdbserver
- jansson
- mod_fcgid
- mosquitto
- ntfs
- ntp
- openssh
- php

2. Teaching order

Teaching progress

Connect the robot to the teaching PC perform teaching after completing the setup of movement amount and speed. Perform teaching on the movement coordinates of the robot and operations in Hand Homing, Direct Move, Hand Alignment, etc. After confirming the training points, save the coordinate data as training data.

Hand Homing | Direct Move | Hand Alignment

Move back to the original coordinate using “Hand Homing”

P.25

Hand Homing — Restore the power.

Move the robot using「Direct Move」

P.26

Direct Move — Move the power-assisted lifting arm to a set position..

Move robot using「Hand Alignment

P.29

Hand Alignment — Orient the terminal vertically or horizontally.

Save coordinate data

P.31

Save the current coordinate value of the power-assisted lifting arm as teaching data.

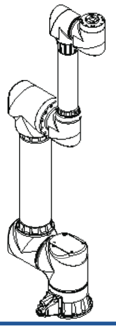
Recover from an unmovable point

P.32

Exit the power-assisted lifting arm from an unmovable position...

Note: Jog, Direct Move, and Hand Alignment operations cannot be performed on in the orthogonal coordinate system at the original position.

original position



Axis values

J1 = 0 deg

J2 = 0 deg

J3 = 0 deg

J4 = 0 deg

J5 = 0 deg

J6 = 0 deg

The original position is a position that cannot be operated with the world coordinate system due to the structure of the power-assisted lifting arm.
Individual operations of J1~J6 are performed through swivel movements in the joint coordinate system, or exit the robot through direct movements in the joint coordinate system

Direct Move

P.32 Recover from an unmovable point

Move Robot using 「Hand Homing」 | Move Robot using 「Hand Alignment」 | Save coordinates data | Recover from unmovable point

Move to the original coordinate using “Hand Homing”

Hand Homing ：Recover the original point

Step 1 Set the move speed.

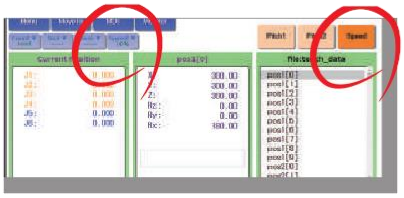
In the Teach Main screen Speed Click Speed .

Speed ▾ 10% Check if the speed is set to 10%..

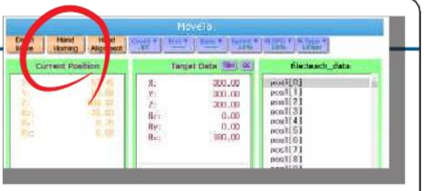
It is recommended to use an operating speed of around 10%..
In case of accelerating operation, please operate after ensuring full safety confirmation..

Speed ▾ 100% of Speed is based on the speed of the longest travel distance of the axis .

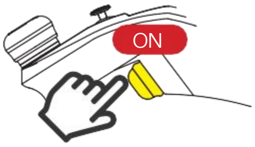
J1 ～ J4 : 5.3 deg/s J5, J6 : 8.0 deg/s



Step 2 Hand Homing Click Hand Homing.



Turn on servo



Step 3 Start moving .

JOG Stick

When use the Jog stick

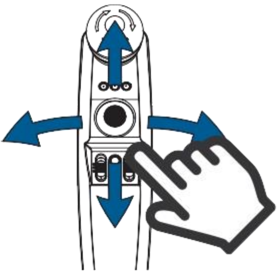
If the control handle of the control set is tilted then start moving.

Tilt the handle of the control set in any direction.

Only move when tilted

【[Stop]: Release the control handle.

Complete]: Notify the completion using a popup screen



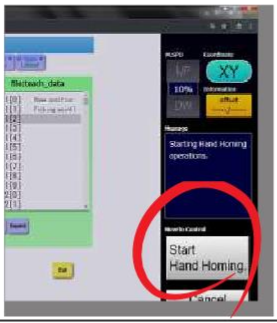
Panel

When use the Control Panel

Start Hand Homing Press the button Start Hand Homing to start moving.

【Stop】 :Cancel Press Cancel.

【Complete】：Notify the completion using a popup screen.



Move to the original coordinate using「Hand Homing」 | Move robot using「Hand Alignment」 | Save coordinates data | Recover from unmovable point

Move robot using「Direct Move」

Direct Move ：Operate the control set back to a preset position.

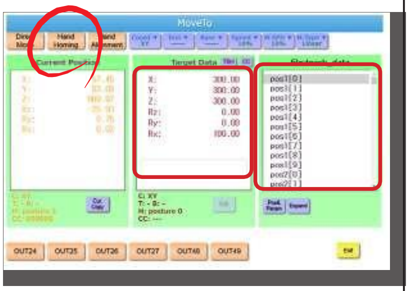
Step 1 Select or input a target position.

- When register teaching data：

➡ file: Select target position from Teach_data .

- When change teaching data or When set up a new target position：

➡ Select target joint for Target Data .



Input using PC keyboard

Press Enter on the PC keyboard to input a value directly...

Input using 10-button panel

Edit Press Edit to input a value using 10-button panel.

- When setup Tool offset, Base offset and posture: :

➡ TBH Press TBH to select a parameter.

T: Tool offset

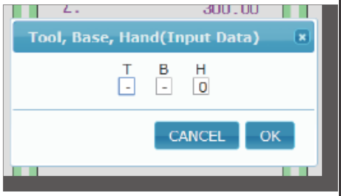
Select an end-device. Setup value: "-" (cancel setup),, "1" ~ "7"

B: Base offset

Select according to the setup status of the power-assisted lifting arm. Setup value: "-" cancel setup ,"1" ~ "3"

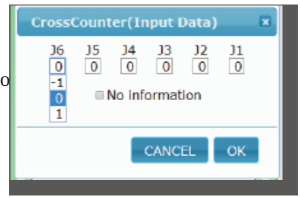
H: posture

Select according to the posture of the power-assisted lifting arm. Giá trị cài đặt: ,"1" ~ "7"

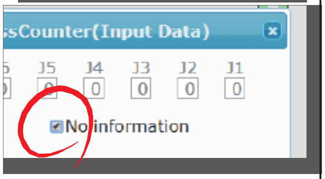


- When setup the cross counter (*1): :

➡ CC Press CC set up the values below within the angular range of each joint from J1 to



Setup value	Angle of each joint
1	180° ~ 540°
0	-180° ~ 180°
F (*2)	-540° ~ -180°



When not using the information of the cross counter,

Please select ☑ No Information 」

In case the cross counter information is not used

*1) The cross counter is set up to convert Position data into characteristic Joint angle data. Setup multiturn arguments of the Position data.
For details, please refer to

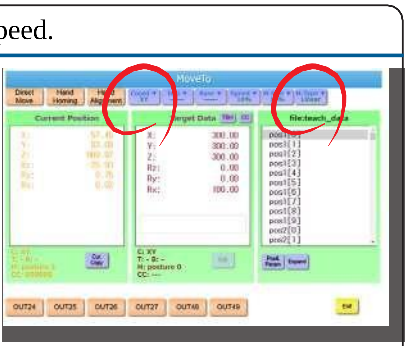
software D Robot Library 2

*2) On the training screen, setting "F" will display "-1"

Move to the original coordinate using 「Hand Homing」 | Move robot using 「Hand Alignment」 | Save coordinates data | Recover from unmovable point

Move robot using「Direct Move」

Step 2 : Select coordinate system, operation method and speed.



・Coordinate system：

Synchronize selected teaching data to the coordinate system you want to operate on..

Coord ▾ XY World coordinate system

Coord ▾ Joint Common coordinate system

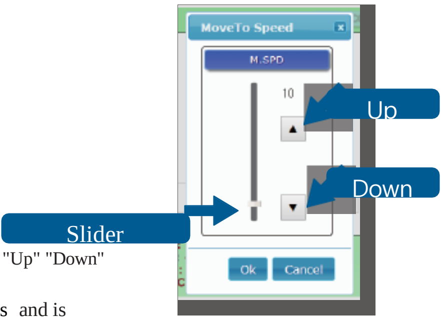
・Operation method：

Coord ▾ XY Select movement mode between 2 points in the world coordinate system.

M.Type ▾ Linear Linear interpolation

M.Type ▾ PTP PTP operation

M.Type ▾ PTP(C2) PTP operation (C2= use cross counter information）



・Movement speed ▲▼ Adjust movement speed (%) using a slider or the buttons "Up" "Down"

100% of Speed is based on the speed of the longest travel distance of the axis and is calculated as so:.

M.Type ▾ Linear 80 mm/s

M.Type ▾ PTP J1~J4：5.3 deg/s
J5, J6：8.0 deg/s

Next page

포인트

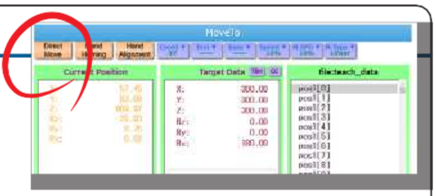
The coordinate values and offset information set in the MoveTo screen are temporary settings for operation verification purposes. They should not be reflected in the training data

Move to the original coordinate using「Hand Homing」 | Move robot using「Direct Move」 | Save coordinates data | Recover from unmovable point

Move robot using「Direct Move」(detailed)

Order 3 Direct Move Click direct move.

Turn on servo.



X: 300.00
Y: 300.00
Z: 300.00
Rz: 0.00
Ry: 0.00
Rx: 180.00

pos1[0]
pos1[1]
pos1[2]
pos1[3]
pos1[4]
pos1[5]
pos1[6]
pos1[7]
pos1[8]

Order 4 Start movement.

JOG Stick

When use the JOG stick

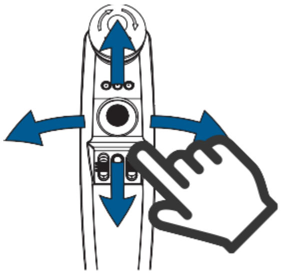
If the control handle of the control set is tilted then start moving.

Tilt the handle of the control set in any direction.

Only move when tilted.

~~【Stop】：Release from the control set handle.~~

【Complete】：Notify the completion using a popup screen



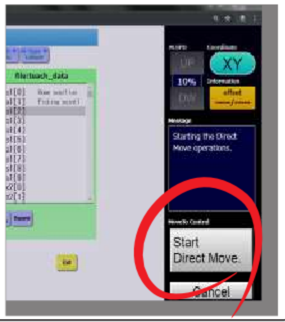
Panel

When use an operation panel

Start Direct Move If press this button then start moving.

【Stop】：Cancel Press cancel.

[Complete】：Notify the completion using a popup screen.



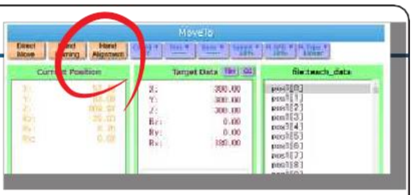
Starting the Direct Move operations.

Move to the original coordinate using 「Hand Homing」 | Move robot using「Direct Move」 | Save coordinates data | Recover from unmovable point

Move robot using「Hand Alignment」

Hand Alignment : Align the orientation of the end effector horizontally or vertically.

Hand Alignment Click Hand Alignment.



Turn on servo.

Order 2 Start moving.

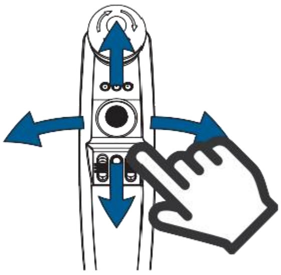
JOG Stick

When use the JOG stick

If the control handle of the control set is tilted then start moving.

Tilt the handle of the control set in any direction.

Only move when tilted.



【Stop】: Release from the control set handle..

【Complete】 : Notify the completion using a popup screen

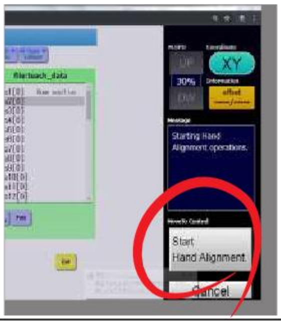
Panel

When use the operation panel

Start Hand Alignment. If press this buttom then start moving

【Stop】 Cancel Press cancel.

【Complete】 : Notify the completion using a popup screen.



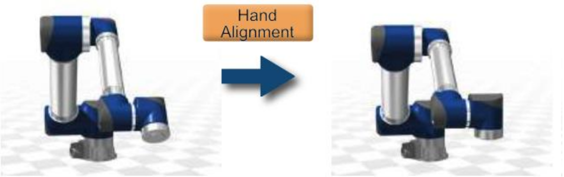
Move robot using「Hand Alignment」

Explain the Hand Alignment operation

Hand Alignment is an operation that maintains the position coordinates (x, y, z) of the end effector flange and adjusts the orientation of the end effector flange. The direction of the Hand Alignment operation varies depending on the orientation of the end effector flange.

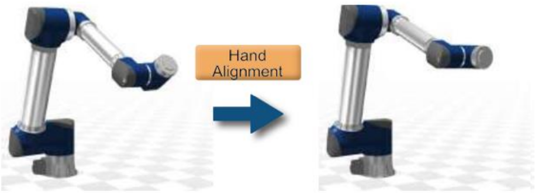
Pattern 1

When end effector Flange is near the downwards direction



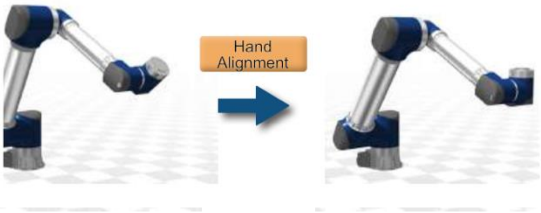
Pattern 2

When end effector Flange is near the direction of the plane



Pattern 3

When end effector Flange is near the upwards direction



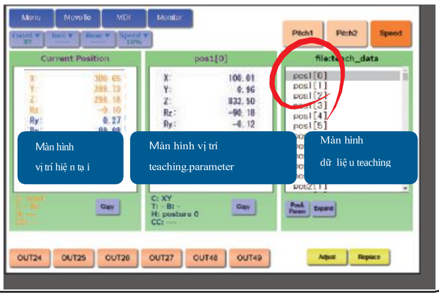
Save coordinates data

Current position screen Save coordinates value of the current position screen using teaching data.

Order 1 Select teaching data of the intended subject.

When you are in the Move To screen, click Exit to return to the Teach Main screen.

Teach Main screen

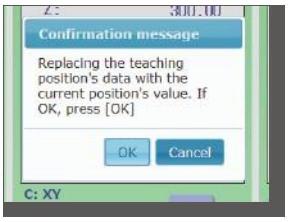


For example：Save the coordinates of the current position screen on pos1[0].

Order 2 Replace Click replace.

Save the coordinates of the current position on pos[0].

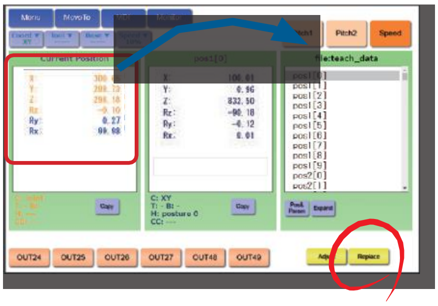
Display on a popup screen.



Confirmation message

Replacing the teaching position's data with the current position's value. If OK, press [OK]

OK Cancel



OK Click ok to save.

If you are done with saving then the popup screen will close.

Move to the original coordinate using 「Hand Homing」 | Move robot using "Direct move" | Save coordinates data

Recover from unmovable point

Restore the power-assisted lifting arm from the non-operational position.

Method 1 Using the JOG stick

Coord ▾ Joint Control individual joints J1~J6 using the JOG stick in a joint-connected coordinate system Coord Joint, and move the power-assisted lifting arm to an operable position (a position that is not at the limit of the orthogonal coordinate system)

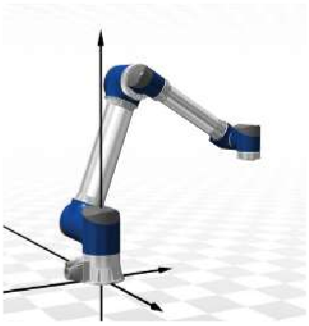
Order 1 Select a joint using the sliding switch L of the JOG stick.

Order 2 Turn on the servo.

Order 3 Tilt the handle of the control set, control the power-assisted lifting arm then move the power-assisted lifting arm to a controllable position.

Example: controllable position

Adjust J1~J6 to have a similar posture to the drawing on the left.
Please confirm the angular positions of the joints at the current position screen before proceeding with the movement.

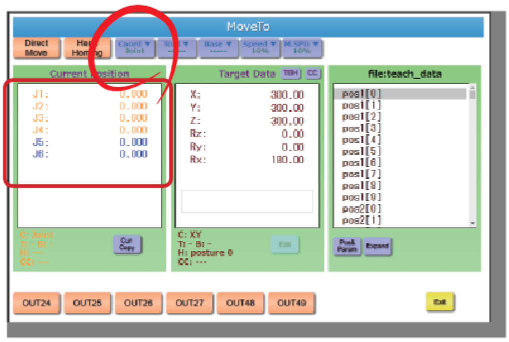


Posture image

Example) value of joints

J1 :	0 deg
J2 :	-20 deg
J3 :	-100 deg
J4 :	0 deg
J5 :	-60 deg
J6 :	0 deg

Move To screen



MoveTo

Current Position	
J1 :	0.000
J2 :	0.000
J3 :	0.000
J4 :	0.000
J5 :	0.000
J6 :	0.000

C: Joint
T: - B: -
H: ---
CC: ---

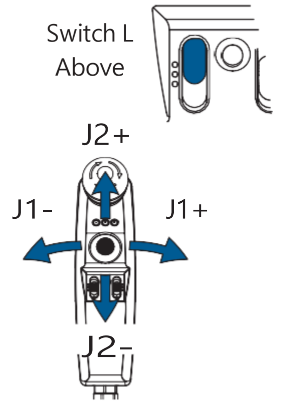
Target Data	
X:	300.00
Y:	300.00
Z:	300.00
Rz :	0.00
Ry :	0.00
Rx :	180.00

C: XY
T: - B: -
H: posture 0
CC: ---

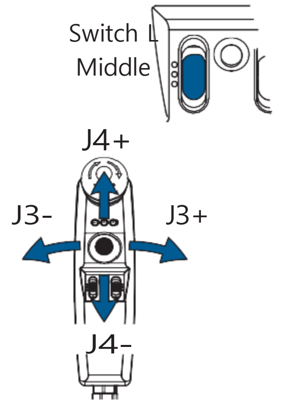
file:teach_data
pos1[0]
pos1[1]
pos1[2]
pos1[3]
pos1[4]
pos1[5]
pos1[6]
pos1[7]
pos1[8]
pos1[9]
pos2[0]
pos2[1]

Joint operation methods

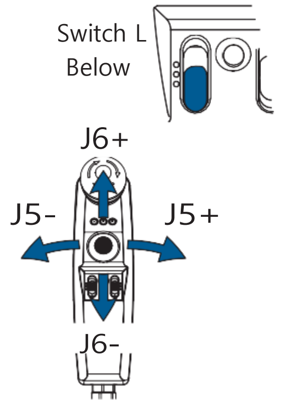
Joint J1, J2

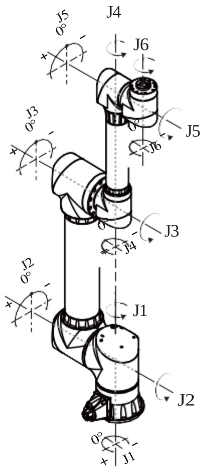


Joint J3, J4



Joint J5, J6





Move to the original coordinate using 「Hand Homing」

Move robot using "Direct move"

Save coordinates data

Recover from unmovable point

Method 2

Move using Direct Move at the teaching point of the joint coordinate system

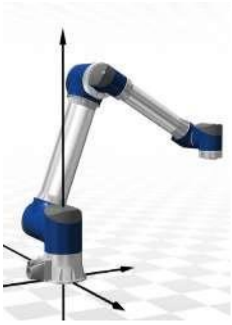
Direct Move

The world coordinate system stores the movable positions at the teaching point of the joint coordinate system (e.g., Joint1 [0]). Even if you are at a non-operational point in the world coordinate system during teaching, after switching to the joint coordinate system, you can easily restore by moving with Direct Move.

Direct Move

After recover, you can change to the world coordinate system and rerun teaching.

For example: Register movable point at Joint1 [0] .

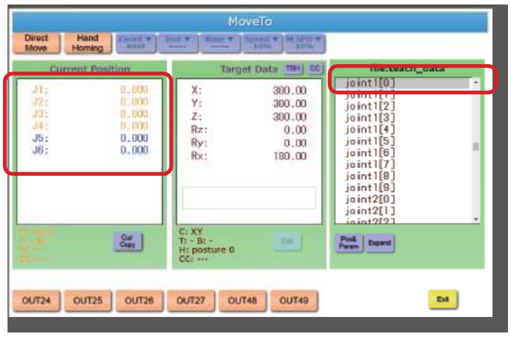


Posture image

Example value of joints

J1 :	0 deg
J2 :	-20 deg
J3 :	-100 deg
J4 :	0 deg
J5 :	-60 deg
J6 :	0 deg

Move To screen



MoveTo

Direct Move | Hand Homing | Coord Joint

Current Position

J1:	0.000
J2:	0.000
J3:	0.000
J4:	0.000
J5:	0.000
J6:	0.000

Target Data

X:	300.00
Y:	300.00
Z:	300.00
Rz:	0.00
Ry:	0.00
Rx:	180.00

C: XY
T: - B: -
H: posture 0
CC: ---

file:teach_data

joint1[0]
joint1[1]
joint1[2]
joint1[3]
joint1[4]
joint1[5]
joint1[6]
joint1[7]
joint1[8]
joint1[9]
joint2[0]
joint2[1]
joint2[2]

포인트!

Use the joint coordinate system to avoid the non-movable points in the world coordinate system.

World coordinate system

There are immovable points depending on the posture, even within the operating range.

Joint coordinate system

Is able to move freely within the operating range.

If you encounter an unmovable point of the power-assisted lifting arm while operating in the world coordinate system Coord XY, Direct Move cannot be used. Switch to the joint coordinate system Coord Joint, use Direct Move, individually operate J1~J6 using the JOG stick, and then move from the unmovable position.

Coord ▾ XY

Direct Move

Coord ▾ Joint

Direct Move

3. Transfer teaching data

1. Transfer from the control set to PC

It is recommended to backup the teaching data into a PC.

The teaching data is stored into the control set.

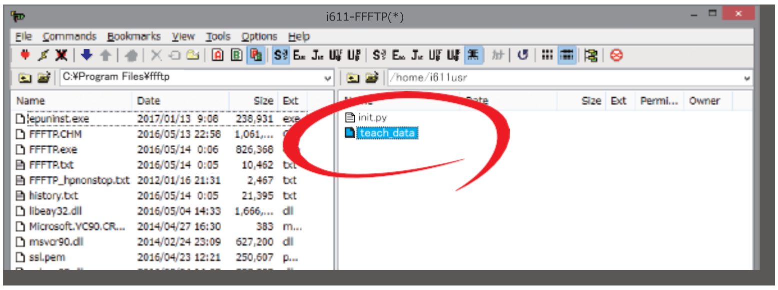
It is possible to integrate the operation programs or teaching data of the robot in case of replacing the control set..

Order 1 「Run FFFTP」.

FFFTP.exe

Save location of teaching data and file name

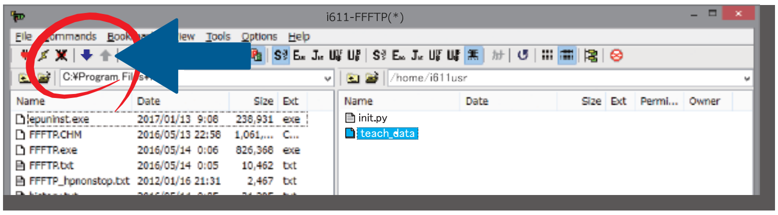
Save path	/home/i611usr
File name	🗋 teach_data



```
i611-FFFTP(*)
C:¥Program Files¥ffftp                          /home/i611usr

Name                    Date              Size      Ext    Name        Date   Size  Ext  Permi...  Owner
epuninst.exe            2017/01/13  9:08  238,931   exe    init.py
FFFTP.CHM               2016/05/13 22:58  1,061,...        teach_data
FFFTP.exe               2016/05/14  0:06  826,368
FFFTP.txt               2016/05/14  0:05  10,462    txt
FFFTP_hpnonstop.txt     2012/01/16 21:31  2,467     txt
history.txt             2016/05/14  0:05  21,395    txt
libeay32.dll            2016/05/04 14:33  1,666,... dll
Microsoft.VC90.CR...    2014/04/27 16:30  383       m...
msvcr90.dll             2014/02/24 23:09  627,200   dll
ssl.pem                 2016/04/23 12:21  250,607   p...
```

Order 2 Transfer file to PC .

Click upload



```
i611-FFFTP(*)
C:¥Program Files¥                               /home/i611usr

Name                    Date              Size      Ext    Name        Date   Size  Ext  Permi...  Owner
epuninst.exe            2017/01/13  9:08  238,931   exe    init.py
FFFTP.CHM               2016/05/13 22:58  1,061,... C...   teach_data
FFFTP.exe               2016/05/14  0:06  826,368   exe
FFFTP.txt               2016/05/14  0:05  10,462    txt
FFFTP_hpnonstop.txt     2012/01/16 21:31  2,467     txt
```

2. Transfer from PC to the control set

!	Please confirm the operation in case of integrating operation programs or teaching data into another control set.	⚠ ⚠
!	Please save to the specified path when transferring teaching data to the control set.	⚠

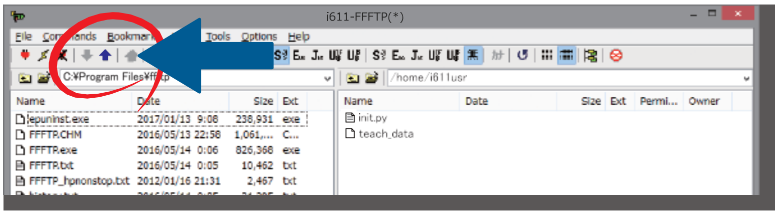
Order 1 Run「FFFTP」.

Save location of the teaching data

Save path	/home/i611usr

Order 2 Transfer file into the control set.

Click upload.



C:¥Program Files¥ffftp

Name	Date	Size	Ext
!epuninst.exe	2017/01/13 9:08	238,931	exe
FFFTP.CHM	2016/05/13 22:58	1,061,...	C...
FFFTP.exe	2016/05/14 0:06	826,368	exe
FFFTP.txt	2016/05/14 0:05	10,462	txt
FFFTP_hpnonstop.txt	2012/01/16 21:31	2,467	txt

/home/i611usr

Name	Date	Size	Ext	Permi...	Owner
init.py					
teach_data					

Notes

C Teaching

5 Coordinate system and posture

1. Coordinate system

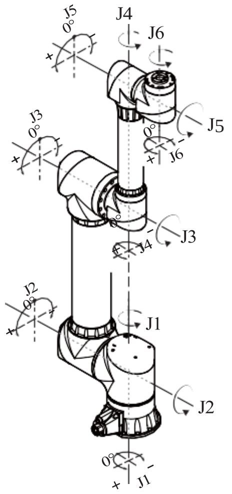
The defined coordinate systems in this product include 5 coordinate systems: joint coordinate system, world coordinate system, base coordinate system, tool coordinate system, and user coordinate system.

Please select the coordinate system (*) that best suits your purpose after reviewing the definitions.

*) The 'Teaching mode' corresponds to the joint coordinate system and the world coordinate system. The base coordinate system and the Tool coordinate system are set with offsets in the world coordinate system and is utilized.

Please consider both ‘unmovable point’ and ‘posture’ of the power-assisted lifting arm when deciding on the coordinate system. The posture includes a total of 8 types based on the combination of angles of the joints J1~J6, defined by the parameter ‘posture’.

Joint coordinate system



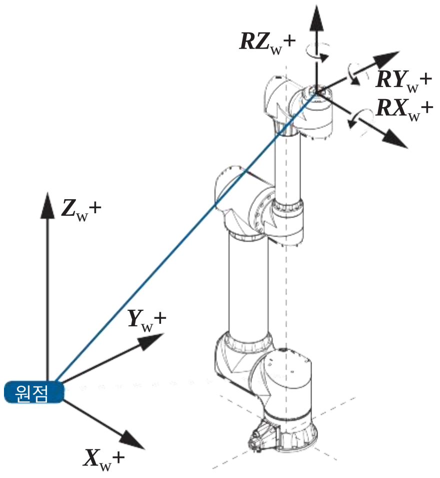
Set individual angles at the joints to determine the posture of the power-assisted lifting arm.
The joints operate independently.

Use when confirming visually with the teaching mode to determine the position or when restoring from an unmovable point.

Because the joint coordinate system moves the power-assisted lifting arm through independent operations of the axes, 'posture' is not used (the parameter posture)

☞ p. 4

World coordinate system



The orthogonal coordinate system that takes an arbitrary point in space to setup the power-assisted lifting arm as the origin, representing the orientation and position of the end effector.

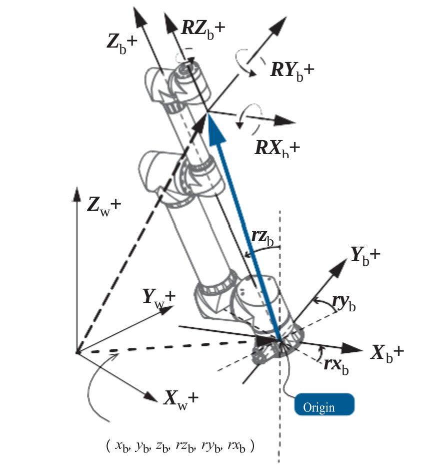
The standard absolute coordinate system in a multi-power-assisted lifting arm system.

The world coordinate system and the base coordinate system is setup in the same way on first setup.

☞ p. 5

1 Coordinate system

Base coordinate system

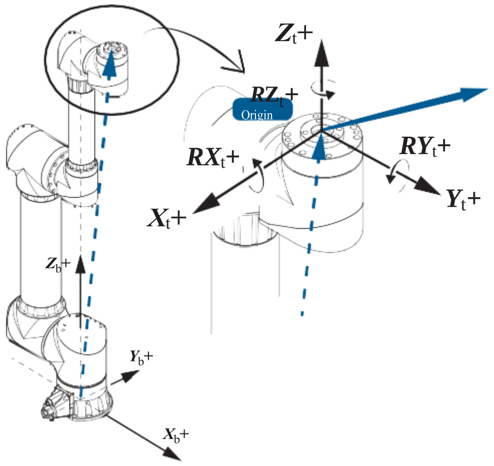


The orthogonal coordinate system that takes the center of the lower flange as the origin, representing the orientation and position of the upper flange.

The world coordinate system and the base coordinate system is setup in the same way on first setup.

☞ p. 6

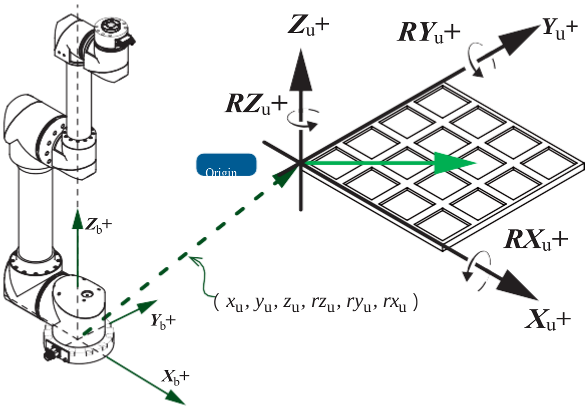
Tool coordinate system



The orthogonal coordinate system that takes the center of the upper flange as the origin. .

☞ p. 7

User coordinate system



The orthogonal coordinate system that applies the base coordinate system, taking a position with arbitrary offset as the origin. It is used during pallet operations.

☞ p. 8

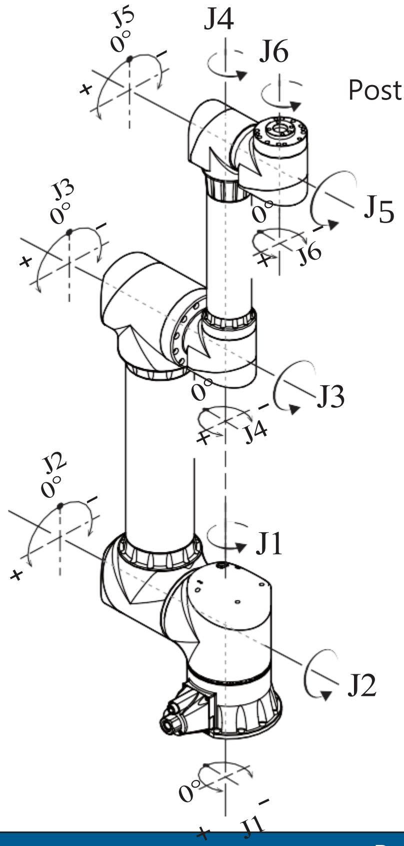
2. Joint coordinate system (standard coordinates)

1. Definition

Joint coordinate system is the coordinate system showing the angles of joints (J1, J2, J3, J4, J5, J6) .

Each joint and angle can be operated separately.

Do not use posture (parameter posture)



Posture at the left image (original point posture)

Axis	Angle
J1	0°
J2	0°
J3	0°
J4	0°
J5	0°
J6	0°

Functions	Parameters		
Standard coordinates	Joint coordinate system		
Current position information	Angles of axes (J1, J2, J3, J4, J5, J6		
Teaching data	200 points Take 10 teaching points as 1 pattern, can use up to 20 patterns（Position teaching points: pos1[0] ～ [9] … pos20[0] ～ [9])		
JOG operation	Maybe		
Direct Move	Commands move through teaching data in a concatenated form		
Hand Homing	Maybe		
Move between 2 points	Operation	Robot program	Teaching

	PTP	move( )
		reljntmove()
			Move To

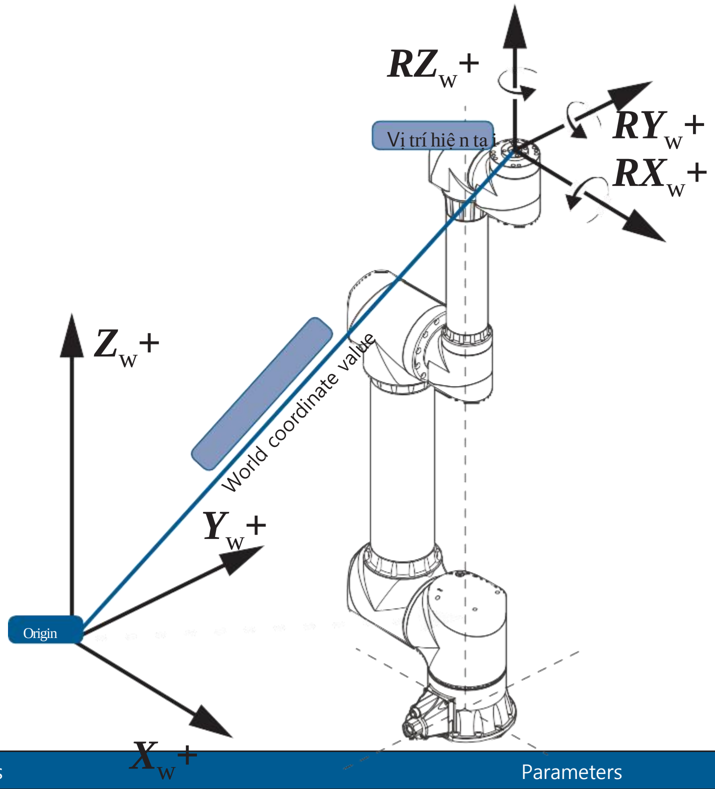
	Line （linear interpolation)	line( )	
	Optline (Optimal linear interpolation)	optline( )	Impossible

3. World coordinate system (standard coordinates)

1. Definition

World coordinate system is a direct coordinate system that takes an arbitrary point of the space where the power lift arm is installed as the origin point, representing the direction and position of the power lift arm above.

The world coordinate system and the base coordinate system are set the same during initial installation.



Functions	Parameters		
Standard coordinate system	World coordinate system		
Current position information	Coordinate value of the upper wing surface of the last segment in the Tool view from the origin of the world coordinate system (x, y, z, rz, ry, rx)		
Teaching data	200 points Take 10 teaching points as 1 pattern, can use up to 20 patterns（Position teaching points: pos1[0] ～ [9] ··· pos20[0] ～ [9])		
JOG operation	Maybe		
Direct Move	Command to move through teaching data of orthogonal coordinate system in world coordinate system		
Hand Homing	Impossible		
Move between 2 points	Operation	Robot program	Teaching
	PTP	move ( )	Move To
	Line （linear interpolation ）	line ( ) relline( )	
	Optline (Optimal linear interpolation)	optline( )	Impossible

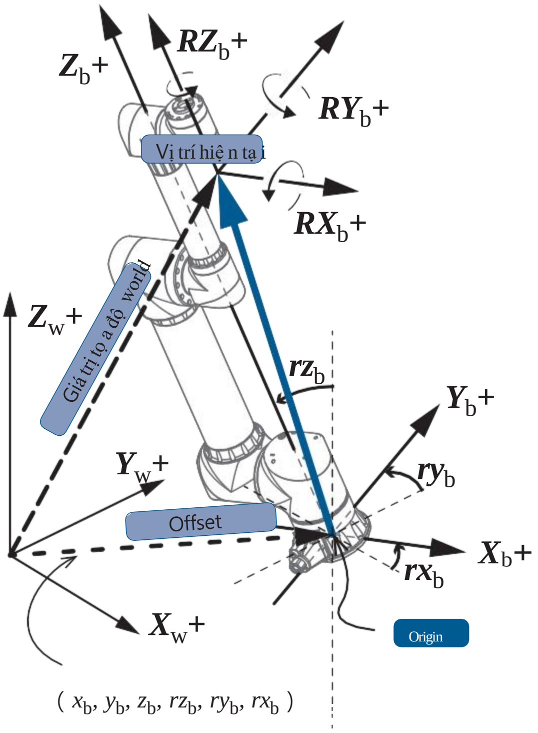
4. Base coordinate system

1. Definition

Base coordinate system is an orthogonal coordinate system that takes the center of gravity of the lower wing as the origin point. Base offset setting matches the power lift arm installation status from the world coordinate system.

Base offset setting is (xb, yb, zb, rzb, ryb, rxb) .

The offset of the initial setting is (xb, yb, zb, rzb, ryb, rxb)=(0, 0, 0, 0, 0, 0), the origin of the base coordinate system is the same as the origin of the world coordinate system.



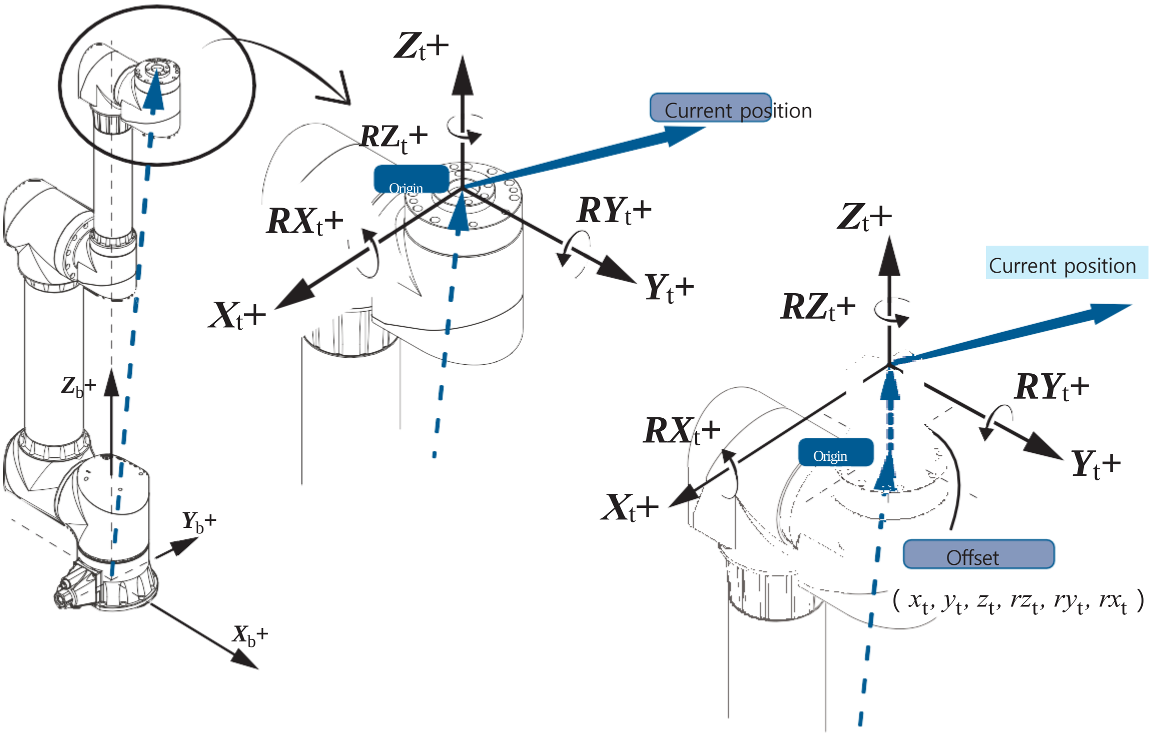
Functions	Parameters		
Standard coordinate system	World coordinate system		
Offset	Up to 3 can be installed Offset of initial setting is (xb, yb, zb, rzb, ryb, rxb) = (0, 0, 0, 0, 0, 0)		
Current position information	Coordinates of the upper flange face of the last section of the Tool except the base offset value from the world coordinate value (x, y, z, rz, ry, rx)		
Teaching data	200 points Take 10 teaching points as 1 pattern, can use up to 20 patterns（Position teaching points: pos1[0] ～ [9] … pos20[0] ～ [9]）		
JOG operation	Maybe		
Direct Move	Command to move through teaching data of orthogonal coordinate system in base coordinate system		
Hand Homing	Impossible		
Move between points	Operation	Robot program	Teaching
	PTP	move ( )	Move To
	Line （linear interpolation）	line ( ) relline( )	
	Optline (Optimal linear interpolation)	optline( )	Impossible

5. Tool coordinate system

1. Definition

Tool coordinate system is a coordinate system that takes the center of gravity of the upper wing as the origin point. Take the last paragraph of the Tool as a standard. Set the offset (xt, yt, zt, rzt, ryt, rxt) according to the installation tool..

The tool coordinate system has a different direction and world coordinate system (base), so please pay attention.

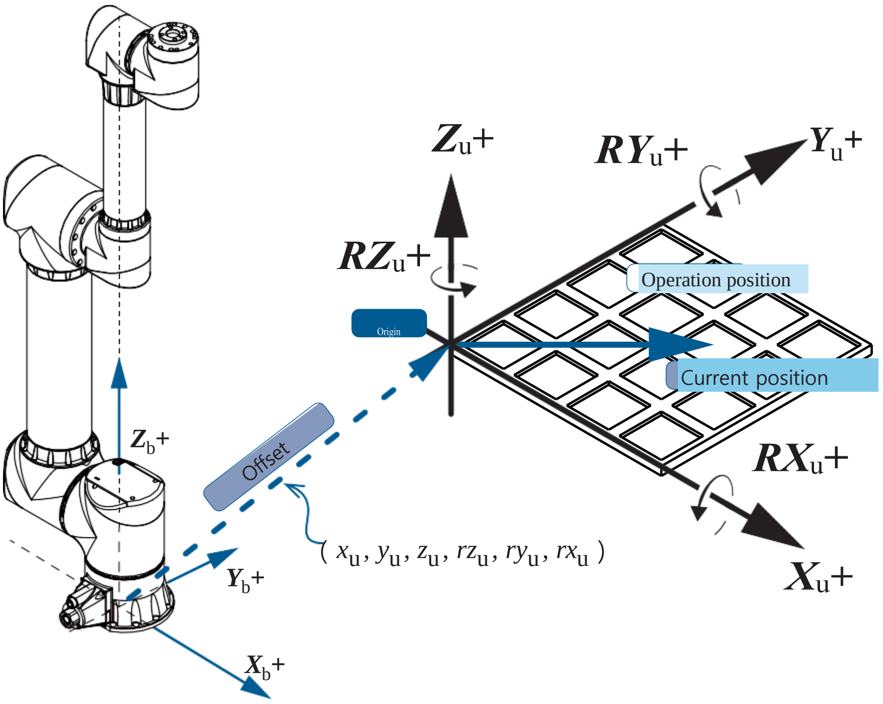


<table>
<tr><th>Functions</th><th colspan="3">Parameters</th></tr>
<tr><td>Standard coordinate system</td><td colspan="3">World coordinate system</td></tr>
<tr><td>Offset</td><td colspan="3">Can be set up to 8</td></tr>
<tr><td>Current position information</td><td colspan="3">Coordinate value viewed from the origin of the Tool coordinate system (x, y, z, rz, ry, rx)</td></tr>
<tr><td>Teaching data</td><td colspan="3">200 points
Take 10 teaching points as 1 pattern, can use up to 20 patterns（Position teaching points : pos1[0] ～ [9] ··· pos20[0] ～ [9])</td></tr>
<tr><td>JOG operation</td><td colspan="3">Maybe</td></tr>
<tr><td>Direct Move</td><td colspan="3">The command moves through the teaching data of the orthogonal coordinate system in the Tool coordinate system</td></tr>
<tr><td>Hand Homing</td><td colspan="3">Impossible</td></tr>
<tr><td rowspan="3">Move between 2 points</td><td>Operation</td><td>Robot program</td><td>Teaching</td></tr>
<tr><td>PTP</td><td>move ( )</td><td rowspan="2">Move To</td></tr>
<tr><td>Line （linear interpolation)</td><td>line ( )
relline( )</td></tr>
</table>

6. User coordinate system

1. Definition

User coordinate system is a coordinate system defined for the user to conveniently operate, suitable for the operating location. The power lift arm can be operated depending on the pallet..

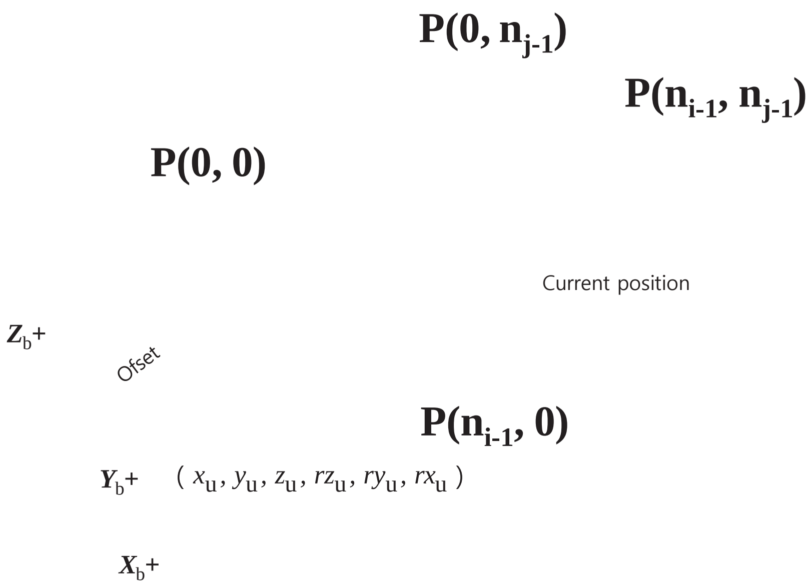


<table>
<tr><th>Functions</th><th colspan="3">Parameters</th></tr>
<tr><td>Standard coordinate system</td><td colspan="3">World coordinate system</td></tr>
<tr><td>Offset</td><td colspan="3">Up to 3 can be installed</td></tr>
<tr><td>Current position information</td><td colspan="3">Coordinates viewed from the origin of a user coordinate system except the user offset value from the base coordinate value (x, y, z, rz, ry, rx)</td></tr>
<tr><td>Teaching data</td><td colspan="3">200 points
Take 10 teaching points as 1 pattern, you can use up to 20 patterns (Position teaching points): pos1[0] ~ [9] ... pos20[0] ~ [9])</td></tr>
<tr><td>JOG operation</td><td colspan="3">Maybe</td></tr>
<tr><td>Direct Move</td><td colspan="3">Command to move through teaching data of an orthogonal coordinate system in the user coordinate system</td></tr>
<tr><td>Hand Homing</td><td colspan="3">Impossible</td></tr>
<tr><td rowspan="3">Move between 2 points</td><td>Operation</td><td>Robot program</td><td>Teaching</td></tr>
<tr><td>PTP</td><td>move ( )</td><td rowspan="2">Move To</td></tr>
<tr><td>Line (linear interpolation)</td><td>line ( )
relline( )</td></tr>
</table>

2. Pallet calculation feature

For finished square pallets in lines n x row nj, automatically calculate the coordinates of the pallet cells (x, y, z, rz, ry, rx) by specifying the quantity Pallet cells and user coordinate values of 4 corners (x, y, z, rz, ry, rx).

The power lift arm operates based on the pallet coordinates P(i, j) (0<=i<= ni-1、0<=j<=nj-1) .
Teaching at least 3 points m P(0, 0)、P(ni-1, 0)、P(0, nj-1) . The 4th teaching point P(ni-1, nj-1) is used to compensate for the shape error of the pallet used..



포인트 !

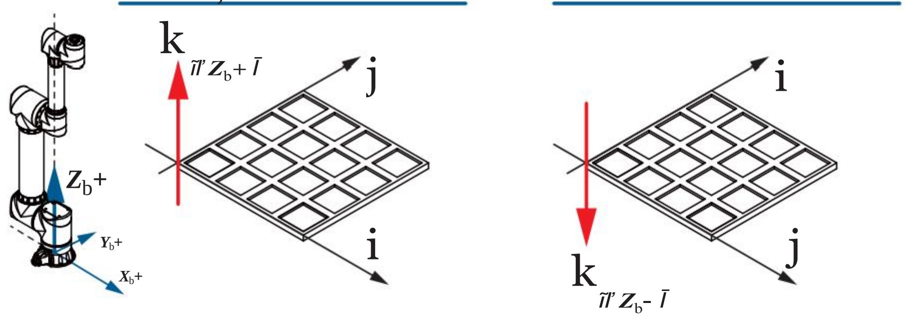
The forward direction of the pallet vertical direction is changed according to the order of the teaching point

dụ ① i × j → For a flat surface (pallet face), it becomes a vertical vector +z.

ví dụ ② j × i : to -z of the opposite direction from example 1

Example: ①x i→ j

Example ②x j→ i



7. Posture

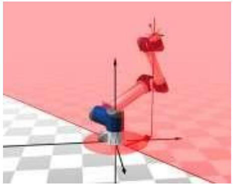
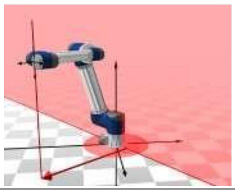
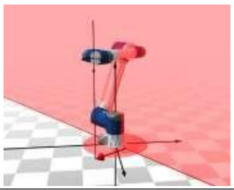
C Teaching

1. Posture

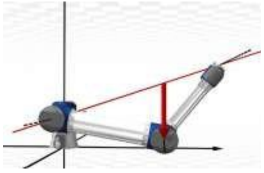
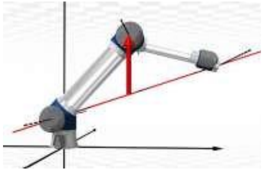
Power lifts have many postures that are determined by (1) arm position and (2) joint angle. Define a total of 8 postures through 3 parameters: shoulder, elbow and wrist..

By switching posture on the operation program, points that cannot be manipulated and points that cannot be moved can be avoided.

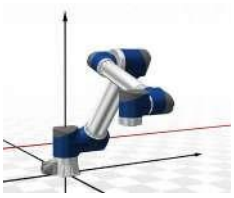
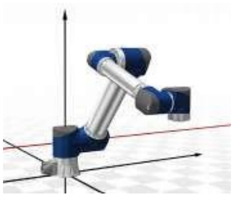
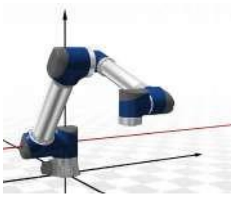
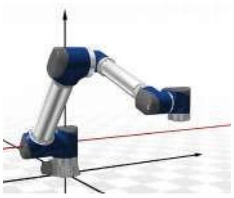
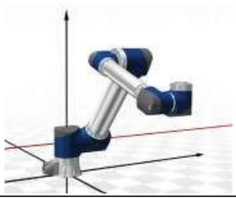
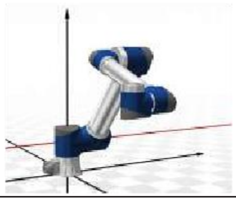
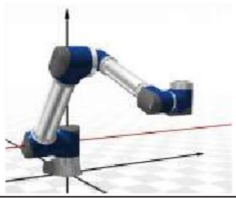
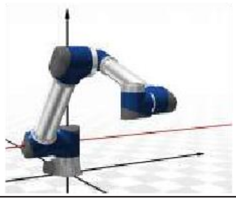
Parameter	Explanation
shoulder (bit0)	Look at the power lift arm from a position where arm 1 is visible to the left of the J1 swivel shaft. If the main part of the upper wing is in front of the plane formed by axis J1 and J2 then it is "1", if it is on the opposite side then is "0".".

bit 0	0	1	
Picture			

Parameter	Explanation
Elbow (bit1)	If the Z coordinate of axis J3 is above the line connecting J2 and J5, it is "1", if it is below, it is "0".

bit1	0	1
Picture		

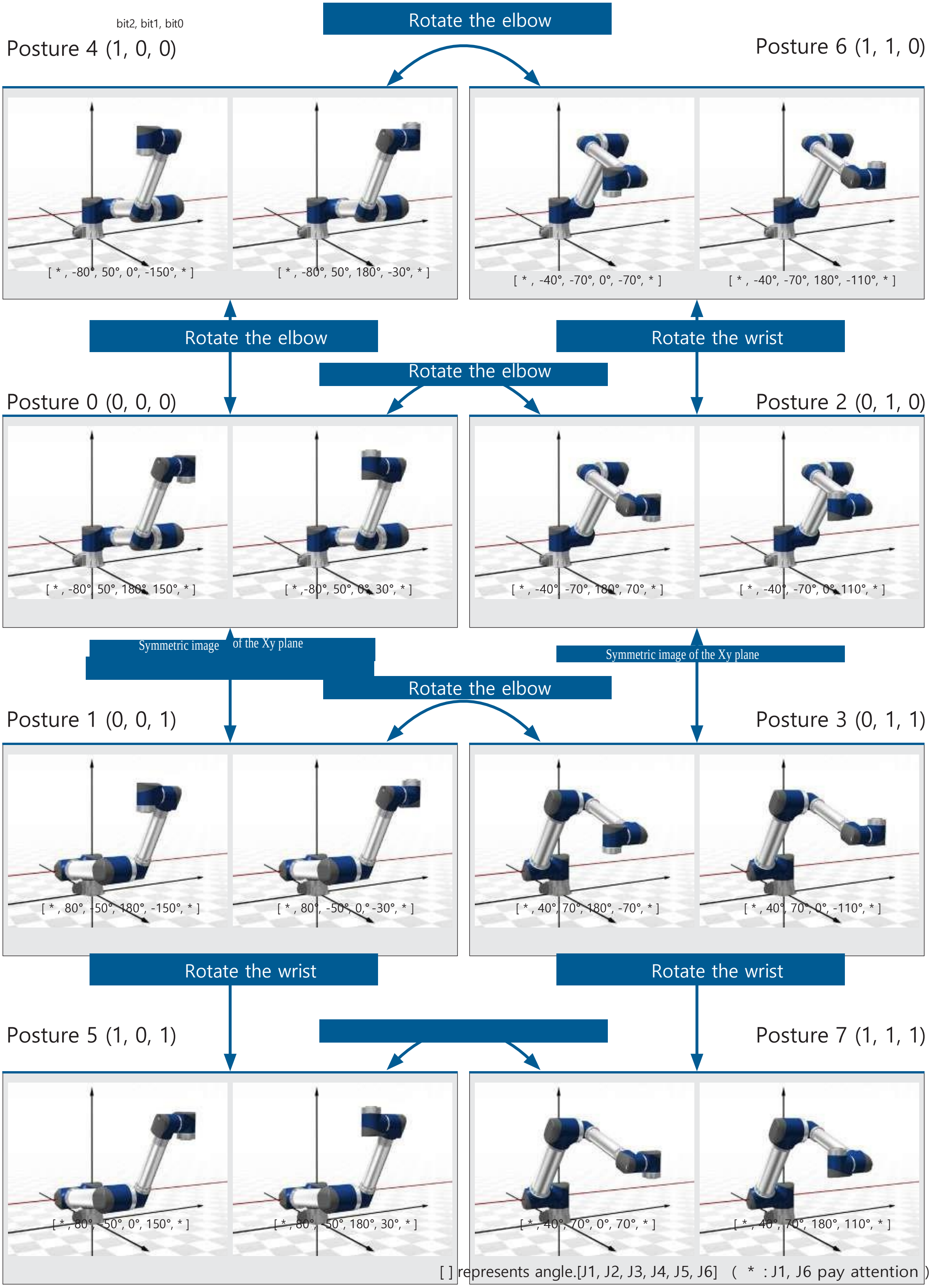
Parameter	Explanation
Wrist (bit2)	Determined by the angle of J5 and bit0. .

bit0	0		1	
J5 angle	-180° ~ 0°	0° ~ 180°	-180° ~ 0°	0° ~ 180°
bit2	1	0	0	1
Pictures	In case the top flange face is facing down When the joint unit of the last segment inserts arm 2 and is on the opposite side of arm 1, "bit2=1", when on the same side, "bit2=0". (The photo below is an example)			
				
	In case the upper wing face is facing up When the joint unit of the last segment inserts arm 2 and is on the opposite side of arm 1, "bit2=0", when on the same side, "bit2=1". (The photo below is an example)			
				

【Calculation formula】: $\text{Posture} = \underbrace{(4 \times \text{bit }2)}_{\text{wrist}} + \underbrace{(2 \times \text{bit }1)}_{\text{elbow}} + \underbrace{\text{bit }0}_{\text{shoulder}}$

【Setting range】: 0 – 7 (8 types)

2. Coordinate system image (reference)



[ ] represents angle.[J1, J2, J3, J4, J5, J6] (* : J1, J6 pay attention)

3. Range of manipulation and posture

The power lift arm has a point where it cannot move on the structure.

To avoid the immovable point, use it in the recommended posture below, above the mounting surface to satisfy conditions 1 and 2._

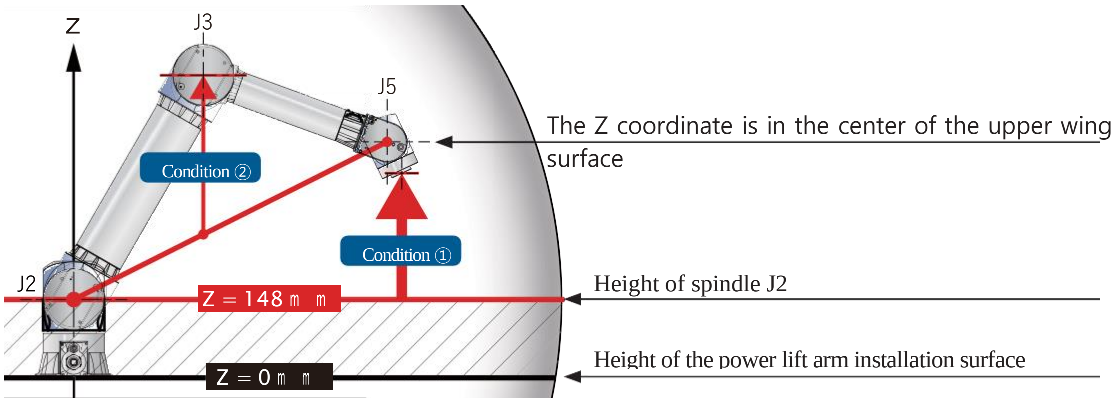
Condition ① : Range

- The range where the Z coordinate of the center of the upper flange surface Z>0, is not related to the angle of J5

Recommended ~~range: Z~~>148（is the range above the height of the rotation axis of J2.)

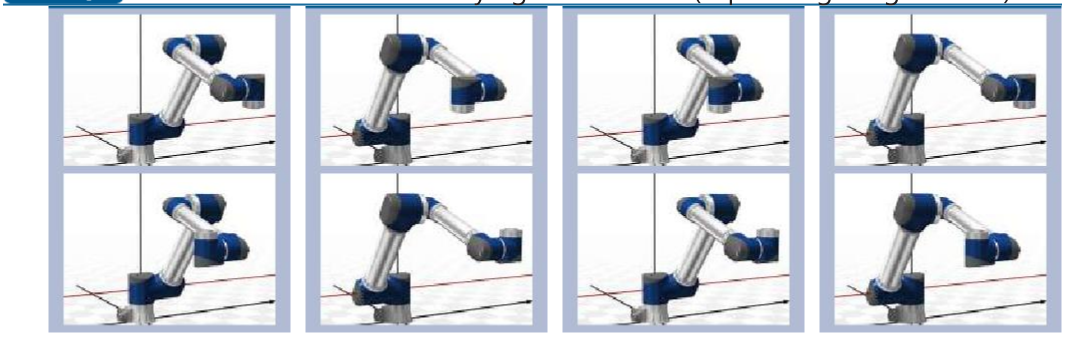
Condition n ② : Posture

- Posture where the coordinates of the rotation axis of J3 are above the lines connecting J2 and J5


Z
J3
J5
The Z coordinate is in the center of the upper wing surface
Condition ②
Condition ①
J2
Z = 148ｍ ｍ
Height of spindle J2
Z = 0ｍ ｍ
Height of the power lift arm installation surface

Posture Posture 2 Posture 3 Posture 6 Posture 7

Condition ② Posture recommends satisfying condition 2 (operating range : Z>0)



Supplement

If the Z coordinate of the upper wing is close to 0, then even if J5 is rotated in a fixed state, J2 and J3 satisfy condition 2, there is a point where the Z coordinate of the wing is above to Z<0.

Please ask the service desk in case of using operating range Z<0.

Important: The point of no movement exists near the position change point of the power lift arm.
For example, even if the angle of J5 satisfying condition 1 and condition 2 is near 0 deg or 180 deg (range is about ± 5 deg), it is still the same.
There are cases where a point that cannot be moved arises.

D SOFTWARE

1. Programming instructions
2. Robot Library
3. Mind map
4. Steps to perform the program

NOTES

D SOFTWARE

1 Programming instructions

⚠ ATTENTION

Complete the check before starting automatic operation.
First, operate the robot at low speed to check whether the motion chain operates safely, then slowly increase the operating speed and check the operation.

Robot motion programs are written in Python language.

1.PC

The following equipment is required to use this product. Please refer to this manual and safety instructions to create the operating system. The software may not function in operating environments other than recommended specifications.

Specifications		
Personal computer (PC)	OS	WindowsR 10 (32bit /64bit) WindowsR 8/8.1 (32bit /64bit) WindowsR 7 (32bit /64bit)
	Languages	Korean, English, Japanese
	CPU	32bit or 64bit processor with speed of 1GHz or higher
	Memory	RAM 1 gigabyte (GB) (32 bit) or RAM 2 GB (64 bit)
	Hard drive capacity	Requires 512 MB or more of space
	Communication function	Wired LAN port (recommended) USB port (*) (if there is no wired LAN port
Screen	Resolution	1366 × 768 pixel or higher
	Color	24 bit color (TrueColor) or higher

*) Separate USB Ethernet adapter required. (Recommended product: LUA3-U2-ATX from Buffalo)

2. Necessary software

python

Python

Compliant with Python2.7.
For Python language and specifications, please refer to general reference books or specialized books.

notepad.exe

Text editor

Python robot programs are written by using a text editor. (Recommended: VSCode) Character codes are UTF-8 and line breaks are LF.

ttermpro.exe

Terminal software Tera Term ）

Control the controller by executing the operating program via Telnet

FFFTP.exe

FTP client software (FFFTP)

Transfer files between PC and controller.

Google Chrome

Web browser（ Google Chrome ）

Instructions are performed on a web browser (Google Chrome). Install on the PC you will be teaching on.
(Google Chrome: 61 and up)

- Microsoft® and Windows® operating systems are trademarks of Microsoft Corporation and its affiliates in the United States.
- “Python” and the Python logo are trademarks of the Python Software Foundation or registered trademarks.
- Google Chrome is a registered trademark of Google Inc..
- Tera Term is the work of Takashi Teranishi and the Tera Term Project. Tera Term is free software distributed under the BSD license.
- FFFTP is a product of Jun Sota and the FFFTP Project.
- FFFTP is free software distributed under the BSD license.
- Copyright of the sample program described in the document belongs to Zeus Company Limited.

This chapter briefly explains typical usage examples of modules, methods, and functions. Select from "Entire process" or "Motion model".

For details on modules and methods, please refer to " Robot Library" �

Search through the entire process

☞ Page 5

Description of the entire program of activities.
Each step of "Initial setting", "Teaching point setting", "Operating condition setting", "Operation definition" and "Ending is explained in detail.

Search in "Motion Models"

☞ Page 6

From the actual operating model, we propose an operating program suitable for your purposes. The operating program using the "Pallet function" is described in "Basic Operations".

Python robot programs distinguish between uppercase and lowercase letters

Search in "Full progress "

Set the robot's dynamic speed and acceleration time.
Motion conditions are optional items, but if omitted,
it will be set as the default value.

Select the desired program block “1. Write a robot program"

▼

Complete the robot program

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

## 1. Initial setting ① #################
    ........
    ........

## 2. Initial setting② #################
    ........


    ........

## 3. Set teaching points ###
    ........


    ........

## 4. Set operating conditions ############
    ........


    ........

## 5. Definition of robot motion ######
    ........


    ........

## 6. End ###################
    ........
```

Block of Program

1. Initial setting (1) ☞ Page 8

Specify the path and character code for the Python interpreter. Import module that uses robots library.

2. Initial setting (2) ☞ Page 9

Create object.
If using an override function to limit the robot's movement speed, define it here.

3. Set the teaching point ☞ Page 10

Set and adjust teaching points.
Saved teaching data can be read and used. If you want to use the Pallet function, specify it here.

4. Set the operation condition ☞ Page 15

Set the robot's movement speed and acceleration time. Motion conditions are optional items, but if omitted, it will be set as the default value.

5. Definition of robot motion ☞ Page 16

Set the robot's movements.
There are anti-cross functions, linked manipulation of I/O input signals and conversion of the coordinate system in

6. End ☞ Page 26

End of robot program

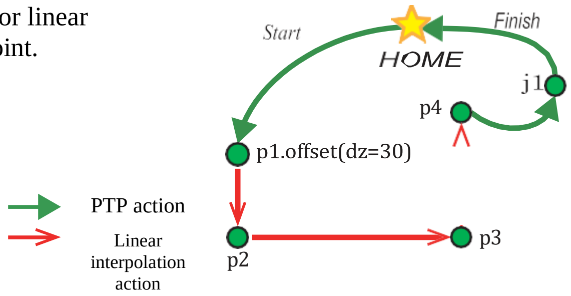
Search in “Motion Models”

Select desired movement pattern "2. Sample program"

Model 1 : Default activity

Page 27

Set the guide point and perform PTP movement or linear interpolation movement towards the specified point.



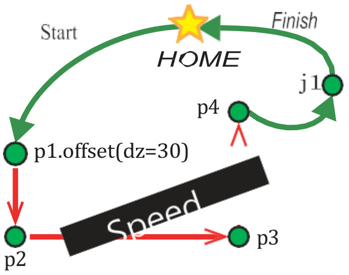
Model 2 : overwrite

Page 28

In the basic motion above, the limit placed on the speed of motion is set by using MotionParam()..

Speed limit operates in percentage (%) set in override mode.

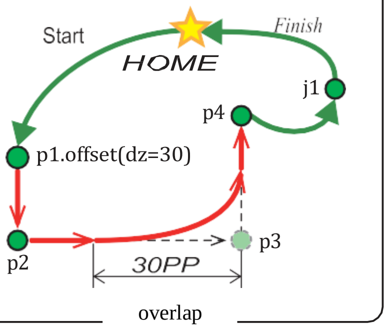
Used when checking the operation of this operating program,.



Model 3 : overlapping

Page 29

At the moment of approaching the target teaching point, the subsequent movement will be overlapped.
The Robot can be moved by performing the following operations without waiting for the completion of the Robot's movement, passing through prepared reference points to avoid obstacles.
The amount of overlap can be set arbitrarily. (Set to 30 mm in the example on the right)

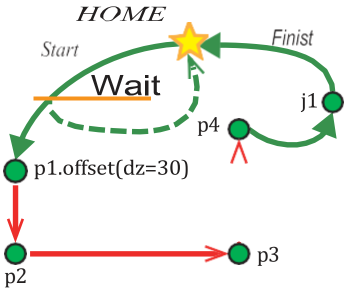


Model 4 : Wait for I/O input

☞ Page 30

The robot's movements are controlled by I/O signals input to the controller from an external device..

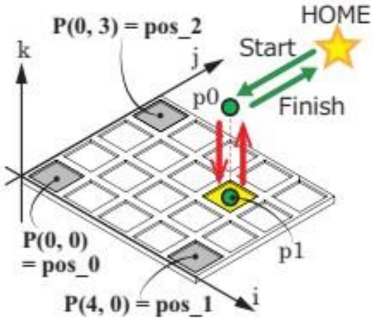
If I/O input is used, it will be possible to start a Robot program pre-registered in the controller via I/O.



Model 5 : Pallet function

☞ Page 31

After determining the number of transfer pallet cells and the coordinates of the four points, the controller will automatically calculate the coordinates of each cell. Set the calculated Pallet coordinates (i, j) as the teaching point.



1. Initial setting (1)

Import modules

Install Korean language processing and specify the Python interpreter path

Using full-width characters without specifying a character code that may cause errors.

Example about the program

```
#!/usr/bin/python
# -*-coding: utf-8 -*-
```

Thông dịch Interpreter

Mã văn bản Text code

Import modules

We can import various modules (standard libraries, robot libraries, customer-made modules) and use the necessary commands to control the robot.

Modules	Functions
i611_MCS	Use the basic functions needed to control the robot
teachdata	Use the teaching data
i611_extend	Using extension functions (Pallet functions)
rbsys	Use administration program
i611_common	Exception handling in methods of class i611Robot (*)
i611_io	Control I/O signals
i611shm	Access shared memory

```
## 1 . Initial setting ①    Import modules ######################
        from i611_MCS import *
        from teachdata import *

        from i611_extend import *

        from rbsys import *

        fromi611_commonimport*

        from i611_io import*

        from i611shm import *
```

*) The Exception class can be imported and used in the i611_MCS module. Loading from i611_common import * in module i611_MCS.

2 Programming instructions

2. Initial setting (2)

Create objects

Robot constructor

Create robot object.

```
    # Robot i611 constructor
        rb = i611Robot( )
```

Definition of world coordinate system

Setting when using world coordinate system.

```
        # Definition of world coordinate system
        _BASE = Base( )
```

Start connecting to the robot, initialize

```
        # Start connecting to the robot, initialize
        rb.open( True )
```

Initialize I/O input/output functions

```
        # Initialize I/O input/output functions (can be omitted when not using I/O)
        IOinit( rb )
```

Overwrite

Set the ratio (%) of operation speed for PTP operation and joint operation.

```
         #Override speed 50%
          rb.override(50)
```

```
    ## 2 . Initial settings②: Set operating conditions ###################################  #
        Robot i611 constructor
            rb = i611Robot( )
        # Definition of world coordinate system
            _BASE = Base( )
        # Start connecting to the robot, initialize
            rb.open( True )
        # Initialize I/O input/output functions (can be omitted when I/O is not used)
             IOinit( rb )
        #Override speed 50%
              rb.override(50)
```

3. Set the teaching points

Definition of teaching point

Position( ) Create an entity that defines the teaching point in the world coordinate system.

Argument	Significance	Variable state	Unit
x, y, z	Position (Descartes coordinate system)	float	mm
rz, ry, rx	Pose (Euler angle based on Z-Y-X)	float	deg
parent	Set the use of the world coordinate system	float	-
posture	Pose	integer	-
multiturn	Cross counter information	long	-

Joint( ) Creates an entity that defines the teaching point of the joint coordinate system.

Argument	Significance	Variable state	Unit
j1, j2, j3, j4, j5, j6	Joint type data for each axis Original value [0.0, 0.0, 0.0, 0.0, 0.0, 0.0] ）	float	deg

```
##３． Set the teaching points ##################################
            p1 = Position( -50, -250, 350, 90, 0, 180 )
            p2 = Position( -300, -250, 350, 90, 0, 180 )
            p3 = Position( -50, -250, 350, 90, 0, 180 )

            j1 = Joint( 10, 30, 10, 0, 5, 30 )
```

The teaching points are set as Position Type or Joint Type.

Omit arguments

Enter arguments to the parameters you want to specify.

(Example)

If the argument after rz is omitted and p = Position (x, y, z), then the parameters after rz are set to their original values.

Adjust the teaching points

replace( )

Replace the world coordinate system object.
(Update the original object)

Argument	Significance	Variable state	Unit
x, y, z	Position (world coordinate system) (original value = 0,0)	float	mm
rz, ry, rx	Pose (angle Z-Y-X Euler) (original value = 0,0)	float	deg
parent	Set the use of world coordinate system	float	-

```
# change of pl value and self-updated
  pl = Position (-50, -250, 350, 90, 0, 180)
  pl.replace( x=100, rz-50 )
```

[100, -250, 350, -50, 0, 180]

shift( )

Moves objects in world coordinates.
(Update the original object)

Argument	Significance	Variable state	Unit
dx, dy, dz	Position (world coordinate system)	float	mm
drz, dry, drx	Pose (Euler angle based on Z-Y-X)	float	deg

```
 # Move pl value and self-updated

 p1.shift( dx=10 )
pl =Position( -50, -250, 350, 90, 0, 180 )
```

Kết quả

[-40, -250, 350, 90, 0, 180)] Result

offset( )

Add the offset coordinate value to the Position coordinate value.
(Create a new object while maintaining the original object)

Argument	Significance	Variable state	Unit
dx, dy, dz	Offset amount of position (Descartes coordinate system)	float	mm
drz, dry, drx	Pose Offset (Euler angle based on Z-Y-X)	float	deg

```
# Create p2 offset while maintaining pl
  pl = Position( -50, -250, 350, 90, 0, 180 )

     p2 = p1.offset( dx=10 )
```

Result

p1 = [-50, -250, 350, 90, 0, 180]
p2 = [-40, -250, 350, 90, 0, 180]

Use teaching data saved in a file

Teachdata( )

Read the teaching data and create an instance of the Teachdata class

Argument	Significance	Variable state	Unit
fname	File name of Teaching data	string	-

get_position( )

Enter teaching data Position coordinate value.

Argument	Significance	Variable state	Unit
key	Position coordinate key name Need	string	-
index	Index of position coordinate Need	integer	-
tool	tool ID collection flag	bool	-
base	base ID collection flag	bool	-
comment	Flag when receiving Comment	bool	-

get_joint( )

Enter teaching data Joint coordinate value.

Argument	Significance	Variable state	Unit
key	Joint coordinate key name Need	string	-
index	Joint coordinate index Need	integer	-
comment	Comment flag	bool	-

```
# Read teaching data files
data = Teachdata( "teach_data" )                    )
# Read teaching points
p1 = data.get_position( "pos1", 0 )
j1 = data.get_joint( "joint1", 0 )
```

"pos1" Load position type data at index [0]

"joint1" Load joint type data at index [0]

get_param()

Enter the parameters of teaching data

Argument	Significance	Variable state	Unit
key	Key name of the parameter Need	string	-
index	Parameter Index Need	integer	-
axis	Axis number in parameter Need	integer	-
comment	Parameter's Comment collection flag	bool	-

Use Pallet functions

init_3() Identify Pallet (3 teaching points)

Argument	Significance	Variable state	Unit
pos_0	[Position] Teaching point in Pallet (origin) Need	float	-
pos_i	[Position] Teaching point on Pallet (direction i) Need	float	-
pos_j	[Position] Teaching point on pallet (direction j) Need	float	-
ni	Number of cells in direction i of the pallet Need	integer	-
nj	Number of cells in direction j of the pallet Need	integer	-

```
# Identify Pallet by 3-point teaching data
pal = Pallet()
pal.init_3( pos_0, pos_1, pos_2, 5, 4 )
```

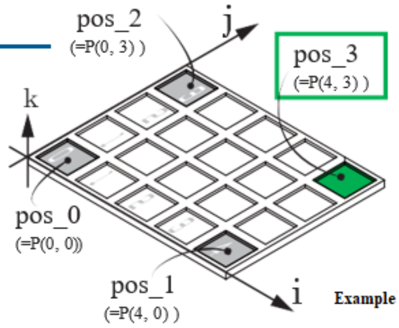
init_4() Identify Pallet (4-point teaching data)

Argument	Significance	Variable state	Unit
pos_0	[Position] Teaching point in Pallet (origin) Need	float	-
pos_i	[Position] Teaching point on pallet (direction i) Need	float	-
pos_j	[Position] Teaching point on pallet (direction j) Need	float	-
pos_ij	[Position] Teaching point on pallet Need	float	-
ni	Number of cells in direction i of the pallet Need	integer	-
nj	Number of cells in direction j of the pallet Need	integer	-

```
# Identify pallet by 4-point teaching data
pal = Pallet()
pal.init_4( pos_0, pos_1, pos_2, pos_3, 5, 4 )
```

Difference between init_3() and init_4()

Correct the Pallet form errors installing the 4th teaching point for use. Robot can be moved more exactly than init_3 ()



Set by init_4()
This is the teaching point to correct the Pallet form

Example of Pallet at size: 5x4

Use Pallet functions

get_pos() Enter the Cell Position

Argument	Significance		Variable state	Unit
i	Index determines cell position in Pallet (i direction)	Need	integer	-
j	Index determines cell position in Pallet (j direction)	Need	integer	-
dk	Offset in vertical direction (defaults to 0 if omitted)		integer	mm

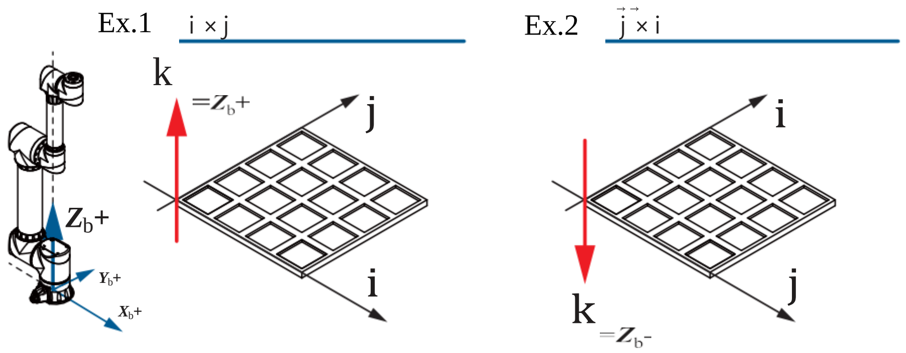
adjust() Adjust the Pallet cell

Argument	Significance		Variable state	Unit
i	Index determines cell position in Pallet (i direction)	Need	integer	-
j	Index determines cell position in Pallet (j direction)	Need	integer	-
di	Offset amount of direction cell position i	Need	integer	mm
dj	Offset amount of direction cell position j	Need	integer	mm

Vertical direction of pallets

The vertical direction of the pallet (+k direction) changes depending on the position of the guide point.

Example ① i × j: Vector z+ points up perpendicular to the pallet plane. Example ② j × i: Vector z - perpendicular to the bottom of the Pallet plane.



4. Set the operation conditions

Set the robot's motion parameters

MotionParam( ) Create an instance of the robot's motion parameters class.

motionparam() Set the operation parameters

Argument	Significance	Variable state	Unit
lin_speed	Speed (Line operation (linear interpolation operation)) Original value：5.0	float	mm/s
jnt_speed	Speed (PTP operation, general operation, optimal linear interpolation operation）Original value：5.0	float	%
acctime	Acceleration time Original value：0.4	float	s
dacctime	Deceleration time Original value：0.4	float	s
posture	Pose Original value：2	integer	–
passm	Line activities Original value：2	integer	–
overlap	Overlapping activities Original value：0.0	float	mm
zone	Complete range of positioning Original value：100	integer	pulse
pose_speed	Speed (pose interpolated motion） Original value：20	float	%
ik_solver_option	Direction of rotation Original value 0x11111111	long	–

If the argument is omitted, the default value will be reset

```
## 4                                                        #
# Set the motion condition in the Motion Param

m = MotionParam( jnt_speed=10, lin_speed=70 )
#  Set the operating condition in MotionParam format
rb.motionparam( m)
```

5. Definition of robot motion

Motion

Linear interpolated motion moves at a constant speed.

home()	Move all axes to Odeg joint coordinates.
move( )	PTP moves at a constant speed. (*)
line( )	Linear interpolated motion moves at a constant speed.
optline()	.

*) As soon as the method is executed, it will work with the motion parameters set in the motionparam method. In case an action parameter is given in the eyebrow method, the next action will be changed.

```
##5. Set up robot movements ##################################
    # Move to each coordinate axis (0,0,0,0,0,0)
    rb.home()
    # Movement
    rb.move( p1.offset(dz=30) )
    rb.line( p2, p3, p4 )
    rb.move( j1 )
      # Move to each coordinate axis (0,0,0,0,0,0)
    rb.move(j1)
```

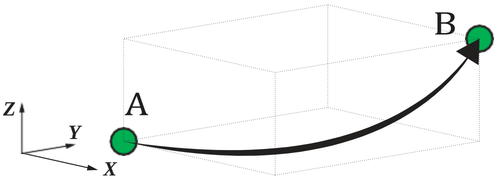
Move the offset coordinates by dz = 30 relative to p1

Move to p2, p3, p4 in the Line action

Move to j1 using PTP operation

PTP operation, linear interpolation operation and optimal linear interpolation operation

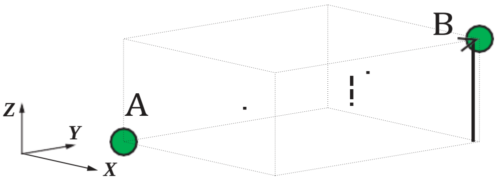
PTP operation (move())



All joints move at a constant speed and angle toward the target coordinates. Motion moves in a smooth curve.

Linear interpolation operation(line())

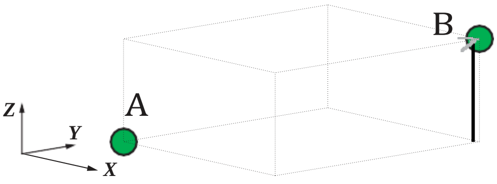
Is the act of moving at a constant speed so that the trajectory goes straight to the destination while simultaneously controlling the X-Y-Z axes.



Optimal linear interpolation operation (optline())

Is the operation of shifting gears and moving at the optimal speed so that the trajectory go straight to the destination while simultaneously controlling the X-Y-Z axes

Speed is indicated in %



Use offset tools

settool() Set tool Offset

Argument	Significance	Variable state	Unit
id	Number of tools Need 0 : Turn off tool Offset 1 - 8 : Choose tool Offset	integer	-
offx	Offset the X-axis tool in the tool coordinate system	float	mm
offy	Offset the Y-axis tool in the tool coordinate system	float	mm
offz	Offset the Z-axis tool in the tool coordinate system	float	mm
offrz	Offset around the Rz axis in the tool coordinate system	float	deg
offry	Offset around the Ry axis in the tool coordinate system	float	deg
offrx	Offset around the Rx axis in the tool coordinate system	float	deg

changetool() Choose tool Offset.

Argument	Significance	Variable state	Unit
tid	Number of tools Need 0 : Turn off tool Offset 1 - 8 : Install tool Offset	integer	-

```
# Number of tools = 1 Register tools
rb.settool( 1, 0.0, 0.0, 137.0, 0.0, 0.0, 0.0 )

# Tools #Change into 1
rb.changetool( 1 )
```

Argument names for tool numbers vary across methods Change tool() and set tool().

Methods	Argument name for tool number
changetool()	tid
settool()	id

I/O input and output

din() Enter I/O

Argument	Significance	Variable state	Unit
*adr	Input port ・When specifying an input port, adr: number of input ports ☐ When reading multiple input ports simultaneously, adr [0]: Number of input ports (start) adr [1]: Number of input ports (End)	string	-

```
# Example 1: Specifying port 15
if din ( 15 ) == '1':
# Example 2: Specifying port 8 and 10
if din ( 8, 10 )[0] == '1':  # When specifying port 10
   . . .
elif din( 8, 10 )[1] == '1':  # When specifying port 9
. . .
elif din( 8, 10 )[2] == '1':  # When specifying port 8
```

dout() I/O Output

Argument	Significance	Variable state	Unit
adr	Output number, number starting addres Need (Setting range: 16 ～ 31)	integer	-
data	Output data from I/O Need Set as bit field in string. '1' = ON '0' = OFF (original value) '*'= constantly	string	-

```
# Specify ON/OFF start address and output data
dout( 16, '11111' )
```

For more information about port numbers, see "Memory Map ".

Set the gate from the bit field

The data part of the doubt(), layout(), shootOut(), wait() methods is a string in bitfield format.

Example) Install output port 16 - 31

```
dout(16, "10001010****1111" )
```

Install port 31 on the port

16 to 1

Set port 16 to 1

Do not change port 20 to 23.

Wait for I/O input

wait( ) Wait until the specified I/O input pattern is reached.

Argument	Significance	Variable state	Unit
adr	Starting number of the input port Need	integer	-
data	Specify data to wait for input "1" = ON "0" = OFF Need	string	-
tm	Limited time Need	float, integer	s

```
# Example 1: List
    if wait( 8, '1', 10 )[0] == 1:
    if wait( 9, '1', 10 )[1] == '1':
    if wait( 9, '1', 10 )[2] > 10:

# Example 2: Keywords
    if wait( adr=1, data='1', tm=10 ) == 1:
```

⚠Attention

🚫	Do not change the reserved ports in init.py.	⚠ (Incident)

Start the robot program according to I/O input

The pre-registered Robot program can be started according to the I/O input of the controller.

File name and position of the setup script

File name	init.py
Save location	/home/i611usr

Do not change the file name and location. The controller includes an example setup script.
Download to PC, modify as desired and use.

Reapply power to the controller and reboot the system.
(When the system boots, "init.py" is executed and the conditions for starting I/O are set.)
· Make sure the jumper connector is connected to the Safety connector.

▪ When the 7 segments indicator appears, press the on switch

Turn on the input signal to the set I/O port.
(Starts execution of the Robot program on the rising edge of the signal.)

```
#!/usr/bin/python

# -*- coding: utf-8 -*-
 from rbsys import Robsys
 if   name     == '    main    ':
         rbs = RobSys()
#This is sample program for initial settings.
         rbs.open()

   —      —     —     —

    #Default assignment
     rbs.assign_din( run=0, stop=1, err_reset=2, pause=3 )

     rbs.assign_dout(running=16,svon=17,emo=18,hw_error=19,
                    sw_error=20,abs_lost=21,in_pause=22,error=23)

    #rbs.set_robtask(  "sample.py"  )

     rbs.close()
```

The filename of the program you want to start
(File names are random)

Port	State name	Command
0	run	Run robot program
1	stop	Slow down to stop
2	err_reset	Reset error
3	pause	Pause

Port	State name	System states
16	running	Robot program state
17	svon	Servo state
18	emo	Emergency stop condition
19	hw_error	System-determined error condition (critical)
20	sw_error	Error state is determined by the system
21	abs_lost	Loss of ABS
22	in_pause	Pause state
23	error	System error status

Commands and system status can be assigned to any I/O port

.

DANGEROUS

Do not use the output signal for safety-critical purposes.
Processed only through software, it cannot guarantee the reliability needed for safety circuits.

The initial installation port is defined in init.py

Signal and port number		Significance
Input	run=0	Run robot program
	stop=1	Slow down to stop
	err_reset=2	Reset errors
	pause=3	Pause
Output	running=16	Robot program status
	svon=17	Servo status
	emo=18	Emergency stop condition
	hw_error=19	System-determined error condition (critical)
	sw_error=20	The error state is determined by the system
	abs_lost=21	Loss of ABS
	in_pause=22	Pause state
	error=23	System error status (*

*) A system-determined error (non-fatal or critical) occurred.).

2 Used to check error status with a control line.

ATTENTION

Before pressing the trigger switch, make sure there are no obstacles within the operating range of the manipulator and ensure safety around the area..

Additional

・When the robot program ends due to an error:

.... The program cannot be restarted until the error is reset.

· The function outputs the status “Program running” and “An error occurred” to I/O:

.... Only valid when the robot program is started from the I/O input.

...........(If you start through a PC terminal emulation program, it will not work)

Convert coordinates

Joint2Position( )

Converts joint coordinate values to position coordinate values.

Argument	Significance
Position type	Position information in list format **Need**

```
# Joint coordinate value
    j10=Joint( 0, 30, 60, 0, 90, 90 )

# Convert to position coordinate value (j10 → convert → p10)
p10=rb.Joint2Position( j10 )
```

Position2Joint( )

Convert from Position coordinates to Joint coordinates.

Argument	Significance
Joint type	Each axis angle in list format Need

```
#Coordinate value of position type
    p10=Position( -50, -250, 350, 90, 0, 180 )

    # Convert to joint type coordinate value (p10 → convert → j10)
    j10=rb.Position2Joint( p10 )
```

Use the overrap operation

asyncm( ) Install the predicted movement part of the robot program.

Argument	Significance	Variable state	Unit
sw	1: Program before operation ON 2: Turn OFF before program operation (default)	integer	-

In the section where the overlapping operation is installed, the next movement will continue when approaching the target teaching point.

With transit points prepared for obstacle avoidance actions, the robot can be moved to perform the next task without waiting for the robot to complete the action.

```
rb.line (p10) # Linear interpolated motion to teaching point p10
rb.asyncm (sw = 1) # Program prediction operation ON (also available in rb.asyncm (1)
rb.line (p20, p21) # Move to teaching points p20 and p21 in the order of performing linear interpolation.

rb.join () # Waiting function for completion of predicted robot program operation

rb.asyncm (sw = 2) # Program prediction operation OFF (also available in rb.asyncm (2)
...
rb.close()
```

포인트

I/O input/output during execution overlaps

If overlapping motion is used, all commands are predicted, including those processed unrelated to robot motion, such as: I/O input/output functions.

To make actions synchronized with robot actions, please use join() method of i611Robot class.

set_behavior( ) Set pause operation (action)..

Argument	Significance	Variable state	Unit
only_hook	Pause only in user_hook method () True ：Valid False：disable（original value）	bool	-
servo_off	Set servo to OFF when paused True ：Valid False：disable（original value）	bool	-
restore_position	When resuming after pausing, the position will return to before pausing True ：Valid False：disable（original value）	bool	-
no_pause	Pause only when the operation stops True ：Valid（Compatible with system version R0.5.0) False：disable（original value）	bool	-

```
# If restarted after a pause, the pose will return to the position before the pause
rb.set_behavior( only_hook=False, servo_off=False, restore_position=True, no_pause=True )
```

enable_interrupt( ) Sets an exception when decelerating to a stop and an emergency stop.

Argument	Significance	Variable state	Unit
eid	Event ID Need 0: An exception occurs when entering the deceleration stop command during operation 1: The exception occurs when an emergency stop command is entered during operation 2: Exception occurs when deceleration stop command is entered during temporary stop 3: Exception occurs when emergency stop command is entered during temporary stop If exception generation is disabled, the robot program will terminate normally.	integer	-
enable	An exception arises Need True ：Valid False：Cancel	bool	-

```
# Example 1: To enable exception generation when entering a deceleration stop command during operation
rb.enable_interrupt( 0, True )

# Example 2: To enable exception generation when there is an emergency stop input during operation
rb.enable_interrupt( 1, True )

# Example 3: To cancel an exception that occurs when entering the deceleration stop command during a pause
rb.enable_interrupt( 2, False )

# Example 4: To cancel an emergency stop exception occurring during a temporary stop
rb.enable_interrupt( 3, False )
```

user_hook( ) Pause the robot program.

Use in paused position.

Pausing on user_hook() is possible only by specifying set_behavior(only_hook = True). If not specified, the robot program method may be paused.

```
...
rb.user_hook()    # Pause the program at this position
...
```

cause_user_error( ) A user-defined error arises.

Argument	Significance	Variable state	Unit
code	Error ID Need Seetting range : 1－99	integer	-
critical	True : A user-defined fatal error has occurred False : A user-defined error occurred (default)	bool	-

```
# In case a user-defined error occurs (error ID: 19)
rb.cause_user_error (19, False)

# If a user-defined fatal error occurs (error ID: 01)
rb.cause_user_error (01, True)
```

release_stopevent( ) Reset the exception event that is occurring

Performed at the beginning of exception handling.

Repeat until the exception is rethrown.

```
try:
    …      # Operation
except Robot_stop:
    rb.release_stopevent()
    …     # Avoidance action
```

Exception handling for methods in class i611Robot ()

Robot_emo( )	Exception occurs when emergency stop (return is not possible)
Robot_error( )	The exception occurred due to an error
Robot_fatalerror( )	Exceptions are raised in case of fatal (irrecoverable) errors.
Robot_poweroff( )	Exception occurs when power is turned off (cannot be recovered)
Robot_stop( )	The exception occurs when decelerating to a stop.

6. End

End of robot program

close( ) Disconnect from the robot.

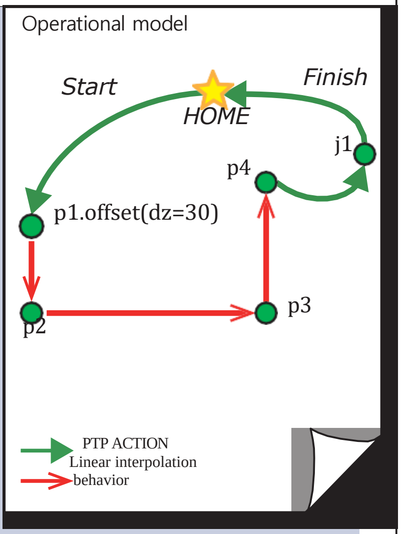
```
    # End
        rb.close()
```

When ending the program or powering off the controller, the close() method must be executed for all using classes.

2. Sample program

Model 1 Default activity

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

## 1. Initial setting ① ##########################################
 # Import the library
 from i611_MCS import *
 from i611_extend import *
 from i611_io import *

 def main():
   ## 2. Initial setting② ######################################
   # Robot i611 constructor
   rb = i611Robot()
      # Definition of world coordinate system
      _BASE = Base()
      # Start connecting to the robot, initialize
      rb.open()
      # Initialize I/O input/output functions (can be omitted when I/O is not
      used)
      IOinit( rb )
         ## 3. Set teaching points ########################
      p1 = Position( 95, -280, 425, -120, 84, -28 )
      p2 = Position( 95, -280, 240, 154, 80, -114 )
      p3 = Position( 300, -280, 240, 159, 86, -156 )
      p4 = p3.copy()
      p4.shift( dz=40 )
      j1  = Joint( 230, -1, -92, 90, 5, 89 )

   ## 4. Set operating conditions ######################### # Set
      the motion condition in the MotionParam constructor m =
      MotionParam( jnt_speed=10, lin_speed=70 )

      # Set the operating condition using the MotionParam type
      rb.motionparam( m )
   ## 5. Definition of robot movement ################## # start
      operation rb.home() rb.move( p1
      ) rb.line( p2
      rb.move( j1.offset(dz=30)
             , p3, p4 )
      rb.home()
             )
   ## 6. End #######
# Disconnect from the robot
      rb.close()

 if __name__== '__main__':
      main()
```



PTP operating speed 10%, linear interpolation operating speed 70mm/si

Go to Home position

PTP operation with 30 mm Z axis offset at guide point p1

Linear interpolation operation follows the order of points p2, p3, p4

PTP activity with teaching point j1

Go to the Home position

In the Python language, paragraphs are separated by indentation. If you copy the PDF file as is, the indentation will not be copied, so indent as in the example by pressing the Spacebar key 4 times or the Tab key once. The Tab key may not work as expected depending on the text editor.

Model 2 Overwrite

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

## 1. Initial setting① ########################################
# Enter library
 from i611_MCS import *
 from i611_extend import *
 from i611_io import *

 def main():
    ## 2. Initial setting② ##############################
    # Robot i611 constructor

    rb = i611Robot()
    # Definition of world coordinate system
        _BASE = Base()
    # Start connecting to the robot, initialize
        rb.open()
    # INITIALIZING I/O I/O FUNCTIONS
    (CAN BE REMOVED WHEN I/O IS NOT IN USE)
        IOinit( rb )
    # Overwrite
        rb.override( 50 )

            ## 3. Set teaching points #######################
         p1  =  Position( 95, -280, 425, -120, 84, -28 )
         p2  =  Position( 95, -280, 240, 154, 80, -114 )
         p3  =  Position( 300, -280, 240, 159, 86, -156 )
        p4  = p3.offset( dz=40 )
        j1  = Joint( 230, -1, -92, 90, 5, 89 )

    ## 4. Set operating conditions ###################### #Đặ t
        motion condition in the MotionParam constructor m =
        MotionParam( jnt_speed=10, lin_speed=70 ) # Set
        the operating condition to type MotionParam
        rb.motionparam( m )

    ## 5. Definition of robot movement ################# # Start
        operation
        rb.home()
        rb.move( p1.offset(dz=30) )
        rb.line( p2, p3, p4 )
        rb.move( j1 )
        rb.home()

    ## 6. End ###################################
        # Disconnect from the robot
        rb.close()

 if __name__== '__main__':
        main()
```

Set Override to 50%

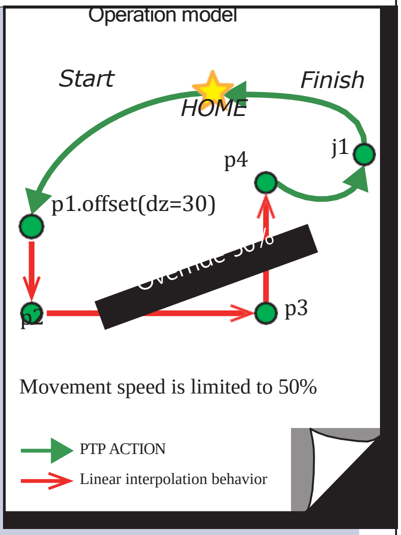
Go to the Home position

PTP operation with 30 mm Z axis offset at guide point p1

Linear interpolation operation follows the order of points p2, p3, p4

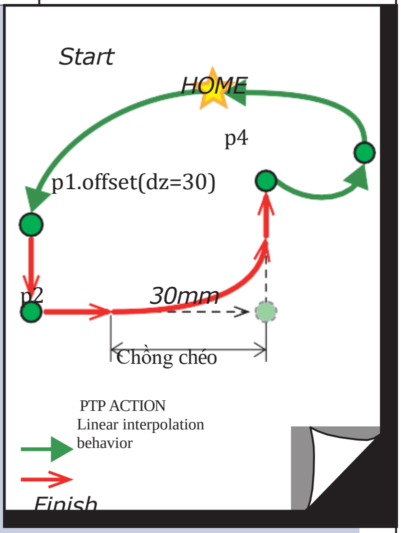
PTP activity with teaching point j1

Go to the Home position



Model 3 Overlapping

Operational model



j1

p3

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

## 1. Initial setting① ########################################
  # Enter library
  from i611_MCS import *
  from i611_extend import *
  from i611_io import *

  def main():
     ## 2. Initial setting② ##############################
     # Robot i611 constructor
        rb = i611Robot()
     # Definition of world coordinate system
        _BASE = Base()
     # Start connecting to the robot, initialize
        rb.open()
     # INITIALIZING I/O I/O FUNCTIONS
     (CAN BE REMOVED WHEN I/O IS NOT IN USE)
        IOinit( rb )
     ## 3. Set teaching points ######################
        p1 = Position( 95, -280, 425, -120, 84, -28 )
        p2 = Position( 95, -280, 240, 154, 80, -114 )
        p3 = Position( 300, -280, 240, 159, 86, -156 )
        p4 = p3.offset( dz=40 )
        j1 = Joint( 230, -1, -92, 90, 5, 89 )


     ## 4. Set operating conditions ################################### # Set the
        motion condition in the MotionParam constructor
        m = MotionParam( jnt_speed=10, lin_speed=70, overlap = 30 )
        # Set the operating condition using the MotionParam type
        rb.motionparam( m )

        ## 5. Definition of robot motion ##############################
        rb.home()
        rb.move( p1 )
        rb.line( p2 )
        # Program prediction
        operation is ON rb.asyncm(
        1 )
        rb.line( p3, p4 )
        rb.join()
        # Program prediction
        operation is OFF
        rb.asyncm( 2 )
        # Coordinates of each axis [0, 0, 0,
        rb.home()
     ## 6. End #################################
        # Disconnect from the robot
        rb.close()

  if __name__== '__main__':
        main()
```

Overlap to 30mm

Go to the Home position

PTP activity with teaching point p1

Linear interpolation with teaching point p2

Program prediction operation ON (preparing to execute overlapping operation)

Switch to overlapping operation p3, p4

Overlapping deactivation positions. Wait for the prediction program to complete

Program prediction operation OFF

Go to the Home position

Model 4 Wait for I/O input

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

## 1. Initial setting① ########################################
# Enter library
from i611_MCS import *
from i611_extend import  *
from i611_io import *

 def main():
    ## 2. Initial setting② ##############################
    # Robot i611 constructor
    rb = i611Robot()
    # Definition of world coordinate system
        _BASE  =  Base()
    # Start connecting to the robot, initialize


        rb.open()
    # Initialize I/O input/output functions
        IOinit( rb )

            ## 3. Set teaching points #######################
         p1  =  Position( 95,  -280, 425,  -120, 84,  -28 )
         p2  =  Position( 95,  -280, 240,  154, 80,  -114 )
         p3  =  Position( 300,  -280, 240,  159, 86,  -156 )
        p4  = p3.offset( dz=40 )
        j1  = Joint( 230, –1, –92, 90, 5, 89 )

    ##4. Set operating conditions ######################### #
        motion condition in the MotionParam constructor
        m  =  MotionParam( jnt_speed=10, lin_speed=70 )
        # Set the operating condition to type MotionParam
        rb.motionparam( m )

    ## 5. Definition of robot movement ############################## # Start
    operation

    rb.home()
    # Wait for I/O input
         if wait( 5, "1", 10 )[0] == 1:

             rb.move( p1.offset(dz=30) )
             rb.line(  p2, p3, p4  )
             rb.move( j1 )
    # When the input time is exceeded
    else:
             rb.home()

    ## 6. End ###################################
    # Disconnect from the robot
        rb.close()

 if __name__== '__main__':
        main()
```

Go to Home position

Set the timeout to 10 seconds until I/O input port number 5 becomes 1. If the input signal arrives before the waiting time ends,

PTP ACTION has a Z-axis offset of 30 mm from the lead point p1

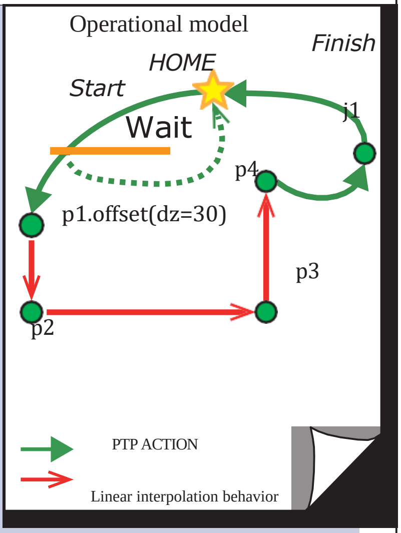
Linear interpolation operation follows the order of points p2, p3, p4

PTP activity with teaching point j1

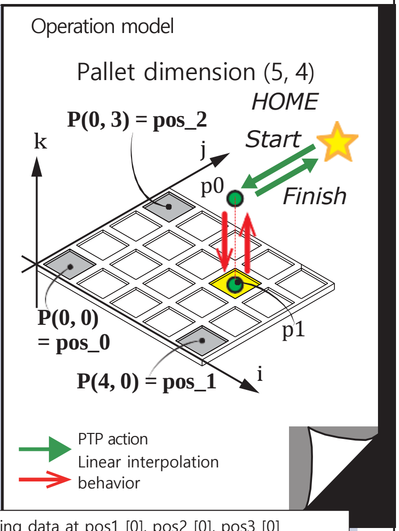
If exceeded

Go to Home position

(In this example, do not move from the Home position)



Model 5 Pallet functions



```
#!/usr/bin/python
# -*- coding: utf-8 -*-

## 1. Initial setting① #########################################
# Enter library
  from i611_MCS import *
  from i611_extend import *

  def main():
     ### 2. Initial setting② ###############################
     # Robot i611 constructor
        rb = i611Robot()
        rb.open()
     # Definition about the world coordinate system
        _BASE = Base()

     ## 3. Set teaching point ################# # Read
        the teaching data file
        data = Teachdata( "teach_data" )
```

Previously taught teaching data at pos1 [0], pos2 [0], pos3 [0]

```
        pos_0 = data.get_position( "pos1", 0 )
        pos_1 = data.get_position( "pos2", 0 )
        pos_2 = data.get_position( "pos3", 0 )
        # Identify Pallet
```

Set the cell position by 3 points to calculate the cell position in the Pallet

```
        pal = Pallet()
        pal.init_3( pos_0, pos_1, pos_2, 5, 4 )
```

Determine the size of the pallet position

```
        # Adjust pallet pal.adjust( 3,
        2, 0.4, -0.3 ) # Search the
        operation position
```

Adjust the position of (i, j) = (3, 2)
(+0.4mm in i direction, -0.3mm in j direction)

```
        p0 = pal.get_pos( 3, 2, 30 )
        p1 = pal.get_pos( 3, 2 )

     ## 4. Set the operation condition ####################################

     # Set the operating condition by using the MotionParam type
        rb.motionparam( jnt_speed=30 )
```

PTP activity rate 30%

```
     ## 5. Definition of robot movement ############################## # Start
     action
        rb.home()
        rb.move( p0 )
        rb.line( p1 )
        rb.line( p0 )
        rb.home()
```

Go to Home position
PTP operation with p0 (30 mm above pallet position (3,2))
Linear interpolation action to p1 (Pallet position (3,2))
PTP operation with p0 (30 mm above pallet position (3,2))
PTP operation with Home position

```
     ## 6. End ##################################
     # Disconnect from robot
        rb.close()

  if __name__== '__main__':
        main()
```

NOTES

D SOFTWARE

2 ROBOT LIBRARY

1. Data type

Variable type

Type (icon)	Significance
string string	String type Enclose in single quotes (') or double quotes (").
float float	Real number type with floating point
integer integer	Integer type
long long	Type of long integer
bool bool	Logical type True / False

Type of data form 1

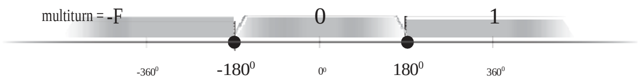
Type (Icon)	Significance
List List	Put the word part in [. . .].
Set Tuple	Is a sequence of elements in (. . .). Tuple elements cannot be changed.
Dictionary Dict	{. . .} separates the key (key) and value (value) called “Dictionary” with a colon (:). (e.g. key:value)
Variable argument Vari. No	Expand the list. Insert a “* (asterisk)” before the argument. The received arguments will be stored in a tuple in the specified order.
Keyword argument Keyword	Expand dictionary. Insert two “* (asterisk)” signs before the argument. The order in which the arguments become "dictionary" at the time they are received is ignored.

Type of data form 2

<table>
<tr><th>Type
(Icon)</th><th colspan="2">Content</th></tr>
<tr><td rowspan="7">Position
[Position]</td><td colspan="2">Position coordinates are expressed using the world coordinate system</td></tr>
<tr><td colspan="2">[x, y, z, rz, ry, rx, parent, posture, multiturn] : List</td></tr>
<tr><td>x, y, z
[mm] float</td><td>Position (Descartes coordinate system)
Original value：0.0</td></tr>
<tr><td>rz,ry,rx
[deg] float</td><td>Shape (Euler angle in Z-Y-X system)
Original value：0.0</td></tr>
<tr><td>parent
[-] float</td><td>Set to use the world coordinate system [*1]
Original value：_BASE Pose [*2]</td></tr>
<tr><td>posture
[-] integer</td><td>Original value：-1
Cross counter information [*3]</td></tr>
<tr><td>multiturn
[-]
long</td><td>Original value：0 x F F 0 0 0 0 0 0
FLAG J6 J5 J4 J3 J2 J1
(There is no cross counter information)

The cross counter is an implementation that uniquely converts position data of type Position to angle data of type Joint.
For example: We need to determine whether the joint angle is -200° or 160°. The cross counter will first be set up, then the movements of the operations confirmed during the teaching process can be reproduced. The value is updated when the angle of each joint is exceeded ±180°. Set the following values according to the angle range of each joint.
The value applied depends on the number of joints. For example, in a 4-axis robot the values J5, J6 are ignored.

・FLAG section: With or without FF cross counter information
：No（Original value）
00：Yes
(Anything other than the above is invalid))

・Part J1 - J6：cross counter value
<table>
<tr><th>Setting value</th><th>Angle of each joint</th></tr>
<tr><td>1</td><td>Rotate more than 180° in + direction</td></tr>
<tr><td>0</td><td>180° ~ 180°</td></tr>
<tr><td>F [*4]</td><td>Rotate more than 180° in - direction</td></tr>
</table>
Relationship between cross counter value and joint angle


</td></tr>
</table>

*1) The setting value is "_BASE ".

*2) There are 8 postures depending on the position, direction and angle of the arms.

(☞ C period Coordinates and pose)
5

*3) Cross counter information can be abbreviated as “CC”.

*4) Displayed as “-1” on the teaching screen.

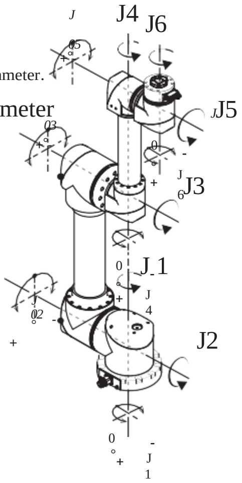
(☞ period 4 Period)

API related to multiturn parameter

Implementing the use_mt() method in the i611Robot class will change the behavior of the following APIs. The multiturn parameter is used in the move() and Location2Joint() methods in the i611Robot class.

This API can be used regardless of whether or not multiturn information is present.

Class	API	Functions	p.
Position	has_mt()	Check cross counter information.	30
	Position() (Hàm tạo)	Create an entity that defines the teaching point in the world coordinate system.	28
	pos2dist()	Position coordinate values are entered in dictionary format.	31
	pos2list()	Position coordinate values are entered in list format.	31
	position()	Convert the Position coordinate value to the Parent coordinate system and import in list format	31
	replace()	Position Replace coordinate values. (Self-updated.)	32
MotionParam	ik_solver_option (Biến số)	Direction of rotation	41
i611Robot	getpos()	Get the current position of the Manipulator operator in the Position type	60
	Joint2Position()	Convert joint coordinate values to position coordinate values..	63
	move()	Move by PTP.	66
	Position2Joint()	Convert from Position coordinates to Joint coordinates..	70
	use_mt()	Cross counter on/off setting	80

Type (Icon)	Content	
Joint [Joint]	Angle of each joint	
	[j1, j2, j3, j4, j5, j6,] : List	
	j1, j2, j3, j4, j5, j6 [deg] float	Joint type data for each axis Original value：[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
MotionParam [MotionParam]	The robot's motion parameters are made up of variables	
	lin_speed [mm/s] float	Speed (linear interpolated motion) Original value：5.0
	jnt_speed [%] float	Speed (PTP motion, Joint motion, optimal linear interpolation operation) Original value：5.0
	acctime float [s]	Acceleration time Original value：0.4
	dacctime float [s]	Deceleration time Original value：0.4
	posture integer [-]	Pose Original value：2
	passm integer [-]	Path movement Original value：2
	overlap float [mm]	Overlap motion Original value：0�0
	zone integer [pulse]	Complete range of positioning Original value：100
	pose_speed float [%]	Speed (pose interpolated motion) Original value：20 When the manipulator operates while changing direction, set the upper limit of the machine head's Euler angular movement speed.
	ik_solver_option [-] long	Direction of rotation Original value : $0\ \mathrm{X}\ \underline{1\ 1}\ \underline{1}\ \underline{1}\ \underline{1}\ \underline{1}\ \underline{1}\ \underline{1}$ (Rsv.) J6 J5 J4 J3 J2 J1 Setting value J1 - J6 0：Shortest line Rotation does not use information from the multiturn parameter. 1：Use information from the multiturn parameter 2：Rotation toward + direction 3：Rotation toward - direction Refer to the picture on the right with + direction and - direction-  Definition of rotation direction

2. Module

1. List of modules

Modules control robot

Modules	Classes	Summary
i611_MCS i611 MCS	Base P. 23 ~	Determine the world coordinate system (The pseudo class will be used in the Position class and the Coordinates class)
	Coordinate P. 24 ~	Handle world coordinate system objects
	Position P. 28 ~	Handling Position of coordinate values in the world coordinate system
	Joint P. 33 ~	Handles joint coordinate values in the joint coordinate system
	MotionParam P. 37 ~	Handle robot motion parameters
	i611Robot P. 47 ~	Handle robot motion

Methods					Lớp
Base( ) p. 23					Base
b2g( ) p. 25	clear( ) p. 25	Coordinate( ) p. 24	copy( ) p. 25	g2b( ) p. 26	Coordinate
inv( ) p. 26	replace( ) p. 27	shift( ) p. 27			
clear( ) p.29	copy( ) p.29	has_mt() p.30	offset() p.30	pos2dict() p. 31	Position
pos2list( ) p. 31	Position( ) p. 28	position( ) p. 31	replace( ) p. 32	shift( ) p. 32	
clear( ) p. 34	copy( ) p. 34	jnt2dict( ) p. 34	jnt2list( ) p. 35	Joint( ) p. 33	Joint
offset( ) p. 35	replace( ) p. 36	shift( ) p. 36			
clear( ) p.43	confdefault( ) p.43	copy( ) p. 44	MotionParam( ) p. 42	motionparam( ) p. 45	Motion Param
mp2dict() p. 46	mp2list( ) p. 46				
abort( ) p.50	adjust_mt() p.50	asyncm() p.51	cause_user_error( ) p.52	changetool() p. 52	i611Robot
check_ready( ) p. 53	close( ) p. 54	disable_mdo( ) p. 54	enable_interrupt( ) p. 55	enable_mdo( ) p. 56	
exit( ) p. 57	get_hw_info() p.57	get_system_port() p.58	get_system_status( ) p. 59	getjnt() p. 59	
getmotionparam( ) p. 60	getpos( ) p. 60	home( ) p. 60	i611Robot( ) p. 49	is_open( ) p. 61	
is_pause( ) p. 61	join() p. 63	Joint2Position( ) p. 63	line( ) p. 64	MCS_version( ) p. 65	
motionparam( ) p. 65	move( ) p. 66	open( ) p. 67	optline( ) p. 68	override( ) p. 69	
pause() p. 69	Position2Joint( ) p. 70	release_stopevent() p. 70	reljntmove() p. 71	relline() p. 72	
restart( ) p. 73	set_behavior( ) p. 74	set_mdo( ) p. 75	settool( ) p. 76	sleep( ) p. 77	
stop( ) p. 78	svoff() p. 78	svstat( ) p. 79	toolmove( ) p. 79	use_mt() p. 80	
user_hook( ) p. 80	version( ) p. 81				

Shading method (▨) is created function.

Modules control robot

Modules	Classes	Summary
teachdata Teach data	Teachdata P. 83 ~	Administration of teaching data
i611_extend i611 Ext.	Pallet P. 93 ~	Handle Pallet functions
rbsys rbsys	RobSys P. 98 ~	Use the system administrator (*)

*) System administrator is the controller's internal program, which has the function of managing the status of the robot program, controlling the teaching status and handling errors..

Module for handling exception

i611_common i611 COM.	Exception P. 108 ~	Exception handling in i611Robot class methods

Functional modules

Modules	Classes	Summary
i611_io i611 IO	(None) P. 113 ~	Functional modules I/O control
i611shm i611 shm	(None) P. 119 ~	Functional modules Access shared memory

Methods					Class
check_format( ) p. 84	close( ) p. 85	flush( ) p. 85	get_coordinate( ) p. 86	get_joint( ) p. 86	Teachdata
get_param( ) p. 87	get_position() p. 88	get_tool() p. 89	is_open() p. 89	open( ) p. 90	
set_joint( ) p. 90	set_param( ) p. 91	set_position( ) p. 92	Teachdata( ) p. 84		
adjust( ) p. 94	get_pos( ) p. 95	init_3( ) p. 90	init_4( ) p. 97	Pallet( ) p. 93	Pallet
assign_din() p. 99	assign_dout( ) p. 100	clear_robtask( ) p. 101	close( ) p. 101	cmd_pause( ) p. 102	RobSys
cmd_reset( ) p. 102	cmd_run( ) p. 103	cmd_stop( ) p. 103	get_robtask( ) p. 104	open( ) p. 104	
req_mcmd( ) p. 105	RobSys( ) p. 98	set_robtask( ) p. 106	version( ) p. 107		

Methods (▨) handled are created functions.

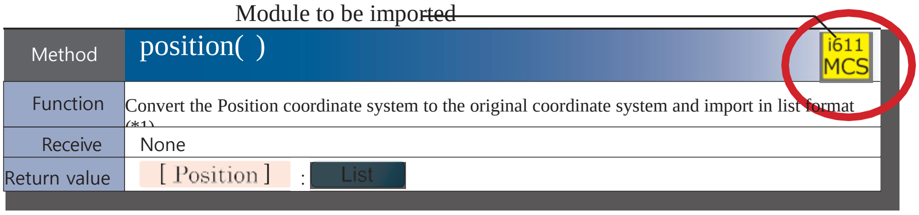
The class inherits the Exception class				Class
Robot_emo() p. 109	Robot_error( ) p. 110	Robot_fatalerror( ) p. 110	Robot_poweroff( ) p. 111	Exception

Functions				Module
din( ) p. 114	dlyOut() p. 115	dout() p. 115	IOinit() p. 116	i611_io
shotOut( ) p. 117	wait( ) p. 118			
shm_read() p. 119	shm_write( ) p. 120			i611shm

2. How to use modules, classes and functions

STEP1 Import module

To use the robot library, import the module in use first. Inside this method there are some modules that need to be entered first. Modules to import are represented by an icon in each method.



Module to be imported

Method	position( ) i611 MCS
Function	Convert the Position coordinate system to the original coordinate system and import in list format (*1)
Receive	None
Return value	[Position] : List

Display and enter modules

Module	Example input program	Inclusion classes
i611_MCS i611 MCS	from i611_MCS import *	Base Coordinate Position Joint MotionParam i611Robot
teachdata Teach data	from teachdata import *	Teachdata
i611_extend i611 Ext.	from i611_extend import *	Pallet
rbsys rbsys	from rbsys import *	RobSys
i611_common i611 COM.	from i611_common import *	Exception
i611_io i611 IO	from i611_io import *	(None)
i611shm i611 shm	from i611shm import *	(None)

STEP2 Prepare to use classes and functions

Use methods of class Base, Coordinate, Position, Joint, MotionParam, i611Robot

Classes	Steps
Base	1. Import module . i611 MCS `from i611_MCS import *`
Coordinate Position	1. Import Module . i611 MCS 2. Define the dummy class. `_BASE = Base()`
Joint	1. Import Module. i611 MCS

MotionParam	1. Import Module. i611 MCS
	2. Create an Entity MotionParam
	`m =MotionParam()`
	Set robot motion parameters if necessary.
	`m = MotionParam(jnt_speed=10,lin_speed=70,overlap=30)`
	(The combed parameters are set to their initial values.)

i611Robot	1. Import Module. i611 MCS
	2. Create an Entity i611Robot.
	`rb =i611Robot()`
	3. Connect to the robot you want to run by using the open method ().
	`rb.open()`
	For methods (*) related to robot movement, set the movement parameters in the robot's MotionParam..

*) The method that accompanies the robot's movements

disable_mdo( )	enable_mdo( )	getjnt( )	getmotionparam( )	getpos( )
home( )	join( )	Joint2Position( )	line( )	motionparam( )
move( )	optline( )	override( )	Position2Joint( )	reljntmove( )
relline( )	set_mdo( )	toolmove( )		

STEP2 Prepare to use class and function

Use the methods of the Teachdata class

Class	Steps

Teachdata	1. Import module. Teach data
	`from teachdata import *`
	2. Create the Teachdata entity.
	`td = Teachdata( )`
	3. Run the open() method on the Teachdata class.
	`td.open( readonly = False )`

Use the methods of the Pallet class

Class	Steps

Pallet	1 . Import module. i611 MCS i611 Ext.
	`from i611_MCS import *`
	`from i611_extend import *`
	2 . Create the i611Robot entity.
	`rb = i611Robot( )`
	3 . Implement the open() method of the i611Robot class.
	`rb.open()`
	4. Create an instance of the Pallet class.
	`pal = Pallet()`

Prepare to use classes and functions

Use the methods of the RobSys class

Class	Steps
RobSys	1 . Import Module. rbsys from rbsysimport * 2 . Create the RobSys entity. rbs = RobSys() 3 . Implement the open() method of the RobSys class. rbs.open() 4 . Import the i611_io module, which has a method (*) that defines the I/O configuration so that the I/O function can be used. (☞p.14) i611 IO

*) I/O setting method

assign_din( ), assign_dout( )

Use the methods of the Exception class

Class	Steps
Exception	1 . Import Module.(*1) i611 COM. fromi611_commonimport* 2 . Use a class that inherits the Exception class. For details, see how to use each layer.

*1) Module imported

Importing i611_common import * in module i611_MCS.

The Exception class is available when importing the i611_MCS module.

Prepare to use classes and functions

Use functions of module i611_io

Module	Steps
i611_io	1 . Import Module. i611 IO fromi611_ioimport* 2 . Initialize I/O . IOinit() 3 . Create the i611Robot entity. rb=i611Robot() 4 . Run the method open() of i611Robot. rb.open() 5 . For methods (*) related to robot movement, set the operating parameters in the robot's MotionParam.

*) For methods (*) related to robot movement, set the operating parameters in the robot's MotionParam (i611Robot class)

disable_mdo( )	enable_mdo( )	getjnt( )	getmotionparam( )	getpos( )
home( )	join( )	Joint2Position( )	line( )	motionparam( )
move( )	optline( )	override( )	Position2Joint( )	reljntmove( )
relline( )	set_mdo( )	toolmove( )		

Use the function of i611_shm module

Module	Step
i611_shm	1 . Import Module. i611 shm from i611_shm import *

Python "time" module

"Time" is a library containing information functions related to time.
Conduct a check of the current system time or generate data in time format. The starting time is 00:00:00 on January 1, 1970. You can use it when importing the following modules.

```
import time
```

3. List of methods

	Methods, functions	Functions	Modules, classes		page
A	abort( )	Stop the robot program	i611 MCS	i611Robot	50
	adjust( )	Accurate cell position on Pallet	i611 Ext.	Pallet	94
	adjust_mt( )	Fix CC value when converting location type coordinate value to string	i611 MCS	i611Robot	50
	assign_din( )	Fix CC value when converting position type coordinate value to string	rbsys	RobSys	99
	assign_dout( )	Assign functions to physical I/O ports and memory I/O output ports	rbsys	RobSys	100
	asyncm( )	Set up the prediction part of the robot program	i611 MCS	i611Robot	51
B	b2g( )	Convert from base coordinate system to world coordinate system	i611 MCS	Coordinate	25
	Base( )	Base class constructor	i611 MCS	Base	23

C	cause_user_error( )
	changetool( )
		User-defined error arises Select tool offset	i611 MCS	i611Robot	52

	check_format( )	Create a formatted version of the teaching data file	i611 MCS	i611Robot	52
	check_ready( )	Check whether the robot can drive automatically or not	Teach data	Teachdata	84
	clear( )	Initialize the world coordinate system instance	i611 MCS	i611Robot	53
				Coordinate	25
	clear( )	Initialize the position coordinate value	i611 MCS	Position	29
	clear( )	Initialize the Joint coordinate value	i611 MCS	Joint	34
	clear( )	Initialize operating parameters	i611 MCS	MotionParam	43
	clear_robtask( )	Unsubscribe from the robot program	i611 MCS	RobSys	101
	close( )	Disconnect from robot	rbsys	i611Robot	54
	close( )	End of teaching data file	i611 MCS	Teachdata	85
	close( )	Terminate connection with system administrator	rbsys	RobSys	101
	cmd_pause( )	Action command: Pause	rbsys	RobSys	102
	cmd_reset( )	Action command: Reset error	rbsys	RobSys	102
	cmd_run( )	Action command: Run the robot program	rbsys	RobSys	103

Inner method (▭) is created function .

Icon module

i611 MCS i611_MCS	Teach data teachdata	i611 Ext. i611_extend	rbsys rbsys	i611 COM i611_common	i611 IO i611_io	i611 shm i611shm

	Methods, functions	Functions	Modules	classes	p.
	cmd_stop( )	Operating command: decelerate to stop	rbsys	RobSys	103
	confdefault( )	Set the initial values of the operating parameters	i611 MCS	MotionParam	43
	Coordinate( )	Constructor of Coordinate class	i611 MCS	Coordinate	24
	copy( )	Copy the world coordinate system object	i611 MCS	Coordinate	25
	copy( )	Copy the Position coordinate value	i611 MCS	Position	29
	copy( )	Copy the Joint coordinate value	i611 MCS	Joint	34
	copy( )	Copy operating parameters	i611 MCS	MotionParam	44
	din( )	Functional I/O input	i611 IO	(None)	114
D	disable_mdo( )	Invalid MDO operation	i611 MCS	i611Robot	54
	dlyOut( )	I/O output after specified time has elapsed	i611 IO	(None)	115
	dout( )	Functional I/O output	i611 IO	(None)	115
	enable_interrupt( )	Set exceptions for deceleration and emergency stops		i611Robot	55
E	enable_mdo( )	Enable MDO activity	i611 MCS	i611Robot	56
	exit( )	Force quit the robot program	Teach data	i611Robot	57
	flush( )	Export updated teaching data to files	i611 MCS	Teachdata	85
F	g2b( )	Convert from world coordinate system to base coordinate system	i611 MCS	Coordinate	26
G	get_coordinate( )	Check the base offset value of the teaching data	Teach data	Teachdata	86
	get_hw_info( )	Check the model name and serial number	i611 MCS	i611Robot	57
	get_joint( )	Check the Joint coordinate value of the teaching data	Teach data	Teachdata	86
	get_param( )	Check parameters of teaching data	Teach data	Teachdata	87
	get_pos( )	Get cell location	i611 Ext.	Pallet	95

Inner method (□) is created function.

Functions

	Methods, functions		Mô-đun	Lớp	p.
	get_position( )	Check the Position coordinate value of the teaching data	Teach data	Teachdata	88
	get_robtask( )	Check the status of the robot program	rbsys	RobSys	104
	get_system_port( )	Check system port status	i611 MCS	i611Robot	58
	get_system_status( )	Check system status and error ID	i611 MCS	i611Robot	59
	get_tool( )	Check the tool offset of the teaching data	Teach data	Teachdata	89
	getjnt( )	Check the current position of the controller according to the joint type.	i611 MCS	i611Robot	59
	getmotionparam( )	Check current operating parameters	i611 MCS	i611Robot	60
	getpos( )	Check the current location of the Manipulator via Position Type	i611 MCS	i611Robot	60
	has_mt( )	Check cross counter information	i611 MCS	Position	30
H	home( )	Move all axes to Joint 0deg coordinates	i611 MCS	i611Robot	60
	i611Robot( )	Constructor of class i611Robot	i611 MCS	i611Robot	49
I	init_3( )	Definition of pallet (Lecture 3 points)	i611 Ext.	Pallet	96
	init_4( )	Definition of pallet (Lecture 4 points)	i611 Ext.	Pallet	97
	inv( )	Perform the inverse transformation to create an object of the Coordinate class	i611 MCS	Coordinate	26
	IOinit( )	Initialize I/O function	i611 IO	(Không có)	116
	is_open( )	Check i611Robot open status	i611 MCS	i611Robot	61
	is_open( )	Check the open status of teaching data	Teach data	Teachdata	8
	is_pause( )	Check the pause status of the robot program	i611 MCS	i611Robot	60
	jnt2dict( )	Check Joint coordinate value in dictionary format	i611 MCS	Joint	34
J	jnt2list( )	Check the Joint coordinate value in list form	i611 MCS	Joint	35
	join( )	Wait for the predicted robot program operation to complete	i611 MCS	i611Robot	63
	Joint( )	Constructor of Joint class	i611 MCS	Joint	33
L	Joint2Position( )	Converts joint coordinate values to position coordinate values	i611 MCS	i611Robot	63
	line( )	Perform linear interpolation	i611 MCS	i611Robot	64
M	MCS_version( )	Check the version of the robot library	i611 MCS	i611Robot	65

Icon module

i611 MCS i611_MCS　Teach data teachdata　i611 Ext. i611_extend　rbsys rbsys　i611 COM. i611_common　i611 IO i611_io　i611 shm i611shm

	Methods, functions	Functions	Modules | Classes	p.
	MotionParam( )	Constructor of MotionParam class	i611 MCS MotionParam	42
	motionparam( )	Set operating parameters	i611 MCS MotionParam	45
	motionparam( )	Set operating parameters	i611 MCS i611Robot	65
	move( )	Carry out PTP activities	i611 MCS i611Robot	66
	mp2dict( )	Get operating parameters in dictionary format	i611 MCS MotionParam	46
	mp2list( )	Enter operating parameters in list format	i611 MCS MotionParam	46
	offset( )	Add offset coordinate value to Position coordinate value (create new object while maintaining itself)	i611 MCS Position	30
O	offset( )	Add offset coordinates to joint coordinates (create a new object while maintaining itself)	i611 MCS Joint	35
	open( )	Start connecting to the robot (initialization)	i611 MCS i611Robot	67
	open( )	Open the teaching data file	Teach data Teachdata	90
	open( )	Initiate communication with the system administrator	rbsys RobSys	104
	optline( )	Perform linear interpolation motion while shifting gears at optimal speed	i611 MCS i611Robot	68
	override( )	Perform override(*)	i611 MCS i611Robot	69
	Pallet( )	Constructor of Pallet class	i611 Ext. Pallet	93
P	pause( )	Pause the robot's movements	i611 MCS i611Robot	69
	pos2dict( )	Enter the Position coordinate value in dictionary format	i611 MCS Position	31
	pos2list( )	Enter the Position coordinate value in list format	i611 MCS Position	31
	Position( )	Constructor of the Position class	i611 MCS Position	28
	position( )	Convert the Position coordinate value to the parent coordinate system and import it in list format	i611 MCS Position	31
	Position2Joint( )	Convert from Position coordinates to Joint coordinates	i611 MCS i611Robot	70
R	release_stopevent( )	Reset the exception event that is occurring	i611 MCS i611Robot	70
	reljntmove( )	Relative motion in the joint coordinate system	i611 MCS i611Robot	71
	relline( )	Perform relative linear interpolations in Cartesian coordinates	i611 MCS i611Robot	72

Inner method(□) is created function.

*) override covers the speed setting.

Methods, functions	Function	Modules	Classes	p.
replace( )	Replace the world coordinate system object (self-updating)	i611 MCS	Coordinate	27
replace( )	Replace Position coordinate value (self-updating)	i611 MCS	Position	32
replace( )	Replace Joint coordinate values (self-updating)	i611 MCS	Joint	36
req_mcmd( )	Check system status and operate command status	rbsys	RobSys	105
restart( )	Generate restart signal from pause	i611 MCS	i611Robot	73
RobSys( )	Function of RobSys class	rbsys	RobSys	98
set_behavior( )	Set pause action (action)	i611 MCS	i611Robot	74
set_joint( )	Update coordinate values to match teaching data	Teach data	Teachdata	90
set_mdo( )	MDO operation settings (*)	i611 MCS	i611Robot	75
set_param( )	Update parameters of teaching data	Teach data	Teachdata	91
set_position( )	Update coordinate values of teaching data	Teach data	Teachdata	92
set_robtask( )	Registration for the robot program	rbsys	RobSys	106
settool( )	Sett tool offset	i611 MCS	i611Robot	76
shift( )	Move objects in world coordinate system (self-updating)	i611 MCS	Coordinate	27
shift( )	Move Position coordinate value (self-updating)	i611 MCS	Position	32
shift( )	Move joint coordinate values (self-updating)	i611 MCS	Joint	36
shm_read( )	Function Read shared memory	i611 shm	(None)	119
shm_write( )	Function Write to shared memory	i611 shm	(None)	120
shotOut( )	Function I/O output at the specified time	i611 IO	(None)	117
sleep( )	Pause processing for a certain period of time	i611 MCS	i611Robot	77
stop( )	Decelerate the robot to stop	i611 MCS	i611Robot	78

*) MDO operation: Is a function that changes I/O output under specified conditions during operation..

Icon module

i611 MCS	i611_MCS
Teach data	teachdata
i611 Ext.	i611_extend
rbsys	rbsys
i611 COM	i611_common
i611 IO	i611_io
i611 shm	i611shm

	Methods, fiunctions	Function	Modules	Classes	p.
	svoff( )	Set servo to OFF	i611 MCS	i611Robot	78
	svstat( )	Check servo status	i611 MCS	i611Robot	79
T	Teachdata( )	Constructor of class Teachdata	Teach data	Teachdata	84
	toolmove( )	Perform relative movement in the tool coordinate system	i611 MCS	i611Robot	79
U	use_mt( )	Set cross counter on/off	i611 MCS	i611Robot	80
	user_hook( )	Pause the robot program	i611 MCS	i611Robot	80
V	version( )	Check the system version	i611 MCS	i611Robot	81
	version( )	Check the version of System Manager	rbsys	RobSys	107
W	wait( )	Function Wait until the specified I/O input pattern is reached	i611 IO	(None)	118

The inner method (□) is the constructor

2 Robot Library

4. Robot Library

i611 MCS

Teach data

i611 Ext.

rbsys

i611 COM.

i611 IO

i611 shm

When ending the program or shutting down the controller, make sure to execute the close() method of all the classes you are using.

The robot program divides between “uppercase” and “lowercase” letters.

「Regarding “Exception”（Exception）

In case the robot program is created incorrectly, an "exception" will occur. If an exception occurs, check the error code and take appropriate action.

Check the results of program execution

Record usage examples provided for each method. You can check the results of the method's execution using the print statement.

Usage example

```
# Load teaching data
    data=Teachdata( './teach_data' )
#  Key: Joint1, index number: Taking in data 0.
    jnt_0=data.get_Joint( 'Joint1', 0 ) print
    jnt_0.jnt2list()
```

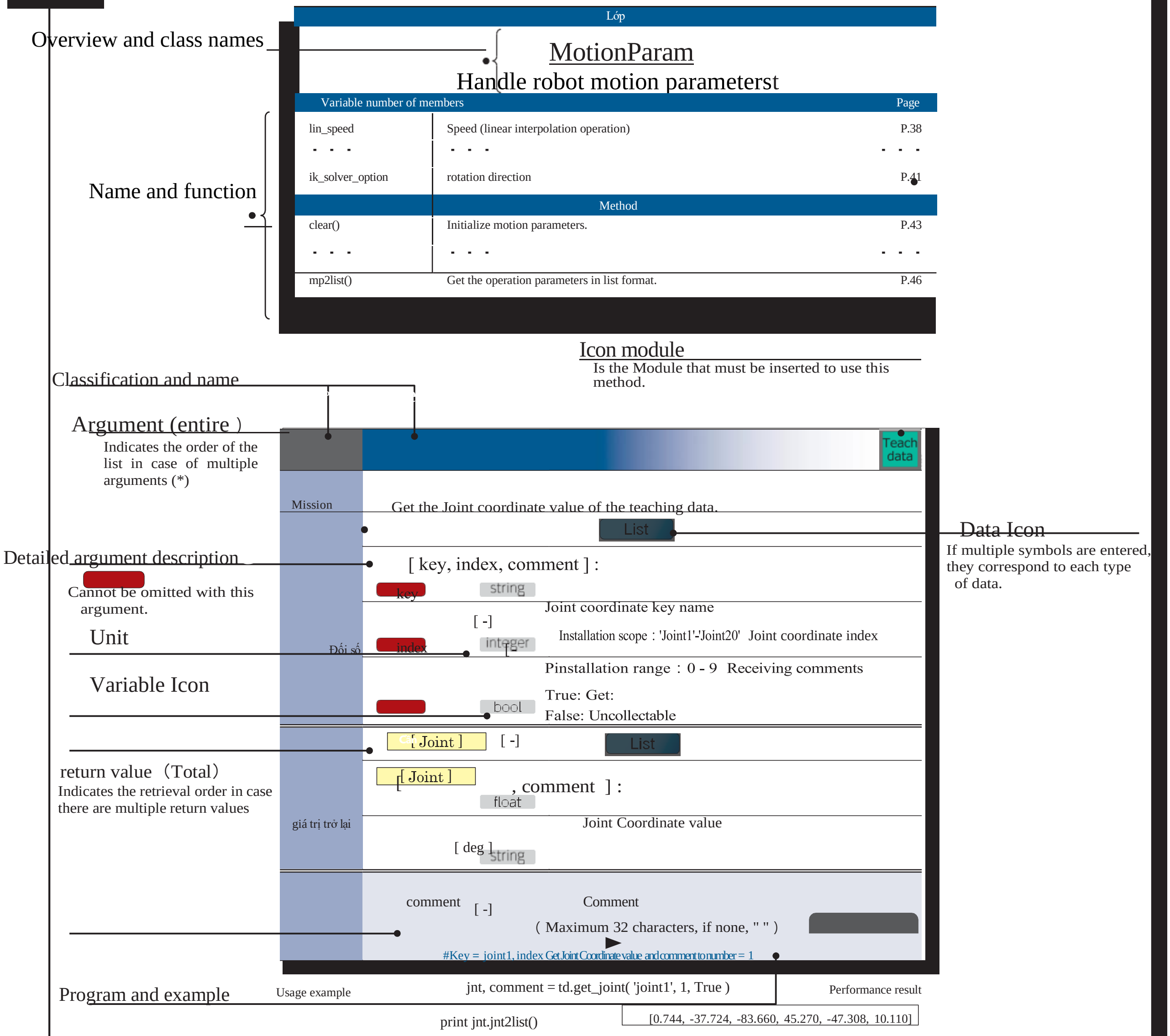
Performance result（This example shows Match Type data）

```
# Check the performance result
[0.744, -37.724, -83.660, 45.270, -47.308, 10.110]
```

（In the following explanation, 「print…」 is omitted, only the performance result is displayed.）

point

How to view the robots library



Overview and class names

Lớp

MotionParam

Handle robot motion parameterst

Name and function

Variable number of members		Page
lin_speed	Speed (linear interpolation operation)	P.38
・・・	・・・	・・・
ik_solver_option	rotation direction	P.41
Method		
clear()	Initialize motion parameters.	P.43
・・・	・・・	・・・
mp2list()	Get the operation parameters in list format.	P.46

Icon module

Is the Module that must be inserted to use this method.

Classification and name

Argument (entire)

Indicates the order of the list in case of multiple arguments (*)

Teach data

Mission

Get the Joint coordinate value of the teaching data.

List

Data Icon

If multiple symbols are entered, they correspond to each type of data.

Detailed argument description

Cannot be omitted with this argument.

Unit

Variable Icon

Đối số

[key, index, comment] :

key string [-] Joint coordinate key name

Installation scope : 'Joint1'-'Joint20' Joint coordinate index

index integer

Pinstallation range : 0 - 9 Receiving comments

bool True: Get:

False: Uncollectable

[Joint] [-] List

return value（Total）

Indicates the retrieval order in case there are multiple return values

giá trị trở lại

[[Joint] float , comment] :

Joint Coordinate value

[deg] string

comment [-] Comment

(Maximum 32 characters, if none, " ")

#Key = joint1, index GetJointCoordinate value and comment to number = 1

Program and example

Usage example

```
jnt, comment = td.get_joint( 'joint1', 1, True )
print jnt.jnt2list()
```

Performance result

```
[0.744, -37.724, -83.660, 45.270, -47.308, 10.110]
```

[Position] [Joint] [MotionParam] Result ☞

Here is an example after implementing this method. Can be tested using print statements.

*) Can be listed. Please refer to the data type for details P.3)

Major Icon

・ : Unomittable arguments

Icon of data type

- List : List
- Tuple : Set
- Dict : Dictionary
- Vari. No : Variable argument
- Keyword : Key words

Icon of variable type

- long : Long
- string : String
- integer : Integer
- float : float
- bool : bool

Unit

- [mm]
- [deg]
- [s]
- [-] (No units)

Icon of data type

- [Position] : Type of position
  [x, y, z, rz, ry, rx, parent, posture, multiturn]
- [Joint] : Match type
  [j1, j2, j3, j4, j5, j6,]
- [MotionParam] : MotionParam type
  [lin_speed, jnt_speed, acctime, dacctime, posture,passm, overlap, zone, pose_speed, ik_solver_option]

1. Module: i611_MCS

Class

Base

Determine the world coordinate system.(*)

Variable of members	
—	—

Method	
—	—

(*) This is a pseudo class used in the Position class and the Coordinate class.

Created functions	Base() i611 MCS
Function	Create instances of the pseudo-class for use in the Position and Coordinate classes of the world coordinate system.
Argument	None
Return value	Self Reference (Base Class Object)
Usage example	_BASE=Base()

Classes

Coordinate

Handles world coordinate system objects.

Variable for members	
x, y, z	Position (absolute coordinates)
rz, ry, rx	Pose (Z-Y-X Euler angle)
parent	Set to use world coordinate system.

Method		
b2g()	Converted from base coordinate system to world coordinate system.	P.25
clear()	Initialize the world coordinate system object.	P.25
copy()	Copy world coordinate system.	P.25
g2b()	Converted from world coordinate system to base coordinate system.	P.26
inv()	Create an object of the Coordinate class to perform the inverse transformation.	P.26
replace()	Replace the world coordinate system object. (self-updated)	P.27
shift()	Moves objects in world coordinates. (self-updated)	P.27

Created functions

Coordinate()

i611 MCS

Creates an instance of the Coordinate class defined in the world coordinate system.

Argument

[x, y, z, rz, ry, rx, parent] : List Keyword

If numbers are omitted, the base value will be set..

Original value：[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, _BASE]

Return value

Self-referencing (Coordinate Object)

Usage examples

#Example 1: Omit arguments. (Original value setting) Performance result

```
CO1=Coordinate()                          →  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >]
```

# Example 2：Keywords (All)

```
CO2=Coordinate( x=1, y=2, z=3, rz=1, ry=2, rx=3, parent=_BASE )
                                          →  [1.0, 2.0, 3.0, 1.0, 2.0, 3.0, < ... >]
```

# Example 3：Keywords (6 words)

```
CO3=Coordinate( x=1, y=2, z=3, rz=1, ry=2, rx=3 )
                                          →  [1.0, 2.0, 3.0, 1.0, 2.0, 3.0, < ... >]
```

# Example 4：Keyword (1word), Unknown value will become the original value.

```
CO4=Coordinate( x=10 )
                                          →  [10.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >]
```

Method	b2g()	i611 MCS
Function	Convert from base coordinate system to world coordinate system.다 .	
Argument	[x, y, z, rx, ry, rz] : List (All arguments are needed to change coordinates.)	
	x, y, z Need [mm] float	Unit (default coordinate system)
	rx, ry, rz Need [deg] float	Pose (Euler angles are based on Z-Y-X)
Return value	[X, Y, Z, RX, RY, RZ] : List	
	X, Y,Z [mm] float	Unit (default coordinate system)
	RX, RY,RZ [deg] float	Pose (Euler angles are based on Z-Y-X)

Usage examples	# Specify all arguments in a list.
	CO1=Coordinate()
	CO1.b2g(1, 2, 3, 4, 5, 6)
		[1.0,2.0, 3.0, 4.0, 5.0, 6.0]

Method	clear()	i611 MCS
Function	Initializes the object in the world coordinate system. Original value：[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, _BASE]	
Argument	None	
Return value	None	

Usage example	# Initial the world coordinate system.
	CO1=Coordinate(1, 2, 3, 4, 5, 6,_BASE)
	CO1.clear()
		Performance result
		[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >]

Method	copy()	i611 MCS
Function	Copy of the world coordinate system.	
Argument	None	
Return value	New copied coordinate system object (Coordinate object))	

Usage example	#Copy of the coordinate system object
	Random coordinates CO1. # Declare CO1.
	CO1=Coordinate(x=1, y=2, z=3, rz=4, ry=5, rx=6, parent=_BASE)
	# Copy CO1 to the new Coordinate object CO1C.
	CO1C=CO1.copy()
		Result
		CO1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >]
		↓
		copy()
		↓ New point object：CO1C
		CO1C : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >]

Method	g2b()		i611 MCS
Function	Convert from world coordinate system to base coordinate system.		
Argument	[X, Y, Z, RZ, RY, RX] : List (All arguments are needed to change coordinates.)		
	X, Y,Z Cần [mm] float	Unit (World coordinate system)	
	RZ, RY,RX Cần [deg] float	Pose (Euler angle based on Z-Y-X)	
Return value	[x, y, z, rz, ry, rx] : List		
	x, y, z [mm] float	Unit (default coordinate system)	
	rz, ry, rx [deg] float	Pose (Euler angle based on Z-Y-X)	

Usage value	# Specify all arguments in a list.
	CO1=Coordinate()
	CO1.g2b(1, 2, 3, 4, 5, 6) →
		Perform.result
		[1.0,2.0, 3.0, 4.0, 5.0, 6.0]	

Method	inv()	i611 MCS
Function	Create an object of the Coordinate class to perform the inverse transformation.	
Argument	None	
Return value	A new Coordinate object is created	

Usage value	# The teaching point in an arbitrary coordinate system must be available in advance.
	# Declare CO1.
	CO1=Coordinate() →
	# Updates equally to the value specified as argument.
	CO1.replace(1,2, 3, 4, 5, 6) →
	CO2 = CO1.inv()
		Perform. result
		[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >]
		[1.0,2.0, 3.0, 4.0, 5.0, 6.0, < ... >]

Method	replace()	i611 MCS
Function	Replace the world coordinate system object. (self-updated)	
Argument	[x, y, z, rz, ry, rx, parent] :	List Keyword
	x, y,z [mm]	Unit (World coordinate system)
	rz, ry, rx [deg]	Pose (Euler angle based on Z-Y-X)
	parent [-] float	Install for world coordinate system usage
	If the argument is omitted, the default value will be set. Initial value : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, _BASE]	
Return value	Self-reference (Coordinate object)	

Usage examples	# Example 1: List (2 items) Any undefined value will become the original value.
	CO2=Coordinate() CO2.replace(7, 8)
	# Example 2: Key words (2 words) Any undefined value will become the original value.
	CO3=Coordinate()
	CO3.replace(x=1, rx=6)
		Result
		→ [7.0, 8.0, 0.0, 0.0, 0.0, 0.0, < ... >]
		→ [1.0, 0.0, 0.0, 0.0, 0.0, 6.0, < ... >]

Method	shift()	i611 MCS
Function	Move objects in world coordinates. (self-updated)	
Argument	[dx, dy, dz, drz, dry, drx] :	List Keyword
	dx, dy, dz [mm] float	Unit (World coordinate system)
	drz, dry, drx [deg] float	Pose (Euler angle based on Z-Y-X)
	If the argument is omitted, it will not move.	
Return value	Self-reference (Coordinate object)	

Usage examples	# Example 1: List (2 items)
	CO1 = Coordinate()
	CO1.replace(1,2,3,4,5,6)
	CO1.shift(80, 70)
	# Example 2 : Key word (1word)
	CO2 = Coordinate()
	CO2.replace(1,2,3,4,5,6)
	CO2.shift(dx=80)
		Result
		→ [1.0,2.0,3.0,4.0,5.0,6.0]
		→ [81.0, 72.0, 3.0, 4.0, 5.0, 6.0]
		→ [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
		→ [81.0, 2.0, 3.0, 4.0, 5.0, 6.0]

Classes

Position

Handle coordinate values Position (*) of the world coordinate system.

Membership variable	
—	—

Method		
clear()	Initialize the Position coordinate value.	P.29
copy()	Copy the Position coordinate value.	P.29
has_mt()	Check cross counter information.	P.30
offset()	Add the offset coordinate value to the Position coordinate value. (create new object while maintaining itself.)	P.30
pos2dict()	Get Position coordinate value in dictionary format.	P.31
pos2list()	Enter Position coordinate values in list format.	P.31
position()	Convert the Position coordinate value to the original coordinate system and import in list format.	P.31
replace()	Replace Position coordinate values. (self-updated)	P.32
shift()	Moving Position of the coordinate value. (self-updated)	P.32

*) These are the coordinates that will be processed in the robot program.
We can handle not only taught coordinates but also untaught coordinates.

Created function

Position()

i611 MCS

Function

Create a version that defines teaching points in world coordinates .

Argument

[Position] [x, y, z, rz, ry, rx, parent, posture, multiturn] : List Keyword

If the argument is omitted, the default value will be set.

Original value：[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, _BASE, -1, 0xFF000000]

- When creating a second or subsequent version, the most recent value will be applied
- Call clear() to reset the recent value and return to the original value.

Usage examples

```
# Example  1: Ignore the argument. (Setting original value)                    Result
    P1=Position() ────────────────→ [0.0,0.0,0.0,0.0,0.0,0.0, < ...>,-1,0xFF000000]

# Eaxample 2: Key words

    P2=Position( x=1, y=2, z=3, rz=1, ry=2, rx=3, parent=_BASE, posture=1 )
     └────────────────────────────→ [1.0, 2.0, 3.0, 1.0, 2.0, 3.0, < ... >, 1, 0xFF000000]
# Eaxample 3: The value with no keyword (1 word) specified becomes the original value.
    P3=Position(rx=103)      ──────→ [0.0, 0.0, 0.0, 0.0, 0.0, 103.0, < ... >, -1,0xFF000000]
```

[Position] For more information, see MotionParam Class (Page 37).

Method	clear() i611 MCS
Function	Initialize the Position coordinate value. Original value：[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, _BASE,-1, 0xFF000000]
Argument	None
Return value	None

Usage example	# Initialize the Random Position object P1
	# P1 Declaration.
	P1=Position(1,2,3,4,5,6)
	# Initialize P1.
	P1.clear()
	Result
	P1 : [1.0,2.0, 3.0,4.0,5.0,6.0, < ...>,-1,0xFF000000]
	↓ clear()
	↓
	P1 : [0.0, 0.0,0.0,0.0,0.0,0.0, < ...>,-1,0xFF000000]

Original value of the posture parameter

posture is a 3-bit value. The original value is “-1”. Any new input will be overwritten.
If not specified, the previous setting value is inherited.

Method	copy() i611 MCS
Function	Copy the Position coordinate value.
Argument	None
Return value	Copy new teaching point object (Position Object)

Usage examples	# Copy any P1 Position object.
	# P1 Declaration.
	P1=Position(x=1, y=2, z=3, rz=4, ry=5, rx=6, 0xFF000000)
	# Copy P1 to new Position object P1C.
	P1C=P1.copy()
	Result
	P1 : [1.0,2.0,3.0,4.0,5.0,6.0,< ... >,-1,0xFF000000]
	↓ copy()
	↓ New point object：P1C
	P1C：[1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >, -1, 0xFF000000]

Method	has_mt()		i611 MCS
Function	Check cross counter information .		
Argument	None		
Return value	res0 [-] bool	Presence or absence of cross counter information True : Yes False : No	
Usage example			

If class Position is created in the program as a parameter with cross counter information omitted, data of type Position without cross counter information will be created

Method	offset()		i611 MCS
Function	Add the offset coordinate value to the Position coordinate value. (create a new object while maintaining the object itself.)		
Argument	[dx, dy, dz, drz, dry, drx] : List Keyword		
	dx, dy, dz [mm] float	Offset quantity of Position (Descartes Coordinates)	
	drz, dry, drx float [deg]	Amount of posture offset (Euler angle series Z-Y-X)	
	If the argument is omitted, Offset is not set.		
Return value	Refer to the maintained Offset value. (Position object)		

Usage examples

```
# Add offset coordinates to the P1 Random Position object.

.# P1 Declaration.
    P1=Position( x=1, y=2, z=3, rz=4, ry=5, rx=6 )

    # Create a new Position object P1ofs offset from P1.
    P1ofs=P1.offset(  dx=100,  drz=-10  )
```

Result

```
P1      : [1.0,2.0,3.0,4.0,5.0,6.0,...]
↓
        offset()
↓
Original Offset:P1
P1      : [1.0,2.0,3.0,4.0,5.0,6.0,...]

New point object : P1ofs
P1ofs : [101.0, 2.0, 3.0, -6.0, 5.0, 6.0, ... ]
```

```
# In case you specify only the values you want to Offset on
the list in P1, create a new Position object
P1ofs.P1ofs=P1.offset( 100, 0, 0, -10 )
```

```
New point object : P1ofs
P1ofs : [101.0, 2.0, 3.0, -6.0, 5.0, 6.0, ... ]
```

Method	pos2dict() i611 MCS
Function	Enter the Position coordinate value in dictionary format.(*1)
Argument	None
Return value	[Position] : Dict

Usage examples	# Render an arbitrary P1 Position object in dictionary format (*2)
	# P1 Declaration.
	P1=Position(1, 2, 3, 4, 5, 6)
	# Export P1 in dictionary format.
	P1.pos2dict()
	Result
	{'parent':<...>,'rx':6.0,'ry':5.0,'rz':4.0,'y':2.0,'x':1.0,'z':3.0,'posture':-1,'multiturn':0xFF000000}

Method	pos2list() i611 MCS
Function	Enter the Position coordinate value in list format.(*1)
Argument	None
Return value	[Position] : List

Usage examples	#Render an arbitrary P1 Position object in list format.(*2)
	# P1 Declaration.
	P1=Position(1, 2, 3, 4, 5, 6)
	# P1 outputs in list format.
	P1.pos2list()
	Result
	[1.0,2.0,3.0,4.0,5.0,6.0,< ... >,-1,0xFF000000]

Method	position() i611 MCS
Function	Convert the Position coordinate value to the original coordinate system and import in list format.(*1)
Argument	None
Return value	[Position] : List

Usage examples	#Render an arbitrary P1 Position object in list format.(*2)
	# P1 Declaration.
	P1=Position(1, 2, 3, 4, 5, 6)
	# P1 outputs in list format.
	P1.position()
	Result
	[1.0,2.0,3.0,4.0,5.0,6.0,< ... >,-1,0xFF000000]

* 1) When setting i611Robot.use_mt (True), the multiturn entry will be added to the return value.

* If i611Robot.use_mt (False) (original value) will not be added.

* 2)This example does not have cross counter information.

Method	replace()	i611 MCS
Function	Replace the Position coordinate value.[*1] (self-updated)	
Argument	[Position] : List Keyword	
	If the argument is omitted, the value will not be updated.	
Return value	Self-reference (Position object)	

Usage examples	# Replace arbitrary Position objects P1, P2, and P3.(*2)
	# Declaration P1, P2, P3.
	P1=Position()
	P2=Position()
	P3=Position()
		Result

	# Example 1: When specifying as a list
	P1.replace(1, 2, 3, 4, 5, 6)
		P1 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >, -1, 0xFF000000]
		↓ replace()
		P1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >, -1, 0xFF000000]

	# Example 2: When specifying two variable arguments
	P2.replace(7, 8)
		P2 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >, -1, 0xFF000000]
		↓ replace()
		P2 : [7.0, 8.0, 0.0, 0.0, 0.0, 0.0, < ... >, -1, 0xFF000000]

	# Eaxmple 3: When specified with two keywords
	P3.replace(x=1, rx=6)
		P3 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >, -1, 0xFF000000]
		↓ replace()
		P3 : [1.0, 0.0, 0.0, 0.0, 0.0, 6.0, < ... >, -1, 0xFF000000]

Method	shift()		i611 MCS
Function	Move the Position coordinate value.[*1] (Self-updated)		
Argument	[dx, dy, dz, drz, dry, drx] : List Keyword		
	dx, dy, dz [mm] float	Amount of movement in position (Descartes Coordinates)	
	drz, dry, drx [deg] float	amount of movement in the pose (Euler angles based on Z-Y-X)	
	Do not move if arguments are omitted		
Return value	Self-reference (Position object)		
Usage examples	# Move Random position P1 object.[*2]	Result	

	# P1 declaration.
	P1=Position(1, 2, 3, 4, 5, 6)
	# Update P1 to the moved location.
	P1.shift(dx=80)
		P1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >, -1, 0xFF000000]
		↓
		shift()
		↓
		First object:P1
		P1 : [81.0, 2.0, 3.0, 4.0, 5.0, 6.0, < ... >, -1, 0xFF000000]	

* 1) When setting i611Robot.use_mt (True), the multiturn entry will be added to the return value.

* If i611Robot.use_mt (False) (initial value) will not be added.

* 2) This example does not have cross counter information.

Class

Joint

Process the angle data of joint coordinate values in the joint coordinate system (*)

Member variable		
j1, j2, j3, j4, j5, j6	Joint angle	
Method		
clear()	Initialize the joint coordinate value .	P.34
copy()	Copy the joint coordinate value.	P.34
jnt2dict()	Enter the joint coordinate value in dictionary format.	P.34
jnt2list()	Enter the joint coordinate value in list format.	P.35
offset()	Add the offset coordinate value to the joint coordinate value. (create a new object while maintaining the object itself.)	P.35
replace()	Replace joint coordinate values. (self-updated.)	P.36
shift()	Move joint coordinate values. (self-updated.)	P.36

*) Joint coordinate value
These are the coordinates processed by the program.
It is possible to handle only taught coordinates but also untaught coordinates.

Created function	Joint()
Function	Create an entity that defines the teaching point of the Joint coordinate system.
Argument	[Joint] [j1, j2, j3, j4, j5, j6] : If the argument is omitted, the default value will be set. Original value：[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Return value	Self-refenrence (Joint object)

Usage examples

```
# Example 1: Ignore the argument. (Set original value.)
                                                        Result quả
    J1=Joint()                                        ► [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

# Example 2: List
    J2=Joint( 1, 1, 1, 1, 1, 1 )                      ► [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

# Example 3: Key words (6)
    J3=Joint( j1=1, j2=2, j3=3, j4=4, j5=5, j6=6 )    ► [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]

# Example 4: Key words (1). Any undefined value will become the
original value.
    J4=Joint( j6 = 6 )                                ► [0.0, 0.0, 0.0, 0.0, 0.0, 6.0]
```

Method	clear()	i611 MCS
Function	Initialize the joint coordinate value. Original value ：[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	
Argument	None	
Return value	None	

Usage example	# Initialize random Joint object J1.
	# J1 declaration.
	J1=Joint(1,2,3,4,5,6)
	# Initialize J1.
	J1.clear()
		Result
		J1 : [1.0,2.0, 3.0, 4.0,5.0, 6.0]
		↓
		clear()
		↓
		J1 : [0.0,0.0, 0.0, 0.0,0.0, 0.0]

Method	copy()	i611 MCS
Function	Copy the joint coordinate value.	
Argument	None	
Return value	Copy the new Joint coordinate object (Joint object)	

Usage examples	#Copy any Joint J1 object.
	# J1 declaration.
	J1=Joint(j1=1, j2=2, j3=3, j4=4, j5=5, j6=6)
	# Copy J1 to new common object J1C.
	J1C=J1.copy()
		Result
		J1 : [1.0,2.0, 3.0,4.0, 5.0,6.0]
		↓
		copy()
		↓ new point object ：J1C
		J1C ：[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]

Method	jnt2dict()	i611 MCS
Function	Enter the joint coordinate value in dictionary format.	
Argument	None	
Return value	[Joint] : Dict	

Usage examples	# Export an arbitrary Joint object J1 in dictionary format..
	# J1 declaration.
	J1=Joint(1, 2, 3, 4, 5, 6)
	# Export J1 in dictionary format.
	J1.jnt2dict()
		Result
		► {'j4': 4.0, 'j5': 5.0, 'j6': 6.0, 'j1': 1.0, 'j2': 2.0, 'j3': 3.0}

Method	jnt2list() i611 MCS
Function	Enter the joint coordinate value in list format.
Argument	None
Return value	[Joint] : List

Usage examples	# Export arbitrary Joint object J1 in list format
	. # J1 declaration.
	J1=Joint(1, 2, 3, 4, 5, 6)
	# Export J1 in list format.
	J1.jnt2list()
	Result
	► [1.0,2.0, 3.0, 4.0, 5.0, 6.0]

Method	offset() i611 MCS
Function	Add offset coordinates to joint coordinates. (Create a new object while maintaining the object itself.)
Argument	[dj1, dj2, dj3, dj4, dj5, dj6] : List Keyword Joint angle compensation amount dj1, dj2, dj3, dj4, dj5, dj6 [deg] : float Original value : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0] If the argument is omitted then the offset is not set. New angle object for each axis (Joint object)
Return value	

Usage examples	# Add J1 offset coordinates to the random Joint object..
	# J1 declaration.
	J1=Joint(1, 2, 3, 4, 5, 6)
	# Create a new joint object J1ofs offset from J1.
	J1ofs=J1.offset(dj1=80, dj6=-30)
	# In case of only specifying a number as the value you want to Offset at J1
	# How to set dj1, dj2 in list form
	J2ofs=J1.offset(80, -30)
	Result
	J1 :[1.0,2.0,3.0,4.0,5.0,6.0]
	↓
	offset()
	↓
	First object:J1
	J1 : [1.0,2.0,3.0,4.0,5.0,6.0]
	New point object : : J1ofs
	J1ofs : [81.0, 2.0, 3.0, 4.0, 5.0, -24.0]
	New point object : J2ofs J2ofs : [81.0, - 28.0, 3.0, 4.0, 5.0, 6.0]

Method	replace()	i611 MCS
Function	Replace the joint coordinate value. (self-updated)	
Argument	[Joint] : List Keyword If the argument is omitted, the value will not be updated.	
Return value	Self-reference (Joint objects)	

Usage examples

```
# Replace arbitrary Joint objects J1, J2, J3.

# Declaration for J1, J2, J3.
    J1=Joint()
    J2=Joint()
    J3=Joint()

    # Example 1: When specified in the list
    J1.replace(1,2,3,4,5,6)

    # Example 2: When specifying two variable arguments
    J2.replace(7,8)

    # Example 3: When specified with two keywords

    J3.replace(j1=1,j6=6)
```

Result

```
J1 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
↓      replace()
J1 : [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]

J2 : [0.0,0.0,0.0,0.0,0.0,0.0]
↓      replace()
J2 : [7.0,8.0,0.0,0.0,0.0,0.0]

J3 : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
↓      replace()
J3 : [1.0, 0.0, 0.0, 0.0, 0.0, 6.0]
```

Method	shift()		i611 MCS
Function	Move joint coordinate values (Self-updated)		
Argument	[dj1, dj2, dj3, dj4, dj5, dj6] : List Keyword		
	dj1, dj2, dj3, dj4, dj5, dj6 [deg] float	Degree of movement of the joint angle Original value : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	
	If the argument is omitted, the amount of movement will not be set.		
Return value	Self-reference (Joint objects)		

Usage examples

```
#Randomly move common object J1.
    # J1 declaration.
    J1.replace( 1, 2, 3, 4, 5, 6 )

    # Move J1 to update itself.
    J1.shift(dj1=80)
```

Result

```
J1 : [1.0,2.0, 3.0, 4.0,5.0, 6.0]
↓
      shift()
↓
First object: J1
J1 : [81.0, 2.0, 3.0,4.0,5.0, 6.0]
```

Class

MotionParam

Handle robot motion parameters.

Membership variable		
lin_speed	Speed (Line operation (linear interpolation operation))	P.38
jnt_speed	Speed (PTP operation, joint operation, optimal linear interpolation operation)	P.38
acctime	Acceleration time	P.38
dacctime	Deceleration time	P.39
posture	Posture	P.39
passm	Operate Pass	P.39
overlap	Gaps overlap	P.40
zone	Complete range of positioning	P.40
pose_speed	Speed (pose interpolated motion)(*)	P.41
ik_solver_option	Direction of rotation	P.41
Method		
clear()	Initialize motion parameters.	P.43
confdefault()	Set the initial value of the operating parameter.	P.43
copy()	Copy operating parameters.	P.44
motionparam()	Set operating parameters	P.45
mp2dict()	Enter action parameters in dictionary format.	P.46
mp2list()	Enter action parameters in list format.	P.46

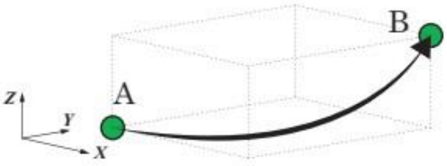
*）The Manipulator head changes direction and operates, then sets the upper limit of the machine head's Euler angular motion speed.

Setting acceleration/deceleration too fast may cause the robot to vibrate.

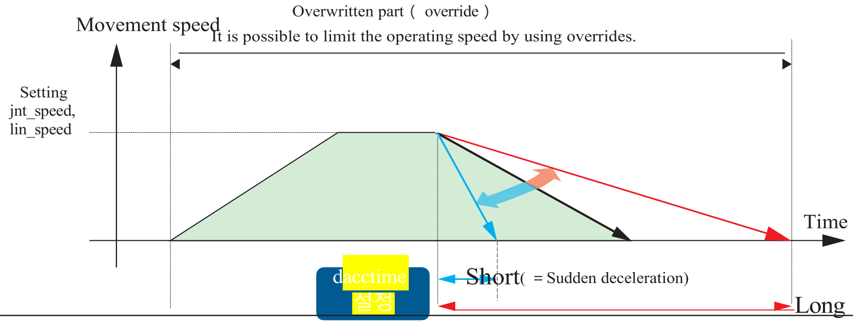
Initial parameter adjustment must be done by gentle acceleration and deceleration to ensure no vibration occurs in the robot.

Member variable of type MotionParam

Member variable	lin_speed		
Function	Speed (Line operation (linear interpolation operation))	Unit/Type Original value	5.0 [mm/s] float
Supplement	Is the act of moving at a constant speed so that the trajectory to the destination is a straight line while simultaneously controlling the X-Y-Z axes synchronously.		Z Y X A B

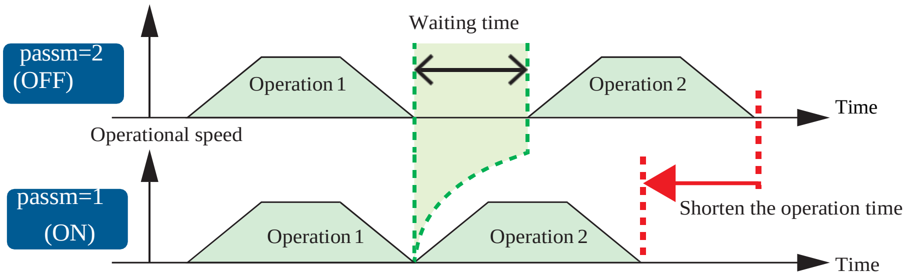
Member variable	jnt_speed		
Function	Speed (PTP operation, joint operation, optimal linear interpolation operation)	Unit/type Original value	5.0 [%]
Supplement	All joints move at a constant speed and angle toward the target coordinates. Motion moves in a smooth curve. The optimal linear interpolation motion also sets the speed according to jnt_speed. Set to % of maximum speed for numeric change.		PTP operation, Joint operation  Optimal linear interpolation operation A B

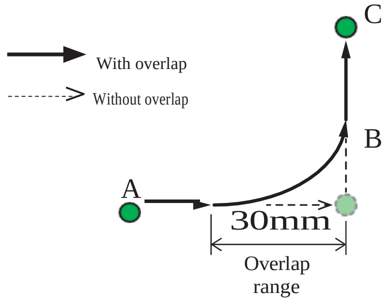
Member variable	acctime		
Function	Acceleration time	Unit/Type Original value	0.4 [s] float
Supplement	Set the time to reach the set speed in lin_speed, jnt_speed. Movement speed Overwritten part (override()) It is possible to limit the operating speed using overrides. Setting jnt_speed, lin_speed Sudden acceleration Time Short (= Sudden acceleration) Long		

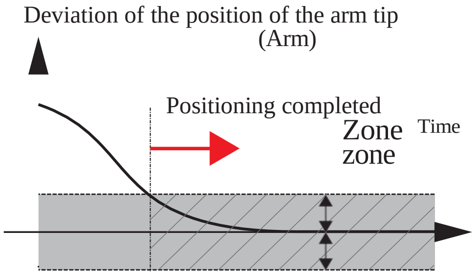
Member variable	dacctime		
Function	Deceleration time	Unit/Type Original value	0.4 [s] float
Supplement	Time from speed set in lin_speed, jnt_speed to deceleration and stopping at target coordinates.  Movement speed Overwritten part（ override ） It is possible to limit the operating speed by using overrides. Setting jnt_speed, lin_speed Time dacctime 설정 Short(＝Sudden deceleration) Long		

Member variable	posture		
Function	Posture	Unit/Type Original value	2 [-] integer
Supplement	Installation range：0 -7 The position of the manipulator is defined as: ①Arm unit ②Joint angle˙ Example: Original value「2」		

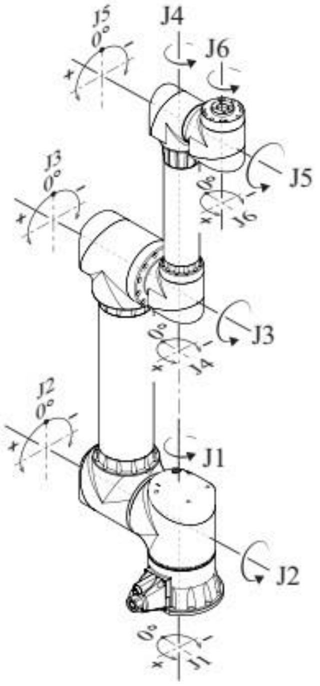
For more information about posture, please refer to the book “Coordinate Systems and posture”.

Member variable	passm		
Function	Pass activity	Unit/Type Original value	2 [-] integer
Supplement	Installation value：1=ON , 2=OFF By enabling the passm operation parameter, you can bypass the waiting time between operations and shorten the overall operation time.  passm=2 (OFF) Waiting time Operation 1 Operation 2 Time Operational speed passm=1 (ON) Shorten the operation time Operation 1 Operation 2 Time If asyncm(1) overlap (predictive read) is used, it will always operate in the ON state regardless of the passm parameter setting.		

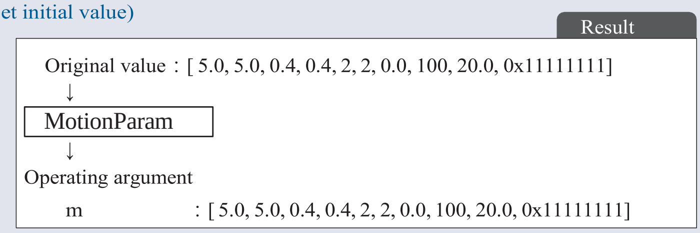
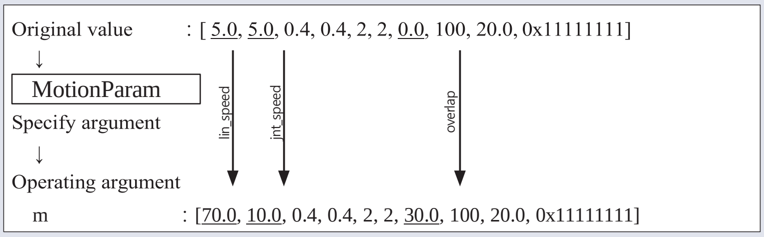
Member variable	overlap		
Function	Overlapping distance	Unit/Type g Original value	0.0 [mm] float
Supplement	As soon as the target point (B) is approached, the next overlapping movement will begin.. The robot can be moved smoothly without stopping motion at the prepared transition point (B) to perform obstacle avoidance actions.	 C With overlap Without overlap B A 30mm Overlap range Example: Overlap = 30mm	

Member variable	zone		
Function	Complete range of positioning	Unit/Type g Original value	100 [pulse] integer
Supplement	Set the encoder pulse range so that the end of the robot arm approaches the target point and determine whether positioning is complete or not	 Deviation of the position of the arm tip (Arm) Positioning completed Zone zone Time	

Member variable	pose_speed		
Function	Speed (Postural interpolation motion)	Unit/Type Original value	20 [%] float
Supplement	Installation value ：100% = 45deg/s When the manipulator head operates and changes direction, set the upper limit of the movement speed of the final Euler angle .		

Member variable	ik_solver_option		
Function	Rotation direction	Unit/Type Original value	0x11111111 [-] long
Supplement	Specify the direction of rotation of each axis when moving to the coordinates specified in the Position type or converting from the Position type to the Joint type. 0 x 1 1 1 1 1 1 1 1 (Rsv.) J6 J5 J4 J3 J2 J1 Setting J1 - J6 0 : This is a rotation that does not use information from shortcuts. 1 : Use information from the multiturn parameter. 2 : Rotate toward direction +. 3 : Rotate toward direction -. Refer to the figure on the right for the +, direction -  J5 0° J4 J6 J5 J3 0° J6 J3 J4 J2 0° J1 J2 0° J1 + -		

Created function	MotionParam() i611 MCS
Function	Create an instance of the robot's motion parameters
Argument	[MotionParam] [lin_speed,jnt_speed,acctime,dacctime,posture,passm,overlap,zone,pose_speed,ik_solver_option] List Keyword If the argument is omitted, the default value will be set. ~~Original value : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20, 0x11111111]~~
Return value	Self-reference (MotionParam object)

Usage examples	#Example 1: Ignore the argument. (Set initial value)
	m=MotionParam()
	
	Result
	Original value : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
	↓
	MotionParam
	↓
	Operating argument
	m : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
	# Example 2: Set the action parameter specified in the argument.
	m=MotionParam(lin_speed=70, jnt_speed=10, overlap=30)
	
	Original value : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
	↓
	MotionParam
	Specify argument
	↓
	Operating argument
	m : [70.0, 10.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]

Method	clear() i611 MCS
Function	
Argument	None
Return value	None

Usage examples	#Initialize a random MotionParam object m.
	# Declaration m.
	m=MotionParam(lin_speed=70, jnt_speed=10, overlap=30)
	
	m.clear()
	
	Result
	m : [70.0, 1.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
	↓
	clear()
	↓
	m : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111] i611 MCS

Method	confdefault()
Function	Change the original value of the operating parameter.
Argument	[MotionParam] : Keyword If the argument is omitted, the default value will be set. Original value : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
Return value	None
Usage examples	m.clear() m.confdefault(lin_speed=70,overlap=30) m.clear() Kết quả Original value before change : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111] lin_speed overlap confdefault ↓ Original value after change : [70.0, 5.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]

보안

Original value setting time of operating parameters

The initial value change setting in the confdefault() method will be reflected the next time when a version of the MotionParam class is created, copied, or deleted.

Method	copy() i611 MCS
Function	Copy operating parameters.
Argument	[MotionParam] List Keyword If specified the argument will change the currently set operating parameter for copying. If the argument is omitted, the current setting operation parameter value will be copied.
Return value	Copied new motion parameters (MotionParam object)

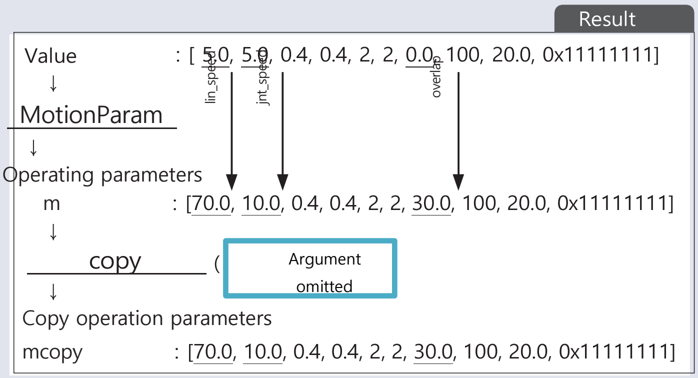
Usage examples

Motion parameters must be preset in MotionParam.

```
m=MotionParam( jnt_speed=10, lin_speed=70, overlap=30 )
```

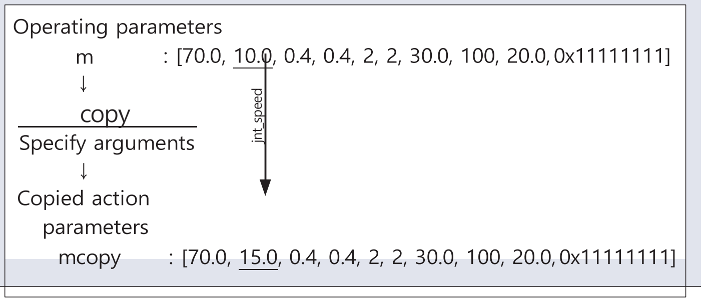
Example 1: Ignore the argument. (The operating parameters are currently set)

```
mcopy = m.copy()
```


Result
Value : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
↓
MotionParam
↓
Operating parameters
m : [70.0, 10.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
↓
copy (
Argument omitted
↓
Copy operation parameters
mcopy : [70.0, 10.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
lin_speed
jnt_speed
overlap

Example 2 : Change and copy action parameters specified in arguments.

```
mcopy = m.copy( jnt_speed=15 )
```


Operating parameters
m : [70.0, 10.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
↓
copy
Specify arguments
↓
Copied action parameters
mcopy : [70.0, 15.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
jnt_speed

Method	motionparam() i611 MCS
Function	Set operating parameters.
Argument	[MotionParam] : List Keyword If the argument is omitted, the default value will be set. Original value : [5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
Return value	Self-reference (MotionParam object)

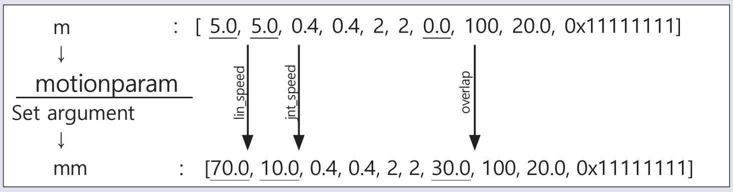
Usage examples

```
# Motion parameters must be preset in MotionParam.
    m=MotionParam()

# Example 1: Ignore the argument. (Set initial value)
```

Result

```
m  : [ 5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
↓
  motionparam  ( Argument is omitted )
↓
mm : [ 5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
```

```
# Example 2: Change operating parameters.
    mm=m.motionparam( lin_speed=70, jnt_speed=10, overlap=30 )
```



```
m              :  [ 5.0, 5.0, 0.4, 0.4, 2, 2, 0.0, 100, 20.0, 0x11111111]
↓
  motionparam
Set argument
↓
mm             :  [70.0, 10.0, 0.4, 0.4, 2, 2, 30.0, 100, 20.0, 0x11111111]
```

Method	mp2dict() i611 MCS
Function	Get action parameters in dictionary format.
Argument	None
Return value	[MotionParam] : Dict

Usage examples	# Example: Preset motion parameters in MotionParam.
	m=MotionParam(lin_speed=70, jnt_speed=10, overlap=30)
	modict = m.mp2dict()
	Result
	{'lin_speed': 70.0, 'zone': 100, 'acctime': 0.400, 'pose_speed': 20.0, 'dacctime': 0.400, 'overlap': 30.0, 'passm': 2, 'jnt_speed': 10.0, 'posture': 2}
	# Get the value directly.
	lin_speed = m.mp2dict()['lin_speed'] → lin_speed=70.0

Method	mp2list() i611 MCS
Function	Get operation parameters in list format.
Argument	None
Return value	[MotionParam] : List

Usage examples	# Example: Preset motion parameters in MotionParam.
	m=MotionParam(lin_speed=70, jnt_speed=10, overlap=30)
	Result
	molist=m.mp2list() → [70.0,10.0,0.400,0.400, 2, 2, 30.0,100,20.0]
	# Only receive the specified quantity.
	molist=m.mp2list()[0:2] → print molist
	[70.0,10.0]
	# Get value directly.
	lin_speed=m.mp2list()[0] → print lin_speed
	lin_speed=70.0

Class

i611Robot

Handling robot movements.

Member variable		
-	-	P.38

Method		
abort()	Stop the robot program.	P.50
adjust_mt()	Fix CC value when converting position type coordinates to string.	P.50
asyncm()	Set up the predicted motion part of the robot program.	P.51
cause_user_error()	A user-defined error arises.	P.52
changetool()	Select Tool Offset.	P.52
check_ready()	Check if the robot can drive itself or not.	P.53
close()	Disconnect from the robot.	P.54
disable_mdo()	Disable MDO action.	P.54
enable_interrupt()	Make exceptions for deceleration stops and emergency stops.	P.55
enable_mdo()	Enable MDO behavior.	P.56
exit()	Force to quit the robot program.	P.57
get_hw_info()	Get the model name and serial number.	P.57
get_system_port()	Get the status of system ports.	P.58
get_system_status()	Get system status and error ID.	P.59
getjnt()	Get the current position of the manipulator in Joint form.	P.59
getmotionparam()	Get current operating parameters.	P.60
getpos()	Get the operator's current position in the Location type.	P.60
home()	Move all axes to joint coordinates 0 degrees .	P.60
is_open()	Check the open status of i611Robot.	P.61
is_pause()	Check the pause status of the robot program.	P.61
join()	Wait for the predicted robot program operation to complete.	P.63
Joint2Position()	Convert joint coordinate values to position coordinate values.	P.63
line()	Perform linear interpolation motion.	P.64
MCS_version()	Download the robot library version.	P.65
motionparam()	Set operating parameters.	P.65
move()	Carry out PTP activities.	P.66
open()	Start connecting to the robot. (reinstall)	P.67
optline()	Linear interpolated motion is performed while shifting gears at optimal speed..	P.68
override()	Override	P.69
pause()	Pause the robot's movements.	P.69
Position2Joint()	Convert from Position coordinates to Joint coordinates.	P.70

(i611Robot method)

Method		
release_stopevent()	Reset an exception event that is occurring.	P.70
reljntmove()	Relative movement is performed in the joint coordinate system.	P.71
relline()	The relative linear interpolation operation is performed in the Descartes coordinate system.	P.72
restart	Emit a continue signal after a pause.	P.73
set_behavior()	Set the operation (action) when paused.	P.74
set_mdo()	Set up MDO behavior.	P.75
settool()	Set up Tool offset.	P.76
sleep()	Pause for a certain period of time.	P.77
stop()	Decelerate the robot to stop.	P.78
svoff()	Set servo to OFF.	P.78
svstat()	Get servo status.	P.79
toolmove()	Relative operation in the Tool coordinate system.	P.79
use_mt()	Set cross counter on/off.	P.80
user_hook()	Pause the robot program.	P.80
version()	Get system version.	P.81

Created function	i611Robot()	i611 MCS
Function	Create an instance of class i611.	
Argument	[host, port] : List Keyword	
	host [-] string	Destination IP address Initial value : '127.0.0.1'
	port [-] integer	Number of connection ports Initial value : 12345
	If you omit the argument, the default value will be set.	
Return value	Return to that class main object.	

Usage examples

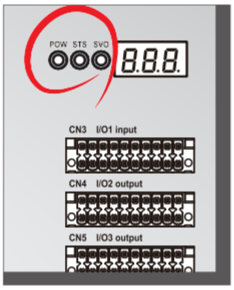
```
# Example 1: Omit argument (Set initial value.)
    rb=i611Robot()

# Example 2: List
    rb=i611Robot(  '127.0.0.1',  12345)

# Example 3: Keyword arguments (specify all)
    rb= i611Robot( host='127.1.1.1', port=10000 )

# Example 4: Keyword arguments (identify the server only)
    rb= i611Robot( host='127.1.1.1' )

# Example 5: Keyword arguments (specify port only)
    rb= i611Robot( port=3000 )
```

Restrict the i611Robot class

The i611Robot class cannot be used in the servo OFF state. (Including status at start-up, emergency stop, main power OFF and system error)
The servo ON state is also indicated by the LED (SVO) on the controller

The i611Robot entity can only be created once in a program.

When looping within a program, create an instance of class i611Robot before the loop.



Method	abort()	i611 MCS
Function	Stop the robot program.(*)	
Argument	None	
Return value	None	
Usage example	rb.abort()	

*) Only activated when the robot is moving.

Method	adjust_mt()		i611 MCS
Function	Fix cross counter value when converting position type coordinates to string.		
Argument	The string is converted to Position coordinates used as the origin [pos, str_x, str_y, str_z, str_rz, str_ry, str_rx] : List		
	pos	[Position]	: Position coordinates are used as the origin
	str_x	[-]	x coordinate string
	str_y	string [-]	y coordinate string
	str_z	string [-]	z coordinate string
	str_rz	string [-]	rz coordinate string
	str_ry	string [-]	ry coordinate string
	str_rx	string [-]	rz coordinate string
Return value	Giá trị bộ Cross-counting is corrected string		

Usage examples	rb =i611Robot()
	rb.open()
	pos = rb.getpos()
	pos_value = pos.position()
	pos_str = [str (round(x,2)) for x in pos_value[0:6]]
	new_mt = rb.adjust_mt(pos, pos_str[0], pos_str[1], pos_str[2], pos_str[3], pos_str[4], pos_str[5]) pos_str += [str (pos_value[7]), "0x%06X" % new_mt]
	print "Position String:%s" % pos_str		

Method	asyncm()	i611 MCS
Function	Set up the predicted motion part of the robot program.	
Argument	sw [-]	1：Program prediction operation is ON 2：Program prediction operation OFF (default value)
Return value	If successful：True [-] integer	
	If it fails: An exception arises . bool	

Usage examples	rb.line(p10) #Perform linear interpolation motion at teaching point p10..
	rb.asyncm(sw=1) # Program prediction operation ON (also available in rb.asyncm (1))
	rb.line(p20,p21) #Move to teaching points p20 and p21 in the order of performing linear interpolation..
	rb.join() # Wait for completion of the robot program's predicted actions.
	rb.asyncm(sw=2) #Program prediction operation OFF (also possible in rb.asyncm (2))
	...
	rb.close()	

Method	cause_user_error()	i611 MCS
Function	A user-defined error arises.	
Argument	[code, critical] : List Keyword	
	code Cần [-] integer	Error ID Installation range ： 1 -99
	critical [-] bool	True ： A user-defined error arises False ： A user-defined error occurred (default value)
Return value	None	

Usage value	# When a user-defined error occurs (Error ID: 19)
	rb.cause_user_error(19, False)
	# User-defined error - When a fatal error occurs (Error ID: 01)
	rb.cause_user_error(01, True)	

About user-defined errors

f you use the error ID (which cannot be optionally omitted) to execute the cause_user_error() method, the Robot program will end when an error occurs..

User-defined error	Error reset method	7-segment LED light
Serious	Reconnect the power source	
Error	' Error reset signal '	

Method	changetool()	i611 MCS
Function	Select Tool Offset.	
Argument	tid Need [-] integer	Tool number 0 ： Tool offset OFF. 1 -8 ： Select Tool offset.
Return value	Successful ： True [-] Unsucessful ： An exception occurred.	
Usage value	# 1 . Preset Tool Offset. rb.settool(1, 0.0, 0.0, 200.0, 0.0, 0.0, 0.0) # Set Tool No.1. # 2 . Select Tool Offset. # Example 1: Specify value rb.changetool(1) # Select Tool No.1 # Example 2: Key word rb.changetool(tid=1) # Select Tool No.1	

Method	check_ready()		i611 MCS
Function	Check whether the robot can work automatically or not.		
Argument	None		
Return value	res	[-] integer	Can drive automatically. 0 : Run the robot program
			Cannot drive automatically. can run programs that do not involve robot movements. 1 : Emergency stop in progress. 2 : servo is OFF. 3 : Not in automatic mode. (JOG bar is connected) 4 : Permission to operate is not granted. 5 : Other errors are occurring.
Usage example	res = i611Robot.check_ready() if res != 0: … # Displays messages and notifications of external output		

How to use check_ready method ()

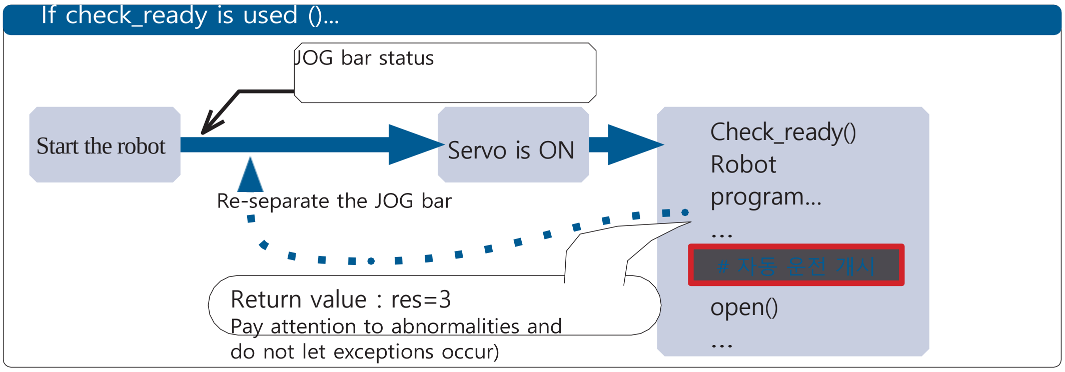
Is a method that confirms whether the open() method of the i611Robot class is implemented in the dictionary

If the return price is not '0', the robot will not work.

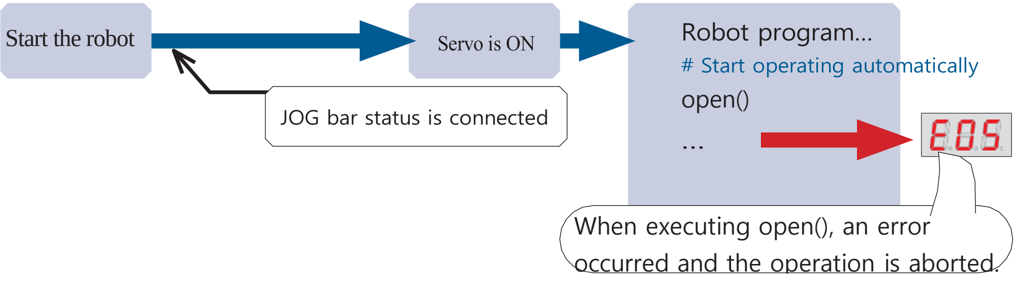
An exception occurred when executing open() or the constructor of class i611Robot. By checking the state in this Method, it is possible to prevent exceptions from occurring.

Example) If running the automatic operation program with the JOG bar connected to the controller...

If check_ready is used ()...



If check_ready() is not used …


Start the robot
Servo is ON
JOG bar status is connected
Robot program...
# Start operating automatically
open()
…
E05
When executing open(), an error
occurred and the operation is aborted.

Method	close() i611 MCS
Function	Disconnect from the robot.
Argument	None
Return value	Successful：True [-] (Only return when successful.)
Usage value	rb.close()

Go to exit () and close ()

Exit() processing is also implemented with close() processing.

Method	disable_mdo	
Function	Turn off MDO behavior.	
Argument	bitfield Cần [-] integer	MDO management number [*1] Set the bit corresponding to the disabled MOD management number. Installation range：0 -255
Return value	Successful：True [-] bool	
	Failed：An exception occurred.	

Usage examples	# Preset MDO operation settings (*2)
	rb.set_mdo(1, 23, 0, 1, 30)
	rb.set_mdo(8, 23, 1, 2, 10)
	rb.enable_mdo(129) # Activate MDO management No. 1 and 8
	# Turn off MDO behavior
	# Example 1: Specify value
	rb.disable_mdo(129) # Disable MDO management No. 1 and 8
	# Example 2: Key words
	rb.disable_mdo(bitfield=129) # Management MDO disabled No. 1 and 8	

* 1) For bit field settings, please refer to “Setting MDO management numbers using bit fields” on page 56.

* 2) See page 75 for more information about set_mdo() and page 56 for Enable_mdo()..

Method	enable_interrupt()		i611 MCS
Function	Set the exception occurrence for emergency stop and deceleration stop.		
Argument	[eid, enable] :	List	
	eid 🟥	[-] integer	Event code 0 : An exception occurred when entering the deceleration stop command during operation 1 : An exception occurs when an emergency stop signal occurs during operation 2 : An exception occurs when entering the deceleration stop command during a pause 3 : An exception occurs when entering the emergency stop state during a pause In case an exception is turned off, the Robot program ends normally.
	enable 🟥	[-]	An exception arises True : activated False : OFF
Return value	res0	[-] bool	True : Successful False : Unsucessful

Usage examples	# Eaxmple 1: An exception is allowed when entering the deceleration stop command during operation.
	rb.enable_interrupt(0, True)
	# Example 2: Allow creating exceptions when an emergency stop command is entered during operation.
	rb.enable_interrupt(1, True)
	# Eaxmple 3: Disable the possibility of an exception when entering a deceleration stop command during a pause.
	rb.enable_interrupt(2, False)
	# Example 4: Disables the feature of an emergency stop exception occurring during a pause.
	rb.enable_interrupt(3, False)		

Method	enable_mdo()		i611 MCS
Function	Enable MDO activity (*1).		
Argument	bitfield ঔ ँ [-] integer	MDO management number [*2] Set the bit corresponding to the active MOD management number. Installation range：0 -255	
Return value	Successful：True [-] bool		
	Unsucessful：An exception occurred.		

Usage examples	# Preset MDO operation settings [*3]
	rb.set_mdo(1, 23, 0, 1, 30)
	rb.set_mdo(8, 23, 1, 2, 10)
	# Enable MDO behavior
	# Example 1: Specify a value
	rb.enable_mdo(129) # Activate MDO management No.1 and 8
	. # Example 2: Key words
	rb.enable_mdo(bitfield=129) # MDO 관리 번호 1, 8 을 활성화합니다 .		

[*] 1) MDO operation is a function that shorts or opens an I/O output under conditions specified in operation.

[*] 2) For bit field settings, refer to “Setting MDO management numbers using bit fields” on page 56.

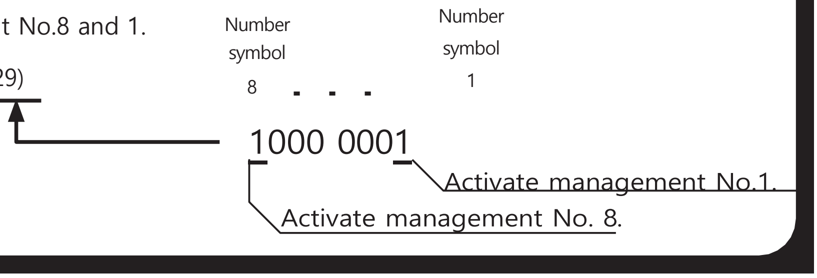
[*] 3) See page 75 for more details about set_mdo().

Set the MDO management number according to the bit field

Enable_mdo() can turn ON/OFF each mdo operation at the same time.

Example) Activate management No.8 and 1.

rb.enable_mdo(bitfield=129)



Numbers can be set at the same time with Enable_mdo()

There are a total of 4 MDOs that can be enabled in the enable_mdo() method. Selection in MDO settings in set_mdo() method.

Even if you run allow_mdo() multiple times, it cannot be enabled more than 5 times at the same time..

Method	exit()		i611 MCS
Function	Force to quit the robot program.		
Argument	res [-]	0 : Normal ending 0 khác : Unusual ending	
Return value	None integer		
Usage examples	rb.exit(0)		

Regarding exit() method

If the robotics program is successfully completed, this Method will not be needed.

- Exit() processing is also implemented with close() processing.
- The exit() method is equivalent to sys.exit() in the standard Python library sys module.

When the robot program is terminated by specifying a non-zero argument

The controller generates a system-determined error and the robot program terminates abnormally. In this error, the 'reset' signal is injected into the recovery I/O.

For port allocation, please refer to “3 Memory Maps”.

Method	get_hw_info()		i611 MCS
Function	Get the model name and serial number.		
Argument	None		
Return value	[model name, serial number] :	List	
	model name [-] string	Model name	
	serial number [-] string	Serial number	
Usage example	model, serial = i611Robt.get_hw_info()		

This method is a static method. This method can be called without specifying an instance of the i611Robot class.

get_system_port()

Function	Get the status of system ports				i611 MCS
Argument	None				
	[running, svon, emo, hw_error, sw_error, abs_lost, in_pause, error, (rsv.)] :				
Return value	running	[-]		Robot program status 1 : Running (Displayed on the controller LED (STS))	List
	svon emo hw_error	[-] [-]	integer	0 : Stopping servo status 1 : Turn on the servo	
	sw_error	[-]	integer	0 Turn off the servo Emergency stop condition 1 : Emergency stop in progress	
	abs_lost in_pause	[-] [-]	integer	0 : None System-determined error condition (critical) 1 : An error occurred	
	error	[-]	integer	0 : None The error state is determined by the system 1 : An error is occurring 0 : None	
		[-]	integer	Loss of ABS 1 : Loss of ABS 0 : None	
			integer	pause status 1 : Pause 0 : No	
				system error condition (*) 1 : A system error occurred 0 : No	
	(rsv.)	[-]	integer	(Expected)	
	# Check the status of each system.		integer		

Usage examples	running = rb.get_system_port()[0] # robot program status
	svon = rb.get_system_port()[1] # servo status
	emo = rb.get_system_port()[2] # emergency stop condition
	hw_error = rb.get_system_port()[3] # Error condition determined by the system (serious)
	sw_error = rb.get_system_port()[4] # Error condition determined by the system
	abs_lost = rb.get_system_port()[5] # Loss of ABS
	in_pause = rb.get_system_port()[6] # pause state
	error = rb.get_system_port()[7] # system error condition

Method	get_system_status() i611 MCS
Function	Get system status and error ID.
Argument	None
Return value	[status, err_id] : List System status [Controller symbol] status [-] integer 1：Start up [2：standby state 3：Loss of ABS] 4：Teaching [ ] 6：The robot program is running [run] 10: A specified system error is occurring 11. System error is occurring (serious) 12. An error is occurring 13. An error is occurring (serious) (88 :ID error) err_id [-] List ID error Error ID is the value displayed on the 7-segment LED display when an error occurs. (usually 0)
Usage examples	# Call by static method status, err_id = i611Robot.get_system_status()

This method is a static method. This method can be called without specifying an instance of the i611Robot class.

Method	getjnt() i611 MCS
Function	Get the current position of the manipulator in Joint form.
Argument	None
Return value	Successful： [Joint] : List Unsuccessful：An exception occurred.

Usage examples	rb.home()
	pos01=rb.getjnt()

Method	getmotionparam() i611 MCS
Function	Get current operating parameters.
Argument	None
Return value	Successful : [MotionParam] : Can reference an element set by the MotionParam class.
	Unsuccessful : An exception occurred.

Usage example	# MotionParam's entity reference. t_lin_speed=rb.getmotionparam().lin_speed
	t_lin_overlap=rb.getmotionparam().overlap

For details about the MotionParam format, please refer to P.37~..

Method	getpos() i611 MCS
Function	Get the current position of the manipulator as position.
Argument	None
Return value	Successful : [Position] : List
	Unsuccessful : An exception occurred.

Usage examples	rb.home()
	pos01=rb.getpos()

Method	home() i611 MCS
Function	Move so that the joint value of all axes is 0 degrees.
Argument	None
Return value	Successful : True [-]
	Unsuccessful : An exception occurs.
Usage examples	rb.home()

Method	is_open()		i611 MCS
Function	Check the open status of i611 Robot.		
Argument	None		
Return value	res0 [-] bool	True : Opening False : Not opening	
Usage examples	if not rb.is_open(): ... #Describe the command that will be performed if not opened.		

Method	is_pause()			i611 MCS
Function	Check the pause status of the robot program.			
Argument	None			
Return values	res0	[-] bool	True : Pause False : No pause .	

Usage example (*)

```
## Track pause and restart status in separate topic.
def thread_fnc(rb): while
    notthread_end:
# check status
        pause_st = rb.is_pause()
        print'Thisstatusis{}.'.format(pause_st)
        print "th:waitstop",din(DIN_STOP)
        if din(DIN_STOP) == "1":
            rb.stop()
        if din(DIN_PAUSE) == "1":
            rb.pause()
        if din(DIN_RESTART) == "1":
            rb.restart()


    # Example) Robot program sample
    try:
        while True:
          # Description of operating programs such as line(), move()
           ・ ・ # Pause by using user hook
            rb.user_hook()
                # ・ ・Description of operating programs such as line(), move() ・ ・
exceptRobot_emo:  # The event handler presses the emergency stop switch # ・ ・ ・ ・
exceptRobot_stop:           # The event handler detects the deceleration stop input # ・ ・ ・ ・
    finally:
        rb.close()
```

Supplement

Supplement of example using is_pause method ()

Prepare to use the is_pause() method

- Create a separate Threads entity.

Example：threadTest=threading.Thread(target=thread_fnc,args=[rb])

- Setup Daemon in separate thread.

Example：threadTest.setDaemon(True)

- Start separate thread.

Example：threadTest.start()

- Set pause operation.（The pause position changes depending on the value of the no_pause flag in set_behavior (). ）

Example： rb.set_behavior(only_hook=False,servo_off=False,restore_position=True,no_pause=False)

- Set the exception interrupt handling for deceleration and emergency stops during operation..

（The default value is all off（False).）

```
Example：rb.enable_interrupt(0,True) #Activate an exception when entering a deceleration stop order during operation
rb.enable_interrupt(1,True) # Enable exception generation when there is an emergency stop input during operation
```

- Set operating parameters.

```
Example：Cnt0 = MotionParam(lin_speed=70, jnt_speed=10,acctime=0.4,dacctime=0.4,overlap=100.0)
      rb.motionparam(Cnt0)
```

- Define I/O.

```
Example：DIN_STOP = 7          # Stop deceleration
    DIN_PAUSE = 8          # Temporary stop
      DIN_RESTART = 9        # Restart.
```

Method	join() i611 MCS
Function	Wait for the expected robot program operation to complete.
Argument	None
Return value	In case of failure: An exception is set out (occurs only if there is an error.)

Usage example

```
rb.line( P10 )          #Linear interpolated motion with teaching point P10

rb.asyncm(sw=1 )        #Program prediction operation ON (start)
rb.line( P20 )          # Perform linear interpolation movements using teaching points
P20.                    # Perform linear interpolation motion to the teaching point
rb.line( P21 )         #Wait for the Prediction Robot program to complete.
P21.
rb.join()

rb.asyncm( sw=2 ) # Program prediction activity OFF (exit)

...

rb.close()
```

포인트!

How to use asyncm () and join() methods

Before implementing asyncm (sw=2), wait until completion of performing the operation command in anticipation of performing the join() method.

Method	Joint2Position() i611 MCS
Function	Convert a Joint coordinate value to a Position coordinate value.
Argument	[Joint] : (Arguments cannot be omitted.)
Return value	Succsessful : List : Unsuccessful : An exception arises.

Usage examples

```
# Joint coordinate value    [ Position ]  List
    j10=Joint( 0, 30, 60, 0, 90, 90 )
# Convert location coordinate values ( j10 → Convert → p10)
    p10=rb.Joint2Position( j10 )
```

Result

```
J10 : [0, 30, 60, 0, 90, 90]
↓ Joint2Position()
      P10 : [125.0,  -717.5,  434.70181275976404,  3.508354649267438e-15,  4.296495291499103e-31,
    180.0, < ... >, 7]
```

Method	line() i611 MCS
Function	Perform linear interpolation.
Argument	[Position] [Joint] [MotionParam] : List Takes one or more motion parameters of type Position, type Joint, and type MotionParam as arguments. If the MotionParam type is taken as an argument, subsequent actions will be executed with the changed motion parameters.
Return value	Successful : True [-] bool Unsuccessful : An exception arises .

Usage example

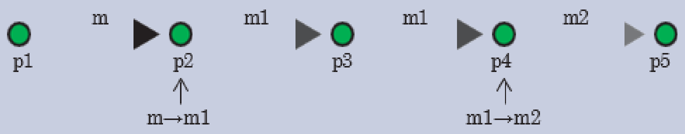
```
# Example 1
# Linear interpolated motion is performed by using position coordinate p10.
# Operating conditions are subject to those given by MotionParam.
rb.line(p10)

#Example 2
#  After arriving at the coordinate position p10, perform a linear interpolation movement to p20.
#  Operating conditions depend on MotionParam settings.
     rb.line( p10, p20 )

# Example 3
# The operating conditions change MotionParam and move to coordinate position p10.
# Then perform a linear interpolation motion with p20.
     mt=m.MotionParam( posture=1, passm=1, overlap=4.8, zone=20, pose_speed=5.0 ) rb.line( mt, p10,
     p20 )
```

Regarding line() and move()

Pre-define the position coordinates of Position type or Joint type.

Example : p1=Position(-50, -250, 350, 90, 0, 180)

```
#
     Change the motion parameters set using the motionparam() method, and move from p1 to p5

#####

   rb.line(p1, p2)                                          Move from p1 to p2 with action parameter m
   m1=m.motionparam( lin_speed=6, jnt_speed=20 )            Move from p1 to p2 with action parameter m
   rb.line(p3, p4)                                          Move the action from p2 to p4 through p3 with action parameter ml
   m2=m.motionparam( lin_speed=7, jnt_speed=40 )            Change the parameter from ml to m2
   rb.line(p5) # move()                                     Move from p4 to p5 with action parameter m2
```



➤ Kết quả
[0, 3, 2, 4]

Method	motionparam() i611 MCS
Function	Set operating parameters.
Argument	[MotionParam] : Keyword
	If the argument is omitted, it will be set to the default value.
Return value	Successful : True [-] bool
	Unsuccessful : An exception arises .

Usage examples	# Example 1: Set as entity of type MotionParam
	m=MotionParam()
	rb.motionparam(m)
	# Example 2: Set as keyword of member variable of type MotionParam
	rb.motionparam(posture=1, passm=1, overlap=4.8, zone=20, pose_speed=5.0)

For a detailed explanation of the MotionParam type, please refer to Page 37~.

Method	move() i611 MCS
Function	Moving PTP
Argument	[Position] or [Joint] or [MotionParam] List
	Take on 1 position coordinate of type Position and type Joint and motion parameters of type MotionParam as arguments. If the MotionParam type is taken as an argument, subsequent actions will be executed with the changed motion parameters.
Return value	Successful : True [-] bool
	Unsuccessful : An exception arises .

Usage examples

```
#Example 1
# PTP moves to the position of p10 coordinate
# Operating conditions are subject to those given by MotionParam.
  rb.move( p10 )

# Example 2
#  Move to coordinate position p10, then PTP moves to p20.
 #  Operating conditions are subject to those given by MotionParam.
      rb.move( p10, p20 )

#Example 3
# Change the operating condition to MotionParam and move to coordinate position p10,
# Then, move PTP to p20.
      mt=m.MotionParam(posture=1,passm=1,overlap=4.8,zone=20,pose_speed=5.0) rb.move(
      mt, p10, p20 )

#Example 4
# Use cross counter information
rb.use_mt(True)
      . . .
      rb.move( p10 )
      . . .
      rb.close()
```

When using cross counter information

Make sure to call use_mt(True) before calling the move() method.
If use_mt(False) is called or use_mt() is not called at all during the operation, that operation will not use the cross counter information.
For more information about cross counter, please refer to P.3, for more information about use_mt(), please refer to P. 80. The settings of cross counter ('ik_solver_option') are used used in the move() method and the Position2Joint() method.

Method	open()		i611 MCS
Function	Start connecting to the robot. (Initialization.)		
Argument	permission [-] bool	The only argument is True. True: Gain permission to operate (default value)	
	If the argument is omitted, it will be set to the default value.		
Return value	Successful : True [-] bool		
	Unsuccessful : An exception arises .		
Usage example	rb.open(True)		

About operation permissions and open() Method

Only a single process in the entire system can have operating privileges.
In case you do not have the right to operate but still use it, you can create many processes but an exception will arise when implementing the method requesting the right to operate.

The number of times open() is performed is only 1 time in the program. After executing close() once, if you execute open() a second time an exception will occur.
When executing a repeat statement, write open() before the repeat statement.

Method	optline()	i611 MCS
Function	Perform linear interpolation movements at optimal speed.	
Argument	[Position] [Joint] [MotionParam] : List	
	Get on 1 position coordinate of type Position and type Joint and type motion parameters MotionParam as argument. If takes type MotionParam as argument, further actions will be executed with changed motion parameters.	
Return value	Successful : True [-] bool	
	Unsuccessful: An exception arises .	

Usage example

```
# Example 1
# # Optimal linear interpolation motion is performed by using position coordinates p10.
# The operating conditions are subject to the given conditions given in MotionParam.
      rb.optline( p10 )

# Example 2
# After moving to coordinate position p10, perform Optimal linear
interpolation motion to p20.
# The operating conditions follow those given in MotionParam.
 Rb.optline(p10,p20)

# Example 3
# Change of operating conditions to MotionParam and move to coordinate position p10.

# Perform the Optimal linear internal with p20.
      mt=m.MotionParam( posture=1, passm=1, overlap=4.8, zone=20, pose_speed=5.0 )
      rb.optline( mt, p10, p20 )
```

!	Optline () speed is set equally to jnt_speed (PTP activity, Joint activity, Optimal linear interpolation motion), not likely to lin_speed (Optimal linear interpolation motion).	⚠

Method	override()	i611 MCS
Function	Run override.	
Argument	ovr Need [%] integer or float	Adjust the speed by applying a multiplier to the robot's motion speed set in motionparam(). (Cannot be omitted) Installation range：0 ～ 200
Return value	Unsuccessful: An exception arises (Return only in case of failure.)	

Usage examples

```
# Set override to 50%.
    rb.override( 50 )
```

Method	pause()	i611 MCS
Function	Pause the robot's movements.(*)	
Argument	None	
Return value	None	

Usage examples

```
## Track the pause state in a separate thread.
def thread_fnc(rb): while
    notthread_end:
        pause_st = rb.is_pause()
        print'Thisstatusis{}.'.format(pause_st)
        print "th:waitstop",din(DIN_STOP)
        if din(DIN_STOP) == "1":
            rb.stop()
        if din(DIN_PAUSE) == "1":
            #Pause.
rb.pause()
        if din(DIN_RESTART) == "1":
            rb.restart()

    # Example) Robot program sample
    try:
        while True:
            #・ ・Describe operating programs such as line(), move()・ ・
            # Pause with user hook
            rb.user_hook()
            #・ ・Describe operating programs such as line(), move()・ ・  exceptRobot_emo:
            # The event handler presses SW to stop urgently #・ ・ ・ ・
    exceptRobot_stop: # The event handler  detects the deceleration stop input #・ ・ ・ ・
    finally:
        rb.close()
```

Method	Position2Joint() i611 MCS
Function	Change the Position coordinate value to the Joint coordinate value.
Argument	[Position] : List Need (Arguments cannot be omitted.)
Return value	Successful : [Joint] : List Unsuccessful : An exception arises .

Usage examples	# Coordinate value of Position type
	p10=Position(-50, -250, 350, 90, 0, 180)
	# Change Joint type coordinates (p10 → change → j10)
	j10=rb.Position2Joint(p10)
	Result
	p10：[-50, -250, 350, 90, 0, 180]
	↓Position2Joint()
	j10：[18.049733949948962, -21.510874537939305, 147.44314399114182, -180.0, 125.9322694532025, 161.95026605005106]

Method	release_stopevent() i611 MCS
Function	Resets an occurring exception event.(*)
Argument	None
Return value	None
Usage examples	try: … # Movement except Robot_stop: rb.release_stopevent() … # Avoidance action

*)・Set up before exception handling.
・The exception occurs continuously until reset.

Method	reljntmove()		i611 MCS
Function	Work relative to the Joint coordinate system		
Argument	[dj1, dj2, dj3, dj4, dj5, dj6] : Keyword		
	dj1	[deg] float	Degree of movement of axis J1
	dj2	[deg] float	Degree of movement of axis J2
	dj3	[deg] float	Degree of movement of axis J3
	dj4	[deg] float	Degree of movement of axis J4
	dj5	[deg] float	Degree of movement of axis J5
	dj6	[deg] float	Degree of movement of axis J6
Return value	None		

Usage examples	# Prepare Joint type coordinate values.
	J1 = Joint(45, 45, -45, -45, 90, 0)
	# Set operating parameters.
	m=MotionParam(jnt_speed=10, lin_speed=70, overlap=30) rb.motionparam(m)
	...
	rb.move(J1)
	# Move J1 to a position 35 degrees offset in the joint coordinate system.
	rb.reljntmove(dj1=35)		

Method	relline()		i611 MCS
Function	Relative linear interpolation motion is performed in the Cartesian coordinate system		
Argument	[dx, dy, dz, drz, dry, drx] :	Keyword	
	dx	[mm] float	Offset amount in axial direction X
	dy	[mm] float	Offset amount in axial direction Y
	dz	[mm] float	Offset amount in axial direction X
	drz	[deg] float	Offset amount around axis Rz
	dry	[deg] float	Offset amount around axis Ry
	drx	[deg] float	Offset amount around axis Rz
Return value	Không có		

Usage example

```
# Prepare coordinate values of type Position.(*)
    P10 = Position( 95, -280, 240, 154, 80, -114 )
# Set operating parameters.
    m=MotionParam( jnt_speed=10, lin_speed=70, overlap=30 ) rb.motionparam(
    m )
    ...

    rb.move(P10)
# Move to an offset position of 15 mm in the X-axis direction in the Cartesian coordinate system..
    rb.relline( dx=15 )
```

*) The teaching point in the example has been simplified. Please use teaching data obtained through actual teaching.

Method	restart() i611 MCS
Function	Send a restart signal from the paused state.
Argument	None
Return value	None Processing returns before restarting according to the actual status. Call from a separate thread apart from the main thread.

Usage examples

```
## Restart in a separate thread.
def thread_fnc(rb):
    while not thread_end:
       pause_st = rb.is_pause()
       print'Thisstatusis{}.'.format(pause_st)
       print"th:waitstop",din(DIN_STOP)
       if din(DIN_STOP) == "1":
          rb.stop()
       if din(DIN_PAUSE) == "1":
          rb.pause()
       if din(DIN_RESTART) == "1":
                  # Restart
          rb.restart()

# Eaxmple) Robot program sample
   try:
      while True:
        # ・ ・ Describe action plans such as
        line(), move()
        # Pause by using user hook
        rb.user_hook()
             # ・ ・ Describe action programs such as line(), move() ・ ・
exceptRobot_emo:                 # The event handler by pressing SW to stop the emergency # ・ ・ ・ ・
   exceptRobot_stop:             #The event handler detects deceleration stop input ・
   finally:
      rb.close()
```

Method	set_behavior()	i611 MCS
Function	Set pause method.	
Argument	[only_hook, servo_off, restore_position, no_pause] :	List Keyword
	only_hook [-] bool	Only pauses are allowed with user_hook(). True : Activate False : OFF (default)
	servo_off [-] bool	Switch servo to OFF when paused. True : Activate False : OFF (default)
	restore_position [-] bool	When restarting after a pause (*1), the position will return to before the pause. (2) True : Activate False : OFF (default)
	no_pause [-]	Only make temporary stops according to the activity classification (*3) True : Activate (Compatible with system version R0.5.0) False : OFF *4) (default)
	If the argument is omitted, it will be set to the default value.	
Return value	None bool	

Usage example	# When restarting after a pause, it will return to the position it was before the pause.
	rb.set_behavior(only_hook=False, servo_off=False, restore_position=True, no_pause=True)	

*1) To continue operation, turn the servo on and then run (run).

*2) Even if the servo is turned on again after a pause and the position is moved, you can continue operation by returning to the position before the pause. If paused during operation, return to the original position and restart.

*3) Action division is immediately after action completion when calling the i611Robot class method.

If you want to pause, periodically call the method associated with the action or insert a user_hook() method.

*4) Distinguish between actions or pauses during actions.

Method	set_mdo()	i611 MCS
Function	Set up MDO behavior.	
Argument	[mdoid, portno, value, kind, distance] :	
	mdoid [-]	MDO management No. Setting range：1 ~ 8
	portno [-] integer	Number of output ports Setting range：0 ~ 12,287
	value [-] integer	Output I/O 0：LOW 1：HIGH
	kind integer [-]	Conditions 1：certain distance from the starting point 2：Reach within a certain range from the endpoint
	distance integer [mm]	Distance Setting range：0.0 ~
Return value	Thành công ：True [-] float	
	Unsucessful：An exception arises . bool	
Usage example	# Example 1: Specify a numeric value rb.set_mdo(1, 23, 0, 1, 30) # Set to MDO management number 1 rb.set_mdo(8, 23, 1, 2, 10) # Set to MDO management number 8 # Eaxmple 2: Key words rb.set_mdo(mdoid=1, portno=23, value=0, kind=1, distance=30) # Set to MDO management number 1 rb.set_mdo(mdoid=8, portno=23, value=1, kind=2, distance=10) # Set to MDO management number 8	

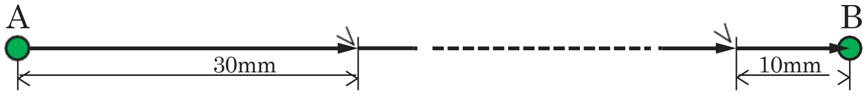
MDO operations

MDO：Middle Digital Out

A function that switches I/O output to LOW/HIGH under specified conditions during operation.

Example) Set output at LOW level.
value=0
kind=1

Example) Set output to HIGH.
value=1 kind=2



Method	settool()	i611 MCS
Function	Install Tool Offset.	
	[id, offx, offy, offz, offrz, offry, offrx] : List Keyword	
	id Cần [-] integer	Tool No.: 0 : Turn off Tool Offset. 1 - 8 : Choose Tool Offset.
	offx [mm] float	Tool Offset amount on the X axis in the Tool coordinate system
	offy [mm] float	Tool Offset amount on the Y axis in the Tool coordinate system
Argument	offz [mm] float	Tool Offset amount on the Z axis in the Tool coordinate system
	offrz [deg] float	The Tool Offset amount is rotated around the Rz axis in the Tool coordinate system
	offry [deg] float	The Tool Offset amount is rotated around the Ry axis in the Tool coordinate system
	offrx [deg] float	The Tool Offset amount is rotated around the Rx axis in the Tool coordinate system
	Original value : [0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	
Return value	Successful : True [-]	
	Unsuccessful : Exception occurs .	

Usage eaxmple	# 1 . Set Tool Offset (Tool No.1). # Example 1: Specify a numeric value
	rb.settool(1, 0.0, 0.0, 200.0, 0.0, 0.0, 0.0)
	# Example 2: Keywords
	rb.settool(id=1, offx=0.0, offy=0.0, offz=200.0, offrz=0.0, offry=0.0, offrx=0.0)
	
	# 2 . Choose Tool No.1 fromTool Offset.
	# Example 1: Specify a numeric value
	rb.changetool(1)
	# Example 2: Keywords
	rb.changetool(tid=1)	

The Tool argument name is used differently in Changetool() Method and settool() Method. ().

Method	Argument name of tool
changetool()	tid
settool()	id

Method	sleep()	i611 MCS
Function	Pause processing. Set pause time as argument.	
Argument	sec [s]	Pause time
Return value	Không có	integer

Usage example

```
# When an Exception is detected, terminate normally
    try.
        # Pause processing for 5 seconds.
        rb.sleep( sec=5 )
        ...
except Robot_emo #Event handler presses SW to stops urgently (cannot recover)
        # Describe the required error handling [e.g., termination handling].
```

*) To enable the Robot_emo() class, describe enable_interrupt() first.·

(☞ enable_interrupt()…P. 55, Robot_emo()…P. 109)

예) enable_interrupt(1,True) # # Activate an exception when an emergency stop command is entered during an operation

Sleep function of sleep() method and Python

Python's sleep function cannot cause an emergency stop exception even if the emergency stop switch is pressed during pause. By using the robot library's sleep() function, robot-related exceptions can be thrown even during sleep.

Method	stop() i611 MCS
Function	Decelerate the Robot to stop.(*)
Argument	None
Return value	None

Usage example

```
## Command to stop deceleration in a separate topic.
def thread_fnc(rb): while
    notthread_end:
        pause_st = rb.is_pause()
        print'Thisstatusis{}.'.format(pause_st)
        print "th:waitstop",din(DIN_STOP)
    # Command to stop deceleration in a separate topic
        if din(DIN_STOP) == "1":
            rb.stop()
        if din(DIN_PAUSE) == "1":
            rb.pause()
        if din(DIN_RESTART) == "1":

            rb.restart()


    # Example) Robot program sample
    try:
        while True:
          #Describe action programs such as line(), move()
          # Pause using user hook
          rb.user_hook()
    # Describe action programs such as line(), move()
exceptRobot_emo:              # Event handler presses SW to stops urgently
    # ・ ・ ・ ・
    exceptRobot_stop:         # Event handler detects input Deceleration stop input detection #
     ・ ・ ・ ・
    finally:
        rb.close()
```

*) In states other than automatic operation, use the following method to stop. The method：abort() can only stop when the robot is operating, and not in other cases. (Go to the next program execution section.))

Related method

Restart Confirmation Method：is_pause ()

Method: Set stop position：set_behavior()

Method	svoff() i611 MCS
Function	Set servo to OFF.
Argument	None
Return value	Successful：True [-] bool Unsuccessful：Exception occurs .

Usage example

```
rb.svoff()
```

Method	svstat()	i611 MCS
Function	Obtain servo status.	
Argument	None	
Return value	state [-] integer	Only worth returning when successful. 1 : ON servo 0 : OFF servo -1 : Emergency stop in progress
Usage example	if rb.svstat() == 1: # ON servo ... elif rb.svstat() == 0: # OFF servo ... elif rb.svstat() == -1: # Emergency stop in progress	

Method	toolmove()	i611 MCS
Function	Relative movement is performed based on the Tool coordinate system.	
Argument	[dx, dy, dz, drz, dry, drx] :	
	dx [mm]	List Keyword The amount of movement in the X axis direction in the Tool coordinate system
	dy [mm] float	The amount of movement in the Y axis direction in the Tool coordinate system
	dz [mm] float	The amount of movement in the Z axis direction in the Tool coordinate system
	[deg] float	The amount of rotation around Rz in the Tool coordinate system
	dry [deg] float	The amount of rotation around Ry in the Tool coordinate system
	drx [deg] float	The amount of rotation around Rx in the Tool coordinate system
	Default : [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] float	
Return value	Successful : True [-]	
	Unsuccessful: Exception occurs .	

Usage example	#Example: Define teaching data of type location [dx, dy, dz, drz, dry, drx] as a list.
	p10=Position(95, -280, 240, 154, 80, -114)
	
	# For example: After moving to coordinate position p10, move a distance dx=15mm according to the Tool coordinate system.
	...
	rb.move(p10) rb.toolmove(
	dx=15)
	...
	rb.close()	

Method	use_mt()	i611 MCS
Function	Set cross counter on/off.	
Argument	mt [-] bool	True : Activate False : OFF (Default: Compatible with R system version 0.5.0)
Return value	None	

Usage example	# Use cross counter information.
	rb.use_mt(True)	

Cross counter method

If cross counter is enabled, the behavior of the underlying API will change

[Created function]

Position()

[Method]

Position class : replace(),pos2list(),pos2dict(),position(),motionparam() Class i611Robot : getpos(),Joint2Position(),Position2Joint(),move()

Class i611Robot : getpos(),Joint2Position(),Position2Joint(),move()

Method	user_hook()	i611 MCS
Function	Pause the robot program.	
Argument	None	
Return value	None	
Usage example	… rb.user_hook() #Pause the program at this position. …	

User_hook() method

Place this Method at the location where you want to pause processing outside of the robot control commands of the Robsys class .

When pausing only a specific part of the robot program, place user_hook() at the location where you want to pause the robot program, in the disabled state “Enable pause only when user_hook” is disabled in set_behavior().

Method	version()		i611 MCS
Function	Get system version.		
Argument	None		
Return value	[res0,major,minor,patch,build,date,option]: res0	List	
	None [-]	bool	True :Successful False: Unsuccessful
	major [-]	integer	Major Version
	minor [-]	integer	Minor Version
	patch [-]	integer	Patch Version
	build [-]	List	Build Version
	date [-]	string	Built date
	option [-]	string	Option
Usage example	rb.version()		[True, 0, 6, 9, 7, u'04:37:07 Nov 28 2017', u'SIM No-spiio ']

NOTES

2. Module: teachdata

Lớp

Teachdata

Teaching data administration

Member variable	
—	—

Method		
check_format()	Get a formatted version of the teaching data file.	P.84
close()	Close the teaching data file.	P.85
flush()	Export updated teaching data to a file.	P.85
get_coordinate()	Enter the Base Offset of the teaching data.	P.86
get_joint()	Get the Joint coordinate value of the teaching data.	P.86
get_param()	Get teaching data parameters.	P.87
get_position()	Get the Position coordinate value of the teaching data.	P.88
get_tool()	Get the Tool offset of the teaching data.	P.89
is_open()	Check the open status of the teaching data file.	P.89
open()	Open the teaching data file.	P.90
set_joint()	Update the matching coordinate value of the teaching data.	P.90
set_param()	Update parameters of teaching data.	P.91
set_position()	Update the Position coordinate value of the teaching data..	P.92

Created functions	Teachdata()	Teach data
Function	Create an instance of the Teachdata class by loading the teaching data.	
Argument	fname [-] string	Name of the teaching data file Original value：'/home/i611usr/teach_data'
Return value	Self-reference（Teachdata class object）	

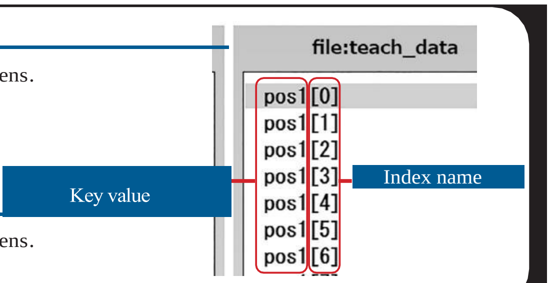
Usage examples	# When a teaching data file is specified
	td=Teachdata(fname = './home/i611usr/teach_data')
	# If argument is omitted
	td = Teachdata()	

「Key」and「Index」

Correspond to what is displayed on the teaching, key and folder screens.

「Key」and「Index」

Correspond to what is displayed on the teaching, key and folder screens.



Method	check_format()	Teach data
Function	Import a formatted version of the teaching data file.	
Argument	fname Need string	Full path to the teaching data file
Return value	ver [-] string	" Version string "
Usage example	ver = Teachdata.check_format("/home/i611usr/teach_data") File name	

An instance of the Teachdata class is not required for the Static Method.

Method	close() Teach data
Function	Close the teaching data file.
Argument	None
Return value	None

Usage example	# End of teaching data file.
	    td.close()

Please implement the close() method at the end of the program.

When opening teaching data in Read/Write mode, export the updated data to a real file and then disable exclusive (exclusive) processing...

Method	flush() Teach data
Function	Export updated teaching data to a file.
Argument	None
Return value	None

Usage example	# Update teaching data files.
	    ..
	    td.flush()
	    ...
	    td.close()

- In case of updating data, when close () will also be when running that data. This should be done when a certain amount of updates have been accumulated, not every update.

- If you are updating data, it will also run internally when you close it ().

Method	get_coordinate()	Teach data
Function	Enter the Base offset of the teaching data.	
Argument	index [-] integer	Base ID Installation range ：0 - 3 0 : Returns the base version 1 -3 : Returns a Coordinates instance of the Base coordinate system.
Return value	baseoffset	Base offset example The corresponding entity is returned according to the ID specified in the argument. index=0 : Base index=1, 2, 3 : Coordinate entity of the corresponding Offset Base

Usage example	# Index number: 1, offset value of base coordinates taken from the teaching data file.
	baseoffset = td.get_coordinate(1)
		Result
		[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, < ... >]

Method	get_joint()	Teach Teach
Function	Get the matching coordinate value of the teaching data.	
Argument	[key, index, comment] : List	
	key Need [-]	Joint Coordinate key value Installation range ：'Joint1' -'Joint20'
	index Need [-] integer	Joint Coordinate index Installation range：0 -9
	comment [-] bool	flag receives Comment True : Received False : Not received .
Return value	[[Joint] , comment] : List	
	[Joint] [deg] float	Joint Coordinate value
	comment [-]	Comment (Maximum 32 characters)
Usage example	#key = joint1, index 번호 = Get the Joint coordinate value and Comment of 1. jnt, comment = td.get_joint('joint1', 1, True) printjnt.jnt2list()	Result [0.744, -37.724, -83.660, 45.270, -47.308, 10.110]

An exception (Robot_error) occurs when data for the specified key and index does not exist.

Method	get_param()		Teach data
Function	Get teaching data parameters.		
Argument	[key, index, axis, comment] :		
	key	[-]	the key List value of the parameter Installation range : 'param1' - 'param4'
	index	string [-]	parameter index Installation range : 0 - 9
	axis	integer [-]	Axis number in parameter Installation range : 1 - 8
	comment	integer [-] bool	Gets the parameter's Comment flag True : Received False : Not received . (Original value)
Return value	param	[-] string	parameter string (Maximum 32 characters/values can be entered on the teaching screen.)

Usage example	#key : "param2", index number : Get the 1st and 2nd teaching data files
	param = td.get_param("param2", 1, 2)		

An exception occurs if data for the specified key, index, and axis does not exist.

Method	get_position()		Teach data
Function	Get the Position coordinate value of the teaching data.		
Argument	[key, index, tool, base, comment] : List		
	key Need	[-]	Key value of Position coordinates Installation range：'pos1' -'pos20'
	index Need	[-] integer	Position coordinate index Installation range：0 -9
	tool	[-]	Tool ID collection flag True ：Received False：Not received .（Original value）
	base	[-]	Facility ID collection flag True ：Received False：Not received .（Original value）
	comment	[-] bool	Comment collection flag True ：Received False：Not received .（Original value）
Return value	[pos, toolid, baseid, comment] : List		
	pos	[mm] float	Position coordinate valueí (object) [Position]
	toolid	[-] integer	Tool ID Return value：0 - 8（0 without tool ）
	baseid	[-] integer	Base ID Return value：0 - 3（0 without tool）
	comment	[-]	Comment （Maximum 32 characters, if there isn't ' ')
Usage examples	# Get data of key = pos1, index number = 1.。 #(Get Tool ID(True), Base ID(True), Comment (True).) pos, toolid, baseid, comment = td.get_position('pos1', 1, True, True, True)		

Result

```
pos       : [21.0, 459.94, 120.61, 53.890, 4.720, -142.88, < ... >, 6]
toolid    : [3]
baseid  : [0]
comment : [test]
```

An exception occurs when data for the specified key and index does not exist.

Method	get_tool()	Teach data
Function	Get the Tool offset of the teaching data.	
Argument	index ঐ◌ঁ [-] integer	Tool ID Installation range：0 -8 (If set to 0, the return value will be [0, 0, 0, 0, 0, 0].)
Return value	[dx, dy, dz, drz, dry, drx]	List
	dx, dy, dz [mm] float	Tool Offset value position (world coordinate system)
	drz, dry, drx [deg] float	Tool Offset value angle (angle Z-Y-X Euler)

Usage example	# Index number: 1, Tool Offset value is taken from the teaching data file.
	tooloffset=td.get_tool(1)
	...
		Result
		[1, u'0.00', u'0.00', u'0.00', u'0.00', u'0.00', u'0.00']

An exception (Robot_error) occurs when data for the specified key and index does not exist.

Method	is_open()	Teach data
Function	Check the open status of teaching data.	
Argument	None	
Return value	res [-] integer	0：Not open 1：ReadOnly mode (read only) 2：Read/Write mode (Read/Write)

Usage examples	# Check the open status of the teaching data file.
	td = Teachdata()
	td.open(readonly=False) if
	td.is_open() == 2:
	return False
	...	

Method	open()	i611 IO Teach data
Function	Open the teaching data file.	
Argument	readonly [-] bool	True : Open in ReadOnly mode (default) False : Open in Read/Write mode (read/write)
Return value	Không có	

Usage example

```
# Open in read-only mode.
 td = Teachdata() td.open(
     readonly=Ture)

# Open in Read/Write mode.
     td = Teachdata()

     td.open(  readonly=False  )
```

- Cannot be opened when the operating mode is “Teaching”..
- If you open in Read/Write mode, other processes cannot open in Read/Write mode.
- If the version of the teaching data file is an unsigned version (R1.0.0 or later), an exception will occur..
- Older versions of the teaching data file can be read, but an error message will appear.

（ It is recommended to convert teaching data files. ）

Method	set_joint()		i611 IO Teach data
Function	Update the Joint coordinate value of the teaching data.		
Argument	[key, index, jnt, comment] : List		
	key Need	[-]	Key Joint name : joint1 -joint20
	index Need	[-] integer	Index Installation range : 0 -9
	jnt Need	[-] float	Updated Joint coordinate value （] Object ）
	comment	[-] string	Comment string (Maximum 32 characters)
Return value	None		

Usage examples

```
# Key value of type Joint : "joint1",index number : 2, Joint type coordinates: jnt, Comment : "home" is updated .
     jnt = Joint( 0, 0, 0, 0, 0, 0 )
     td.set_joint("joint1",2,jnt,"home")
```

- Can only be used on data that already exists.
- If there is no specific key or index, an exception will occur.
- An exception occurs if not in Read/Write mode.
- If you run the Flush() command after calling this Method, the file will be updated.

Method	set_param()	Teach data
Function	Update parameters of teaching data.	
Argument	[key, index, axis, paramstr comment] :	
	key [-]	Key List Param name : param1 ~ param4
	index [-]	Index Installation range : 0 -9
	axis integer [-]	Axis Installation range : 1 -8
	paramstr integer [-]	Parameter string (Maximum 32 characters)
	comment string [-] string	Comment string (Maximum 32 characters)
Return value	None	

Usage examples	# key : param2, index No. : 3, number of axis : 1, parameter "1.00" updated
	td.set_param("param2", 3, 1, "1.00")	

- Can only be used on data that already exists.
- If there is no specific key or index, an exception will occur.
- An exception occurs if not in Read/Write mode.
- If you run the Flush() command after calling this Method, the file will be updated.

Method	set_position()		Teach data
Function	Update the Position coordinate value of the teaching data.		
Argument	[key, index, pos, tooloffset, baseoffset, comment] : List		
	key Cần	[-] string	Key Position name : pos1 ~ pos20
	index Cần	[-] integer	Index Installation range : 0 - 9
	pos Cần	[-] float	Update Location coordinate value [Position] (Object)
	tooloffset	[-] integer	The Tool Offset ID is used to determine this Position coordinate value Installation range : 0 - 8 Original value: 0 (Do not use Base Offset).)
	baseoffset	integer [-]	The Base Offset ID is used when determining this Position coordinate value Installation range : 0 - 3 Original value: 0 (Do not use Base Offset.)
	comment	[-] string	Comment string (Maximum 32 characters)
Return value	None		

Usage example	# Value type Position : pos2, Index number : 2, Tool ID : 1 , Base Offset : Not used, Comment “work” updated
	pos = Position(95, -280, 425, -120, 84, -28)
	td.set_position("pos2", 2, pos, 1, 0, "work")		

- Can only be used on data that already exists.
- If there is no specific key or index, an exception will occur.
- An exception occurs if not in Read/Write mode.
- If you run the Flush() command after calling this Method, the file will be updated.

3. Module: i611_extend

Class

Pallet

Performs the function of pallets.

Member variable	
-	-

Method		
adjust()	Fix cell position.	P.94
get_pos()	Enter the position of the cell.	P.95
init_3()	Definition of pallet (Lecture 3 points)	P.96
init_4()	Definition of pallet (Lecture 4 points)	P.97

Created funtions	Pallet() i611 Ext.
Function	Pallet function: creates a Pallet class entity.
Argument	None
Return value	Back to the private class object

!	For practical operations, use the points taught to determine the coordinates of each Pallet.

Method	adjust()		11
Function	Fix cell position.		
Argument	[i, j, di, dj] : List		
	i [-] integer	Index determines the cell's position in the Pallet (i direction)	
	j [-] integer	Index determines the cell's position in the Pallet (j direction)	
	di [mm] integer	The offset value of the cell position in direction i	
	dj [mm] integer	The offset value of the cell position in direction j	
Return value	None		

Usage example

```
# For example: Determine the coordinates of the 4 corners of the Pallet and identify the Pallet.. (*)
    pos_0=Position( -250, -250, 400 )
    pos_1=Position( -170,-250, 400 )
    pos_2=Position( -250, -180,400 )
    pos_3=Position( -170,-180,400 )
    pal.init_4( pos_0, pos_1, pos_2, pos_3, 8, 7 )

  # Set the Pallet offset value.
  pal.adjust( 4, 4, 10, 10 )

    rb.close()
```

*) For practical operations, use the points taught to determine the coordinates of each Pallet.

Method	get_pos() i611 MCS i611 Ext
Function	Get the cell's position.
Argument	[i, j, dk] List
	i Cần [-] integer Index determines the cell's position in the Pallet (direction i)
	j Cần [-] integer Index determines the cell's position in the Pallet (direction j)
	dk [mm] integer Offset value in vertical direction Original value : 0
	If an offset value is set, the offset coordinates will be taken from the Pallet coordinates in the k direction of the cell. If the dk argument is omitted, the original value is set.
Return value	[Position] Coordinates of the cell at position (i, j) in the Pallet

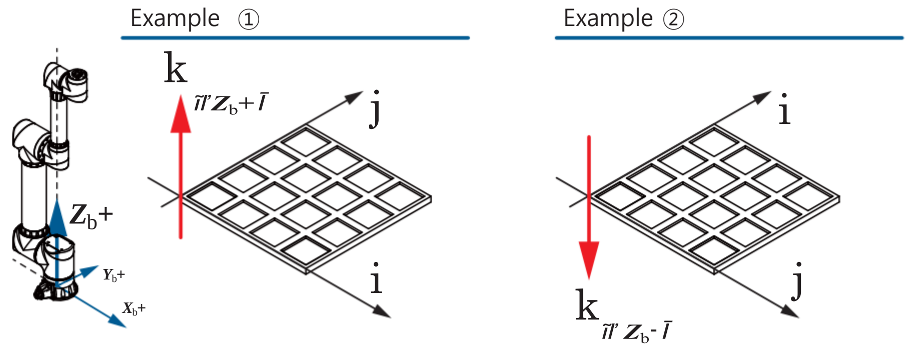
Usage example	# Example: Determine the Pallet and the coordinates of the 4 corners of the Pallet. (*)
	pos_0=Position(-250, -250, 400)
	pos_1=Position(-170,-250,400)
	pos_2=Position(-250, -180,400)
	pos_3=Position(-170,-180,400)
	pal.init_4(pos_0, pos_1,pos_2, pos_3, 8, 7) pal.adjust(4, 4, 10, 10)
	
	# Gets the coordinates of the specified index in the Pallet, including offset.
	p00=pal.get_pos(0, 0, 10)
	
	# Go to the coordinates obtained from get_pos ().
	rb.move(p00)
	rb.close()

*) For practical operations, use the points taught to determine the coordinates of each Pallet..

Vertical direction of pallets

The positive sign direction of the vertical direction of the pallet (k direction) changes depending on the position of the teaching point..

Example ① i × j: Vector z + perpendicular to the plane (Pallet plane).
→ →
Example ② j × i: z - opposite to the example①.
→ →



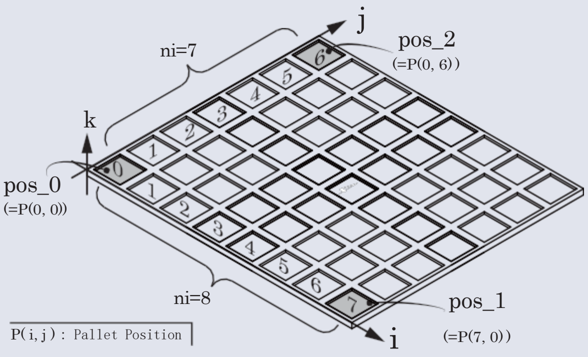
Example ①

Example ②

Method	init_3()	i611 MCS i611 Ext
Function	Identify Pallets (3-point guide)	
Argument	[pos_0, pos_i, pos_j, ni, nj] : List	
	pos_0 need [-] float	[Position] Pallet teaching point (original)
	pos_i need [-] float	Pallet teaching point (direction i) [Position]
	pos_j need [-] float	Pallet teaching point (direction j) [Position] Number of cells arranged in direction i of the Pallet
	ni need [-] integer	Number of cells arranged in direction j of the Pallet
	nj need [-] integer	Only return when successful
Return value	res [-] bool	True : Successful

Usage example

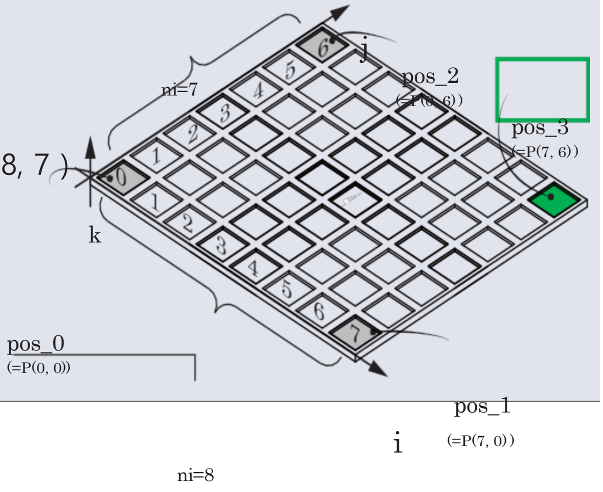
```
# Example: Determine the coordinates of the three vertices of the Pallet (*)
    pos_0=Position( -250, -250, 400 )
    pos_1=Position( -170, -250, 400 )
    pos_2=Position( -250, -180, 400 )

# Identify Pallets using 3-point guidance data.
    pal.init_3( pos_0, pos_1, pos_2, 8, 7 )

    rb.close()
```



*) For practical operations, use the points taught to determine the coordinates of each Pallet.

Method	init_4()				i611 MCS i611 Ext.
Function	Xác định Pallet (hướng dẫn 4 điểm)				
Argument	[pos_0, pos_i, pos_j, pos_ij, ni, nj] :		List		
	pos_0	Cần	[-] float	[Position]	Pallet teaching point (original)
	pos_i	Cần	[-] float	[Position]	Pallet teaching point (direction i)
	pos_j	Cần	[-] float	[Position]	Pallet teaching point (direction j)
	pos_ij	Cần	[-] float	[Position]	Pallet teaching point
	ni	Cần	[-] integer		Number of cells arranged in direction i of the Pallet
	nj	Cần	[-] integer		Number of cells arranged in direction j of the Pallet
Return value	res		[-] bool		Only return when successful True : Successful
Example					

```
    # Example: Determine the coordinates of the 4
         vertices of the Pallet (*)
  pos_0=Position( -250, -250, 400 )
  pos_1=Position( -170, -250, 400 )
  pos_2=Position( -250, -180, 400 )
  pos_3=Position( -170, -180, 400 )

# Identify Pallets using guidance data 4 point

pal.init_4( pos_0, pos_1, pos_2, pos_3, 8, 7 )


  rb.close()
```



P(i,j) : Pallet Position

*) For practical operations, use the points taught to determine the coordinates of each Pallet.

4. Module : rbsys

Class

RobSys

System administrator control

Member variable	
-	-

Method		
assign_din()	Allocate functions to input ports of physical I/O and memory I/O.	P.99
assign_dout()	Assign functions to the output ports of physical I/O and memory I/O.	P.100
clear_robtask()	Unsubscribe from the robot program.	P.101
close()	Terminate the connection with the system manager.	P.101
cmd_pause()	Action command: pause	P.102
cmd_reset()	Action command: reset error	P.102
cmd_run()	Action command: Run the robot program	P.103
cmd_stop()	Action command: stop deceleration	P.103
get_robtask()	Get the status of the robot program.	P.104
open()	Initiate communication with the system administrator.	P.104
req_mcmd()	Get system status and command status.	P.105
set_robtask()	Register for the robot program.	P.106
version()	Get version information of System Manager.	P.107

❗ The RobSys class is an administrative program interface for I/O control and task management in configuration scripts. (init.py).
In the robot program, use the Method of the i611Robot class and do not use the RobSys class.

Constructor	RobSys()	rbsys
Function	Create robot system instances.	
Argument	host [-] string	Specify the IP address of the connection destination Original value : '127.0.0.1'
	If the argument is omitted, it will be set to the default value.	
Return value	Return to the class object itself	

Usage examples

```
# Example 1: Omit argument (set to initial value)
    rbs = RobSys()

# Example 2: Keywords
    rbs = RobSys( host='127.1.1.1' )
```

Method	assign_din()	i611 IO	rbsys
Function	Assign functions to input ports of physical I/O and memory I/O.		
Argument	[run, stop, err_reset, pause] :	Keyword	
	run	[-] integer	Run the robot program Original value：-1
	stop	[-] integer	Reduce speed to stop Original value：-1
	err_reset	[-] integer	Reset errors Original value：-1
	pause	[-] integer	Pause Original value：-1
	Installation range： ・Physical I/O : 0 - 15 ・Memory I/O ：32 - 12287 ・If no value is assigned, specify -1. ・If the argument is omitted, it is set to Initial Value. ・Duplicate assignments cannot be made to the same I/O. ・Please refer to “3 Memory Maps” for assigned physical I/O. ・Setting range 0 - 15 corresponds to IN1 to IN16 (CN3: I/O connector 1) of the input port signal name. ・Low— Hi input detection input signal		
Return value	Successful：True [-]	bool	
	Unsuccessful：Exception occurs		

Usage example	# Specify in init.py: (recommended settings below ）
	rbs.assign_din(run=0, stop=1, err_reset=2, pause=3)		

Method	assign_dout()		11 rbsys
Function	Assign functions to output ports of physical I/O and memory I/O.		
Return value	[running, svon, emo, hw_error, sw_error, abs_lost, in_pause, error] : Keyword		
	running [-] integer	robot program status Original value : -1	
	svon [-] integer	Servo status Original value : -1	
	emo [-] integer integer	emergency stop condition Original value : -1	
	hw_error [-] integer	System-determined error condition (critical) Original value : -1	
	sw_error [-] integer	The error state is determined by the system Original value : -1	
	abs_lost [-] integer	Status of ABS loss Original value : -1	
	in_pause [-] integer	Pause status Original value : -1	
	error [-]	system error condition (*) Original value : -1	
Examples	Installation range : ・Physical I/O : 16 - 31 ・Memory I/O : 32 - 12287 ・If no value is assigned, specify -1. ・If the argument is omitted, it is set to Initial Value. ・Duplicate assignments cannot be made to the same I/O. ・Please refer to “3 Memory Maps” for assigned physical I/O. ・Setting range 16 - 31 corresponds to signal names O1 to O16 (CN4: I/O connector 2, CN5: I/O connector 3) ・When the output is ON, the corresponding output port becomes Hi bool		
	Successful : True [-]		
	Unsuccessful: Exception occurs		

*) A system-determined error status (non-fatal or fatal) occurs. Used to test 2 error states on one control line.

If the robot program is started without using the system manager, (*)the running output is not displayed. If running outputs, please reboot through system manager.

*) For example, the case of starting with terminal software.

Method	clear_robtask()	rbsys
Function	Unsubscribe from the robot program.	
Argument	None	
Return value	res0 [-]	True : Successful False : Unsuccessful

Usage examples	# Successful
	if rbs.clear_robtask()[0] == True:
	# Unsuccessful
	if rbs.clear_robtask()[0] == False:	

The clear_robtask() method does not stop the Robot program while it is running.

Method	close()	rbsys
Function	Terminate the connection with the system manager.	
Argument	None	
Return value	None	
Usage example	rbs.close()	

Method	cmd_pause() rbsys	
Function	Action command: pause	
Argument	None	
Return value	res0 [-]	True : Successful False : Unsuccessful
Usage example	rbs.cmd_pause()	

Usage time md_pause()

Pausing can be done if cmd_pause() is executed in an action command or at the time of executing the user_hook() Method.

To continue working, execute with cmd_run ().

Method		
Function	Action command: reset error	
Argument	None	
Return value	res0 [-]	True : Successful False : Unsuccessful
Usage example	rbs.cmd_reset()	

Errors can be reset by cmd_reset()

	Type of errors			Is it possible to cancel via cmd_reset()?
E88	E**	system-determined error		○
c88	c**	system-determined error	Critical	×
u88	u**	user-determined error		○
r88	r**	user-determined error	Critical	×

Method	cmd_run() rbsys	
Function	Action command: Run the robot program (If paused, the program will be reconnected.)	
Argument	fname [-]	Framework name
	If the argument is omitted, the robot program specified with set_robtask() will be executed.	
Return value	None	

Usage example	# Example 1: When executing the robot program specified with set_robtask(), the argument will be ignored.
	rbs.set_robtask('sample.py')
	rbs.cmd_run()
	# Example 2: When specifying a filename as an argument
	rbs.cmd_run('sample.py')	

Method	cmd_stop() rbsys	
Function	Motion command: stop deceleration	
Argument	res0 [-] bool	True : Successful False : Unsuccessful
Return value	None	
Usage example	rbs.cmd_stop()	

Relationship between cmd_stop() and allow_interrupt() in class i611Robot

The behavior of the robot program after deceleration to stop varies depending on the Enable_interrupt() (deceleration interrupt setting) of the i611Robot class.

Interrupt speed deceleration enable_interrupt(eid, enable)	Robot program after stopping deceleration
Valid enable_interrupt(0, True)	Exception occurred
Disable enable_interrupt(0, False)	Normal interrupt

In case the validity expires, the Robot program will end abnormally.

Method	get_robtask()	rbsys
Function	Get the status of the robot program.	
Argument	None	
Return value	[res0, res1, res2] : List	
	res0 [-] bool	True : Robot program in progress False : Stop
	res1 [-] integer	Process ID of the currently running robot program or last run
	res2 [-] string	File name of the registered robot program
Examples	# Get the execution status of the robot program rb.get_robtask()[0] # Get the process ID of the robot program that is running or last executed rb.get_robtask()[1] # Get the file name of the registered robot program rb.get_robtask()[2]	

Method	open()	rbsys
Function	Initiate communication with the system administrator.	
Argument	None	
Return value	None	
Usage example	rbs.open()	

Method	req_mcmd()		rbsys
Function	Get system status and command status.		
Argument	None		
Return value	[running, svon, emo, hw_error, sw_error, abs_lost, in_pause, error] :		
	running	[-]	the robot program status List 1：Displayed on the controller LED (STS) is running)
	svon emo hw_error	[integer	0：Stopping servo status 1：서보 ON
	sw_error	[-] integer [-]	0：서보OFF emergency stop condition 1：Emergency stop in progress
	abs_lost in_pause	[-] integer [-]	0：None System-determined error condition (critical) 1：An error is occurring
	error	integer [-]	0：None System-determined error condition 1：An error is occurring
		[integer	0：None Status of Loss of ABS 1：Loss of ABS
		integer	0：None Stopping status
		integer	1：Stopping 0：None system error condition (*)
		integer	1：A system error occurred 0：None

Usage examples

```
# Check your system status individually
    running = rbs.req_mcmd()[0]          # robot program status
    svon = rbs.req_mcmd()[1]              # servo status
    emo = rbs.req_mcmd()[2]               # emergency stop condition
    hw_error = rbs.req_mcmd()[3]  # System-determined error condition (critical)
    sw_error = rbs.req_mcmd()[4]          # System-determined error condition
    abs_lost = rbs.req_mcmd()[5]          #  Status of ABS loss
    in_pause = rbs.req_mcmd()[6]          # pause status
    error = rbs.req_mcmd()[7]             # system error condition
```

Method	set_robtask() rbsys	
Function	Register for the robot	
Argument	fname [-]	Program file name
Return value	res0 string [-]	True : Successful False : Unsuccessful (The specified file does not exist)
Usage example	rbs.set_robtask('sample01.py')	

Robot programs can only be registered using the set_robtask() method. Cannot be used when starting the robot program.

Relationship between I/O allocation and Python ports

Function	Related method	Number of Python ports	
		Physical I/O (*2)	System I/O
Run the robot program	cmd_run()	0	4288
Reduce speed to stop	cmd_stop()	1	4289
Reset errors	cmd_reset()	2	4290
Pause	cmd_pause()	3	4291

Function	Related method	Number of Python ports	
		Physical I/O (*2)	System I/O
The state of the robot program	req_mcmd()[0]	16	4160
Servo state	req_mcmd()[1]	17	4096
The state of emergency stop	req_mcmd()[2]	18	4097
The state of error according to what the system determined (critical)	req_mcmd()[3]	19	4098
The state of error according to what the system determined	req_mcmd()[4]	20	4161
The state of losing ABS	req_mcmd()[5]	21	4099
The state of pause	req_mcmd()[6]	22	4162
The state of system error (*1)	req_mcmd()[7]	23	4163

*1) A system-determined error status (non-fatal or fatal) occurs. Used to test 2 error states on one control line.
*2) We recommend assigning this I/O. For details on memory I/O, please refer to “3 Memory Maps”.

Method	version()		rbsys
Function	Get the system manager version information.		
Argument	None		
Value	[res0, major, minor, patch, build] : List		
	major	[-] bool	True : Successful False : Unsuccessful
		[-] integer	Principal version
	minor	[-] integer	Principal version
	patch	[-] integer	patched version
	build	[-] integer	built version
Examples	version=rbs.version() print version		

5. Module : i611_common

Class

Exception

Exception handling of Method of class i611Robot.

The class inherits the Exception class		
Robot_emo	Exception occurs during emergency stop (cannot be recovered)	P.109
Robot_error	The exception arose due to an error	P.110
Robot_fatalerror	Exceptions arise in case of fatal (irrecoverable) errors)	P.110
Robot_poweroff	Exception occurred when power off (irrecoverable)	P.111
Robot_stop	Exception occurs when decelerating to stop	P.112

The Exception class can be used simply by importing the i611_MCS module. Load from i611_common import in the i611_MCS module.

Class	Robot_emo() i611 COM. i611 MCS
Function	Exception occurs during emergency stop (cannot be recovered) Event handling during emergency stop.
Argument	None
Value	None

Examples

Preparation

```
-
## 1 . Initial setup ① Import the module #####################
    from i611_MCS import *
## 2 . Initial settings ②: Set operating conditions ############################### #
    Robot constructor i611
        rb = i611Robot()
    # Determine the world coordinate system
        _BASE = Base()
    # Start connecting to the robot, initialize
        rb.open( True )
    # Exception triggered due to emergency stop input during operation
        rb.enable_interrupt( 1, True )
...
# Exception handling
# Example 1: Exception detection and normal termination
      try:
      ...
except Robot_emo:  # Exceptions are made for emergency stops

# Required error handling (termination handling) technique
# Example 2: Exception detected, termination of error
try:   ...
      except Robot_emo: # Exceptions arise during emergency stops
```

E.14 Arising (Refer to P. 57)

Added description of Robot_emo class

Operate the robot program while pressing the emergency stop switch

Code try: ... except: in action program・・・

・Case not included :

The program ends with an error condition. Display controller・ E.13 .

Case included :

The program ends normally.(*)

*) Write a program so that the robot program ends within 5 seconds when the emergency stop switch is pressed.
If it exceeds 5 seconds, it will end in an error condition. display 7seg:
E.10

Class	Robot_error()	i6 COM.	11
Function	The exception arose due to an error This is the event handler when an error occurs..		
Argument	None		
Return value	None		

Usage examples

```
# Preparation
## 1 . Initial setup ① Import the module #####################
     from i611_MCS import *

## 2 . Initial settings②: Set operating conditions ############################## #
     Robot i611 constructor
          rb = i611Robot()
          # Determine the world coordinate system
          _BASE = Base()
     # Start connecting to the robot, initialize
          rb.open( True )
...

# Exception handling
# Exception detection and normal termination
        try:
        ...
          except Robot_error: # Exception raised due to error # Error handling technique (termination
          handling) required
```

Class	Robot_fatalerror()	i611 COM.	11 i6 MCS
Function	Exceptions occur when a fatal (unrecoverable) error occurs) This is the event handler for when a fatal error occurs.		
Argument	None		
Return value	None		

Usage examples

```
# Preparation
## 1 . Initial setup ① Import the module #####################
     from i611_MCS import *

## 2 . Initial settings②: Set operating conditions ##############################  #
     Robot i611 constructor
          rb = i611Robot()
                                        # Determine the world coordinate system
          _BASE = Base()
     # Start connecting to the robot, initialize
          rb.open( True )
...

# Exception handling
        # Exception detection and normal termination
        try:
        . . .
        except Robot_fatalerror: # Exceptions arise when there is a fatal error
        # Required error handling (termination handling) technique
```

Class	Robot_poweroff() i611 MCS
Function	Exception occurs when power is turned off (cannot be recovered) This is the power-off event handler.(*)
Argument	None
Return value	None

Usage examples

```
# Preparation
## 1 . Initial setup ① Import the module #######################
     from i611_MCS import *

## 2 . Initial settings②: Set operating conditions ################################ #
     Robot i611 constructor
          rb = i611Robot()
# Determine the world coordinate system
          _BASE = Base()
     # Start connecting to the robot, initialize
          rb.open( True )
…

# Exception handling
# Exception detection and normal termination
        try:
        . . .
      except Robot_poweroff: # Exception raised on power off # Error handling technique (termination handling) required
```

*) Occurs when the controller is visible PoF .

If you want to turn off the power, please complete the process until then.

Class	Robot_stop() i611 MCS
Function	Exception occurs when decelerating to stop This is the event handler for deceleration to stop.
Argument	None
Return value	None

Usage examples

```
# Preparation ---------------------------------------------------------------------------------
## 1 . Initial setup ① Import the module ######################
     from i611_MCS import *

## 2 . Initial settings②: Set operating conditions ############################## #
     Robot i611 constructor
          rb = i611Robot()
     # Determine the world coordinate system
          _BASE = Base()
     # Start connecting to the robot, initialize
          rb.open( True )
     # Enable an exception to be raised when entering “Deceleration Stop” during operation
          rb.enable_interrupt( 0, True )
…

# Exception handling------------------------------------------------------------------------
# Example 1: Exception detection and normal termination
        try:
        …
        except Robot_stop: # Event handler checks “Stop deceleration”
        # Required error handling (termination handling) technique
# Example 2: Detect exceptions and terminate errors
        try:
        …
        except Robot_emo: # Event handler checks “Stop deceleration”
        rb.exit(1)  # At the end there is errors occur
```

E.14 Occuring . (Refer to P. 57)

Add description of Robot_stop class

Action program behavior when deceleration stop occurs

Try code: … except: in the action program ・ ・

・ If not described :

The program ended with an error. Display controller E.16 .

・ If described :

The program ends normally.

6. Module : i611_io

Class	
(None) I/O Control	
Member variable	
—	—
Functions	
din()	I/O input P.114
dlyOut()	I/O output after specified time has elapsed P.115
dout()	I/O output P.115
IOinit()	Initialize I/O P.116
shotOut()	I/O output during specified time P.117
wait()	Wait until the specified I/O input pattern is reached P.118

Function	din()	i611 IO
Function	I/O input	
Argument	[*adr] : List	
	*adr [-] string	Input port ・When specifying an input port adr : input port number ・When specifying multiple port ranges adr[0] : Input port number (start) adr[1] : Input port number (end)
Return value	[*port] : string	
	*port [-]	Return execution result of input port to '0' or '1'' If multiple input ports are specified, will be received as list.
Usage example	# Example 1: Specify port number 15 if din (15) == '1': # Example 2: Specify port 8 to 10 if din (8, 10)[0] == '1': # When specifying port number 10 … elif din(8, 10)[1] == '1': # When specifying port number 9 … elif din(8, 10)[2] == '1': # When specifying port number 8	

⚠ ATTENTION

🚫	Do not use preset ports in init.py	⚠ (Incident)

Function	dlyOut()			i611 IO
Function	I/O output after specified time has elapsed			
Argument	[num, dat, tim]: List Keyword			
	num ঔ	[-] integer	Delay output port number	
	dat	[-] string	Output data from I/O Set as bit field in string (☞ See “Setting ports in bit fields” on page 116) '1' = ON '0' = OFF (Orignal value) '*' = Without change	
	tim ঔ	[s] integer	Delay time	
Return value	None			

Usage examples	# Example 1: List (output port: 8, data output: ON, delay time: 10 seconds))
	dlyOut(8, '1', 10)
	# Example 2: Keyword (output port: 1, data output: OFF, delay time: 10 seconds)
	dlyOut(num=1 ,dat='0', tim=10)			

Function	dout()			i611 IO
Function	I/O output			
Argument	[adr, data] : List Keyword			
	adr ঔ	[-] integer	Output port start number Installation range : 16 ~ 31	
	data ঔ	[-] string	Output data from I/O Set as bit field in string (☞ See “Setting ports in bit fields” on page 116) '1' = ON '0' = OFF (Original value) '*' = Without change	
Return value	None			

Usage examples	# Specify ON/OFF start address and output data (ports 20, 19, 18, 17, 16 ON)
	dout(16, '11111')			

Function	IOinit()	i611 IO
Function	Initialize I/O (*)	
Argument	[IPaddress, port] : List	
	IPaddress [-] string	IP Address Original value : '127.0.0.1'
	port [-] integer	Number of port Original value : 12345
	If the argument is omitted, it will be set to default	
Return value	None	

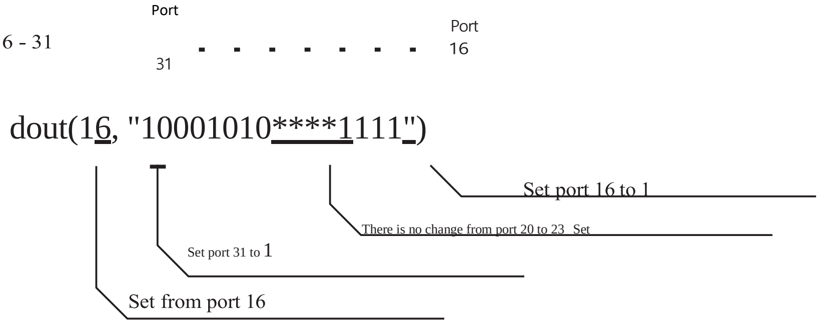
Usage examples	# Example 1: If argument is omitted (set to Initial Value)
	IOinit()
	# Example 2: List (all)
	IOinit('127.0.0.1', 12345)	

*) IOinit() does not affect what is stored.
Initializes the interface to access memory I/O.

Set the port in the bit field

The data part of the doubt(), layout(), shootOut(), wait() methods is a string in bitfield format

Example) Output port settings 16 - 31


Port
31
Port
16
dout(16, "10001010****1111")
Set port 16 to 1
There is no change from port 20 to 23 Set
Set port 31 to 1
Set from port 16

Function	shotOut()	i611 IO
Function	I/O output during specified time (When the time set in tm elapses, it will return to the previous I/O output)	
Argument	[adr, data, tm]: List Keyword	
	adr Cān [-] integer	Output port address number
	data Cān [-] string	Output data from I/O Set as bit field in string (☞ See “Setting ports in bit fields” on p 116) '1' = ON '0' = OFF (Original value) '*' = Without change
	tm Cān [s] integer	Output time
Return value	None	

Usage examples	# Example 1: List (output port: 8, data output: ON, output time: 10 seconds)
	shotOut(8, '1', 10)
	# Example 2: Keywords (output port: 1, data output: OFF, output time: 10 seconds))
	shotOut(adr=1, data='0', tm=10)	

Function	wait()	i611 IO
Function	Wait until the specified I/O input pattern is reached	
Argument	[adr, data, tm] : List Keyword	
	adr ঔ◌ঁ [-] integer	Input port address number
	data ঔ◌ঁ [-] string	Specify the data waiting for input '1' = ON '0' = OFF (Original value : 0)
	tm ঔ◌ঁ [s] float integer	End time
Return value	[res0, res1, res2] : List	
	res0 [-] integer	Result 1 : Matches input values 0 : Time out -1: Other errors
	res1 [-] string	Input value
	res2 [s] float integer	Time passes until the condition is established

Usage examples

```
# Example 1: List
    if wait( 8, '1', 10 )[0] == 1:
    if wait( 9, '1', 10 )[1] == '1':
    if wait( 9, '1', 10 )[2] > 10:

# Example 2: Keywords
    if wait( adr=1, data='1', tm=10 ) == 1:
```

7. Module : i611shm

Class	
(Not available) Access shared memory.	
Member variable	
-	-
Functions	
shm_read()	Read shared memory. P.119
shm_write()	Write to shared memory. P.120

Function	shm_read()		i611 shm
Function	Read shared memory.		
Argument	[index, num] : List		
	index	[-] integer	Readable shared memory address Installation range : 0x0100 -0x3800 「 Please refer to “Memory Map
	num	[-] integer	Number of variables read continuously Original value : 1
	If the num argument is omitted, it will set the default value		
Return value	res	[-] string	The string is separated by commas Return the value of the number specified with num as a comma-separated string

Usage examples	# Current command value of J1 (Joint coordinates)
	val_list = shm_read(0x3050, 6).split(',')
	joint0 = float(val_list[0])		

About comma separation

Use a comma separator (,) as the argument to .split(). This allows you to separate values.

```
val_list = shm_read( 0x3050, 6 ).split( ',' )
```

Function	shm_write()		i611 shm
Function	Write to shared memory		
Argument	[index, num] : List		
	index	[-] integer	Writable shared memory address Installation range : 0x1800-0x23F8 Please refer to “3 Memory Maps”” ■
	num	[-] integer	List or set of values to read consecutively
Return value	None		
Usage example	shm_write(0x1800, 10) shm_write(0x1C00, (3.5, 4.3))		

Writable memory and shares

Memory address	Type of table	Quantity
0x1800 -0x1BFC	integer (4byte)	256 bytes
0x1C00 -0x23F8	float (8byte)	256 bytes

D Software

3 Memory map

1. Start

Share memory and I/O memory are RAM (Variable memory) .

All memory contents will be deleted when power off. After starting the system, all values are initialized to 0.

When starting the system, everything except for the user block in the share memory is updated with the current new information.

Save memory content

Recommendations to save memory content to a file.

When saved, call Flush () or Close () after writing. (Flush (): Forced to delete the file buffer)

2. Share memory

1. Share memory structure

Data block	Offset	byte	Content
Title	+0x0000	256	Title in share memory
Memory I/O	+0x0100	1024	Contents like Memory I/O
(Reserve)	+0x0500	768	_
System administration	+0x0800	4096	System administration area
User	+0x1800	4096	User area (4 -byte integer and float type)
Manager controls	+0x2800	4096	Default processing area

Access to share memory

Access to the share memory, use the robot library SHM_READ (), SHM_WRITE ().

Shm_write (), can only be used with the user block. Other areas are only reading areas.

2. Memory map (share memory）

Title block

R : Only read ｜ R/W : Both Read and Write

Offset	Data	byte	Data type	Name of variable	Cont ent	Update cycle	Users access
+0x0000	(Reserve)	－	－	－	－	－	－
+0x0008	Update counter	4	unsigned short	update_counter	"1" ：Original update	1ms	R
+0x000C	The flag is updating	2	unsigned short	now_updating	Become 1 in the original update process.	1ms	R
+0x000E	(Reserve)	－	－	－	－	－	－

Memory I/O block

Offset	Data	byte	Data type	Name of variable	Con tent	Update cycle	Users
						access	
+0x0100	Digital In/Out	4	unsigned int	dio_io	I/O X16 input, X16 output	1ms	R
+0x0104	Hand Digital In/Out	4	unsigned int	dio_handio	I/O input (ARM) X8, X4 output	1ms	R
+0x0108	Reserve）	－	－	－	－	－	－
+0x0300	System SI (Input) 0	4	unsigned int	mio_si0	Servo status, emergency stop	1ms	R
+0x0304	Reserve）	－	－	－	－	－	－
+0x0308	System SI (Input) 2	4	unsigned int	mio_si2	The operating status of the user program	1ms	R
+0x030C	Reserve）	－	－	－	－	－	－
+0x0318	System SL (Output) 2	4	unsigned int	mio_sl2	Input implementation	1ms	R
+0x031C	Reserve）	－	－	－	－	－	－
+0x0320	User In/Out (output input)	480	unsigned int	mio_pi0[120]	User area	Frequent	R

2 Share memory

System management block

R : Only read | R/W : Both Read and Write

Offset	Data	byte	Data type	Name of variable	Content	Update cycle	Users access
+0x0800	robtask name	32	char (string)	robtask_name[32]	"Name of the user program registered in set_robtask"	When updated	R
+0x0820	running program name	32	char (string)	running_name[32]	Run the user program name	When updated	R
+0x0840	running program pid	4	unsigned int	running_pid	"pid of the user program is running"	When updated	R
+0x0844	assign_din(run)	2	short	assign_port[0]	Input gate (run) or -1	When updated	R
+0x0846	assign_din(stop)	2	short	assign_port[1]	Enter the designated port (stop) or -1	When updated	R
+0x0848	assing_din(err_reset)	2	short	assign_port[2]	Input allocation port (err_reset) or -1	When updated	R
+0x084A	assign_din(pause)	2	short	assign_port[3]	Enter the designated port (pause) or -1	When updated	R
+0x084C	assign_out(running)	2	short	assign_port[4]	Running or -1 output gate	When updated	R
+0x084E	assign_out(svon)	2	short	assign_port[5]	Output allocation port (svon) or -1	When updated	R
+0x0850	assign_out(emo)	2	short	assign_port[6]	Output assignment port (emo) or -1	When updated	R
+0x0852	assign_out(hw_error)	2	short	assign_port[7]	The output allocation port (hw_error) or -1	When updated	R
+0x0854	assign_out(sw_error)	2	short	assign_port[8]	The output allocation port (sw_error) or -1	When updated	R
+0x0856	assign_out(abs_lost)	2	short	assign_port[9]	The output allocation port (abs_lost) or -1	When updated	R
+0x0858	assign_out(in_pause)	2	short	assign_port[10]	The output allocation port (in_pause) or -1	When updated	R
+0x085A	assign_out(error)	2	short	assign_port[11]	Running or -1 output gate	When updated	R
+0x085C	（Reserve）	–	–	–	–	–	–

User block

Offset	Data	byte	Data type	Name of variable	Content	Update cycle	Users access
+0x1800	intval0	4	integer	intval0	User variable (integer)	Not following the cycle	R/W
+0x1804	intval1	4	integer	intval1	User variable (integer)	Not following the cycle	R/W
+0x1808	intval2 - intval255	1016	integer	intval(n)	User variable (integer)	Not following the cycle	R/W
+0x1C00	floatval0	8	double	floatval0	User variable (dynamic commas)	Not following the cycle	R/W
+0x1C08	floatval1	8	double	floatval1	User variable (dynamic commas)	Not following the cycle	R/W
+0x1C10	floatval2 - floatval255	2032	double	floatval(n)	User variable (dynamic commas)	Not following the cycle	R/W
+0x2400	(Reserve)	–	–	–	–	–	–

Share memory and number of user blocks can be recorded

Memory address	Type of variable	Quantity
[redacted]		

*) That is 4 bytes.

Control management block

Control status (csts)

R : Only read | R/W : Both Read and Write

Offset	Data	byte	Data type	Name of variable	Content	Update cycle	Users access
+0x2800	errcode	2	unsigned short	errcode	Số lỗi nghiêm trọng	When arising	R
+0x2802	bTeachMode	2	unsigned short	bTeachMode	Cờ trong giờ giảng dạy	When updated	R
+0x2804	bSPILargeFrame	2	unsigned short	bSPILargeFrame	Cờ giao tiếp SPI cho khung lớn	When updated	R

Operation information (rbcfg)

Offset	Data	byte	Data type	Name of variable	Content	Update cycle	Users access
+0x2C00	manip_type	36	char (string)	manip_type	Operation information	No update	R
+0x2C24	manip_serial	36	char (string)	manip_serial	Continuing operation	No update	R
+0x2C48	format_version(major)	4	unsigned int	format_version[0]	Data structure version	No update	R
+0x2C4C	format_version(minor)	4	unsigned int	format_version[1]	Data structure version	No update	R
+0x2C50	format_version(patch)	4	unsigned int	format_version[2]	Data structure version	No update	R
+0x2C54	parameter_version(major)	4	unsigned int	parameter_version[0]	Data structure version	No update	R
+0x2C58	parameter_version(minor)	4	unsigned int	parameter_version[1]	Data structure version	No update	R
+0x2C5C	parameter_version(patch)	4	unsigned int	parameter_version[2]	Data structure version	No update	R
+0x2C60	(Reserve)	–	–	–	–	–	–

Robot status (rbsts)

R : Only read | R/W : Both Read and Write

Offset	Data	byte	Data type	Name of variable	Content	Update cycle	Users access
+0x3000	Command value (Descartes coordinates)	8	double	cmdx	Current command value	1ms	R
+0x3008	Command value (Descartes coordinates) Y [mm]	8	double	cmdy	Current command value	1ms	R
+0x3010	Command value (Descartes coordinates) Z [mm]	8	double	cmdz	Current command value	1ms	R
+0x3018	Command value (Descartes coordinates) Rz [deg]	8	double	cmdrz	Current command value	1ms	R
+0x3020	Command value (Descartes coordinates) Ry[deg]	8	double	cmdry	Current command value	1ms	R
+0x3028	Command value (Descartes coordinates) Rx[deg]	8	double	cmdrx	Current command value	1ms	R
+0x3030	(Reserve)	－	－	－	－	－	－
+0x3040	Arm posture (rescue flag)	4	unsigned int	posture	Posture (0 ～ 7)	1ms	R
+0x3044	(Reserve)	－	－	－	－	－	－
+0x3048	Specific score information	4	unsigned int	singular	Is there a special feature in the current position?	1ms	R
+0x304C	Multi -rotation information	4	unsigned int	multiturn	Multiple rotation information on each axis	1ms	R
+0x3050	Command value (Joint) J1[deg]	8	double	joint[0]	Current command value	1ms	R
+0x3058	Command value (Joint) J2[deg]	8	double	joint[1]	Current command value	1ms	R
+0x3060	Command value (Joint) J3[deg]	8	double	joint[2]	Current command value	1ms	R
+0x3068	Command value (Joint) J4[deg]	8	double	joint[3]	Current command value	1ms	R
+0x3070	Command value (Joint) J5[deg]	8	double	joint[4]	Current command value	1ms	R
+0x3078	Command value (Joint) J6[deg]	8	double	joint[5]	Current command value	1ms	R
+0x3080	(Reserve)	－	－	－	－	－	－
+0x3090	Speed	8	double	velocity	Current command value	1ms	R
+0x3098	Abnormal speed	4	unsigned int	vel_error_axes	The shaft has a speed error	When arising	R
+0x309C	Soft limit	4	unsigned int	softlimit	Is there near the limit of each axis	1ms	R
+0x30A0	Location right before TURNING OFF servo J1 [Rad]	8	double	joint_svon_to_svoff[0]	Location right before TURNING OFF	When arising	R
+0x30A8	Location right before TURNING OFF servo J2[rad]	8	double	joint_svon_to_svoff[1]	Location right before TURNING OFF	When arising	R
+0x30B0	Location right before TURNING OFF servo J3[rad]	8	double	joint_svon_to_svoff[2]	Location right before TURNING OFF	When arising	R
+0x30B8	Location right before TURNING OFF servo J4[rad]	8	double	joint_svon_to_svoff[3]	Location right before TURNING OFF	When arising	R
+0x30C0	Location right before TURNING OFF servo J5[rad]	8	double	joint_svon_to_svoff[4]	Location right before TURNING OFF	When arising	R
+0x30C8	Location right before TURNING OFF servo J6[rad]	8	double	joint_svon_to_svoff[5]	Location right before TURNING OFF	When arising	R
+0x30D0	(Reserve)	－	－	－	－	－	－
+0x30E0	Location storage flag right before TURNING OFF the servo	4	unsigned int	b_saved	Is the location right before TURNING OFF valid?	When arising	R
+0x30E4	(Reserve)	－	－	－	－	－	－
+0x37E8	(Reserve)	－	－	－	－	－	－
+0x37F0	(Reserve)	－	－	－	－	－	－
+0x37F8	(Reserve)	－	－	－	－	－	－

3. Memory I/O

1. Memory map (Physical I/O)

R : Only read ｜ R/W : Both Read and Write

Type	Address	Content	Connected terminal （Signal name）	Python port number		Users access
Physical Dido of 4 -byte controller (I/O connector)	0L	Digital input CN3: I/O 1 connector (input)	2A (IN1)	0	0x0000	R
			2B (IN2)	1	0x0001	R
			3A (IN3)	2	0x0002	R
			3B (IN4)	3	0x0003	R
			4A (IN5)	4	0x0004	R
			4B (IN6)	5	0x0005	R
			5A (IN7)	6	0x0006	R
			5B (IN8)	7	0x0007	R
			6A (IN9)	8	0x0008	R
			6B (IN10)	9	0x0009	R
			7A (IN11)	10	0x000A	R
			7B (IN12)	11	0x000B	R
			8A (IN13)	12	0x000C	R
			8B (IN14)	13	0x000D	R
			9A (IN15)	14	0x000E	R
			9B (IN16)	15	0x000F	R
	0H	Digital output CN4: I/O 2 connector (output)	2A (O1P), 2B (O1N)	16	0x0010	R/W
			3A (O2P), 3B (O2N)	17	0x0011	R/W
			4A (O3P), 4B (O3N)	18	0x0012	R/W
			5A (O4P), 5B (O4N)	19	0x0013	R/W
			6A (O5P), 6B (O5N)	20	0x0014	R/W
			7A (O6P), 7B (O6N)	21	0x0015	R/W
			8A (O7P), 8B (O7N)	22	0x0016	R/W
			9A (O8P), 9A (O8N)	23	0x0017	R/W
		Digital output CN5: I/O 3 connector (output)	2A (O9P), 2B (O9N)	24	0x0018	R/W
			3A (O10P), 3B (O10N)	25	0x0019	R/W
			4A (O11P), 4B (O11N)	26	0x001A	R/W
			5A (O12P), 5B (O12N)	27	0x001B	R/W
			6A (O13P), 6B (O13N)	28	0x001C	R/W
			7A (O14P), 7B (O14N)	29	0x001D	R/W
			8A (O15P), 8B (O15N)	30	0x001E	R/W
			9A (O16P), 9A (O16N)	31	0x001F	R/W
ArmDIDO 8 byte (I/O connector arm)	1L	Digital input	6A (I1)	32	0x0020	R
			6B (I2)	33	0x0021	R
			5A (I3)	34	0x0022	R
			5B (I4)	35	0x0023	R
		(Reserve)	–	36 - 47	0x0024 -0x002F	–
	1H	Digital output	3A (O1)	48	0x0030	R/W
			3B (O2)	49	0x0031	R/W
		(Reserve)	–	50 - 63	0x0032 -0x003F	–
	2	(Reserve)	–	64 - 95	0x0040 -0x005F	–
(Reserve)	3 - 127	–	–	96 -4095	0x0060 -0x0FFF	–

2. Memory map (System I/O)

R : Only read | R/W : Both Read and Write

Type	Address	Content	Python port number access		Users
System SI 16 bytes	128	Servo status	4096	0x1000	R
		emergency stop	4097	0x1001	R
		Error caused by the system (serious)	4098	0x1002	R
		ABS loss	4099	0x1003	R
		（Reserve）	4100 -4103	0x1004 -0x1007	－
		(Reserve)	4104 -4111	0x1008 -0x100F	－
		(Reserve)	4112 -4127	0x1010 -0x101F	－
	129	(Reserve)	4128 -4159	0x1020 -0x103F	－
	130	Robot program status	4160	0x1040	R/W
		Error status determined by the system	4161	0x1041	R/W
		Pause status	4162	0x1042	R/W
		System error status (*)	4163	0x1043	R/W
		System status	4164 -4167	0x1044 -0x1047	R/W
		error code	4168 -4175	0x1048 -0x104F	R/W
		(Reserve)	4176- 4183	0x1050 -0x1057	－
		(Reserve)	4184- 4187	0x1058 -0x105B	－
		(Reserve)	4188	0x105C	－
		(Reserve)	4189 -4191	0x105D -0x105F	－
	131	(Reserve)	4192 -4223	0x1060 -0x107F	－
System SL 16 bytes	132	(Reserve)	4224 -4255	0x1080 -0x109F	－
	133	(Reserve)	4256 -4287	0x10A0 -0x10BF	－
	134	Run the robot program	4288	0x10C0	R/W
		Decelerate to stop	4289	0x10C1	R/W
		reset the error	4290	0x10C2	R/W
		Pause	4291	0x10C3	R/W
		(Reserve)	4292 -4319	0x10C4 -0x10DF	－
	135	(Reserve)	4320 -4351	0x10E0 -0x10FF	－
User Input/Output 480 bytes	136-255	For users (*)	4352 -8191	0x1100 -0x1FFF	R/W

*) Error has been defined by the system (not serious or serious).

About I/O Memory

Physical I/O and System I/O are collectively referred to as I/O memory.
IEO () function only creates the interface to access I/O memory. Does not affect memory content.

The relationship between I/O allocation and Python port

Input

Functions	Related methods	Python port number	
		Physical I/O (*2)	Digital I/O
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]

In

Functions	Related methods	Số cổng Python	
		Physical I/O (*2)	Digital I/O
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]

*1) Error status caused by the system determined (not serious or serious).

Used to check 2 errors with 1 control line.

*2) This is recommended settings.

D Software

4 Steps to implement the program

The whole process

1. How to start robot program

Step 1 Prepare teaching points.

Prepare teaching points in advance.

Also, please refer to the guidebook "Teaching ".)

tep 2 Write robot program

Create a robot program in the text editor on PC.

Establish prepared teaching points for step 1 for the robot program.

(「 Please refer to "programming guide" and "2 robot libraries".)

DStep 3 Transfer robot program to controller.

File transmission between PC and controller with FTP client software.

(Please refer to the user manual "guide c" to know the connection method.)

ước 4 Run the robot program.

There are two methods to start.

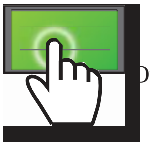
Method 1 「When starting from" I/O input "

Run the robot program from I/O input (hardware signal).

Set the port I/O 5 init.py.

Mainly used in automatic operation at production facilities.

(Please refer to "1 programming guide" and "2 robot libraries".)



Or

(Init.py is prepared in the controller.)

Step 2 「When starting from "terminal software"

Use terminal software to connect with the controller and run the transferred robot program.

ttermpro.exe

Mainly used for testing and removing holes.

(Please refer to the user manual "guide c" to know the connection method)

1 The whole process

⚠ ATTENTION

🚫	Do not write a robot program in Init.py to start operating automatically as soon as the power is turned on.	⚠ ⚠ ⚠ ⚠
❗	When combining or sharing robots with a machine that can damage the robot, we recommend that you test all functional programs and motion individually, outside the working area of another machine.	⚠ ⚠ ⚠ ⚠

포인트!

The relationship between the promoting method of the robot program and the log output

The different diary data export method depends on the implementation method of the robot program. 「

When starting from "I/O input”

Standard output, error output is stored in a file.

▪ Robot program

Output information	Save location	File name
Standard output	/opt/i611/log	userporg_out�log
Error output		userporg_err�log

Output information	Save location	File name
Standard output	/opt/i611/log	sys_out�log
Error output		sys_err�log

「When starting from "terminal software”

▪ init.py

Not stored in log files.

All output data, including error information, is displayed on the screen of the section

▪ Robot program (xxx.py)

Users can freely create robot movements with a robot library. Users can freely name the file. (xxx.py)

▪ System program (init.py)

Set description for control on I/O input by using Robsys layers in the robot library. Do not change the file name.

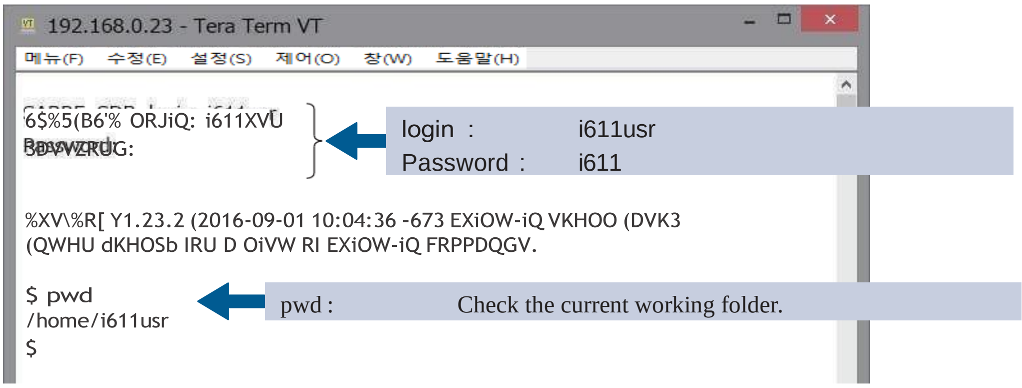
When starting from "I/O input"

Please refer 「1 Programming guide」, 「2 Robot library」.

When starting from "terminal software"

Start communicating with the controller.

(For example, the screen is Windows 8.1)

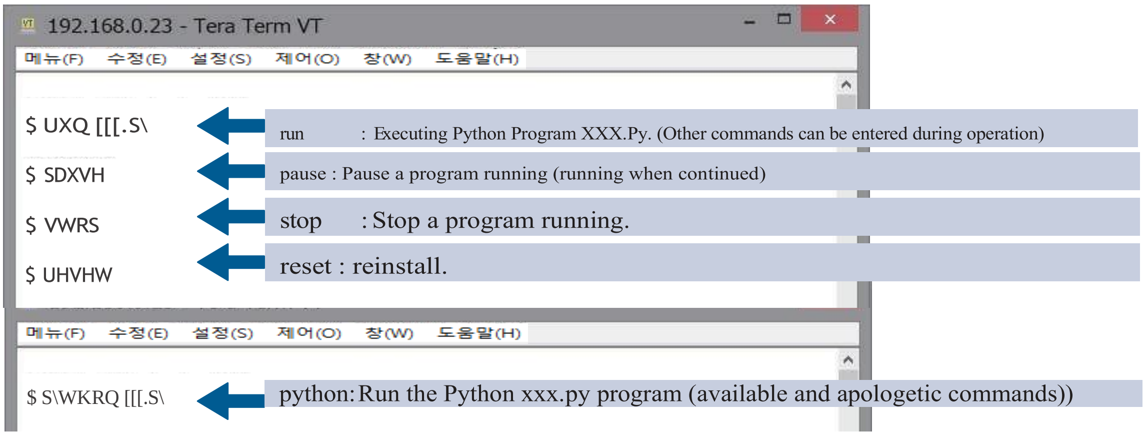

192.168.0.23 - Tera Term VT
메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

```
6$%5(B6'% ORJiQ: i611XVU
3DVVZRUG:

%XV\%R[ Y1.23.2 (2016-09-01 10:04:36 -673 EXiOW-iQ VKHOO (DVK3
(QWHU dKHOSb IRU D OiVW RI EXiOW-iQ FRPPDQGV.

$ pwd
/home/i611usr
$
```

login : i611usr
Password : i611

pwd : Check the current working folder.

Run the program and start control.


192.168.0.23 - Tera Term VT
메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)
메뉴(F) 수정(E) 설정(S) 제어(O) 창(W) 도움말(H)

```
$ UXQ [[[.S\
$ SDXVH
$ VWRS
$ UHVHW
```

run : Executing Python Program XXX.Py. (Other commands can be entered during operation)

pause : Pause a program running (running when continued)

stop : Stop a program running.

reset : reinstall.

```
$ S\WKRQ [[[.S\
```

python: Run the Python xxx.py program (available and apologetic commands))

The notes when using the Print () statement

Do not use too many print () statements when executed via I/O. If the error exceeds the controller capacity (C06) occurs due to this error, delete the log file and delete the print () statement. If you need to log in a large capacity, the log file should be written by using Rotatingfilehandler, Timedrotatingfilehandler of the log library instead of using print () ..

Related path:
https://docs.python.org/ko/3/library/logging.handlers.html#logging.handlers.RotatingFileHandler
https://docs.python.org/ko/3/library/logging.handlers.html#logging.handlers.TimedRotatingFileHandler

DOCUMENTS

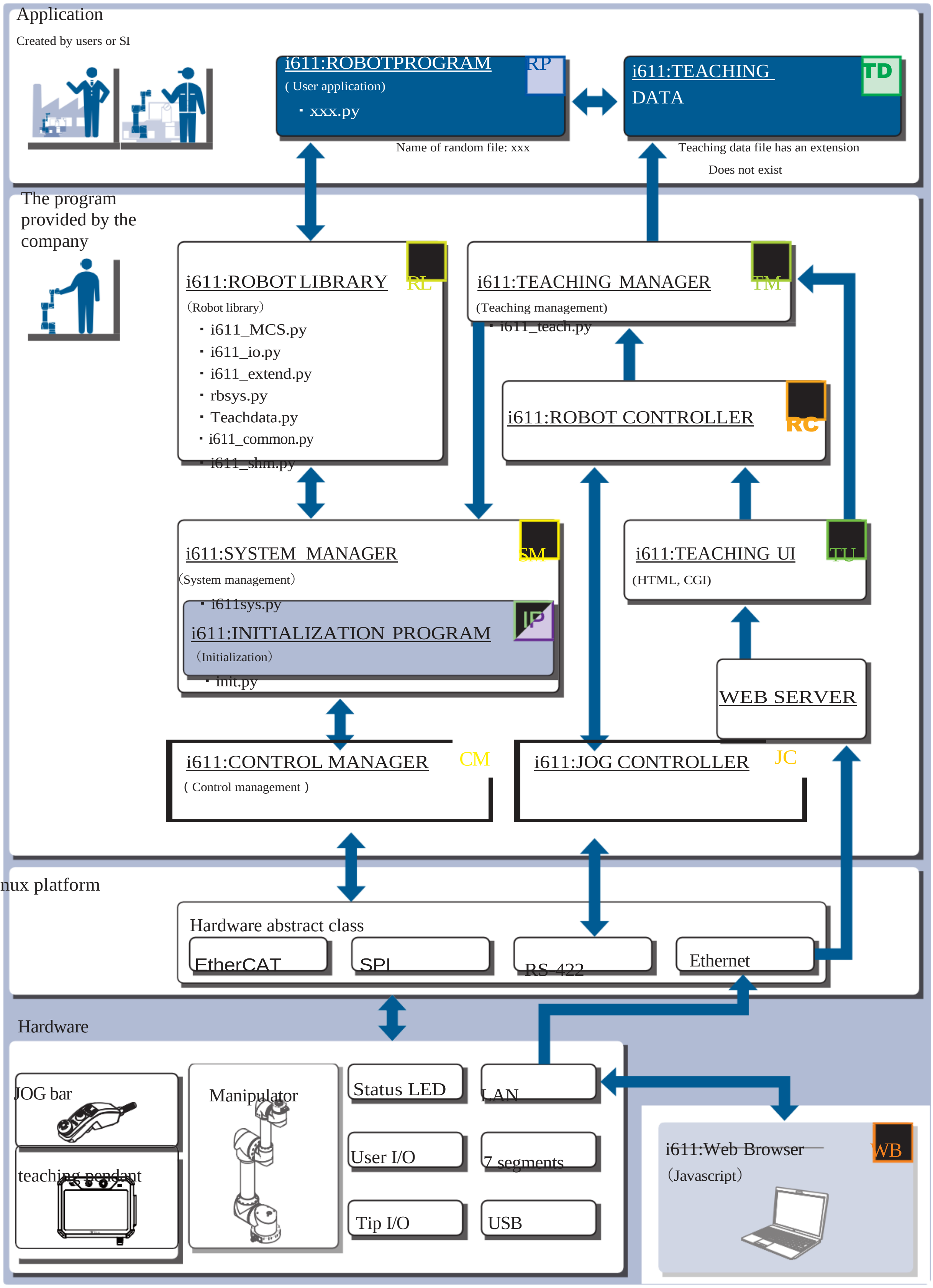
1. Block diagram
2. Maintenance
3. Terminology
4. Solving problems

MEMO

Z Data section

1 BLOCK DIAGRAM

1. System block diagram

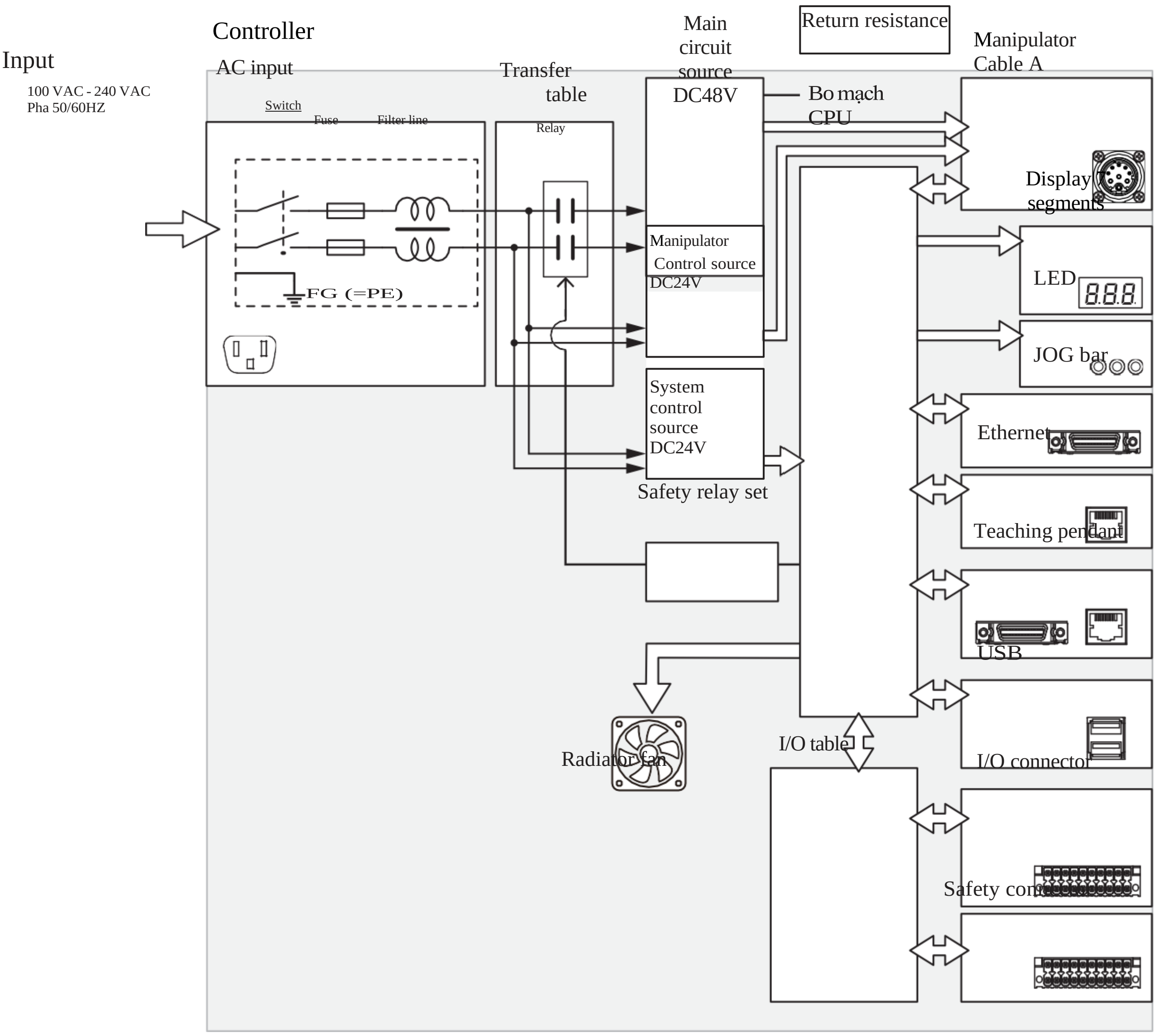


2. Program list

Program name	Summary
i611:ROBOT PROGRAM RP	Created by users or SI. This is a controller control robot program ..
i611:TEACHING DATA TD	Created by users or SI. This is the file to save the coordinate information set by teaching.
i611:INITIALIZATION PROGRAM IP	Provided by the manufacturer. Can be modified by users or SI. This is the file set to explain the I/O settings and how to run the robot program ..
i611:SYSTEM MANAGER SM	Provided by the manufacturer. Make status management, control the status of teaching and handle errors of the robot program
i611:ROBOT LIBRARY RL	Provided by the manufacturer. This file contains different modules necessary to program the movement of the robot ..
i611:CONTROL MANAGER CM	Provided by the manufacturer. Status management, including booting and stopping the system as well as handling errors. Calculate the relationship between the angle and the posture (the World coordinate system) of each joint of the machine in real time and perform the control/speed control. This is an important part that makes up the movement.
i611:TEACHING MANAGER TM	Provided by the manufacturer. Used to locate the controller, the output of the coordinates is stored in a file and allows the use of the output coordinates in the user program
i611:ROBOT CONTROLLER RC	Provided by the manufacturer. Control user interface and JOG behavior.
i611:JOG CONTROLLER JC	Provided by the manufacturer. Control the JOG bar. Get and process vibration signals, led lights and horns from Robot Controller.
i611:TEACHING UI TU	Provided by the manufacturer. This is a browser user interface to establish teaching or operating robots.
i611:Web Browser WB	Use Chrome Browser provided by Google JavaScript operation for teaching.

2. Hardware block diagram

1. Control block diagram



Z Data section

2 MAINTENANCE

1. Checking

1. Points to note when checking

⚠ ATTENTION

❗	To use this product safely and for a long time, perform checks to prevent problems and ensure safety..	⚠
🚫	Do not use petroleum-based products when cleaning plastic parts.	⚠ (Deformation/discoloration)

Start checking after performing the following preparation steps.

1. Place “check” signs on controls and entrances to secure areas to prevent other workers from working inside while moving.
2. Workers ensure priority control by locking controls before work and carrying the key.
3. Ensure enough space and light for work.
4. The inspection is performed by someone who has completed a special training course on industrial robots.
5. Arrange the person on duty in a position to observe the entire situation and prepare for an immediate emergency stop.
6. Check the method of transmitting signals to each other.
7. Store inspection records for over 3 years.

Based on inspection time and operating time

15h/ day × 20 days / month ×3 months ＝about 1,000h

Make sure to perform daily or regular checks to detect any abnormalities.

If there is a problem, it must be repaired immediately or take necessary measures to prevent the problem and ensure safety.

Try to perform outside the range of motion as much as possible, if working within the range of motion is unavoidable then take safety measures before doing it.

It is recommended to perform maintenance and inspection of the entire system according to the system integrated maintenance plan. Do not perform major testing (insulation resistance measurement).

2. Daily inspection and periodic inspection

Daily inspection；Perform before operation

Before turning on the power (Check the checklist below before turning on the power.)

Test item (content)	Measures when there are abnormalities
1. Is the power cable securely connected?	Be sure to connect.
2. Are operator cables plugged in and locked?	Be sure to connect.
3. Are the I/O connectors and safety connectors securely connected??	Be sure to connect.
4. Are the controller connections loose?	Tighten the bolts securely.
5. Are the upper flange mounting bolts loose?	Tighten the bolts securely.
6. Are there any cracks or chips in the plastic part of the manipulator?	Stop using and contact a service center.
7. Are there any foreign substances such as powder or oil?	Check for any problems, clean and remove.
8. Are there any objects in the area of operation?	Remove objects to avoid interference.
9. Are the controller's intake and exhaust ports clogged with dust?	Please clean and remove.
10. Are there any cracks or chips on the JOG bar (optional product)?	Use products that do not have cracks or chip.
11. Are there any cracks or chips on the teaching pendant (optional product)?	Use products that do not have cracks or chip.
12. Are the main cable of the teaching device and the communication cable securely connected?	Be sure to connect.
13. Is the cable likely to be destroyed or damaged?	Use cables that are not torn or damaged.
14. Is the cable submerged in oil or water?	Keep clean from oil and water.
15. Is the power supply and voltage normal?	Check to see if there are any problems.
16. Is there a strange smell?	Stop using and contact a service center.
17. Is there oil, moisture, dust or foreign matter on the connector on the front of the controller?	Keep clean from oil and water.
18. Are operating temperatures and humidity within the range of usage conditions?	Use within the range of usage environments.
19. Are any devices or equipment connections loose or misaligned?	Check to see if there are any errors.
20. Are there any foreign substances in moving parts such as joints and end devices?	Check to see if there are any errors.

After turning on the power (Turn on the power while paying attention to the robot.)

Checking items (content)	Measures when there are abnormalities
21. Are there any unusual movements, strange sounds, strange smells when the power is turned on?	Be sure to connect.

During control (while running a program)

Checking items (content)	Measures when there are abnormalities
22. Is there an error in the position of the manipulator?	Are the bolts on the base or end-effector loose? Has the position of the fixture changed?
23. Does the controller operate abnormally, vibrate abnormally, or emit strange sounds or smells due to program operations?	Contact the service center.

1 Checking

Periodic inspection; Conduct more detailed inspections than daily inspections once a month

Manipulator

Checking items (content)	Measures when there are abnormalities
1. Are the bolts on each part of the manipulator loose?	Tighten the bolts securely.
2. Are the connector fixing bolts or the bolts on the connection terminal block loose?	Tighten the bolts.
3. Are there any strange noises coming from the coupling part (reducer)?	Contact the service center.

Controller

Checking items (content)	Measures when there are abnormalities
1. Are the controller's intake and exhaust filters dirty?	Clean or replace with new parts.

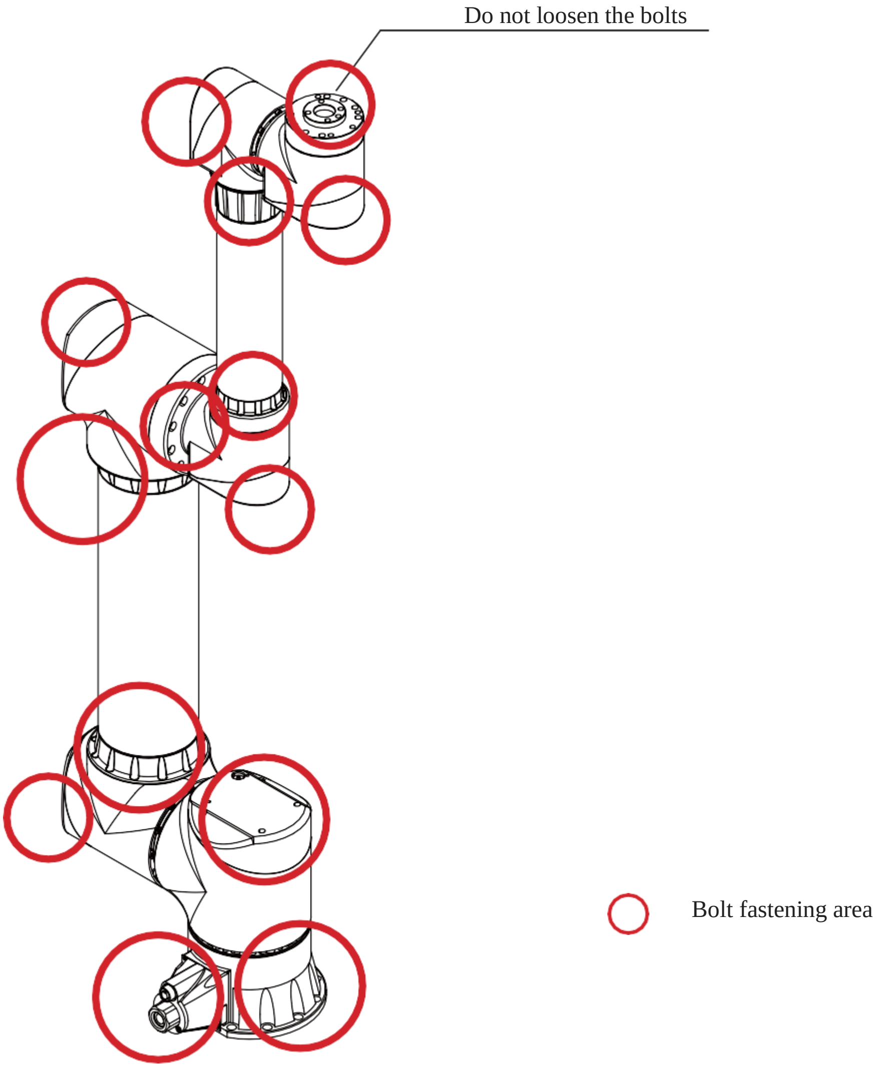
Teaching pendant (Optional)

Checking items (content)	Measures when there are abnormalities
1. Are there strange sounds coming from the Teaching pendant's speakers?	Contact the service center.
2. Is the Teaching pendant filter dirty?	Contact the service center.

2. Maintenance

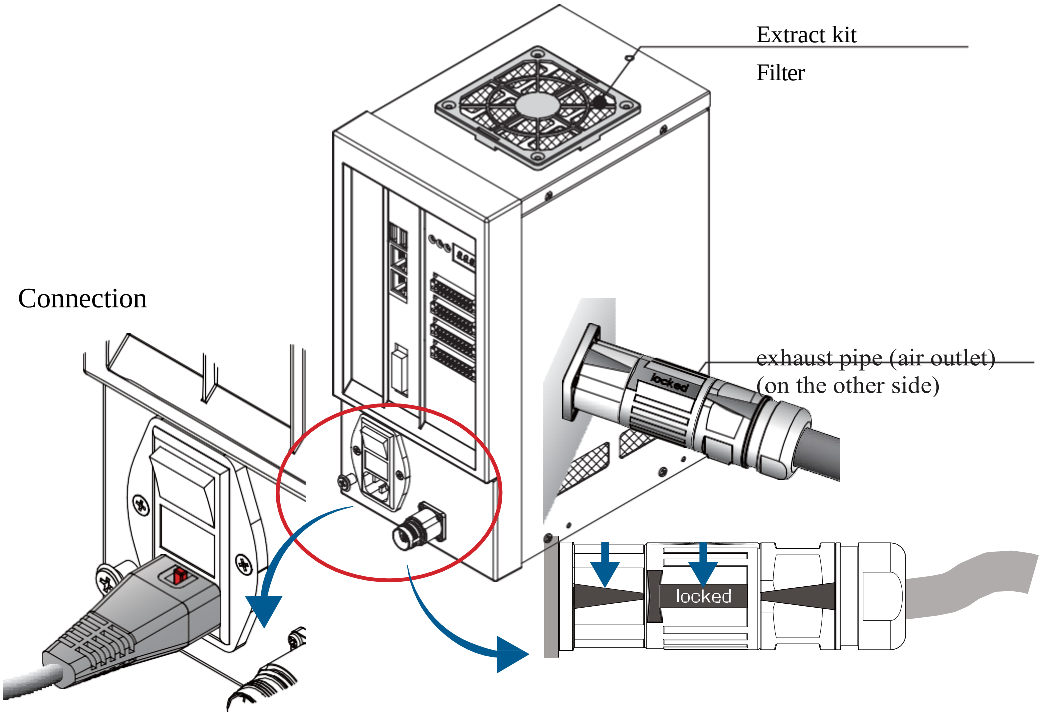
1. Manipulator

Checking items	Content
【Before driving】 external inspection	Check to see if any foreign substances such as powder or oil have seeped into the arm or joint area. Check to see if the bolts are loose. Make sure the encoder cover is not damaged.
【During driving】 Noise, position error	Check for any abnormalities in the steering sound. Make sure no location errors occur.



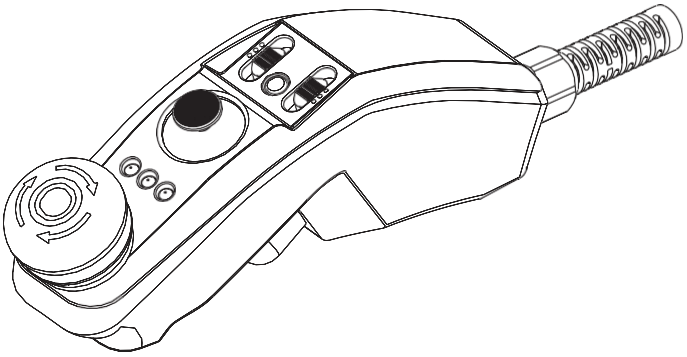
2. Controller

Checking items	Content
Cooling fan exhaust kit	Check if the exhaust port is blocked by dust or foreign objects. Air vents are located on the left and right sides of the controller.
Cooling fan extract kit	Check that the air intake port is not blocked by dust or foreign objects. If blockage or filter damage is observed, replace the filter.
Connection	Check if it is securely attached as shown in the picture below. Make sure there is no dust or foreign objects.



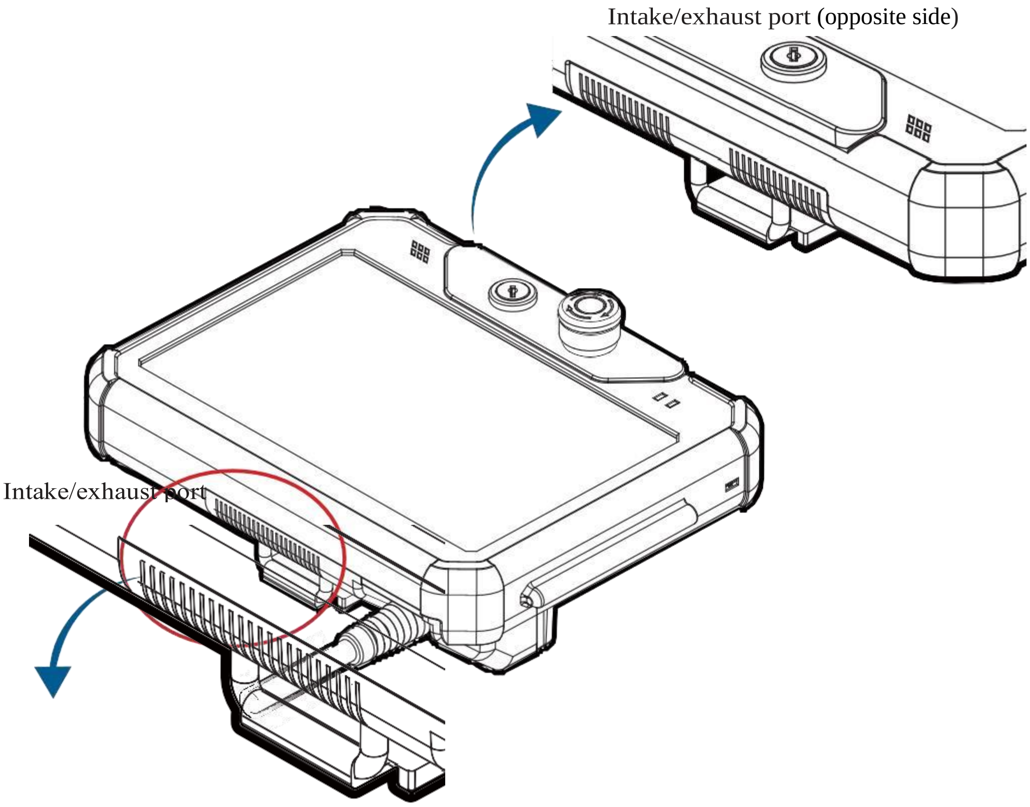
3. JOG bar

Checking items	Content
External inspection	Make sure there are no cracks or crevices.



4. Teaching pendant

Checking items	Content
External inspection	Make sure there are no cracks
	Check that the air intake and exhaust ports are not blocked by dust or foreign substances. If blockage or filter damage is observed, contact a service center.
Connection	Make sure they are securely fastened. Make sure there is no dust or foreign objects.



MEMO

Z Data section

3 TERMINOLOGY

3. Terminology

1. Terminology

Data section

A		
ABS Encoder	Absolute (absolute) encoder. The detector can output angle data to the outside. Location information is not lost even after power off	
ABS Loss	The encoder of the joint part loses absolute position information. This mainly happens when the brake is released while the controller is turned off. ABS Need to go back to the roots.	
ABS Zero Return	Active recovery from ABS loss. Reset the encoder of the joint to its original position with the control set to zero and regenerate the angle data. .	
API	Abbreviation for Application Programming Interface. software interface.	
Arm	The aluminum box between the joints. Length varies for each model.	
Asynchronous System	A system that allows simultaneous processing of other tasks during the exercise movement process. Predicts the target point and allows switching to the target point during movement.	
B		
Base coordinate System	The coordinate system is located on the robot's base floor Basically, when the offset is 0, it matches the world coordinate system.	
C		
C.CODE	This code combines controller and controller. Assign to each object. Abbreviation for Connection Code.	
CN1 Connector	Connector on the controller side for connecting the operation cable.	
CN2 Connector	Connector on controller side for connecting JOG stick or dummy connector.	
Command queue	Function that specifies the order of multiple commands, such as file transfers, command line processing, or session end commands	
Controller	A device that comprehensively controls the complex movements of the manipulator	
Control Manager	State management includes starting and stopping the system, error handling, calculations to determine the relationship between the angles of each joint of the robot and the position and posture of the robot's fingertips (WORLD coordinate system) and acceleration/deceleration control.	
Crossover counter	Settings to convert position type position data to unique joint type angle data. Set in the multidimensional parameter of the Position Type data. The value is updated when the angle of each joint exceeds ± 180°.	
E		
Euler angles	Represents the position relationship of two orthogonal coordinate systems.	

F		
FTP	Is a communication protocol for transferring files over the network (short for File Transfer Protocol).	
Factory default settings	Factory reset state.	
First Arm	Middle arm J2 - J3 .	
H		
Home Position	Position where all joints of the manipulator are at 0 degrees.	
Home Position	Position 0deg on each axis in the joint coordinate system (0, 0, 0, 0, 0, 0).	
Hold	Slow down to a stop, stop and wait. A state in which a robot program can be reused without termination.	
I		
I/O Start	Operation to start a robot program with a command entered into physical I/O or memory I/O.	
Initialization Program	Initial setup program (init.py)	
Initial Value	Initial default settings	
J		
Jog Stick	The device allows manual operation of the robot. Use during teaching.	
Joint	Moving parts of manipulatorn. Integrated structure of motor and encoder.	
Jumper Connector	Connector for automatic mode. Accessory.	
Joint coordinate System	The coordinate system is set by the joint axis. This is the coordinate system used to determine the position and posture according to the angles of the J1 axis to the J6 axis of the robot.	

L		
Linear Interpolation	While synchronously controlling the X-Y-Z axes, the composite trajectory will move so that it becomes a straight line. Similar to Line's operation.	
Line Motion	While synchronously controlling the X-Y-Z axes, the composite trajectory will move so that it becomes a straight line. Similar to linear interpolation.	
M		
Manipulator	Among the parts of the robot, the part that performs physical movements. Includes multiple arms and joints.	
Manipulator Cable	Controller and controller cable connection.	
Mechanical Home Position	Posture when all joints of the operator are aligned with the 0 mark. Posture when returning to ABS power	
Memory I/O	Generic term for controller physical I/O and system I/O.	
MDO Middle Digital Out	Function to switch I/O output to LOW/HIGH under specified conditions during operation..	
Multiturn	Cross counter information	
O		
Orthogonal coordinate system	X-Y-Z axis coordinate system. WORLD coordinate system. Base coordinate system. Generic term for all Cartesian coordinate systems, including user coordinate systems.	
Override	Override the setting value by multiplying the speed setting value by the ratio (%).	
P		
Parent Coordinate System	The information is used to establish guide points in the Cartesian coordinate system relative to the WORLD coordinate system. Used in Location class and Coordinate class.	
Physical I/O	The controller connects to the Tip I/O port	
Point To Point Motion	Action in which all joints move in a smooth curve at a constant speed toward a target coordinate	
Position	Location information	

Posture	The operator's posture information is represented by numbers 1 to 8.	
R		
Resume	Stop condition. Restart the activity in the hold state.	
Robot	The general term includes manipulator and controller.	
Robot Library	The file contains many modules needed to program the robot's movements..	
Robot Position	Controller end coordinates (Position type).	
S		
Safety Connector	An interface connector connects to a separate external protection device to cut off the robot's drive power and stop the manipulator operation in case of abnormal phenomena.	
Safety Plug	Similar to interlock plugs. The plug may block the steering control circuit for safety	
Second Arm	Middle arm J4 - J5	
Servo communication	Communication between 6 joints of manipulator and controller. Send and receive operating commands and status monitoring data.	
Slow down to Stop	Stop when decelerating by servo control.	
Step stop	Define a motor movement command as a single execution step, stopping each time the movement is completed.	
Synchronous System	A system that makes other tasks wait until the target point is reached during locomotor movement	
System I/O	System and program ports	
System Manager	Implement user robot program status management system control, teaching status control and error handling..	

T		
Task coordinate System	The coordinate system is determined by the task	
Teaching	The real task is to move the controller to remember the operations of the control program. Set the necessary information for operation using the JOG bar and PC.	
Teaching Data	Teaching point data file	
Teaching Manager	Teaching task control work (i611_teach.py)。	
Teaching Parameter	Teaching point information. Includes coordinates and posture values.	
Teaching Pendant Tablet	Tablet computers have the ability to change many settings for teaching	
Teaching Point	The operator's coordinates are set by the instructor. Postures are included.	
Tool Coordinate System	The coordinate system is established based on the tool, the last part of the manipulator	
Tool Flange	The top flange of the mechanical interface with the tool contact surface is flat, as is the distal (end) flange.	
Tool I/O	The instrument's electrical interface is mounted at the top of the controller	
Top Flange	The top flange of the mechanical interface with the flat tool contact surface. Similar to the tool flange..	
U		
User Robot Program	User-generated robot motion program. (= robot program)	
W		
Work	Products and parts are being processed.	
World coordinate System	Coordinate system located on the ground or work floor Since the initial offset is 0, it matches the base coordinate system.	

Z Data section

4 Solving problems

1. Configure error Log

The robot saves error log when detecting abnormalities.
If an error occurs, download the error log to your PC and contact the service center.

The Error Log file is a compressed file in .tgz format. Log compression error

Error Log folder：/opt/i611/log

ndstatus	Status log
userprog_out.log	User robot program output (print output)
userprog_err.log	User robot program output (exception output)
sys_out.log	System administrator output (print output)
sys_err.log	System administrator output (exception output)

（If the error log file size exceeds 200kB, it will be divided and saved..)

2. How to receive error Log

Step 1 Connect to PC.

Please refer teaching .

ttermpro.exe

FFFTP.ex

Step 2 Select location to save errror log

When not using USB memory

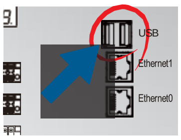
Download to computer.

（The error log is stored in the controller.）

When using USB memory

Download USB memory

Insert the USB memory into the controller



（Controller）

Step 3 Start a terminal program and connect to Telnet.

Run get_log.sh in /opt/i611/tools

Step 4 Receive error log

When not using USB memory

Transfer the error log generated in the /tmp controller from the FTP client to the PC. .

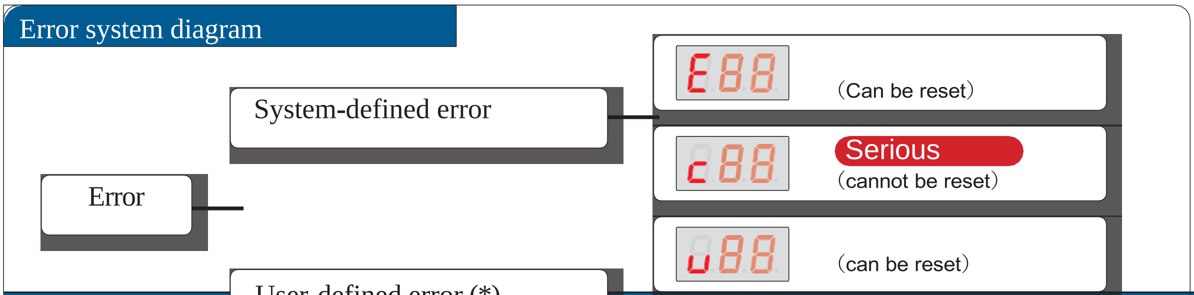
When using USB

Error log is created in USB memory..

Example of error log file：Log_SN16110017A_20170123_102914.tgz

1. Type of error

Errors are classified into four types.
Check the error type or code. Please refer to the troubleshooting guide. Errors are displayed on a 7-segment LED on the front of the controller.


Error system diagram

Type of error		
System-defined error	Robot will not operate until an error is resolved	
	User program	An exception arises.
	How to reset	Eliminate the cause of the error and enter the error reset command (cmd_reset()) or I/O (it will enter standby state after reset.)
System-defined error E88	The robot will not operate until the power is turned back on. The internal workings of the controller continue.	
	User program	Forced exit.
	How to reset	Remove the cause of the error and turn the power back on.
User-defined error Serious (*) c88	Occurs when a user program calls a dedicated API. The robot will not operate until the error is resolved.	
	User program	An exception occurred.
	How to reset	Eliminate the cause of the error and enter the error reset command (cmd_reset()) or I/O (enter standby state after reset.)
System-defined error Serious (*) r88	The robot will not operate until the power is turned back on. The internal workings of the controller continue.	
	User program	An exception occurred.
	How to reset	Remove the cause of the error and turn the power back on.

*) The two-digit number of the user-defined error code. Write appropriately for the application you use.

2. Danh sách lỗi do hệ thống xác định

Error code		Meaning
E01	E01	init.py not found.
E02	E02	An error occurred in init.py.
E03	E03	The robot program is not running.
E04	E04	The robot program has not been established yet.
E05	E05	Mode in which the robot program cannot be executed..
E06	E06	The robot motion API was used before open() of the i611Robot class was executed.
E07	E07	The robot program is executed when the ABS origin is lost.
E08	E08	The robot program terminated abnormally.
E09	E09	The Robot program executes open() in class i611Robot during an emergency stop.
E10	E10	The Robot program executes open() in i611Robot class while the servo is OFF.
E11	E11	The robot program has not been authorized to operate.
E12	E12	The robot program cannot communicate with the system administrator.
E13	E13	There are no exceptions for emergency stops.
E14	E14	Abnormal termination of the Robot program's exit() method.
E15	E15	The robot program ended with an exception.
E16	E16	There are no exceptions to stopping deceleration.
E17	E17	System termination process not completed
E18	E18	Unable to access memory I/O.
E19	E19	Instances of the i611Robot class are created multiple times in a single process.
E20	E20	open() in class i611Robot was opened multiple times in one process.
E21	E21	The inconsistent API call occurred in another thread.
E40	E40	The teaching process ended abnormally..
E53	E53	Usage of home directory (/home/i611usr) has exceeded the limit..
E99	E99	Another error occurred.

3. List of system-defined errors (Serious)

Error code		Meaning
c01	c01	System Manager failed to start.
c02	c02	System administrator terminated abnormally.
c03	c03	The system administrator cannot communicate with the controlling administrator.
c04	c04	An error occurred in JOG operating mode.
c05	c05	Control Manager terminated abnormally.
c06	c06	There is no more storage space left on the controller.
c10	c10	(Matching) circuit is broken.
c11	c11	(Matching) overcurrent arises.
c12	c12	(Matching) A brake error has occurred. (When servo OFF → ON)
c13	c13	(Matching) Excessive torque detected..
c14	c14	(Matching) overload (heat) occurs.
c15	c15	(Matching) Drive voltage has decreased..
c16	c16	(Matching) An AC power abnormality has occurred.
c17	c17	(Matching) A servo communication error has occurred.
c18	c18	(Matching) Servo 1 ON indicator error has occurred. (does not operate normally.)
c19	c19	(Matching) Servo 2 ON indicator error has occurred. (Z, not detected.)
c20	c20	(Matching) ABS loss: Absolute encoder value cannot be detected.
	c21	(Match) ABS loss: An error occurred in the absolute encoder value..
	c22	(Match) ABS loss: Unable to detect the incremental encoder value..
c23	c23	(Matching) ABS loss: Progressive encoder is broken
c24	c24	(Matching) ABS loss: The battery voltage of the progressive encoder has decreased.
c25	c25	(Matching) An error occurred during the state change condition.
c26	c26	Abnormalities arise at Tip I/O.
c28	c28	An error occurred during internal screen processing.
c29	c29	The cooling fan has stopped.

System-defined error（Serious）

Error code		Meaning
c30	c30	Regenerative resistor 1 error has occurred.
c31	c31	Main circuit relay is broken.
c32	c32	Wiring error detected in “emergency stop circuit”.
c33	c33	Wiring error detected in “mode circuit”.
c34	c34	An error has occurred in the control power supply.
c35	c35	Detects thermal anomalies in resistors to prevent intrusion.
c36	c36	Regenerative resistor 2 error has occurred.
c37	c37	A secondary regenerative resistor error has occurred.
c39	c39	The robot's communication has been cut off.
c40	c40	A backup signal mismatch has occurred in the “gate circuit”.
c41	c41	There has been a mismatch in the duplex signal in the “mode circuit”.
c42	c42	An error occurred depending on the state transition time.
c43	c43	A communication error occurred due to an interruption.
c44	c44	Overspeed slave error occurred.
	c58	Error occurred in the SPI circuit.
c59	c59	An error was detected in the robots definition file.
c60	c60	An operation error has occurred.
c89	c89	(Match) An error occurred in the EtherCAT communication packet..
c91	c91	(Match) Abnormal detection of abnormal position and speed deviations.
c92	c92	(Match) Match parameter error detected.
c93	c93	(Match) An encoder communication error has occurred..
c94	c94	(Joint) The control panel is too hot.
c95	c95	(Match) A synchronization error occurred in EtherCAT communication.
c96	c96	(Match) An error occurred during control synchronization.
c98	c98	Power supply is cut off.
c99	c99	Another error occurred.

4. How to handle errors

Type of error	Error handling instructions	
System-defined error E88	The robot will not operate until the error is resolved. Please refer to the following troubleshooting solutions.	
	User program	An exception occurred.
	How to reset	Eliminate the cause of the error and issue an error reset command (cmd_reset()) or I/O input (go to wait state after reset.)
System-defined error c88	The robot will not operate until power is restored. Since the internal operation of the controller continues, data cannot be imported from the outside. If you do not see improvement, contact a service center.	
	User program	Forced exit.
	How to reset	Remove the cause of the error and turn the power back on

E01	init.py not found.	
	Reason	init.py is missing from controller /home/i611usr/.
	Measure	Copy init.py from /opt/i611/tools/ to /home/i611usr/.

E02	An error occurred in init.py.	
	Reason	There is an error in the init.py code.
	Measure	Check init.py.

E03	The robot program is not running.	
	Reason	The robot program is not specified properly.
	Measure	Checks the specified file name. Example) rbs = RobSys() rbs.open() rbs.set_robtask ('filename.py')

E04	The robot program has not been established yet.	
	Reason	Missing specified by rbs.set_robtask('filename.py') or specified a file that does not exist..
	Measure	Check the filename specified in rbs.set_robtask('filename.py').

E05	Become a mode that the Robot program cannot perform.	
	Reason	The jumper connector (accessory) is not connected to CN2 on the controller.
	Measure	Make sure the jumper connector is connected to CN2. The robot program cannot be run if the JOG bar is connected to CN2 or if nothing is connected.

E06	The robot motion API was used before open() of the i611Robot class was executed.	
	Reason	There is no description of the i611Robot class in the robot program. or the i611_MCS module cannot be imported.

	Measure	Robot program
		rb=i611Robot( )
		rb.open( )
		Check to see if it has been described above

E07	Execute the Robot program while losing the ABS base point.	
	Reason	Return of ABS origin does not take place. or the brake is released when the controller is turned off.
	Measure	Proceed to adjust the operating origin of ABS. ・"Recovering ABS base point for operation" in the procedure book ・Arm Moduley User Guide 「Teaching C」

Chương trình robot chấm dứt bất thường.	
Nguyên nhân	Ngoại lệ không mong đợi đã xảy ra trong chương trình robot.
Biện pháp	Kiểm tra chi tiết lỗi trong chương trình robot và sửa lỗi.

E09	The robot program executed open() of class i611Robot during an emergency stop..	
	Reason	The robot program starts by pressing the emergency stop switch.
	Measure	Release the emergency stop switch.

E 10	The robot program executed open() of the i611Robot class while the servo was OFF.	
	Reason	When the servo was turned off, open() of the i611Robot class was executed.
	Measure	Please turn on the servo.

		The robot program is not allowed to operate yet..
E 11	Reason	Duplicate calls or open() with rb=i611Robot( ) rb.open(permission=False) in class i611Robot.
	Measure	Check if rb=i611Robot( ) rb.open(permission=Ture) (← rb.open() is included in the Robot program. Or make sure you don't call multiple open().

		The robot program cannot communicate with the system administrator.
E 12	Reason	An unexpected error occurred.
	Measure	Re-apply power to the controller. If there is no improvement, contact the service center..

		There are no emergency stop exceptions.
E 13	Reason	There is no description of the try statement or the try statement is missing an except description.
	Measure	In the Robot program check to see if it is included try: . . . except Robot_emo: or not .

		Abnormal termination of the Robot program's exit() method.
E 14	Reason	In a robot program, the argument to the exit() method specifies a non-zero value.
	Measure	On normal termination, the exit() method's argument is set to 0. rb=i611Robot() . . . rb.exit (0)

		The robot program ended with an exception.
E 15	Reason	An exception occurred due to a programming error.
	Measure	Check error details and fix the faulty program.

E 16	There are no exceptions for deceleration to stop.	
	Reason	There is no description of the try statement or the try statement is missing an except description.
	Measure	In the Robot program check to see if it is included try: . . . except Robot_stop: or not .

E 17	The system has not completed the shutdown process.	
	Reason	Handle interrupt (Robot_emo) when emergency stop time expires.
	Measure	Set the emergency stop interrupt processing to complete within 5 seconds and terminate the program.

E 18	Unable to access memory I/O.	
	Reason	The I/O connector is not connected properly or the Control Manager is disconnected abnormally.
	Measure	Check the connection status of the I/O connector.

	Thực thể của lớp i611Robot được tạo nhiều lần trong một quy trình.	
	Nguyên nhân	Trong chương trình robot, một số thực thể của lớp i611Robot được mô tả.
	Measure	Check if you are calling multiple instances of the i611Robot class in a single process.

E 20	open() of the i611Robot class is executed multiple times in a single process.	
	Reason	Described some open() of the i611Robot class in the Robot program.
	Measure	Please check if open() of class i611Robot is called multiple times in one process..

E 21	The inconsistent API call occurred in another thread.	
	Reason	Calling a method that has been prohibited from being called in another thread or without creating an entity.
	Measure	Create an entity and call. The API of the i611Robot class can be called from another thread: abort(), stop(), pause() and restart().

E40	The teaching process ended abnormally..	
	Reason	Teaching Manager terminated abnormally.
	Measure	Re-apply power to the controller. If there is no improvement, contact the service center..

E53	Usage of home directory (/home/i611usr) has exceeded the limit..	
	Reason	The controller's home directory is incomplete.
	Measure	Free up space in the folder, move unnecessary files in /home/i611usr to /home/i611usr/ex.

E99	Another error occurred.	
	Reason	An unexpected error occurred.
	Measure	Re-apply power to the controller. If there is no improvement, contact the service center..

MEMO

MEMO

Service center

Zeus: 132 Annyeongnam-ro, Hwaseong-si, Gyeonggi-do

e-mail : zero@globalzeus.com