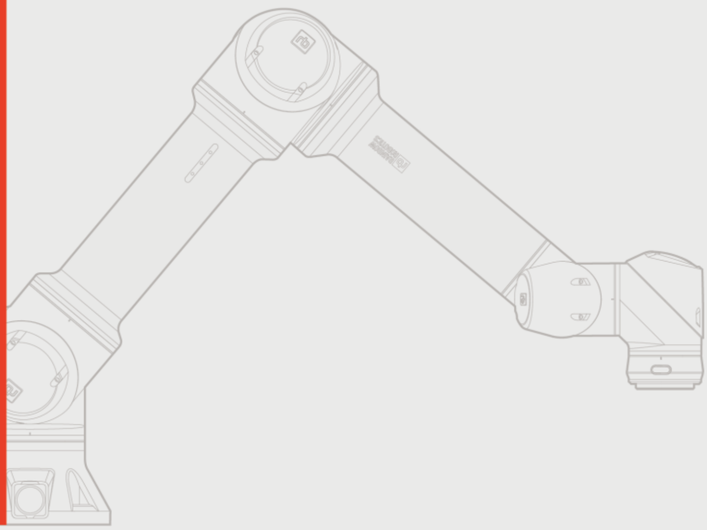

# RB Series

Collaborative Robot

## User Manual
(English)

Ver : v3.9.1



Collaborative Robot

# RB Series
# USER MANUAL

RAINBOW ROBOTICS

UPDATE : 2021 / 09

# INDEX

# PREFACE

Before installing this product, please read the user manual thoroughly. Please follow the instructions in the manual according to the installation procedure. The contents of this manual are based on the version of the manual when it was written, and the information about the product may have been changed without notifying the user in advance. If you are unsure about the requirements, recommendations or safety procedures described in this manual, please contact Rainbow Robotics. Some illustrations in this manual are intended to help you understand the concepts and installation of the system and may differ from actual products.

# CHAPTER 1. INTRODUCTION

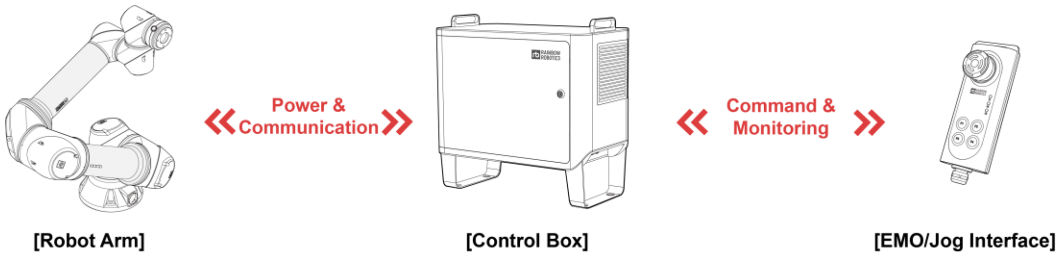
## 1.1 RAINBOW ROBOTICS' COLLABORATIVE ROBOTS

The RB product line from Rainbow Robotics is a series of collaborative robots. The RB series is designed to be easily accessible and usable to anyone. The RB series is specialized to perform regular, continuous, and repetitive tasks in small and dense human environments across various fields without any additional safety devices. The RB series is the robotic solution to increase productivity for your business.

- Intuitive Usability: It is easy to set up and operate an RB robot. Experts and non-experts alike can use it effectively through the intuitive graphical User Interface (UI) configuration.
- Convenience and safety: The RB series has External and self-collision detection systems, which minimize accidents and injuries while providing a safe working environment for the operator.
- Space Efficiency: An RB can be applied to all types of production lines, regardless of space. Users may use it in many different environments due to versatile orientations that allow it to be installed on a variety of surfaces.

## 1.2 System Configuration

The system configuration of an RB is illustrated in the figure below.

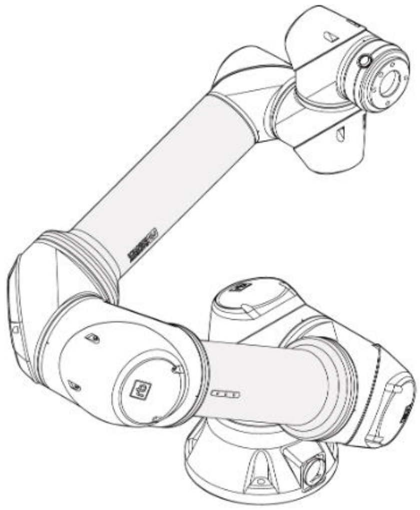



< Stand-type control box system configuration >

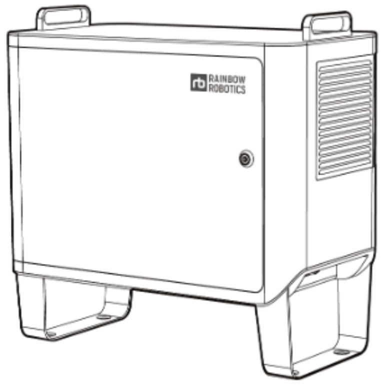
- Robot Arm: The Robot Arm is an industrial collaborative robot that can be used for the repetition of simple tasks, carrying small objects, or assembling parts. It can be used with various third-party robotic grippers and as well as various types of tools.
- Control Box: The Control Box controls the movement of the robot arm according to user programs contained on the Teaching Pendant/Tablet PC. Digital and analog input/output ports are available for connecting various devices and tools.
- Estop/Jog Interface(For stand-type control box): With the emergency stop switch, the robot operation can be stopped. It comes with simple program flow control buttons such as Play/Stop.
- Teaching Pendant/Tablet PC (optional): The Teaching Pendant/Tablet PC is an external device on which a user can create programs and operate the system. It is used to setup, program, and send commands to the robot arm.

The RB contains system components and accessories as described below.

- Stand-type control box components

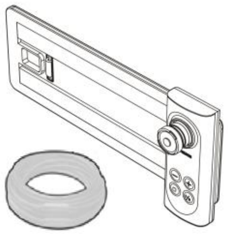


<Robot Arm>





<Control Box>



<Interface / Cables>



<User Manual>

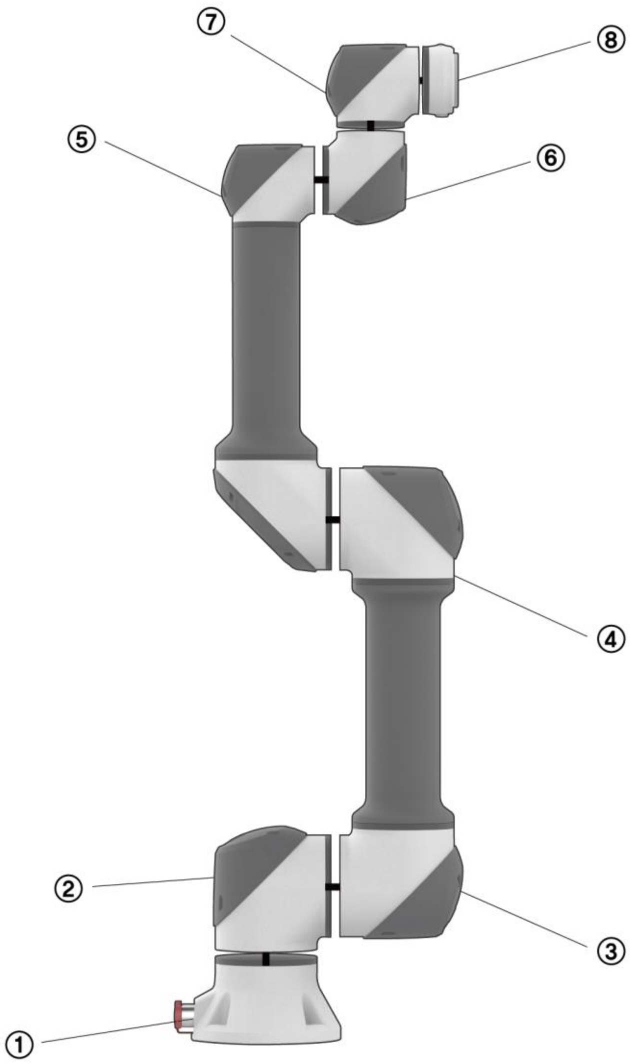
<Tablet PC>

| | | |
|---|---|---|
| Robot | Robot Arm | 1 EA |
| System Components | Control Box | 1 EA |
| | Estop/Jog Interface | 1 EA |
| | Power cable | 1 EA |
| | Cable between Robot Arm-Control Box | 1 EA |
| | Optional: Tablet PC, Tablet cover, USB cable | 1 EA |
| Other | User Manual(Electronic Document) | 1 EA |

※ Please use the 5 meter Power cable, Estop/Jog interface cable and RobotArm-ControlBox connection cable provided by the manufacturer. For the user LAN shield cables/IO cables/USB cables/external extension cable for the electric line passing models, less than 3 meter is recommended.

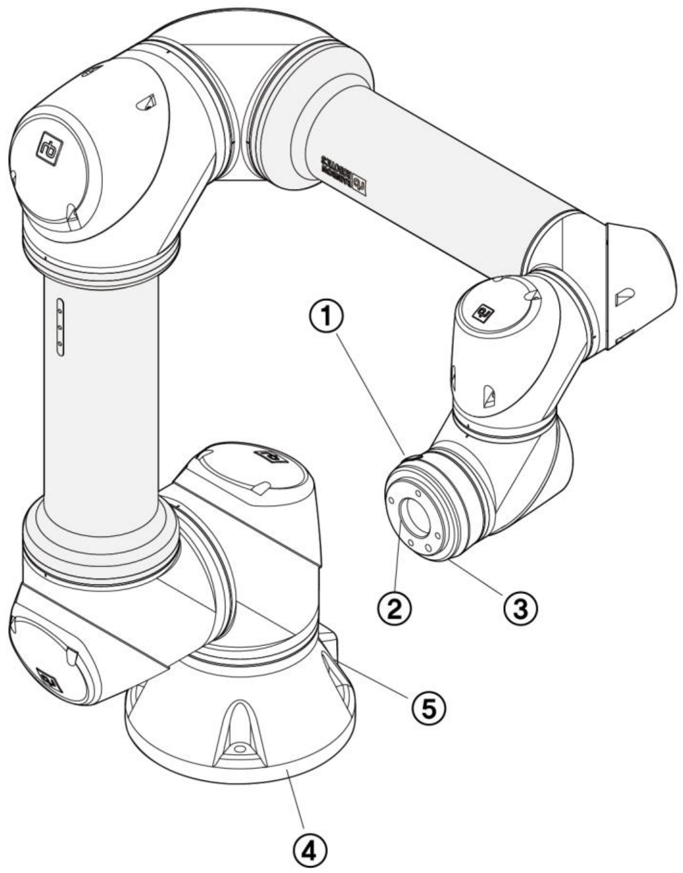
## 1.3 Robot Arm

- RB Series Joint Description



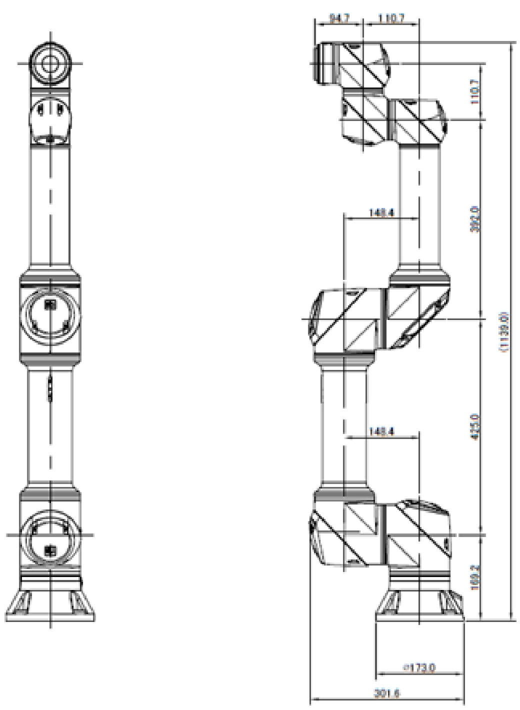

| No. | Name |
|---|---|
| ① | Base |
| ② | Base Joint |
| ③ | Shoulder Joint |
| ④ | Elbow Joint |
| ⑤ | Wrist 1 Joint |
| ⑥ | Wrist 2 Joint |
| ⑦ | Wrist 3 Joint |
| ⑧ | Tool Flange |

■ RB Series Component Description

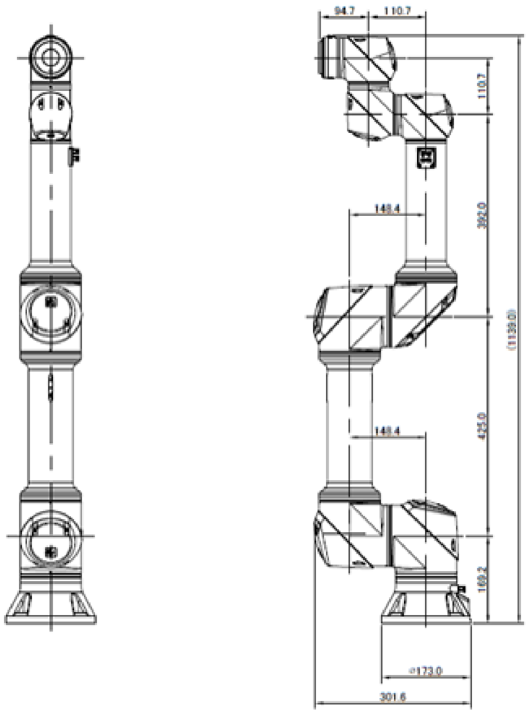



| No. | Name | Description |
|---|---|---|
| ① | Teaching Button | Button for direct-teaching |
| ② | Tool Flange | Part for mounting tool or gripper |
| ③ | Tool I/O | I/O ports to control tool or gripper |
| ④ | Base | Part for mounting the robot arm |
| ⑤ | Connector | Connector for robot-arm cable |

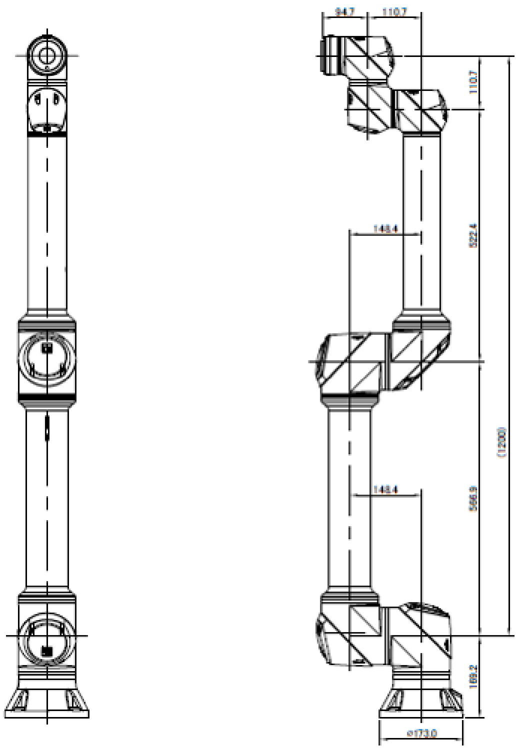
■ Dimensions



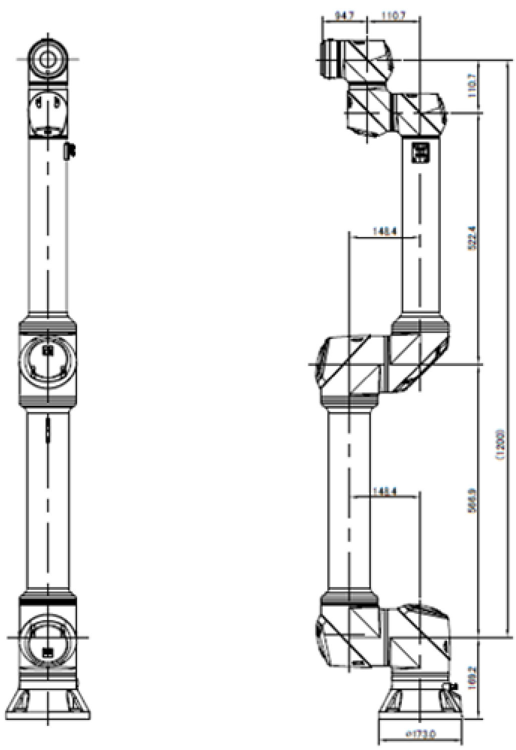

[RB5-850E]



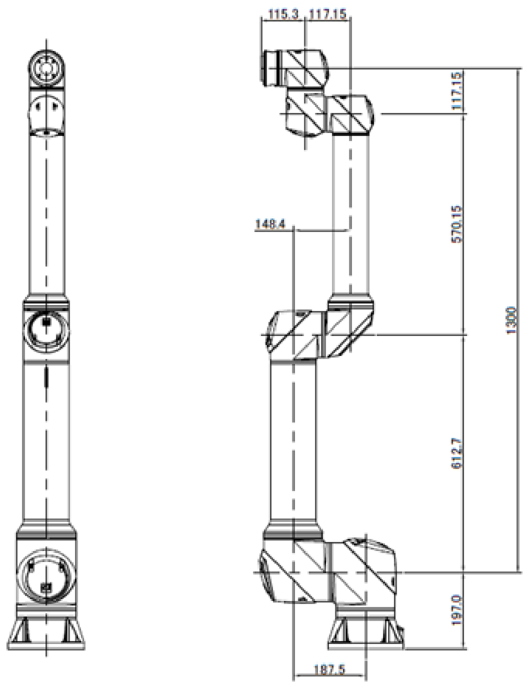

[RB5-850EA1/A2]



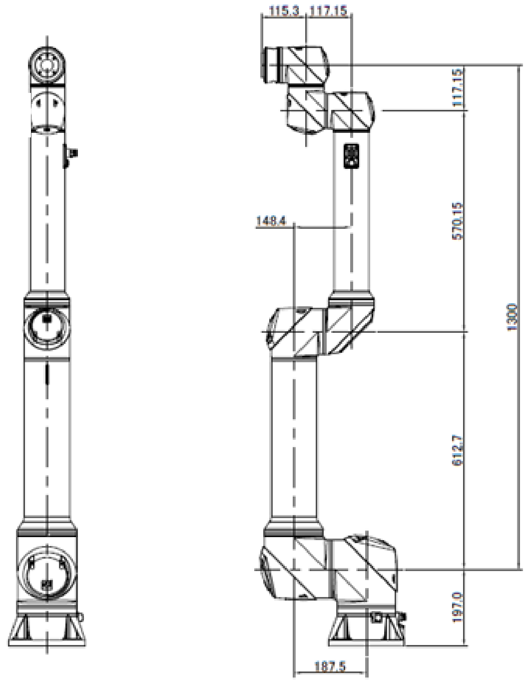

[RB3-1200E]




[RB3-1200EA1/A2]

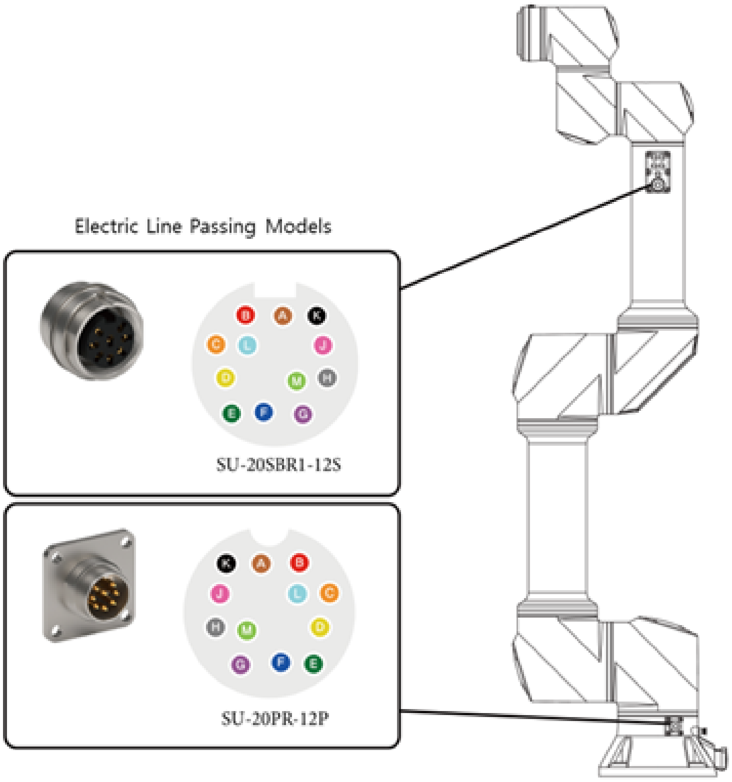



[RB10-1300E]

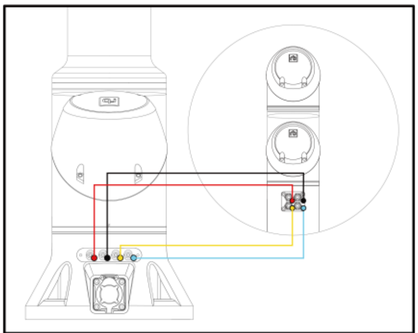



[RB10-1300EA1/A2]

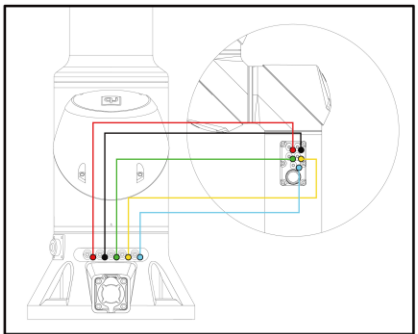
■ Configuration of the embedded Pneumatic-tubing/Electric-line in RB5-850EA#, RB3-1200EA#, and RB10-1300EA#.



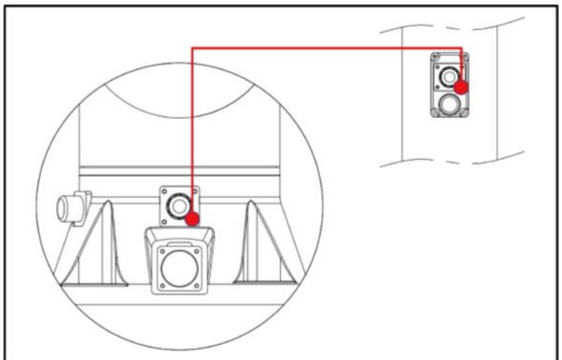

[Embedded pneumatic tubing connection]



[RB5-850EA1, RB3-1200EA1]



[RB5-850EA2, RB3-1200EA2, RB10-1300EA2]



[RB10-1300EA1]

※ RB5-850EA#, RB3-1200EA#, RB10-1300EA# model's pneumatic lines and wire lines are provided as shown in the following table, please refer to the picture above for use.

| Model Name | Pneumatic Lines | Wire Lines |
|---|---|---|
| RB5-850EA1 | 4ea(4Ø Pneumatic Tube) | None |
| RB5-850EA2 | 5ea(4Ø Pneumatic Tube) | 12 Pin(AWG28) |
| RB3-1200EA1 | 4ea(4Ø Pneumatic Tube) | None |
| RB3-1200EA2 | 5ea(4Ø Pneumatic Tube) | 12 Pin(AWG28) |
| RB10-1300EA1 | 1ea(8Ø Pneumatic Tube) | 12 Pin(AWG28) |
| RB10-1300EA2 | 5ea(4Ø Pneumatic Tube) | 12 Pin(AWG28) |

## Warning

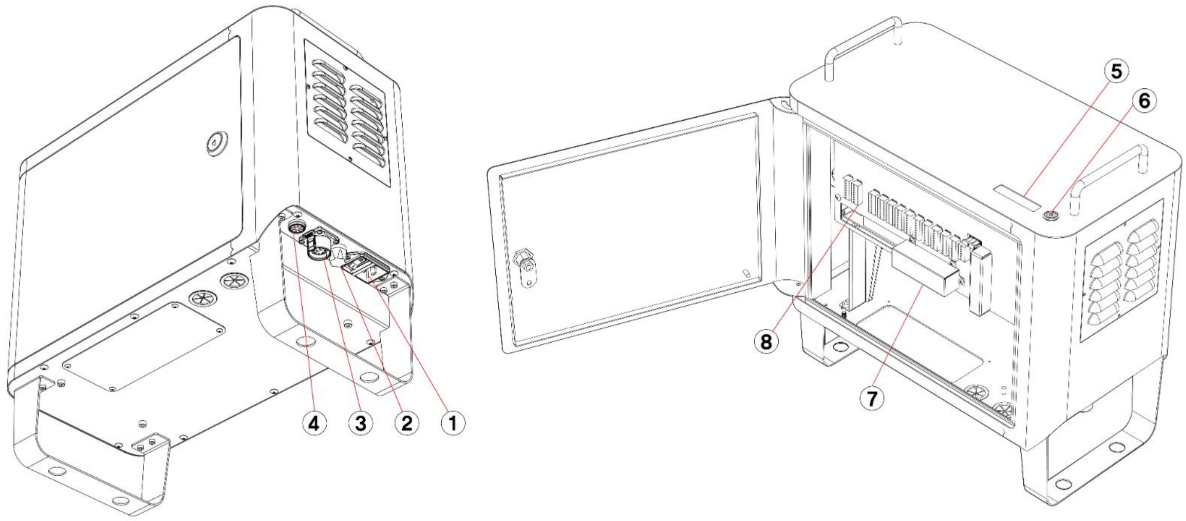
Warning:

1) In case of pneumatic / electric wire passing models, if passing air or power over the defined standard, hardware may be damaged.

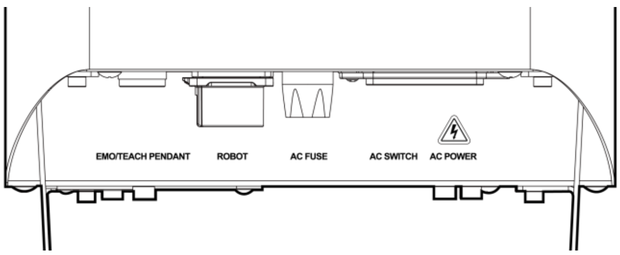
## 1.4 Control Box

The front and lower/inner sections of the control box are illustrated in the figure shown below.

- Stand-type control box




<Bottom section> <Top section>



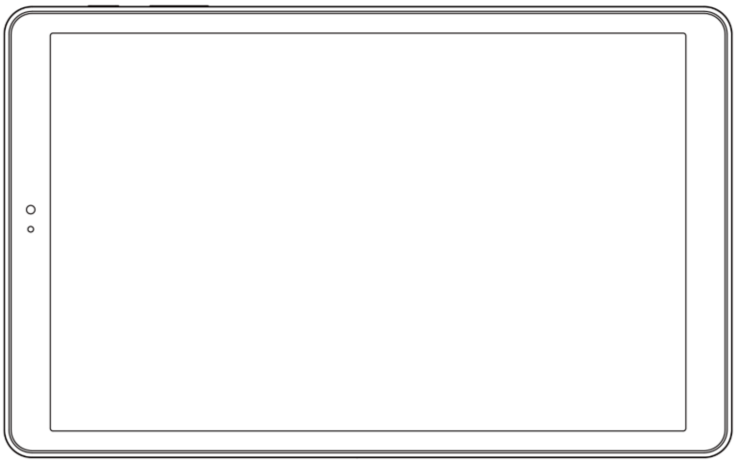

<Main connector/socket>

| No. | Name |
|---|---|
| ① | AC Power Socket (AC POWER) |
| ② | AC Power Switch (AC SWITCH) |
| ③ | Connector for Robot Arm cable (ROBOT) |
| ④ | Connector for TEACH PENDANT<br>(EMO/TEACH PENDANT) |
| ⑤ | LCD |
| ⑥ | Main Switch |
| ⑦ | USB/LAN connectors |
| ⑧ | I/O ports |

## 1.5 TEACHING PENDANT TABLET PC (OPTIONAL)

The teaching pendant/tablet PC is an optional accessory. It MUST be purchased separately.

- For stand-type control box

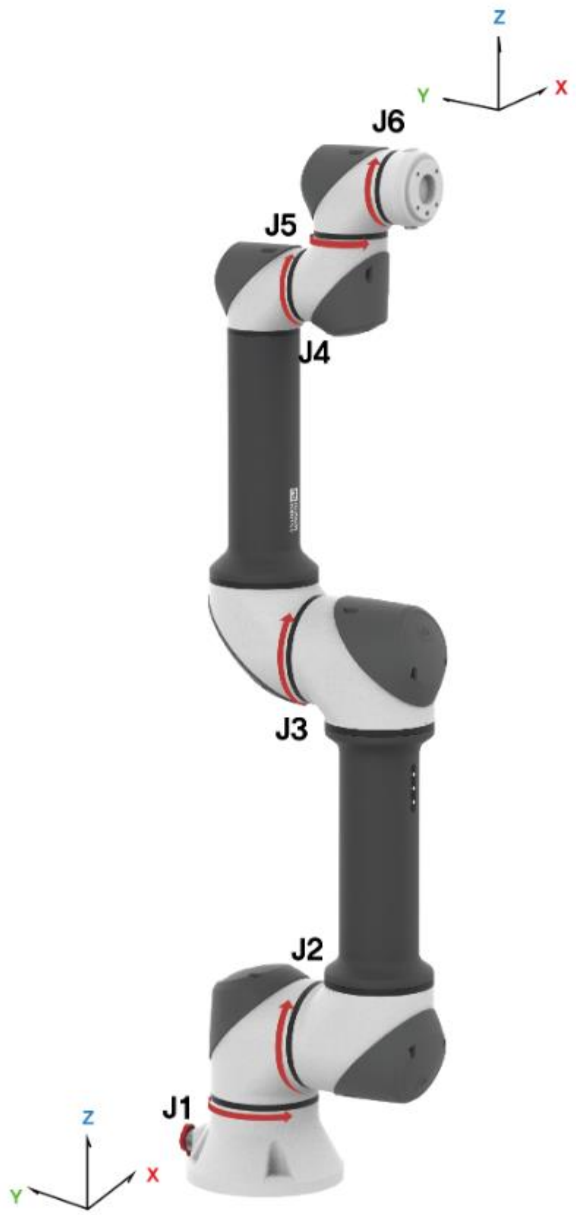



※ Purchasing the tablet PC is not required. The App to operate the RB series can be downloaded from Rainbow Robotics' website.

※ Tablet setup is required for use with RB products. See the Appendix.

## 1.6 Joint Limit

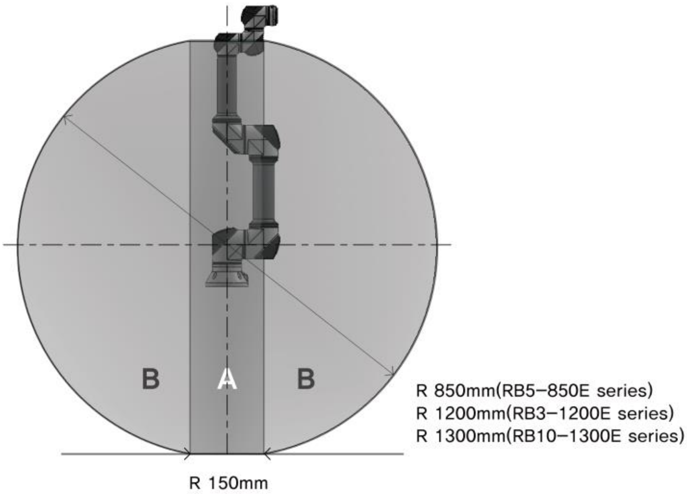
An RB robot consists of six joints. The axes of rotation and joint limits are illustrated in the following section.




| Joint | Range |
|---|---|
| J 1 | ± 360 ˚ |
| J 2 | ± 360 ˚ |
| J 3 | ± 165 ˚ |
| J 4 | ± 360 ˚ |
| J 5 | ± 360 ˚ |
| J 6 | ± 360 ˚ |

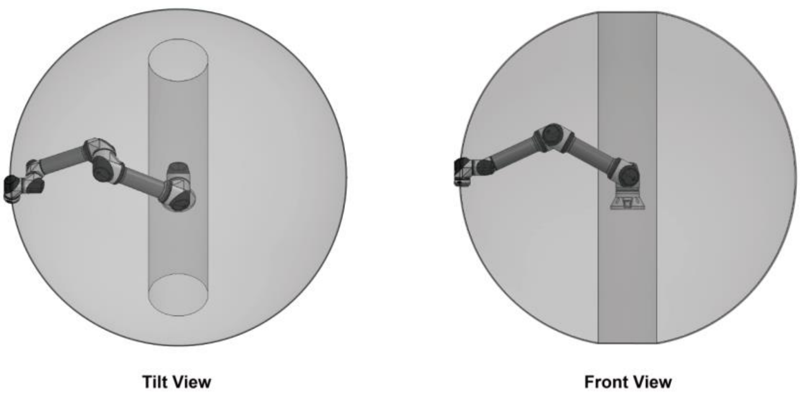
## 1.7 WORKSPACE

The maximum radius is an 850mm workspace for the RB5-850E Series. The maximum radius is 1300mm For the RB10-1300E Series and is 1200mm for the RB3-1200E Series. The area A in the figure below represents the kinematic singular area. This means that any motion programmed in the inertial coordinate system (e.g. programming a motion using the Move L command) may not work properly. The robot may stop itself or move faster than designated. Programing the motion in the joint coordinate system (e.g. Move J) is recommended in this particular area.



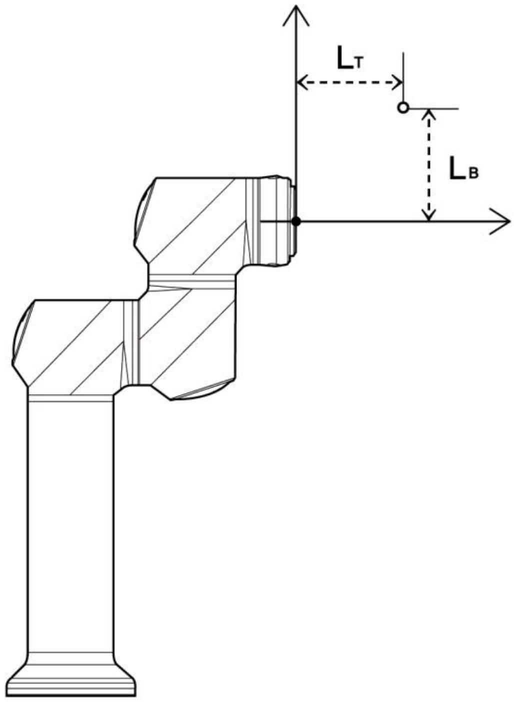

A. The kinematic singular area. Limits some motion programmed in the inertial coordinate system (e.g. Move L).

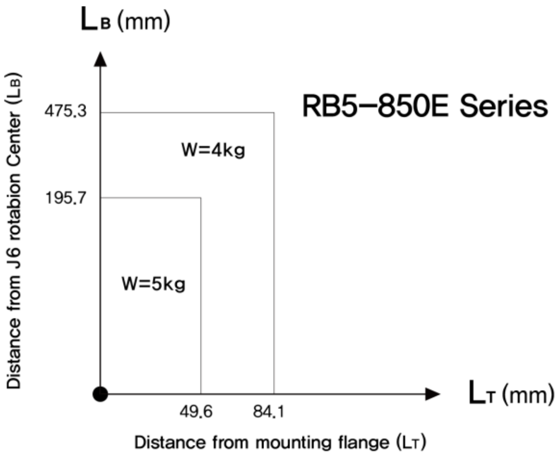
B. Workspace of the RB.

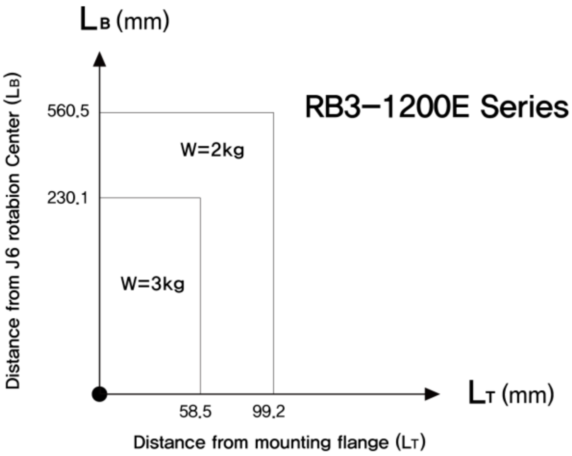


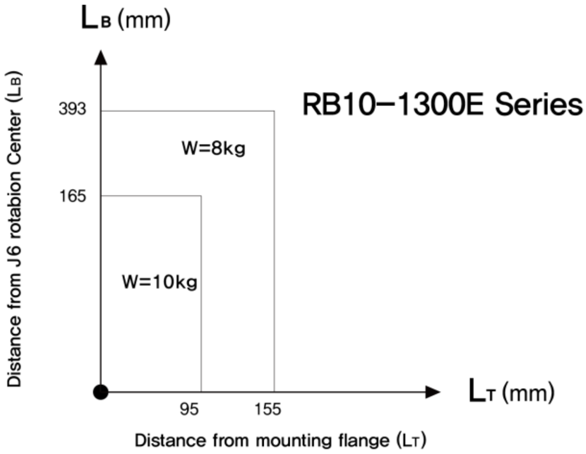
## 1.8 MAXIMUM LOAD CAPACITY

The maximum payload of the robot arm depends on the distance between the tool flange and the center of mass of the payload. The maximum payload according to this distance is as follows.



# CHAPTER 2. SAFETY & PRECAUTIONS

## 2.1 SAFETY INDICATIONS

The following safety notices are used in this manual.

### Danger

Danger :

Failure to follow instructions marked with this symbol may result in severe harm, which could result in serious injury or death.

### Warning

Warning :

Failure to follow the instructions with this symbol may result in an accident, which could result in serious injury to the user.

### Caution

Caution :

Failure to follow directions marked with this symbol may result in damage to the product or injury to users.

## 2.2 GENERAL SAFETY WARNING & PRECAUTIONS

This section contains general hazards, warnings, and cautions that will be repeated or further explained elsewhere in this manual.

**Danger**

Danger:

1) Robots and electrical equipment must be installed in accordance with the instructions from **Chapter 4, Installation.**

**Warning**

Warning:

1) Robot users and robot application system manufacturers should be familiar with the contents of this manual. In addition, they should complete operational training.
2) Please ensure enough space is provided for the robot arm to move freely
3) When using the robot, do not wear loose clothes or jewelry. Long hair should be tied so that it does not get caught in the joints of the robot.
4) Never operate a broken or a faulty robot.
5) If a fatal error occurs in the software, immediately hit the emergency switch to stop the robot, then contact your supplier or Rainbow Robotics.
6) Check that the robot installation angle, tool setting, safety setting, etc. are entered correctly.
7) Please do not connect safety equipment to the general use I / O ports in the back of the control box. Safety equipment should only be used with safety related I / O ports.
8) Please be careful about the movement of the robot when using the pendant for teaching.
9) During the operation of the robot, please do not enter the operating range of the robot. In addition, please do not touch the robot while it is operating.

10) Please do not modify the robot without the support of Rainbow Robotics. Rainbow Robotics (hereinafter "the manufacturer") assumes no responsibility for any problems caused by user's modification or modification of the product.

11) Both the robot arm and the control box generate heat when used for a long time. Do not touch the robot after long use. If the user needs to touch the robot, please turn off the controller and allow the robot to cool down before touching.

12) When the robot collides with an external object, a considerable amount of kinetic energy is generated. This kinetic energy is proportional to the speed of the robot and the payload.

13) Please confirm that you are using the recommended installation settings for the robot. The teaching or collision detection functions may not work properly if the robot arm's mounting orientation, tool weight, tool center of gravity, length, safety configuration, etc. are not entered correctly.

14) The teaching function should only be used in a safe environment. Do not use this function when there are hazards nearby.

15) Before using the teaching function, input the relevant information (tool length, weight, center of gravity, etc.) accurately. Not entering the relevant specifications will cause malfunctions when using the direct teaching function.

16) If the robot joints move at an unsafe speed when using the direct teaching function, the user can force the robot to stop with the emergency switch for their safety.

## Warning

Warning:

1) Robotic arm and control box generate heat during operation. Do not touch the robot arm during operation or immediately after operation as continuous contact with the robot arm may cause it to malfunction. Before manipulating or touching the robot arm, please check the temperature on the UI screen or turn off the robot arm. Please wait at least 1 hour to cool it down before touching.

## Caution

Caution:

1) When using with a machine or another robot that can damage the robot arm, it is recommended to test all functions separately before use. The manufacturer is not responsible for any programming errors, damage to the RB, or damage to other machines due to robot malfunctions.

2) Do not expose the robot to strong magnetic fields as the robot may be damaged.

Warning:

1) Attach a warning label to the location where there is a danger of electric shock from the electric device.

2) Do not tear, damage, or remove the cover. Be careful when handling parts or devices with a label attached, as well as surrounding components.

3) To avoid electric shock, do not touch the internal electric parts.

## 2.3 Usage & Functionality

The robot arm is intended to be used for transferring and assembling objects by utilizing tools and should only be operated in the environment specified in the description. It is possible to work without a physical protective barrier. However, a safety mechanism should be used after performing the risk evaluation for the whole system. The use of the robot in any of the following applications and environments is considered improper use, and the manufacturer is not liable for any direct or indirect damage to the robot.

- Use in any potentially explosive environment
- Medical and life related uses
- Human and animal transport
- Any use without risk assessment
- Any use in places where the performance of the safety function is insufficient
- Any use beyond performance / environmental specifications

※ Not limited to above items.

## 2.4 Potential Safety Issues

Additional protective measures must be taken if the final system is deemed unsafe or unable to adequately reduce risk. Users should consider the following potential risks:

- Injury (stenosis), which may occur when a finger is caught between the gears, etc.,
- Injury (stabbing, penetrating) by sharp edges or edges of the tool.
- Injury (stabbing, penetrating, falling) caused by objects located near the robot.
- Injury that can occur when working with toxic and harmful substances (skin damage, dyspnea)
- Injury caused by collision with the robot (stump, fracture)
- Injury that may occur due to not fully fastening objects
- Injury from an object that has detached or dropped from the tool mount

※ Potential risks that may occur depending on the final system may be different.

## 2.5 Liability Limitations

This manual does not cover all peripherals that affect safety. The system installer must comply with safety requirements in accordance with national safety regulations and the laws of the country where the robot arm is installed. The robotic arm consists of an end-coupled system of peripherals. This manual also does not cover all peripherals, including the design, installation, operation and safety of the final system. The final system to which the robot arm is applied must be designed and installed to meet safety requirements in accordance with the regulations and laws of the country where the system is installed.

The operator or the installer of the final system containing the robot arm is responsible for:

- Risk assessment of the final system
- Risk assessment of whether to add additional safeguards
- Ensuring that the system is properly designed, configured, and installed
- Definition of usage for the system
- Identification of important markings and contacts for use and safety
- Providing technical documents, such as manuals

※ Not limited to the above items. Complying with the safety instructions in this manual does not imply that you can avoid all risks that may occur.

## 2.6 Shipping & Transportation

At least two people are required for transportation. Any damages to the robot incurred during shipment or transportation are excluded from the warranty.

**Warning**

Warning:

1) Be careful not to damage the product during transportation. Damages incurred during transportation will void the warranty.

2) When transporting the robot arm, strong vibration or shock may damage the system. The robot must be transported using the packaging box provided by the manufacturer.

## 2.7 EMERGENCY STOP

The emergency stop button can be used to forcefully stop the robot arm if there is an emergency. By pressing the emergency stop button, the user will stop commands sent from the robot control box and terminate any motion.

The section below describes how the emergency stop button for stand-type control box works.

- Emergency Stop

  Users can stop the robot arm immediately by pressing the EMERGENCY STOP button.


- Re-Activating from Emergency Stop

  Turning the EMERGENCY STOP button in clockwise direction.

## 2.8 USER SAFETY

For the user's safety, please note the following:

- Powerless robot operation

In case of an emergency, or in any situation without power, the user can move the robot arm by forcing the joints into a different configuration (forced back driving). To perform forced back driving, the user must push or pull the robot arm firmly. Please ensure that the robot is not powered-on while performing forced back driving.

**Caution**

Caution:

1) If excessive force is applied to the joints in the non-powered state, please be aware that the driving part may be overloaded. The manufacturer is not responsible for any damage caused by excessive force.

## 2.9 SAFETY CONTROLLER

The Safety Control System of Rainbow Robotics' RB5-850E Series, RB3-1200E Series, RB10-1300E Series follows ISO 13849-1 Cat3. PLd.

## 2.10 Risk Assessment

A risk assessment is important when creating a system that uses robots, including the RB. The safety factors to be considered when operating the robot depend on the configuration and integration of the robot arm into the whole system. As such, the robot alone cannot be used for risk assessment.

Please refer to the guidelines of ISO 12100 and ISO 10218-2, as well as the technical specifications of ISO / TS 15066 in order to carry out the risk assessment of robots.

A risk assessment must be performed immediately after robot arm installation. This assessment is to determine and configure safety settings. Determining the need for additional emergency stop buttons, as well as adding protective measures for the surrounding environment, are the key tasks of risk assessment.

The safety-related functions of the collaborative robot can be configured in the safety configuration menu. The menu provides the following functions.

- I / O settings: The control box can be set to output safety information through the output terminal.
- Speed control: Allows the user to control the moving speed of the robot arm.
- Collision Detection Sensitivity Adjustment: When the robot collides with a nearby object, it will stop. However, the user can control the sensitivity at which the arm detects the collision.

If the above safety-related functions do not sufficiently reduce the risk, or if any risks cannot be eliminated, please add an additional safeguard to eliminate the risks. The manufacturer is not responsible for accidents caused by risks that do not comply with the relevant requirements provided by international standards, risks that do not comply with the relevant requirements provided by national laws and regulations, as well as any risk that is not reviewed in the above risk assessment.

# CHAPTER 3. SAFETY FUNCTIONS

## 3.1 INTRODUCTION

This chapter contains important safety information, which must be read and understood by the integrator of the RB Series collaborative robots before the robot is powered on for the first time.

RB Series can protect users and devices by providing various safety functions and safety device interfaces. Safety functions and interfaces meet Category 3, Performance Level d (PL d) as described in ISO 13849-1 and Hardware Fault Tolerance 1, Safety Integrity Level 2 (SIL 2) as described in IEC 62061.

**Caution**

Caution:

1) Depending on the case of the robot installation, the system integrator must perform a risk assessment, and accordingly, the workspace must be configured using safety monitoring functions and interfaces.

2) If a fault is found in the robot's safety function or interface, Stop Category 0 is initiated.

3) Examples of defects include broken cables in the emergency stop circuit, incorrect wiring of additional safety devices, and non-overlapping wiring of additional safety devices (refer to Section 5 of this chapter).

4) System integrators and operators must be aware that there is a safety monitoring function that the robot performs internally, and the safety detection function is not only the operation of the emergency stop switch, the operation of the protective stop device, but also the position of the robot arm during task execution. The robot can be stopped in the designated stop mode even for movements above the physical limit, such as speed, momentum, etc. (See Section 3 of this chapter for the safety monitoring function).

5) System integrators and operators should consider the time and stopping distance between the robot stopping due to the operation of the error and safety monitoring functions described above. The system integrator must conduct a risk assessment considering the stopping distance and time (see Section 4 of this chapter).

6) System integrators and operators are aware of the fact that there is a safety monitoring function to limit the movement of the robot's joints and the robot/TCP, and must select the range of motion of the robot. TCP refers to the position to which the offset is added from the center point of the end of the robot arm.

## Danger

Danger :

1) The system integrator must conduct a risk assessment before applying power to the robot, and if it is used differently from that determined by the risk assessment or if different parameters are used, a risk that is not sufficiently reduced may occur.
2) When connecting additional safety devices, the power of both the robot and the control panel must be cut off.
3) When installing an additional safety device, measures must be taken to ensure that there is no problem when using it mechanically. For example, when using a light curtain, it must be firmly fixed to the floor/fixture, and movement and vibration must not occur during robot operation.
4) All safety function interfaces are set to 24V. Be careful when connecting devices with different voltages as it may cause equipment damage and fire.
5) The signal from the device mounted on the Tool Flange is not included in the safety function. Do not connect the safety device to the Tool Flange cable.

## 3.2 Stop Category

The safety function allows the robot to initiate three types of stop categories defined according to IEC 60204-1.

| Stop Category | Description |
|---|---|
| 0<br>[STO] | Immediately, the robot is turned off and stopped.<br>* Joint brake wear may occur, which may shorten the life of the robot. Do not use it unless it is unavoidable.<br>**Because the power of the robot is cut off, it is necessary to activate it when using it again after removing the danger. |
| 1<br>[SS1] | All joints of the robot are decelerated to the maximum and stopped, and then the power is cut off to stop.<br>*Since the power of the robot is cut off, it is necessary to activate it when using it again after removing the danger. |
| 2<br>[SS2] | All the joints of the robot are decelerated to the maximum, stop, and then enter the SOS state.<br>*SOS: Maintains the current position while the robot is powered on and activated, and starts Stop Category 0 when a position change is detected.<br>** Since the power is not cut off, it can be used immediately after removing the danger. |

**Caution**

Caution:

1) In accordance with ISO10218-1 5.5.2 and 5.5.3, a suitable stop category for emergency stop and protective stop should be selected.
2) In case of emergency stop, you must select from stop category 0 or 1.
3) In case of emergency stop, activation is required.
4) In case of protection stop, at least one must be selected from stop categories 0 and 1.
5) For additional protection stops, stop category 2 can be used.

## 3.3 FUNCTIONAL SAFETY

The manufacturer recommends the following conditions are met for the installation location. The safety functions of the collaborative robot RB Series are used to reduce the risk of the robot system determined by risk assessment.

The parameters of the safety function are set at the factory, and the system integrator can change some items according to the risk assessment. Position and speed items are defined based on the base of the robot.

The following is the safety function specifications provided by RB Series.

| | Safety Function | PL & Category |
|---|---|---|
| **Safety stopping functions** | SF.1 STO (Safe Torque Off) | PL d, Category 3 |
| | SF.2 SS1 (Safe Stop 1) | PL d, Category 3 |
| | SF.3 SS2 (Safe Stop 2) | PL d, Category 3 |
| **Safety monitoring functions** | SF.4 SOS (Safe Operating Stop) | PL d, Category 3 |
| | SF.5 SLP (Safely-Limited Position) | PL d, Category 3 |
| | SF.6 SLS (Safely-Limited Speed) | PL d, Category 3 |
| | SF.7 SLA (Safely-Limited Acceleration) | PL d, Category 3 |
| | SF.8 SLI (Safely-Limited Increment) | PL d, Category 3 |
| | SF.9 SLT (Safely-Limited Torque) | PL d, Category 3 |
| | SF.10 RPL (Robot Position Limit) | PL d, Category 3 |
| | SF.11 TSL (TCP Speed Limit) | PL d, Category 3 |
| | SF.12 CBPL (Control Box Power Limit) | PL d, Category 3 |
| **Emergency stop** | SF.13 EMS1 (Emergency Stop1) | PL d, Category 3 |
| | SF.14 EMS2 (Emergency Stop2) | PL d, Category 3 |
| **Protective stop** | SF.15 PRS (Protective Stop) | PL d, Category 3 |
| | SF.16 HSS (Hard Safeguard Stop) | PL d, Category 3 |
| | SF.17 SSS (Soft Safeguard Stop) | PL d, Category 3 |

- STO(Safe Torque Off): This function prevents force-producing power from being provided to the motor. Power, that can cause rotation, is not applied to the motor. This safety sub-function corresponds to an uncontrolled stop in accordance with stop category 0 of IEC 60204-1.

- SS1(Safe Stop 1): This function is specified as either a) SS1-d (Safe-Stop 1 deceleration controlled) initiates and controls the motor deceleration rate within selected limits to stop the motor and performs the STO function when the motor speed is below a specified limit; or b) SS1-r (Safe-Stop 1 ramp monitored) initiates and monitors the motor deceleration rate within selected limits to stop the motor and performs the STO function when the motor speed is below a specified limit; or c) SS1-t (Safe Stop 1 time controlled) initiates the motor deceleration and performs the STO function after an application specific time delay. This safety sub-function corresponds to a controlled stop in accordance with stop category 1 of IEC 60204-1. Above three candidates, our system uses SS1-t.

- SS2(Safe Stop 2): This function is specified as either a) SS2-d (Safe Stop 2 deceleration controlled) initiates and controls the motor deceleration rate within selected limits to stop the motor and performs the safe operating stop function when the motor speed is below a specified limit; or b) SS2-r (Safe Stop 2 ramp monitored) initiates and monitors the motor deceleration rate within selected limits to stop the motor and performs the safe operating stop function when the motor speed is below a specified limit; or c) SS2-t (Safe Stop 2 time con-trolled) initiates the motor deceleration and performs the safe operating stop function after an application specific time delay. This safety sub-function SS2 corresponds to a controlled stop in accordance with stop category 2 of IEC 60204-1. Above three candidates, our system uses SS2-t.

- SOS(Safe Operating Stop): This function prevents the motor from deviating more than a defined amount from the stopped position. The PDS (SR) provides energy to the motor to enable it to resist external forces. This description of an operational stop function is based on implementation by means of a PDS (SR) without external (for example mechanical) brakes.

- SLP(Safely-Limited Position): This function prevents the motor shaft (or mover, when a linear motor is used) from exceeding the specified position limit(s).

- SLS(Safely-Limited Speed): This function prevents the motor from exceeding the specified speed limit.

- SLA(Safely-Limited Acceleration): This function prevents the motor from exceeding the specified acceleration and/or deceleration limit.

- SLI(Safely-Limited Increment): This function prevents the motor shaft from exceeding the specified limit of position increment within specified time.

- SLT(Safely-Limited Torque): This function prevents the motor from exceeding the specified torque (or force, when a linear motor is used) limit.

- RPL(Robot Position Limit): This function prevents the robot arm's TCP (tool center point) or body frame exceeding the specified spatial region.

- TSL(TCP Speed Limit): This function prevents the robot arm's TCP speed exceeding the specified speed.

- CBPL(Control Box Power Limit): This function prevents the Control Box's power consumption exceeding the specified limit.

- EMS1(Emergency Stop1): This function activates the stop mode when the emergency stop switch of the Teaching Pendant Unit is activated. The stop mode is the SF.2.

- EMS2(Emergency Stop2): This function activates the stop mode when the special I/O ports of the Control Box are activated. Those ports are provided for users to connect their own switch devices. The stop mode is the SF.2.

- PRS(Protective Stop): This function activates the stop mode when the special I/O ports of the Control Box are activated. Those ports are provided for users to connect their own protective devices. The stop mode is the SF.2.

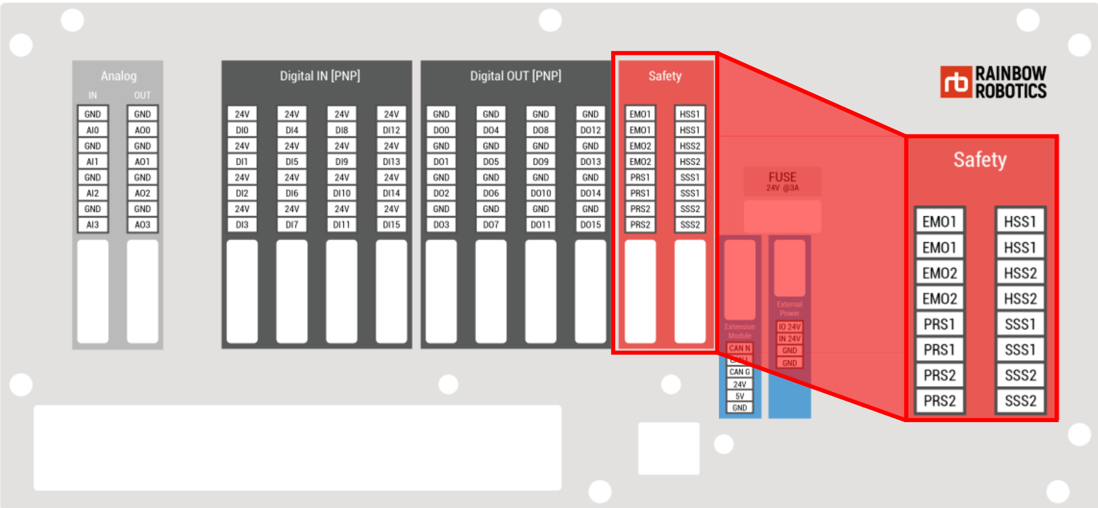
- HSS(Hard Safeguard Stop): This function activates the stop mode when the special I/O ports of the Control Box are activated. Those ports are provided for users to connect their own protective devices. The stop mode is the SF.1.

- SSS(Soft Safeguard Stop): This function activates the stop mode when the special I/O ports of the Control Box are activated. Those ports are provided for users to connect their own protective devices. The stop mode is SF.3.

## 3.4 Safety Device Mounting Location

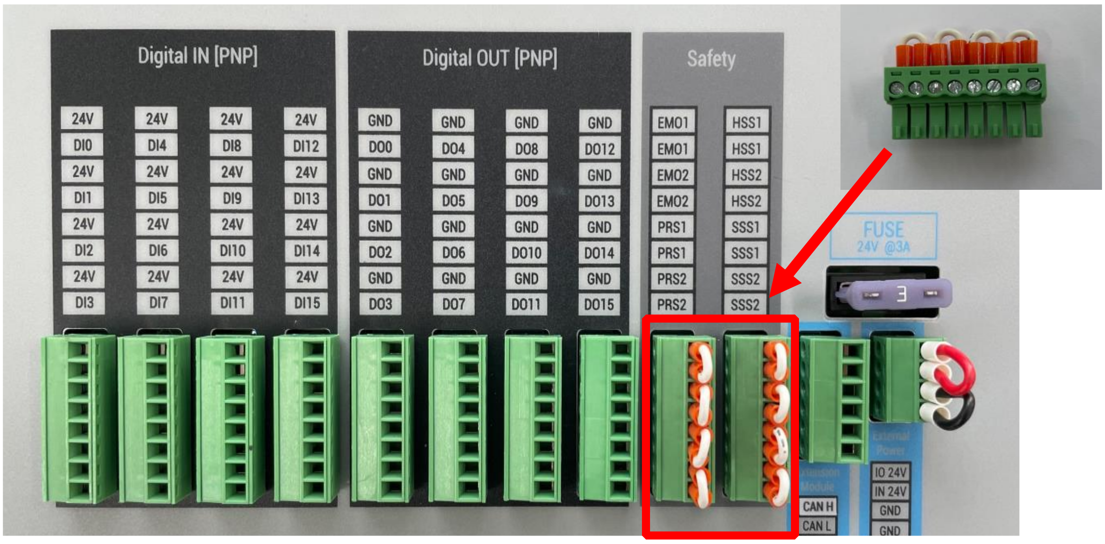
In addition to the basic emergency stop switch, the RB Series can be equipped with additional safety devices required by the system integrator through risk assessment.

The safety-dedicated contact terminal consists of 16 ports. This terminal is a redundant dedicated contact input terminal.




The additional ports can be equipped with 4 equipment. The robot is delivered with a default configuration, which enables operation without any additional safety equipment.

When using without connecting an external safety device, connect and use the basic contact input as shown below.



Safety device port specifications are as follows.

- EMO

This port is used when it is necessary to install an extra emergency stop switch through risk assessment.

The emergency stop switch should be used as a product conforming to IEC 60947-5-5.

Emergency stop generated through EMO is designated as stop category 1.

- PRS

This port is used to connect one or more protective stop devices through risk assessment.

Protective stop devices must be used in accordance with 5.3.8.3 of ISO 10218-2.

Protection stops occurring through PRS are designated as stop category 1.

- HSS

This port is used to connect one or more protective stop devices through risk assessment.

Protective stop devices must be used in accordance with 5.3.8.3 of ISO 10218-2.

Protection stops occurring through HSS are designated as stop category 0.

- SSS

This port is used to connect one or more protective stop devices through risk assessment.

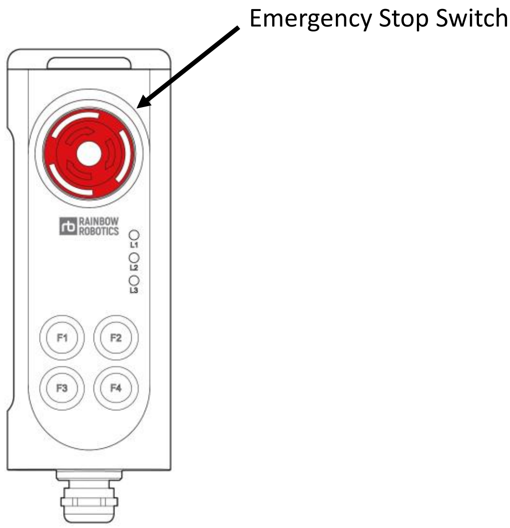
Protective stop devices must be used in accordance with 5.3.8.3 of ISO 10218-2.

Protective stops that occur through SSS are designated as stop category 2.

## 3.5 Emergency Stop Switch

The collaborative robot RB Series allows the operator to use the emergency stop switch to stop the robot in preparation for an emergency situation.

In the event of an emergency, the robot must be stopped immediately by pressing the emergency stop switch on the top of the pendant.




### Caution

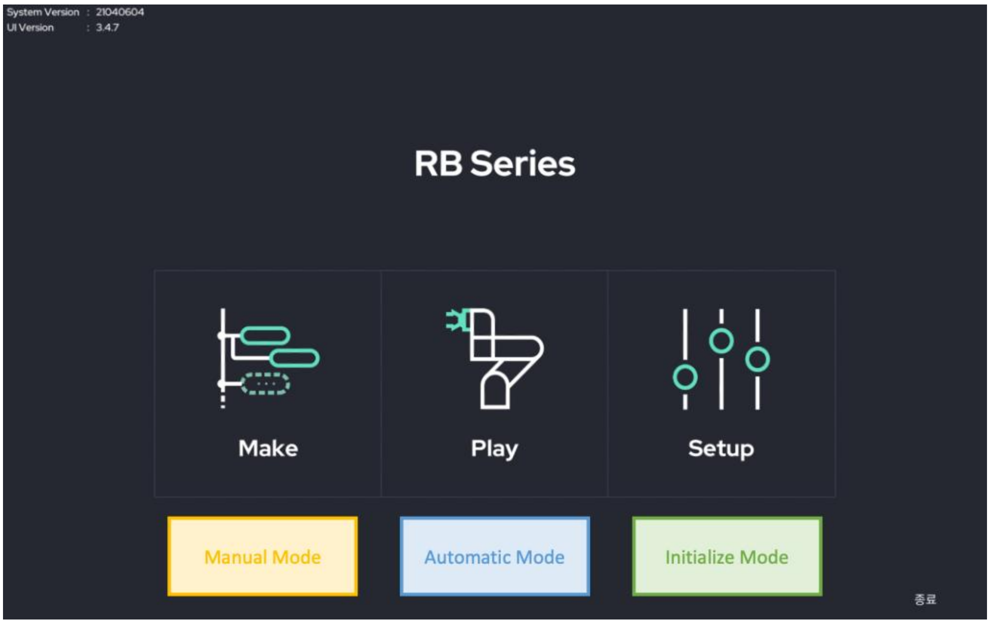
Caution:

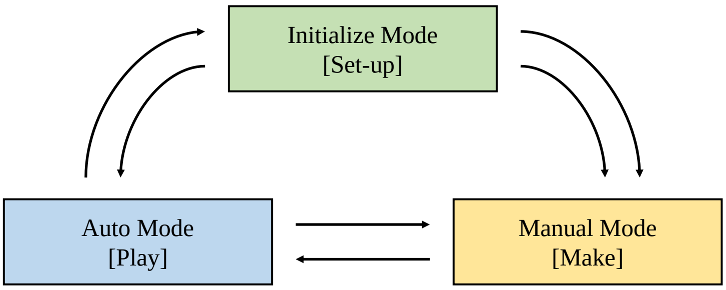
1) The emergency stop switch is designated as stop category 1.
2) You can cancel the emergency stop function by turning the emergency stop switch clockwise.
3) If you need an additional emergency stop switch, you can use it through the control panel.
4) Emergency stop should not be used as a risk reduction method, but should be used as a secondary protective device.

## 3.6 Operation Mode

The operation mode of the collaborative robot RB Series is composed as follows.

When entering the automatic mode, you must access it through a password.




**Caution**

Caution:

1) The password for entering automatic mode is not set at the time of shipment. Set up and use a password so that no one else can access it.
2) Before entering Auto Mode, the user must remove the dangerous situation and check the status of the emergency stop switch and the protective stop device.
3) In any case, the user must correctly grasp the installation state of the robot and complete the setting before operating the robot.

- Initialize Mode [Set-up]

Peripheral device settings or robot status can be set before the robot moves.

Overall settings for robot motion such as workspace, TCP offset, and payload can be made.

At this time, power is not supplied to the motor. The motor can be powered through the activation action.

- Auto Mode [Play]

The robot is in a state where only predefined tasks are performed without user intervention. At this time, power is supplied to the motor.

The motion of the robot programmed through the simulation function can be verified through simulation, and the robot can be driven by the verified program by converting it to a real state.

At this time, the robot is performing pre-set safety functions, and the user can monitor the status of the robot and peripheral devices through the Play window.

- Manual Mode [Make]

It is a state in which the robot is operated through direct actions of the user. At this time, power is supplied to the motor.

Direct teaching, program creation and modification, and manual operation of peripheral devices can be performed, and the robot can only be operated at the moment the user operates the tablet through the safe speed slide bar.

When you release your hand from the safety slide, the robot will stop moving immediately.

## Cautions

Caution:

1) In case of manual operation, the safety slide function must be set.
2) At initial shipment, the safety slide function is deactivated.
3) In addition, when using a 3-position enabling device, it must be used in accordance with 5.8.3 of ISO 10218-1.

## 3.7 OPERATING ENVIRONMENT

In order to keep the robot in a safe state for a long time, it must be used in the following environment.

| | |
|---|---|
| Maximum allowable operating temperature | 50˚C |
| Maximum permissible storage temperature | 60 ˚C |
| Minimum allowable operating temperature | 0˚C |
| Minimum allowable storage temperature | -5˚C |
| Maximum permissible humidity | 80% |
| Lowest permissible humidity | 20% |

## 3.8 MAINTENANCE OF SAFETY FUNCTIONS

In order to keep the robot in a safe state for a long time, it is necessary to continuously check the safety functions.

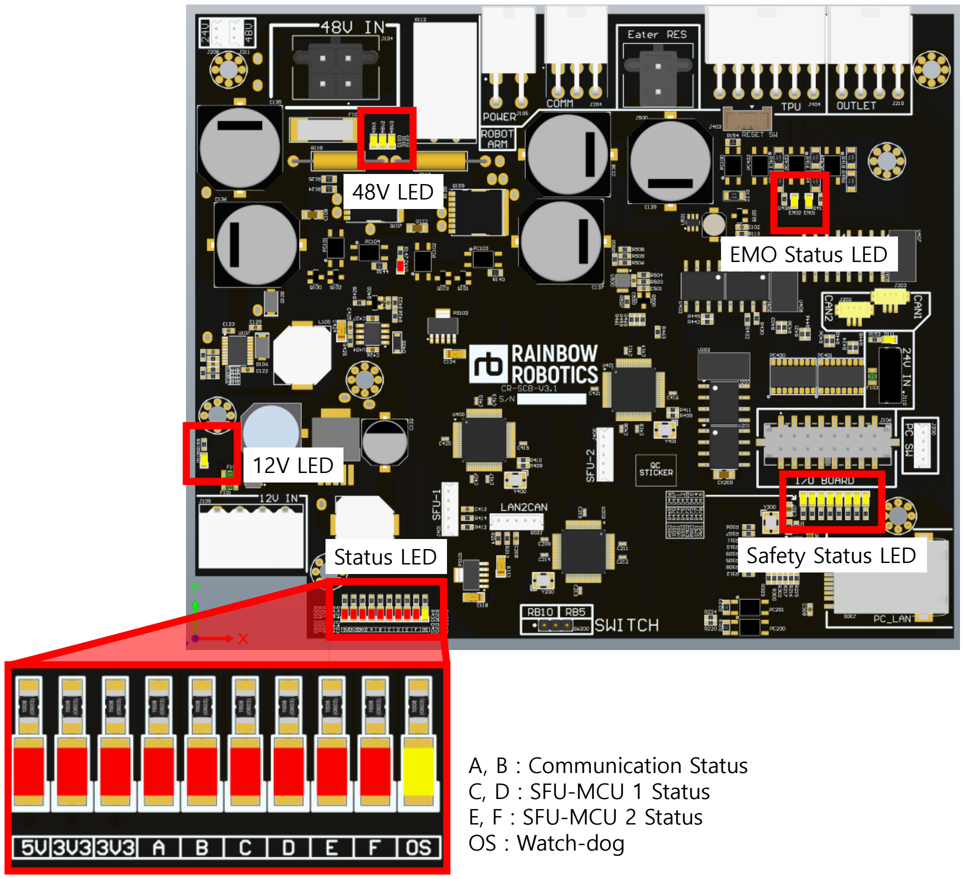
Safety Inspection Managers need periodic inspections for the following items. If during the inspection you find a problem that cannot be solved by yourself, contact the manufacturer.

<table>
<tr><th>Inspection target</th><th></th><th>Check List</th><th>Period</th></tr>
<tr><td rowspan="2">Pendant</td><td>Safety Function</td><td>Check whether the emergency stop switch mounted on the pendant is working properly.</td><td>1 Month</td></tr>
<tr><td>Cable</td><td>Check the condition of the connection cable between the pendant and the control box.</td><td>1 Month</td></tr>
<tr><td rowspan="7">Control Box</td><td rowspan="4">Interface</td><td>Check whether the EMO port to which the safety device is connected is working properly.</td><td>1 Month</td></tr>
<tr><td>Check whether the PRS port to which the safety device is connected is working properly.</td><td>1 Month</td></tr>
<tr><td>Check whether the HSS port to which the safety device is connected is working properly.</td><td>1 Month</td></tr>
<tr><td>Check whether the SSS port to which the safety device is connected is working properly.</td><td>1 Month</td></tr>
<tr><td rowspan="2">Power</td><td>Check the normal output of 24V voltage connecting the safety device.</td><td>1 Month</td></tr>
<tr><td>Check if the 24V fuse is inserted normally.</td><td>1 Month</td></tr>
<tr><td>Cable</td><td>Check the condition of the connection cable between the safety device and the control box.</td><td>1 Month</td></tr>
</table>

## ■ Safety Function Board Specification

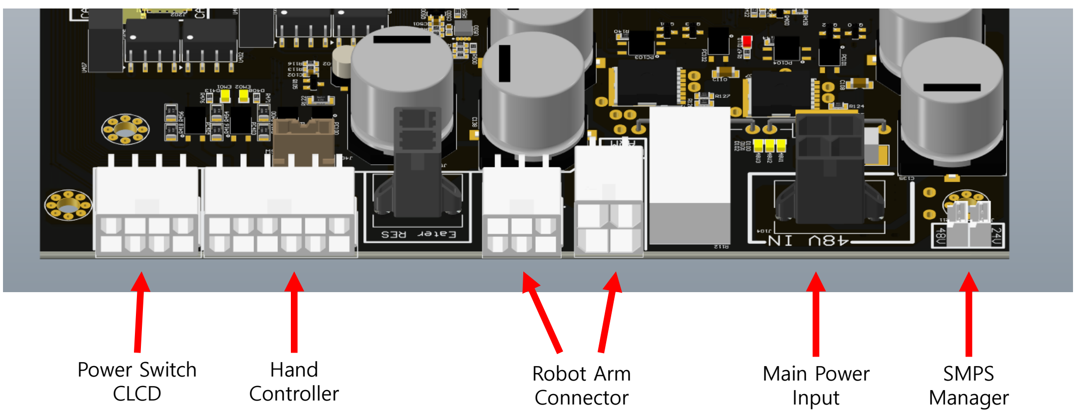
Inside the control box, there is a built-in safety function board to drive the RB Series.

The information of the LED indicating the operation status of the board is as follows.




A, B : Communication Status
C, D : SFU-MCU 1 Status
E, F : SFU-MCU 2 Status
OS : Watch-dog

Connector information connected to the board is as follows.



## 3.9 APPLIED STANDARDS

| Standard | Title |
|---|---|
| **IEC 61508-1:2010** | Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1: General requirements |
| **IEC 61508-2:2010** | Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 2: Requirements for electrical/electronic/programmable electronic safetyrelated systems |
| **IEC 61508-3:2010** | Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements |
| **IEC 61508-4:2010** | Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 4: Definitions and abbreviations |
| **IEC 61508-5:2010** | Functional safety of electrical/electronic/programmable electronic safety-related system – Part 5: Examples of methods for the determination of safety integrity levels |
| **IEC 61508-6:2010** | Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 6: Guidelines on the application of IEC 61508-2 and IEC 61508-3 |
| **IEC 61508-7:2010** | Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 7: Overview of techniques and measures |
| **IEC 60204-1:2016** | Safety of machinery – Electrical equipment of machines – Part 1: General requirements |
| **IEC 61000-6-1: 2016** | Electromagnetic compatibility (EMC) – Part 6-1: Generic standards – Immunity standard for residential, commercial and light-industrial environments |
| **IEC 61000-6-2: 2016** | Electromagnetic compatibility (EMC) – Part 6-2: Generic standards – Immunity standard for industrial environments |
| **IEC 61000-6-7: 2014** | Electromagnetic compatibility (EMC) – Part 6-7: Generic standards – Immunity requirements for equipment intended to perform functions in a safety-related system (functional safety) in industrial locations |
| **IEC 61326-3-1: 2017** | Electrical equipment for measurement, control and laboratory use – EMC requirements – Part 3-1: Immunity requirements for safety-related systems and for equipment intended to perform safety-related functions (functional safety) – General industrial applications |
| **IEC 61800-5-1: 2007** | Adjustable speed electrical power drive systems – Part 5-1: Safety requirements –Electrical, thermal and energy |
| **IEC 61800-5-2: 2016** | Adjustable speed electrical power drive systems – Part 5-2: Safety requirements – Functional |

| | |
|---|---|
| **IEC 62061:2005** | Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems |
| **ISO/TS 15066: 2016** | Robots and robotic devices — Collaborative robots |
| **ISO 10218-1: 2011** | Robots and robotic devices — Safety requirements for industrial robots — Part 1: Robots |
| **ISO 10218-2: 2011** | Robots and robotic devices — Safety requirements for industrial robots — Part 2: Robot<br>systems and integration |
| **ISO 12100:2010** | Safety of machinery — General principles for design — Risk assessment and risk reduction |
| **ISO 13849-1: 2015** | Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design |
| **ISO 13849-2: 2012** | Safety of machinery — Safety-related parts of control systems — Part 2: Validation |

# CHAPTER 4. INSTALLATION

## 4.1 INSTALLATION PRECAUTION

Robot installers must install and operate the robots in accordance with the guidelines of ISO 12100 and ISO 10218-2, and installers must comply with the relevant requirements of international standards such as ISO / TS 15066 and national laws. The manufacturer is not responsible for any accidents caused by risks that do not comply with the relevant requirements provided by international standards, risks that do not comply with the relevant requirements provided by national laws and regulations, or those caused by failure to review the risk assessment in the previous chapter.

## 4.2 Installation Location

The manufacturer recommends the following conditions are met for the installation location.

- Building with seismic design
- No leakage
- No flammable or explosive material
- Constant temperature and humidity
- Limited dust inflow

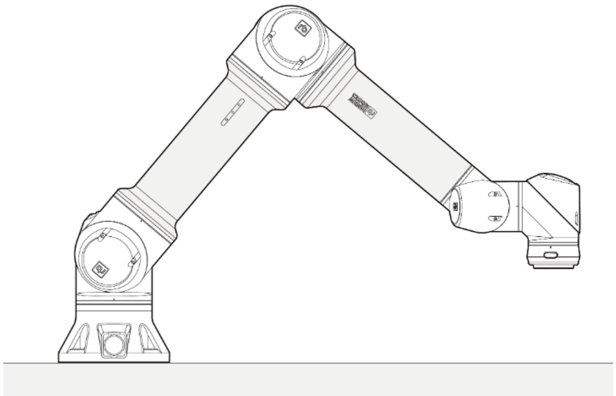
**Caution**

Caution:

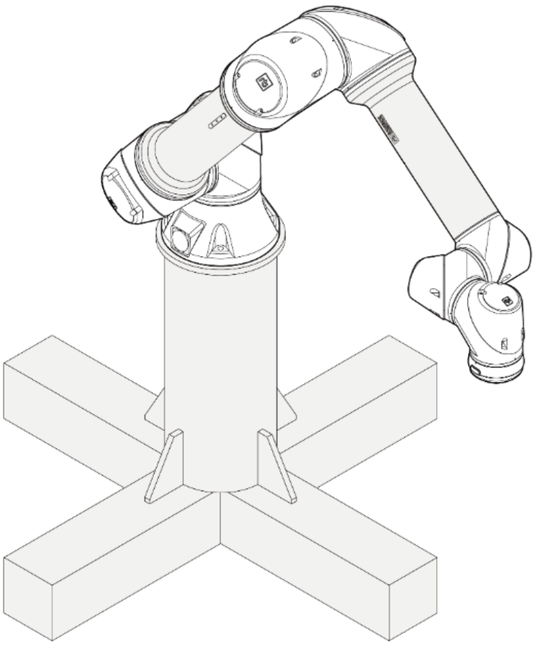
1) If the system is not installed in a location that matches the recommendations, the performance and lifespan of the robot may be reduced.

## 4.3 EXAMPLES OF INSTALLATION

The robot arm can be installed on a horizontal surface (e.g. a table), a wall, the ceiling, or any other angle. However, the user must set the angle of installation in the system-setup when installing on a surface that is not a horizontal surface.



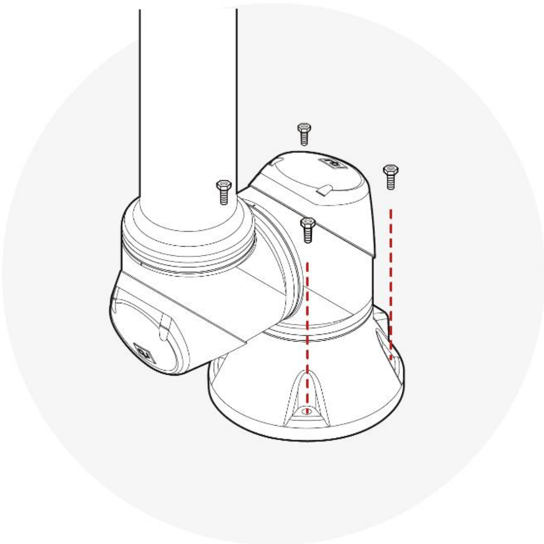
<Installation on the horizontal surface>



<Installation on the fixed post>

## 4.4 MOUNTING THE ROBOT

It is recommended to use four M8 30mm bolts for robot arm installation.



**Warning**

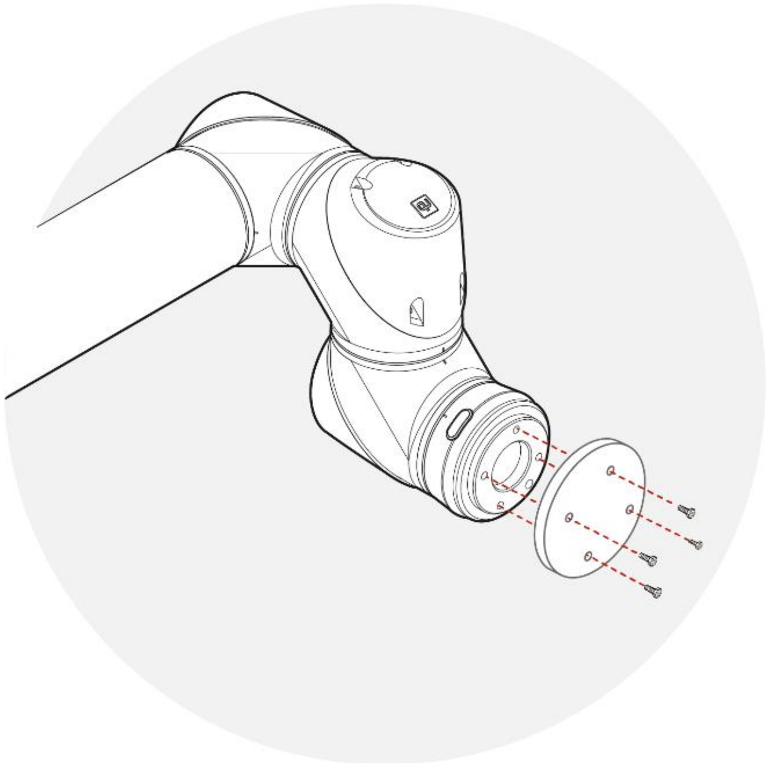
Warning:

1) When attaching the robot, fix it firmly so that the bolts do not come loose.
2) Install the robot on a sturdy surface that can withstand the combined weight of the robot and the load generated by the robot.
3) Please ensure that the mounting surface on the robot arm is completely in contact with the surface that it is mounted upon
4) Never disassemble the bolts that are assembled in the robot. Ensure that all bolts are securely fastened before operating the robot arm.
5) If the bolts are not fastened properly, or if a bracket etc. is installed incorrectly, the product may become damaged, or the safety of the user may be seriously affected.

## 4.5 Tool Connection

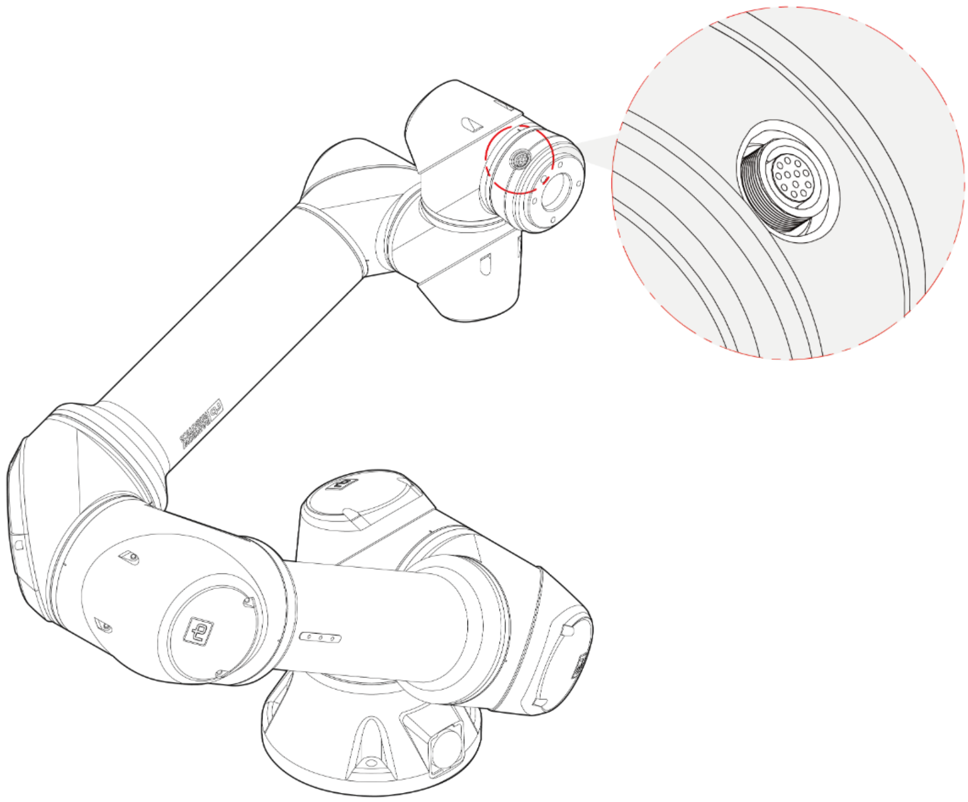
Use four M6 bolts to secure the tool to the tool flange.

- Tools and M6 bolts are not included in the product.
- The connection methods may be different between tools. Please contact the tool manufacturer for further details.

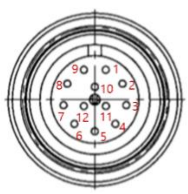


- After fixing the tool to the tool flange, connect the necessary cables to the I/O ports on either the tool I/O or the control box I/O.
- The tool connector uses the Samwoo SW-10W-12 (P) connector.

■ The tool I/O has a 12-pin connector.



■ The port specifications in the tool flange are as follows.

| Port | Layout | Pin num | Signal (Non-E Version) | Signal (E Version) |
|---|---|---|---|---|
| Tool I/O |  | 1 | Digital Output A | Digital Output A |
| | | 2 | Digital Output B | Digital Output B |
| | | 3 | 0/12/24 VCC | 0/12/24 VCC |
| | | 4 | Ground | Ground |
| | | 5 | Digital Input A | Digital Input A |
| | | 6 | Digital Input B | Digital Input B |
| | | 7 | Analog Input A | Digital Input C |
| | | 8 | Analog Input B | Digital Input D |
| | | 9 | RS485+ | RS485+ |
| | | 10 | RS485- | RS485- |
| | | 11 | Common Ground | Digital Input E |
| | | 12 | Common Ground | Digital Input F |

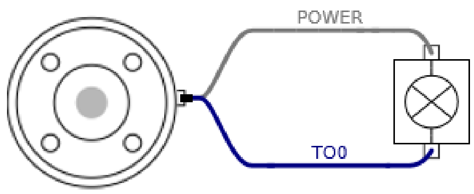
■ The internal power supply can be set to 0V, 12V, or 24V on the I/O tab of the GUI.

| - | Min | Nominal | Max | Unit |
|---|---|---|---|---|
| 24V mode | - | 24 | - | V |
| 12V mode | - | 12 | - | V |
| Current Supply* | - | - | 2000 | mA |

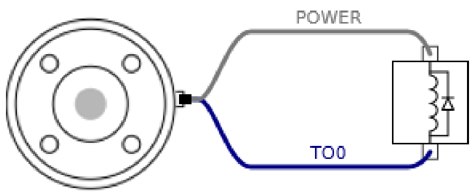
■ The tool connector uses NPN ('sinking') for the digital output. When the digital output is enabled, the corresponding port is connected to GND (ground). When the output is deactivated, the corresponding port becomes 'open' (open-collector / open-drain). The electrical specifications are as follows.

| - | Min | Nominal | Max | Unit |
|---|---|---|---|---|
| Voltage when opened | 0 | - | 24 | V |
| Current through GND | 0 | - | 2000 | mA |

■ The image shown below illustrates how to turn on/off a load with 12V or 24V. The voltage level can be specified in the Tool Out (TO0) block



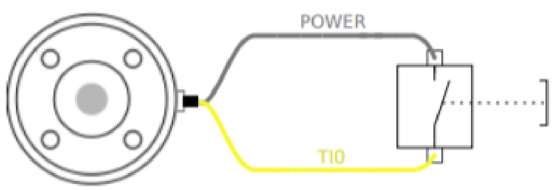

※ It is strongly recommended to use a diode to protect the tool using an inductive load.




■ The tools digital inputs use PNP and pull-down resistors. Therefore, when the input port is not connected (floating), the corresponding input port is read as low (0). Electrical specifications are as follows.

| - | Min | Nominal | Max | Unit |
|---|---|---|---|---|
| Input Voltage | 0 | - | 24 | V |
| Logic Low-Voltage | - | - | 9 | V |
| Logic High-Voltage | 10 | - | - | V |

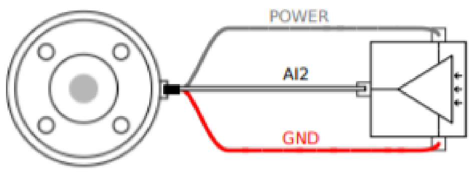
■ The figure shown below illustrates how to use the digital input for a simple switch.



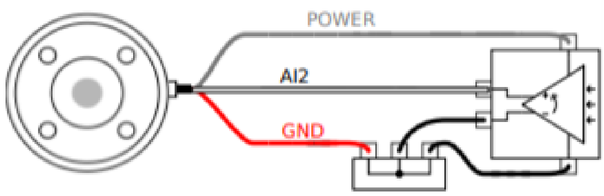
■ The tool analog input measures the voltage in a non-differential manner. The measurement categories are as follows.

| – | Min | Nominal | Max | Unit |
|---|---|---|---|---|
| Input Voltage | 0 | – | 10 | V |
| Resolution | – | 12 | – | bit |

■ The figure below shows how to connect an analog sensor with non-differential voltage output characteristics to the tool flange.




■ The figure below shows how to connect an analog sensor with differential voltage output characteristics to the tool flange. Connecting the negative output of the sensor to GND (ground) works the same as the non-differential light sensor.




## Caution

Caution:

1) For further details regarding technical specification and wire connection, please refer to Appendix D.
2) The cross-sectional view related to the tool flange is illustrated in Appendix C.

■ The tool flange supports RS485 serial communication and supports the following serial communication standard.

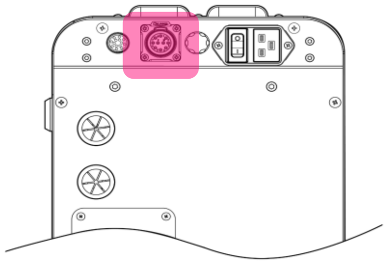
| | |
|---|---|
| Baud-Rate | 9600, 19200, 38400, 57600, 115200, 1M, 2M. |
| Stop Bit | 1,2 |
| Parity | None, Even, Odd |

## 4.6 Cable Connection

The cable connection for stand-type control box describes as follows.

- Connecting the robot arm to the robot control box using the robot arm cable.

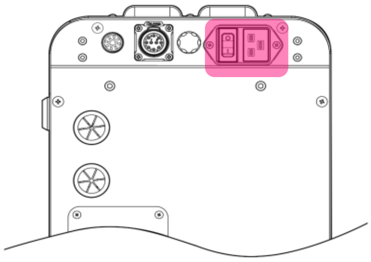
  Please connect the female connector to the robot arm and the male connector to the control box. Please check whether pins in the connector are bent or not.



<Connecting Part for Robot Arm cable>

- Connecting the power cable to robot control box.

  Connect the power cable to the power terminal as shown in the figure below.



<Connecting Part for AC power cable>

The specification of the power system is as follows.

| | |
|---|---|
| Input Voltage | 100 ~ 240 VAC |
| Input Frequency | 50 ~ 60 Hz |

**Caution**

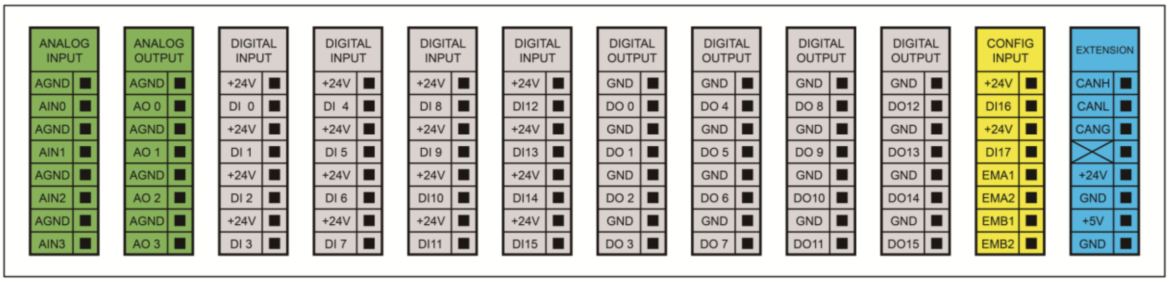
Caution:

1) Do not unplug the robot cable, power cable, or teaching pendant while the robot is turned on.

2) In the use of AC power, the peripherals should share a common ground.

## 4.7 Robot Control Box I/O Overview

To connect other external devices to the robot control box, please connect the I/O from the control box to the corresponding device. The I/O of the control box is very flexible, so it can be used to connect with various equipment such as relays, PLCs, and emergency stop buttons. The layout of the electrical interface inside the robot control box is as follows.




The specifications of the power and digital I / O provided by the control box are as follows. All digital I/O is compliant with the IEC 61131-2 standard.

※ IEC 61131-2: IEC standard for programmable controllers

| Port | Parameter | Min | Type | Max | Unit |
|---|---|---|---|---|---|
| | | | Digital Output | | |
| [ DOx ] | Current | 0 | - | 1 | A |
| [ DOx ] | Voltage Drop | 0 | - | 0.5 | V |
| [ DOx ] | Current Leak | 0 | - | 0.1 | mA |
| [ DOx ] | Type | - | PNP | | Type |
| [ DOx ] | IEC 61131-2 | - | 1A | - | Type |
| | | | Digital Input | | |
| [ DIx ] | Voltage | -3 | - | 30 | V |
| [ DIx ] | OFF Range | -3 | - | 5 | V |
| [ DIx ] | ON Range | 11 | - | 30 | V |
| [ DIx ] | Current(11-30V) | 2 | - | 15 | mA |
| [ DIx ] | Type | | PNP+ | | Type |
| [ DIx ] | IEC 61131-2 | - | 1 | - | Type |

**Caution**

Caution:
When tightening the I/O wiring, please turn off the power to the control box in advance. Any damage to the product caused by the user's carelessness (24V power shorts, incorrect wiring, etc.) is not covered by the product's warranty.

## 4.8 Safety Input Configuration

For the safety of users, all safety-related I/O must be configured with multiple backups. Safety devices and equipment must be installed in accordance with the instructions in Chapter 2 Safety and Chapter 3 Installation.

**Danger**

Danger:

1) Never connect a safety signal to a PLC other than a safety PLC. Failure to follow these warnings could result in unsafe operation, resulting in serious injury or casualty. The safety signal and general I/O signal must be separated.

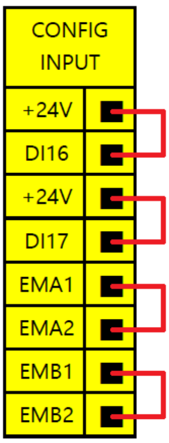
**Warning**

Warning:

1) Inputs and outputs of all safety classes are redundant. It is necessary to isolate the channel so that the safety function is not activated due to signal failure. The safety functionality must be confirmed before installing the robot. The safety functionality should also be checked periodically for abnormalities.

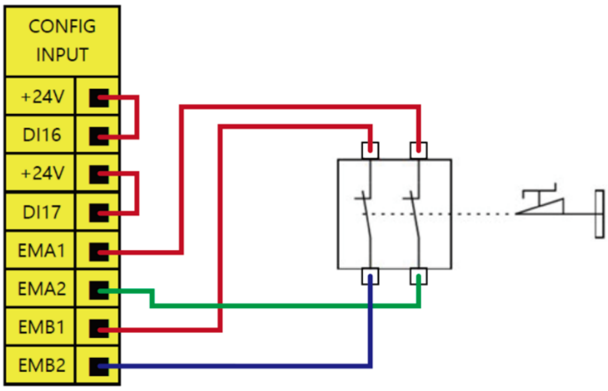
- Initial Safety Configuration

The robot will be shipped with the initial safety configuration set to default, so that the users can use it without further configuration. The initial safety configuration is as follows.



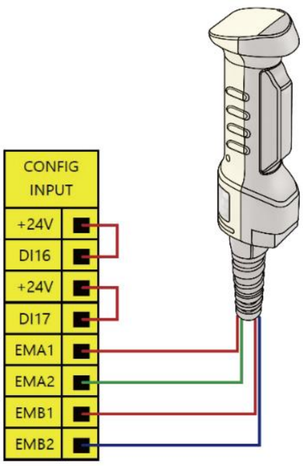
■ Safety protection stop and automatic restart

An example of a safety protection device would be a door switch that stops the robot when the door is opened. The figure shows how to configure these features:




■ Enabling Device Input (Option)

Connect the active device input interface using the 3-position switch. When the position activation switch is in the operating position (middle position), the robot starts moving. If the 3-position activation switch is pressed, the switch is in the inoperative position and the robot arm will not move. Rainbow Robotics does not provide an Enabling Device. An Enabling Device is available as an option if the user needs. To configure the feature, refer to the following configuration:

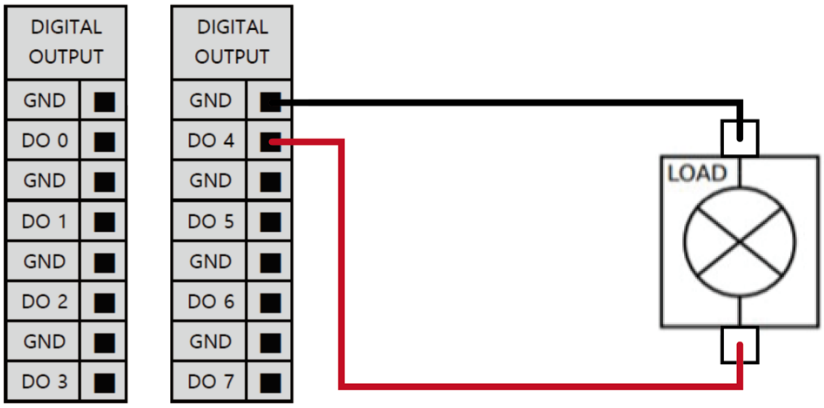


## 4.9 General Purpose Digital I/O Configuration

All Digital I/O can be used as general purpose digital I/O. To use other external equipment with the robot, connect the I/O from the robot control box with the corresponding equipment. The universal digital I/O can be used to configure devices such as relays or PLC systems. In this configuration, the output is always LOW unless the program is running. The following subsections are examples.

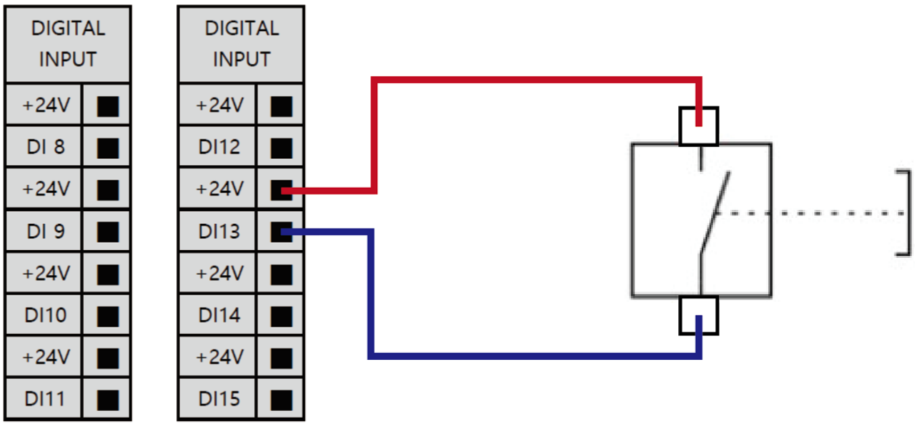
- Electric load control with digital output

  The figure below shows a way to control electric load by using the digital output.



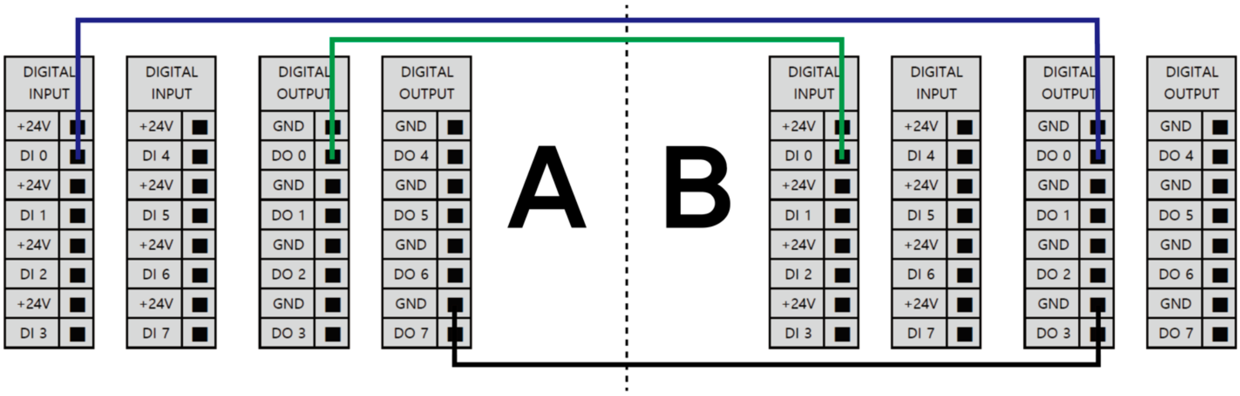

- Control of digital input with a button

  The figure below shows a simple way of connecting a button to the digital input.



■ Communication with other system or PLC

If another other system provides PNP and uses a common ground, the digital I/O can be configured to communicate with the other system. Its connection is shown in the figure below.




**Warning**

Warning:

For the details in the technical specification and wire connection, please refer to Appendix D.

## 4.10 General Purpose Analog I/O Configuration

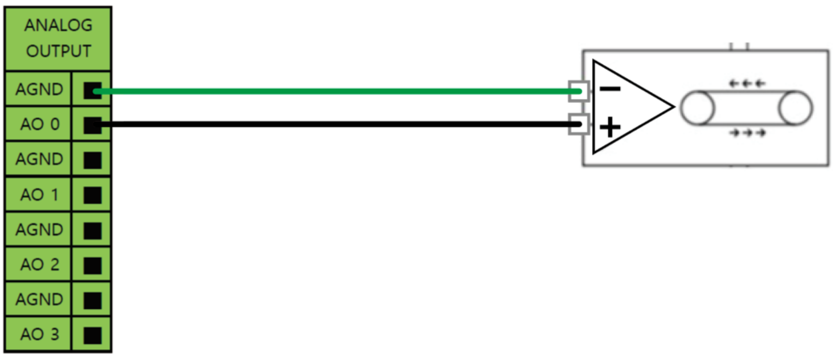
The following methods are recommended for high reliability.

- Use analog GND closest to I/O.
- Equipment and control box use the same GND. Analog I/O is not isolated from the robot control box.
- Use shielded or twisted-pair cable. Connect to the GND shield on the Power (J12) terminal.

| Port | Parameter | Min | Type | Max | Unit |
|---|---|---|---|---|---|
| Voltage mode Input | | | | | |
| AIx - AG | Voltage | 0 | - | 10 | V |
| AIx - AG | Resolution | - | 16 | - | Bit |
| Voltage mode Output | | | | | |
| AOx – AG | Voltage | 0 | - | 10 | V |
| AOx – AG | Resolution | - | 16 | - | Bit |

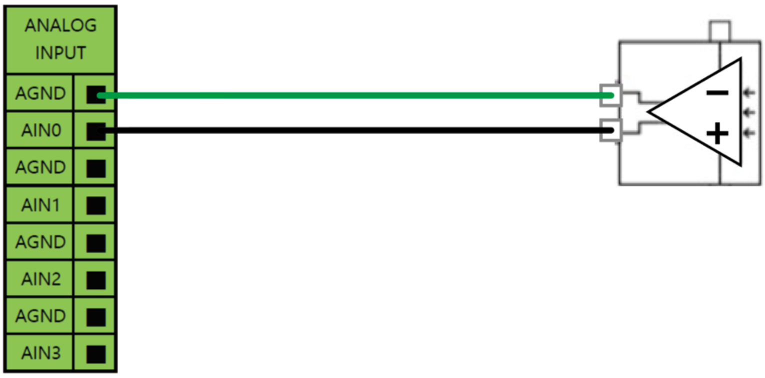
■ Analog output

The analog output can be used to control speed of conveyor. The figure below illustrates a simple demonstration.

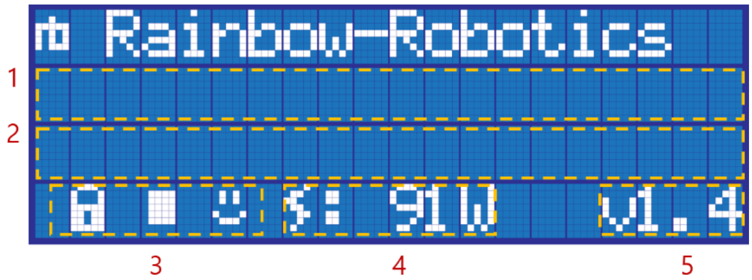


■ Analog input

The figure shown below illustrates a simple connection to an analog sensor. The output value of the analog sensor can be used by the control box as analog input.




■ LCD Status Display




1. Display Box (1): Displays information about system status

   : Please Wait (the main PC in the control box is booting up)

   : Normal Operation (the main PC in the control box is ready)

2. Display Box (2): Displays information about robot operation and status.
3. Action Icon: Definition : lock () or release() state, play() or stop() state, crash() or safe() state.
4. Power Consumption: Indicates the total power consumption in watts (W).
5. System Version Information: System version information.

# CHAPTER 5. GET STARTED

## 5.1 CONTROL BOX ON/OFF

Procedure for turning control box On/Off is as follows

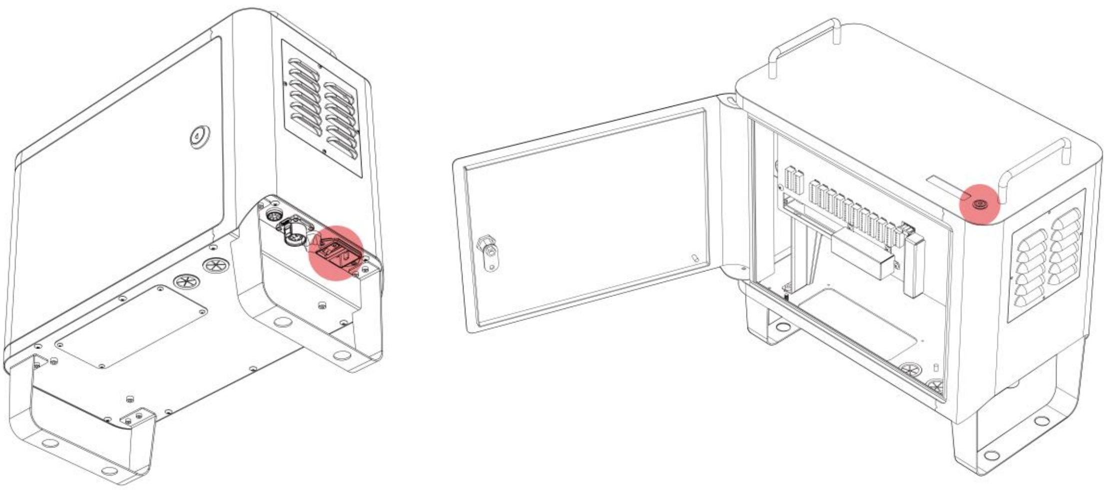
- Stand-type control box On/Off

Press the AC power switch at the bottom of the control box to apply AC power.

Press the main power switch at the top of the control box to turn on the main power.

"Please Wait" is displayed in the LCD screen of the control box. This indicates that the control box is being booted.

When the control box is changed to the enabled state, the LCD message is displayed as "Normal Operation".



To turn off the power, press the main power switch during few seconds.

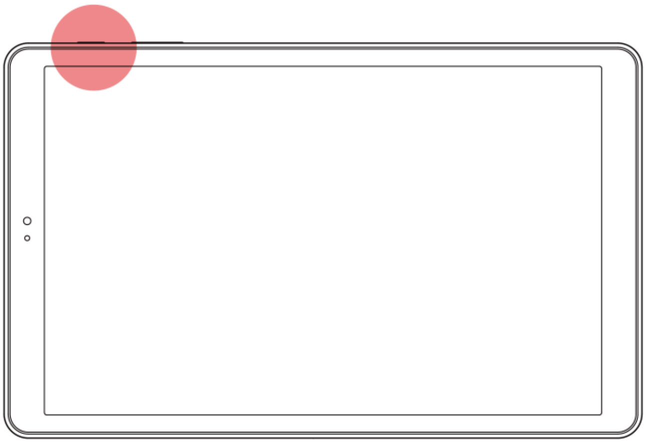
**Caution**

Caution: Control box uses AC 100 to 240V single phase (50 to 60 Hz).

## 5.2 Teaching Pendant/PC On/Off

If the user uses the teaching pendant provided by Rainbow Robotics, the teaching pendant and cover are provided. To turn on the teaching pendant, press the power button on the top left corner.

- For stand-type control box



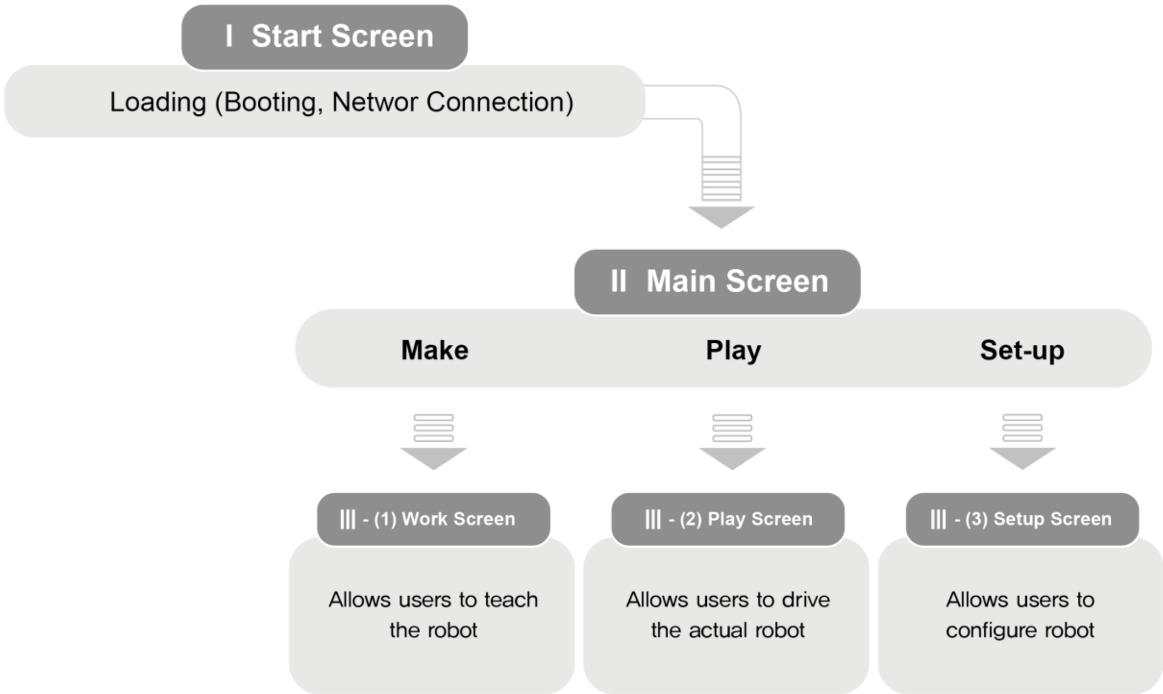
**Caution**

Caution: Make sure that the teaching pendant is connected to the control box before running the application provided by the Rainbow Robotics. Do not perform unnecessary operations while the system is booting, as it may cause problems with the system.

# CHAPTER 6. SOFTWARE OVERVIEW

## 6.1 UI STRUCTURE

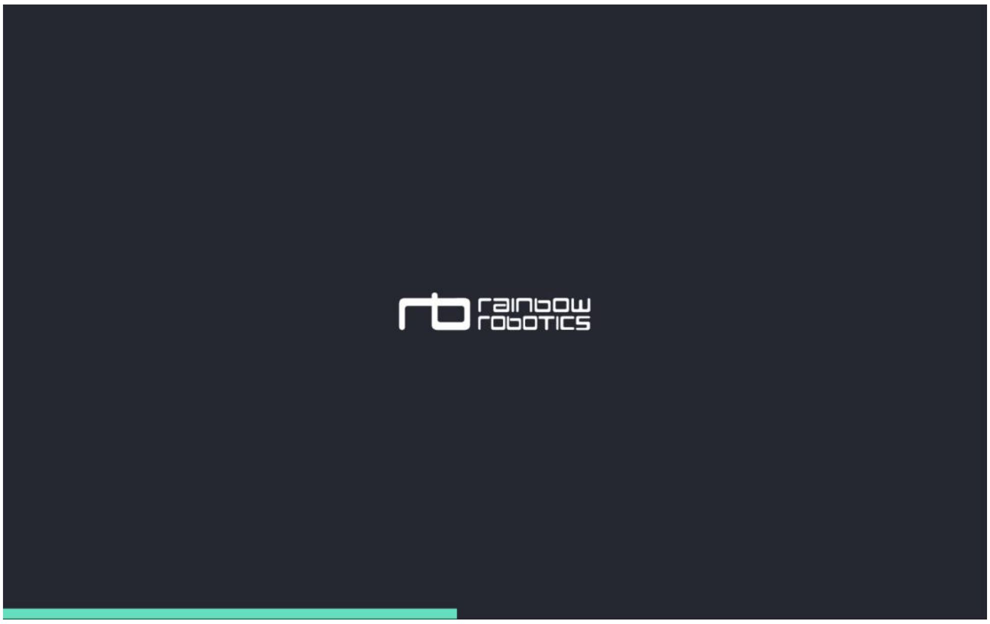
The UI (User Interface) program is divided into three screens as follows. Each section allows the user to enter necessary steps.



## 6.2 Startup Screen Display

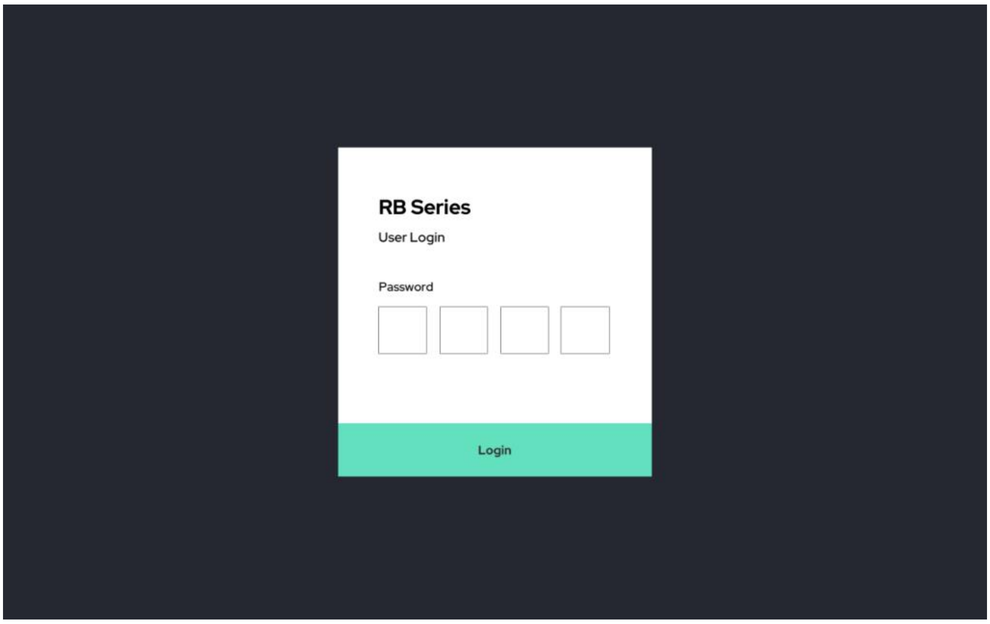
- Intro

The below image shows the start screen. The start screen will occur while the application is loading its processes.




- Login **(Factory-Default login password: 0000)**

To set up the password or to enable automatic login, please go to the "Setup-System-Password" menu.

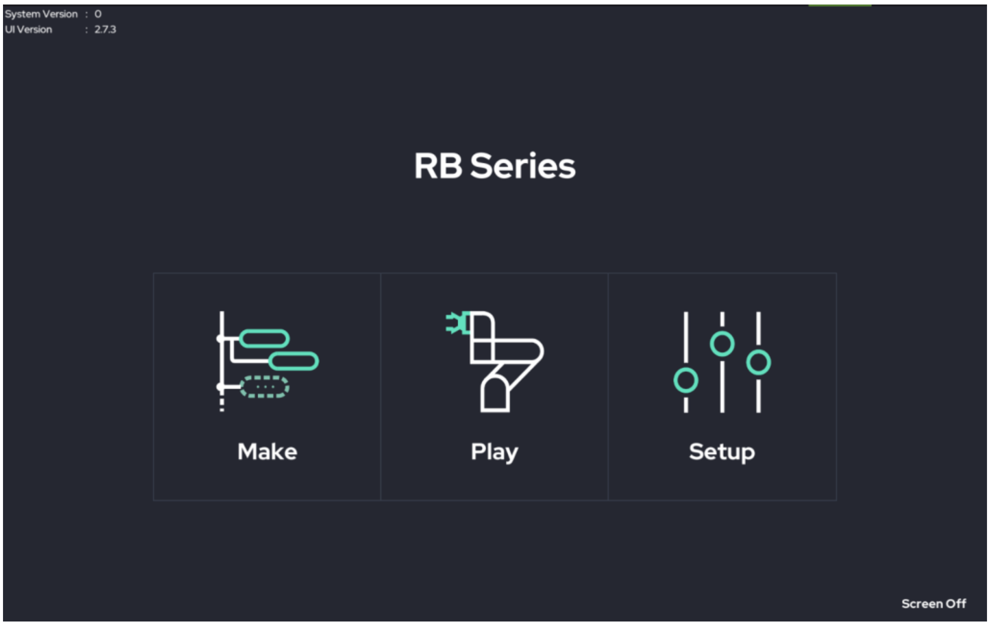


## 6.3 MAIN SCREEN DISPLAY

The UI has three main menus.

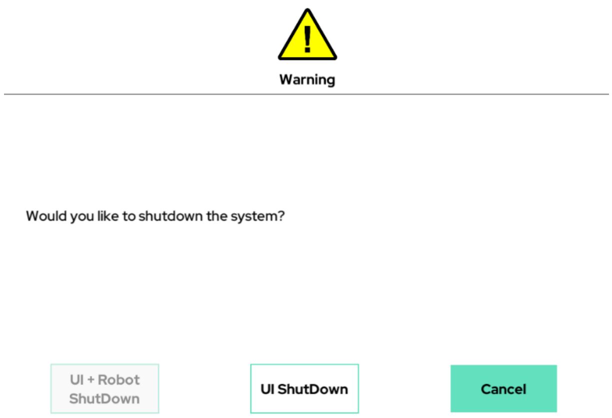
- Make : for programming robot motion and tasks.
- Play : for running motion and tasks pre-programmed in the Make menu.
- Setup : for setting up parameters.

In the main screen, users can create programs for the robot (Make), move the robot (Play), or set settings (Setup) through each relevant menu.




- Power Off

  When the power button (bottom right) is pressed, the power off dialog will pop up. If the user presses the UI Shutdown button, the application closes. If the robot is activated and Tablet is connected to the tablet PC, the power of the robot will be turned off as well.



# 6.4 Make

■ Make

The Make screen is the interactive menu to program the robot. Programming the collaborative robot will also be referred to as "teaching." Teaching the robot requires the use of the icons at the top of the screen. Moving the robot requires the use of icons at the right. Moving the robot by one of these icons will also be referred to as "jogging." Editing the teaching program requires the use of icons at the left.

- Left Icons: Copy & Paste, Save, Delete, Add Comment, etc.
- Right Icons: Jog/Jogging, Move Left/Right/Forward/Backward, etc.
- Middle Icons: Program Functions, etc.
- Bottom Icons: Save/Load, Play, Motion Speed Adjustment, etc.

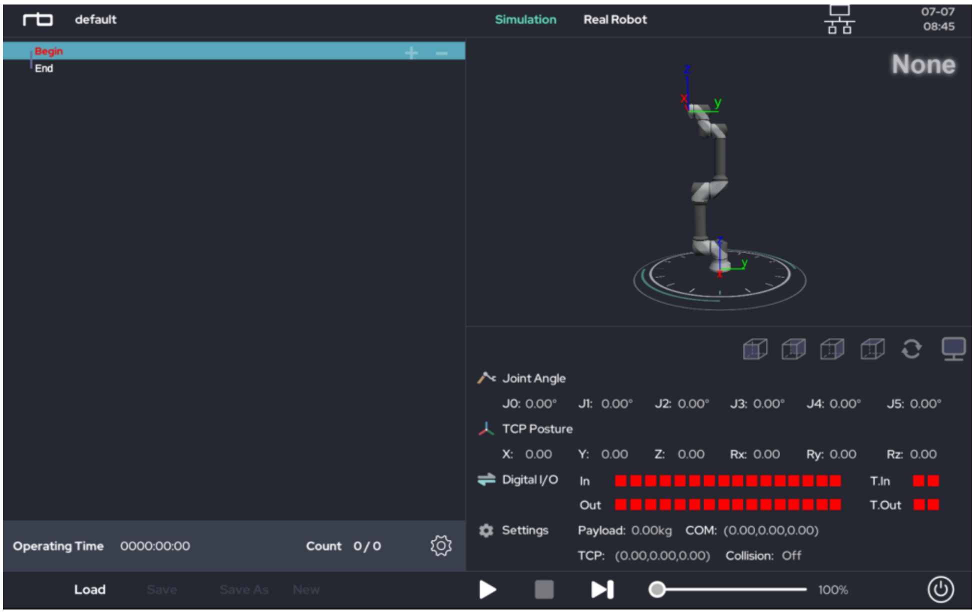

※ For more details about icons and configuration, please refer to Chapter 7.

※ In the Make menu, the robot will not move unless a button is pressed and held. This feature can be removed in Setup-Interface.

# 6.5 PLAY

■ Play

The Play screen allows the user to load and run a teaching program. The Play screen only allows for physical movement of the robot (unlike Make, which allows for simulation). A program loaded into the Play screen will repeat the number of times specified in Setup-Interface. The Operating time at the bottom left of the screen shows the time elapsed.

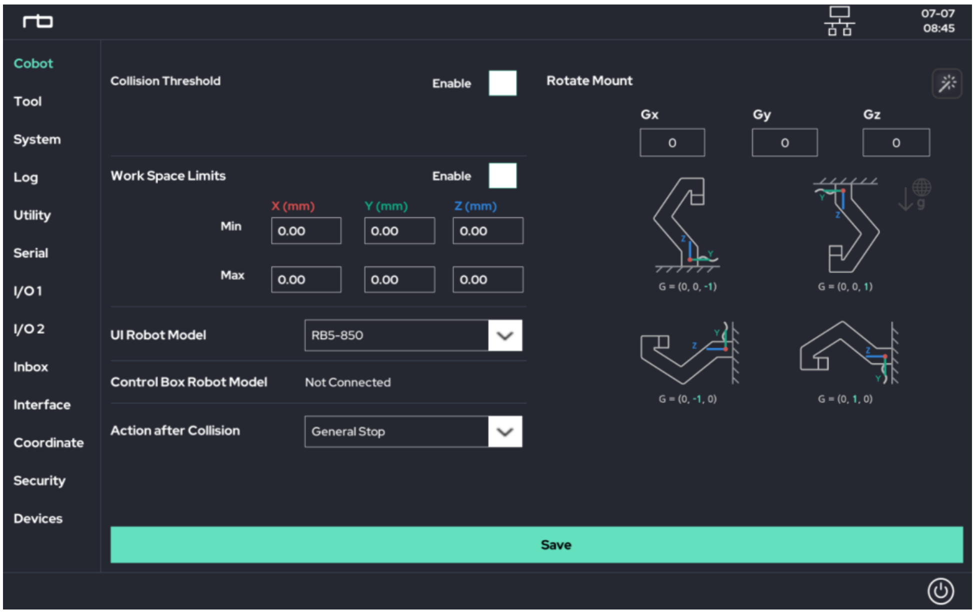



※ Please refer to Chapter 8 for more details about Play mode.

# 6.6 SETUP

■ Setup

The Setup screen allows the user to see/change the robot's default values, such as sensitivity for collision detection, orientation of the robot installation, range of workspace, tool settings, system log, I/O, coordinate system, etc.

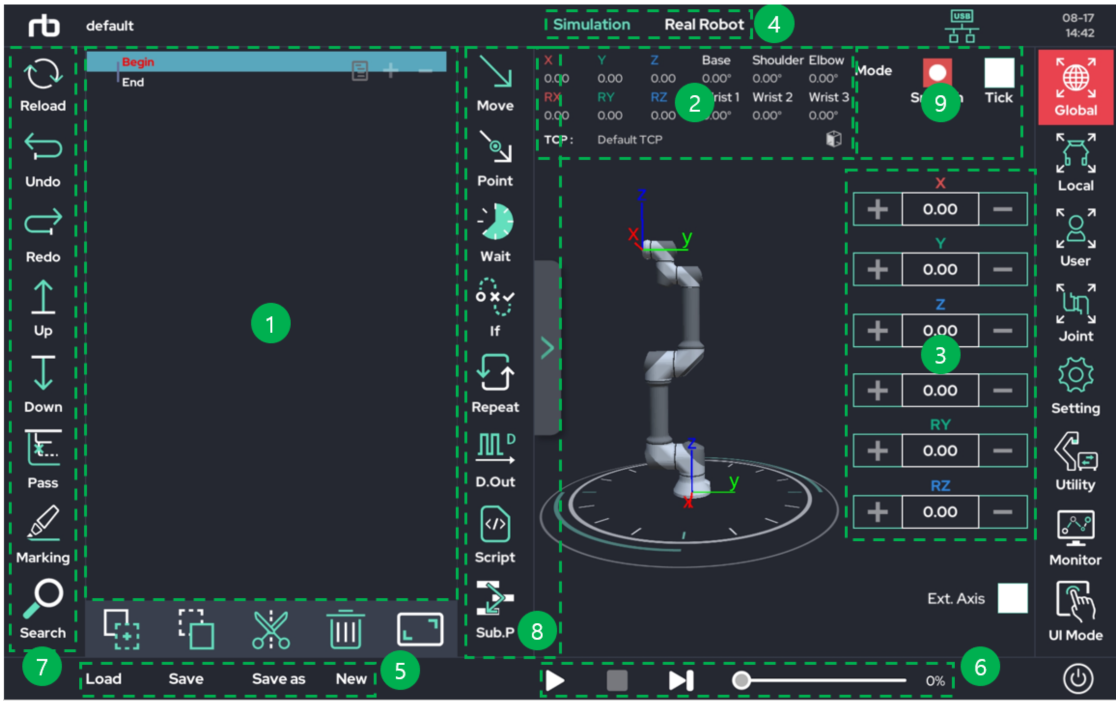



※ Please refer to Chapter 9 for more details about Setup.

# CHAPTER 7. PROGRAMMING GUIDE

## 7.1 ICONS AND ACTION SCREEN

■ Description of components in Make screen display

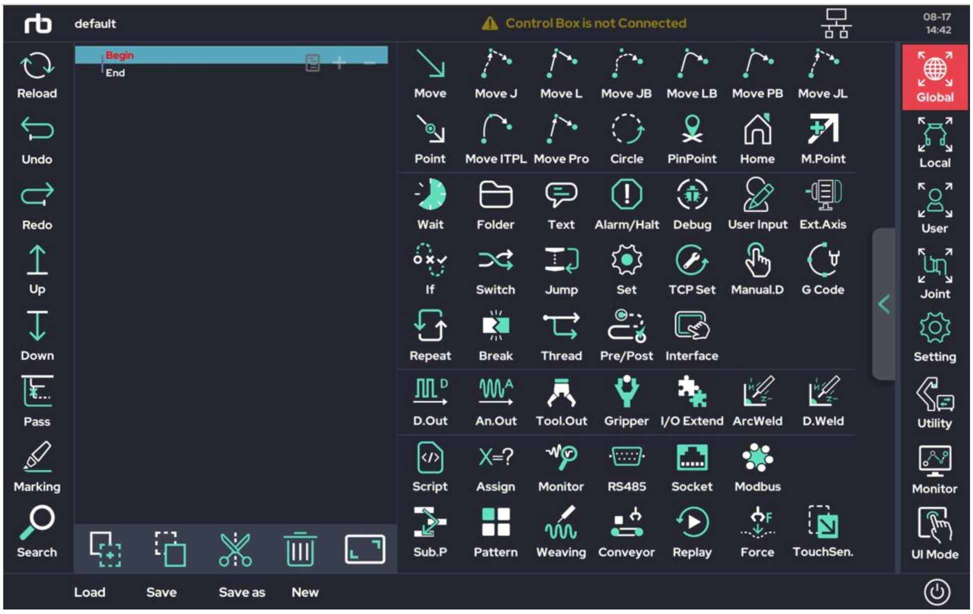



| Num | Description |
|---|---|
| ① | Show the program list in tree form. |
| ② | Shows the angle of each joint of the robot arm and the Cartesian coordinate position of the TCP. |
| ③ | TCP Jog: can change the Cartesian coordinate position. (base/tool/user defined coordinate system Selectable)<br>Joint: Jog: can change the angle of each joint. |
| ④ | Button to switch to Simulation or Real mode.<br>※ Real mode must be selected to drive the real robot arm. |
| ⑤ | Can import saved projects and create new projects. |
| ⑥ | Starts or stops the program and exists Motion Speed Adjustment Bar. |
| ⑦ | Various editing tools are located, such as Copy/Paste/Annotations. |
| ⑧ | Can adjust the motion speed of the robot arm (even when the robot is running). |
| ⑨ | Determines Jog Method – either Smooth Mode or Tick Mode. |

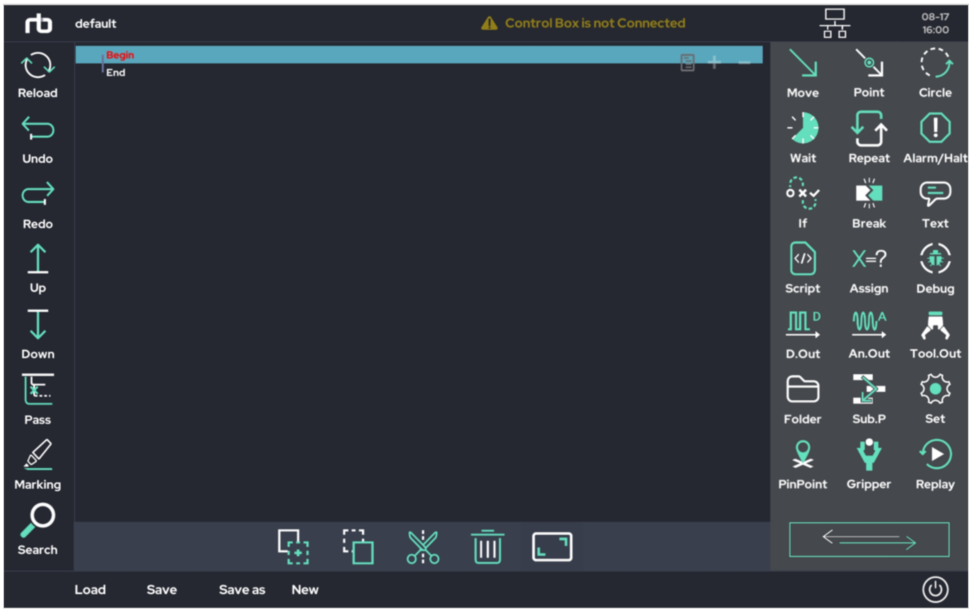
※ Teaching: programming RB's motion by means of moving the robot by hand

※ TCP (Tool Center Point): The point defined for the tool center point within the robot's base coordinate system. It may also be the origin of the end-effector.


<Basic View Mode>




<Icon Extended View Mode>

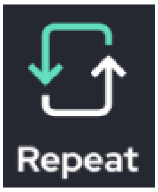
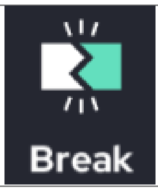
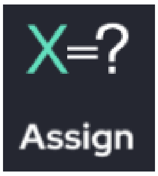
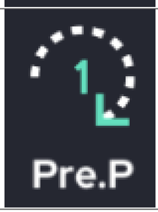


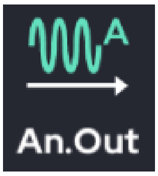

<Program-only Mode>

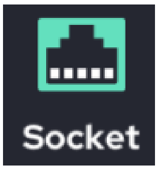
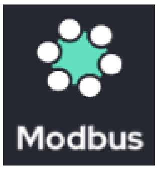
■ Description of icons used in teaching (Upper part in Make page)

※ A detailed description of each function is explained in later chapter

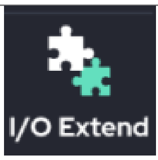
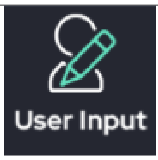
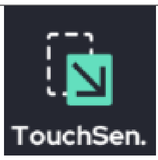
| Icon | Description |
|---|---|
|  | This icon is used to set motion property of the robot. The core algorithms for seven types of motion properties are pre-programmed.<br>In MoveJ, each joint moves independently to reach a given target joint angle in a given time.<br>In MoveL, the TCP linearly moves to reach a given target position and orientation in a given time. In this motion, each joint angle to move is calculated by built-in algorithms.<br>MoveJB, MoveLB, MovePB, MoveJL, and MoveITPL are advanced motions using MoveJ or MoveL. |
|  | This icon is a sub item of Move. It is used to set the target values of motion.<br>In MoveJ and MoveJB, a desired joint angle value can be set.<br>In MoveL, MoveLB, MovePB, MoveJL, and MoveITPL, a desired TCP position (x,y,z) and orientation (Rx,Ry,Rz) can be set in Cartesian coordinates. |
|  | This icon is used to design a circular motion. This motion can be generated as an arc passing through three points given by a user, or a circle defined by the center and the axis of rotation. |
|  | This icon is used to let the robot pause shortly. The robot will pause for a given time set by a user. With a conditional statement like IF, a user can have the robot pause when a condition is true. |
|  | This icon is used to create a conditional statement. A user can generate separate motion program branches depending on conditions using If, Else if, and Else. |
|  | This icon is used to create another type of conditional statement. For the Switch statement, a user defines each case. |

| | |
|---|---|
|  | This icon is used to repeat a specific section in the motion program. A user can set a specific number of times to repeat. A user can also repeat indefinitely until a condition becomes false, or repeat indefinitely until a condition becomes true. |
|  | This icon will force quit a loop. It is used as a sub item of Repeat. |
|  | This icon ends the program. It is typically used with a conditional statement to force an end to the motion program in a situation. |
|  | This icon is used to declare variables. It has four types; variable, array, point and string. The variable stores a single number, array stores multiple numbers, point stores posture information, and string stores words. |
|  | This icon allows a user to program manually. It is typically used when a user-specific calculation and substitution are needed. |
|  | This icon allows a user to add a comment or memo to the program. |
|  | This icon bundles the commands of the created program into the sub items of the folder. |
|  | This icon allows a user to import other user programs and insert them to the current program. The imported programs cannot be edited in the current program. |
|  | This icon is used to run a specific command or program only for one time at the beginning of a program. |

| | |
|---|---|
|  | This icon allows a user to run a command or program in parallel to main program. Note that motion commands cannot be used in the thread. |
|  | This icon generates a message pop-up during operation. It can be used when a situation requires a confirmation during program execution. |
|  | This icon allows a user to check the current value of variables (which is assigned by 'Assign' action) or parameters. Information is displayed within a pop-up. |
|  | This icon allows a user to temporarily change the values of parameters located in Setup menu at the current program. |
|  | This icon is used to control Digital output ports located in the control box. A user can select a port and define its output signal (High, Low, or Bypass) |
|  | This icon is used to generate voltage through an Analog output port located in the control box. Each Analog output can generate a voltage in a range between 0V to 10V. |
|  | This icon is used to specify two digital outputs located in Tool flange. A user can setup the digital output to generate 0V, 12V, or 24V. |
|  | This icon is used when a gripper manufactured in third-party company is attached at the robot. Built-in functions let a user quickly install and use the gripper. |
|  | This icon is used to send data to the port located in the Tool flange or Control Box via RS485/RS232. Please refer to Setup-serial for protocols. |

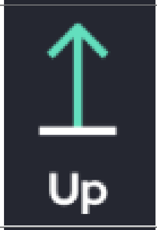
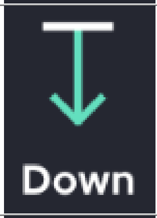
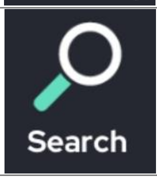
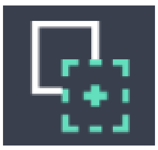
| | |
|---|---|
|   | This icon is used for socket communication. A maximum of five connections are allowed. |
|   | This is for ModBus Client function. This icon allows a user to connect to other ModBus server. The program can access to a specific IP address in order to request and receive data.<br>The protocol of the ModBus Server is provided separately in the user manual Appendix. |
|   | This icon allows the robot to work as a conveyor system. When the moving speed and direction of the conveyor is defined, the robot follows the conveyor. MoveL, MoveLB, MovePB, MoveITPL and Circle can be programed on top of the conveyor's motion. |
|   | This icon is used to set up a task after the program ends. Note that motion commands cannot be included in here. |
|   | This function inserts another pre-made program file (teaching file) into the current document in a modifiable form.<br>It is similar to Sub.P, but files added using Sub.P are not modifiable. However, the programs added using the Template function can be modified in the current program. |
|   | This function is used to declare variables (single variables, arrays, point variables, etc.) that you want to observe in real time while the program is running. Variables declared in the Monitor function can be viewed by clicking the monitor icon on the right side of the Make / Play page. |
|   | PWM (Pulse Width Modulation) output function. Input the frequency and duty ratio of the PWM pulse to send PWM signal through digital output port. |

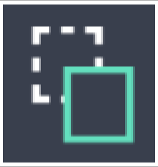
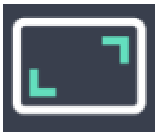
| | |
|---|---|
| Pattern | This function allows a user to define repetitive behavior. By defining information about the space in which to perform the repetitive actions, as well as defining the repetitive actions to be performed at each location, the robot will perform the same action at every point. Palletizing can be implemented through this function. |
| PinPoint | This is a function that can be used by storing certain posture/position information as a variable and then referring to (calling) another motion function. However, this function itself does not move to that position. |
| Jump | This function can jump program flows to a specific location/line. It can return to the starting position, or control program flow discontinuously with a specific line number. |
| Replay | This function re-play recorded motion through motion recording function. Motion recorded through direct instruction or et cetera can be replayed with J or L type through this function. |
| Weaving | This is a dedicated feature for weaving operation. Actions such as Move L or Circle set below the weaving action are combined with the weaving options set. |
| Force | This is a function for force control. You can select the desired direction of force action and coordinate system. The actions included in the lower force control are automatically given the force control function. |
| ArcWeld | This function is dedicated macro function for arc welding. It is a dedicated function that binds functions which can be implemented through normal D.out or Wait function so that they can be used quickly in macro form. Setting up the arc welder is done in device on the Setup page. |
| TCP Set | The TCP settings feature is the ability to temporarily change TCP values by recalling pre-saved TCP values during program execution. User can save TCP values in advance from the Tool List on the Setup page. |

| | |
|---|---|
|  | Manual direct teaching is a feature that allows user to pause and use direct - teaching while the program is running. |
|  | The G code feature is a feature that you can use if you have placed the G code file in the specified folder. The robot will implement the path to that G code. |
|  | This function is built-in so that other products such as HMI and PLC can be used easily. Users can select the product they want to use and select detailed functions to use the selected product. |
|  | This function is available when adding an I/O expansion module. It is possible to set the digital/analog output of the I/O expansion module. The setup method is the same as the existing D. output and An.output. |
|  | The user input function pauses during program execution to allow users to change the value of a variable/arrangement/point/character/global/ROM by entering it. You can change the value, ignore it, and skip it, depending on your situation. |
|  | It is intended for use in welding applications. Detects the movement of the base material and allows welding to be carried out by reflecting the direction of movement and movement. |
|  | It is a function to move the robot to 'Project Home Posture' or 'Joint Zero Posture'. The movement type can be selected from among MoveJ and MoveL. |
|  | This function is for using the digital welding machine. After selecting the brand digital welding machine user wants to use, user can easily use the digital welding machine by selecting the mode and option. |
|  | Unlike the General/Non-Stop thread that runs concurrently with the program, it is a function that calls the event General/Non-Stop thread that is executed when called from the main program. |

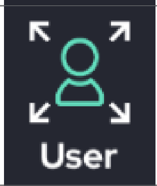
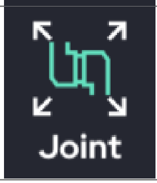
| | |
|---|---|
|  | This function is used when you want to change the main program to another project. |
|  | It is a function to operate the robot joint and the external axis at the same time. |
|  | This function is for controlling external axis in addition to the robot. Up to 6 can be added. |

■ Icon description for editing (Left side in Make screen)

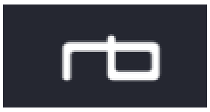
| Icon | Description |
|---|---|
|   | It recalls the currently open file.<br>※If you press Reload button without pressing Save button, you will be able to blow up the last saved file, so be careful. |
|   | It reverses your last action.<br>You can do up to 50 times. |
|   | It reverses your last Undo.<br>You can do up to 50 times. |
|   | It can raise the command one by one. |
|   | It can lower the command one step at a time. |
|   | This is Annotation function. It prevents the selected command from running. Annotated commands exist in the program but not executed |
|   | It can mark the highlight (marking) in the desired program line. Therefore, you can underline important program lines. |
|   | This is a function to search for functions used within the program. However, you can only search in English. |
|  | It can copy the selected command and you can the copied command to a different location. |

| | |
|---|---|
|  | It can paste the copied or cut command into the selected location. |
|  | It can cut the selected command. This command can be pasted to a different location. |
|  | It can delete the selected command. |
|  | It can change edit mode to zoom mode. |

■ Jog and other utilities (Right side of Make screen)

| 아이콘 | 설명 |
|---|---|
|  | It can move TCP's position relative to a global coordinate system fixed to the base. |
|  | It can move the position of TCP based on the local coordinate system (tool coordinate system) fixed to TCP. |
|  | It can move the position of TCP based on the user-defined coordinate system (User Coordinate). |
|  | It can allow to move each joint of the robot arm separately |
|  | This is a collection of special features which can view status and set-up values such as I/O status information of the system, user-coordinate Setting information and current/temperature information of the robot. |
|  | This is a collection of settings such as User coordinate system settings, automatic TCP find, and other easy-to-use settings with a Jog. These settings can also be set in the Setup menu by default. |
|  | It is a window for real-time observation of the values selected variables through the Monitor function. In addition to the selected variables, system variables that need to be checked frequently are also displayed. |
|  | This function allows the user to select the UI mode. Users can select UI mode according to their level and environment. |

- System function button

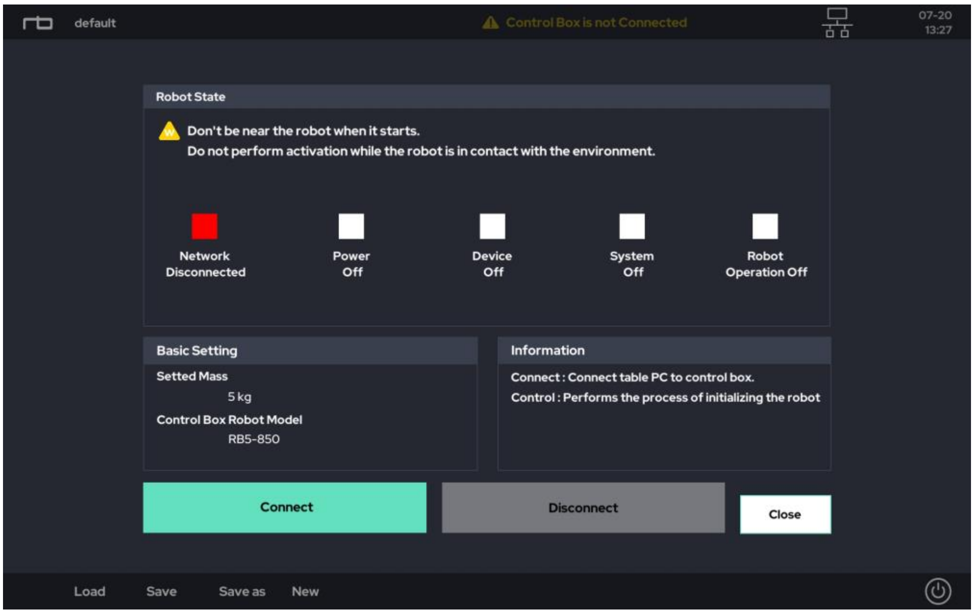
| | |
|---|---|
|  | This icon is used to move to home screen & another page. It is located in the top left. |
|  | This icon is used to power off the UI. When the tablet PC is connected to the robot, the robot will also be turned off. It is located in the bottom right.<br>Screen Lock function is included in here. |

# 7.2 CREATE TEACHING ENVIRONMENT

Robot teaching (programming) is available only in the Make screen. Please open the Make screen from the Play or Setup screen via the  button located at the top of the UI. It is also possible to move to the Make screen from the Home screen.

- Connect Tablet PC to Control Box

 This icon is in the bottom left of each screen. The robot control box and tablet PC must be connected before teaching. When this icon is pressed, the following screen is displayed.



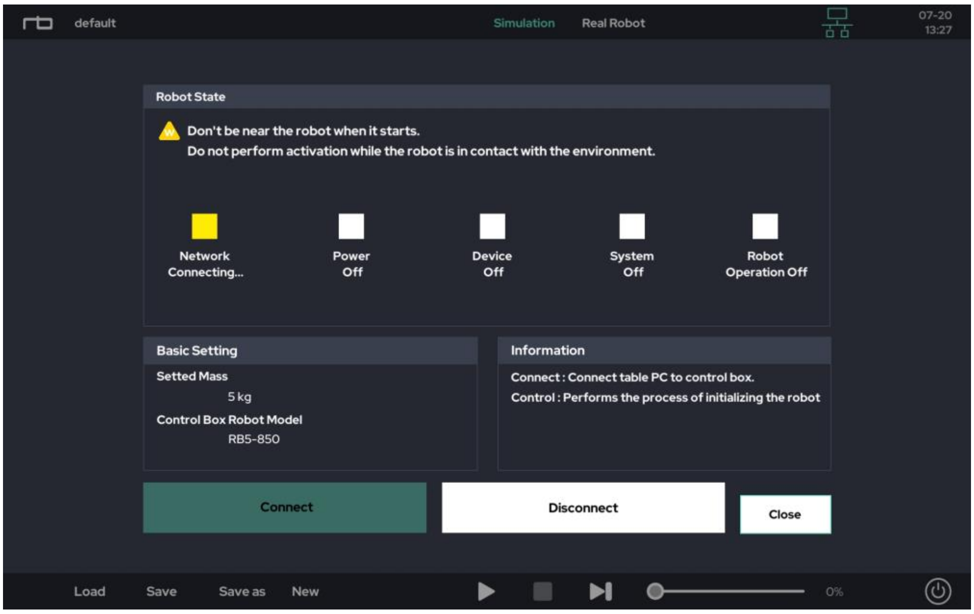

Press the 'Connect' button to link the tablet PC to the robot control box.

- 'Connect' button: Will connects the tablet PC to the robot control box.


Caution:

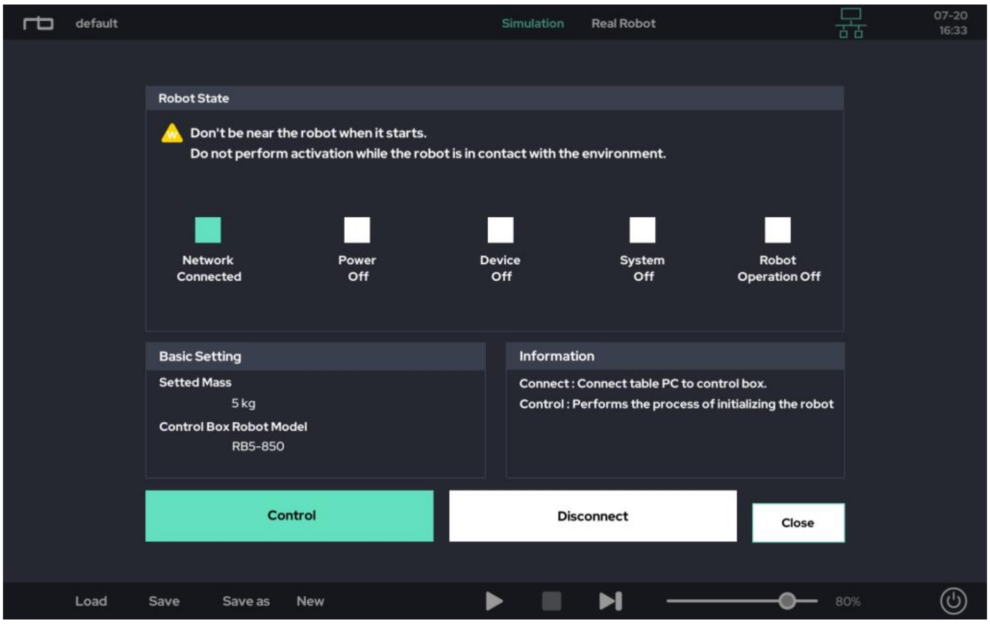
1) Make sure that the control box is turned on and that the emergency stop switch is turned off. If the control box is not on, the light beneath 'Device Off' will turn red.

The figure below shows a display when the tablet PC and control box are being connected.




'Network Connecting' lights yellow when the tablet PC is trying to connect to the control box.

'Network Connected' becomes blue when the table PC and control box are connected properly. The 'Control' button is also activated once more.

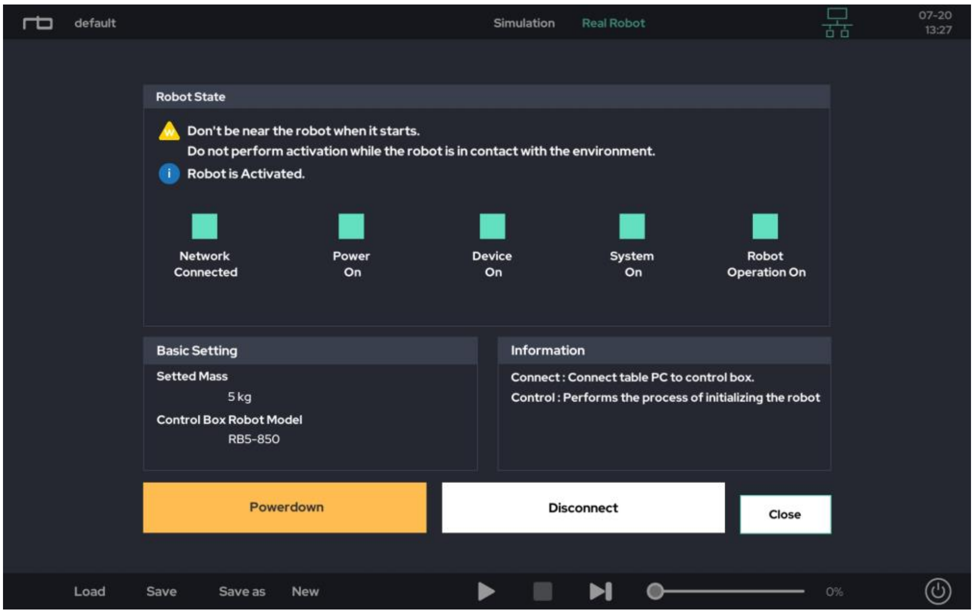


After ‘Network Connected’, press the ‘Control’ button to activate the robot control system.

- ‘Control’ button: Will initialize the robot arm for operation.

During initialization, the mechanical joint brake is released. Unlocking the joints will generate a clicking sound.

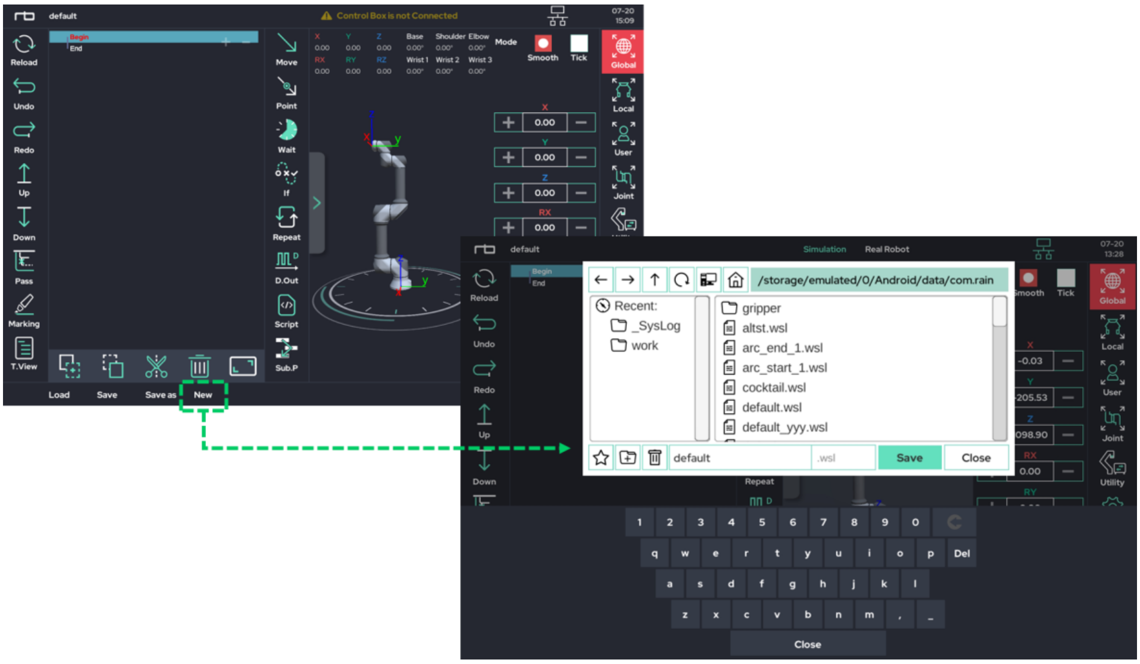
All lights become blue when the robot is ready.




※ When ‘Robot Operation On’ is still in red, please follow the instructions contained within the message popup.

■ Create New Project

Press the ‘New’ button at the bottom of the screen to create a new project and can give the file a name.

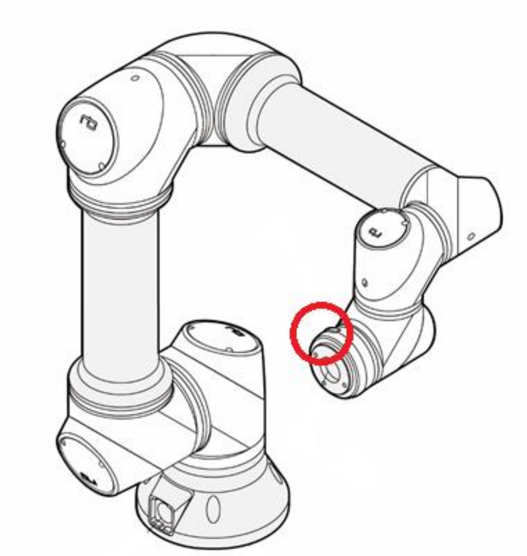



The default name of a new project is ‘default’. Please type a name for the new project and press the ‘Save’ button in the dialog. Note that the new project is not created if the ‘Save’ button is not clicked.

## 7.3 Teaching (Programming)

- Ways to Move the Robot
  - Direct-Teaching: When a user manually rotates each joint to change the pose of the robot.
  - Jogging: When a user uses the jog buttons in the UI to move the robot.

- Direct-teaching



The ‘Gravity Compensation’ algorithm allows the robot to keep its pose when set by a user. For ‘Direct-Teaching’, a user must press and hold the mechanical button located on the tool flange. Pressing this button allows each joint to move freely. The red circle in the figure above indicates the location of the button.

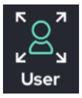
**Warning**

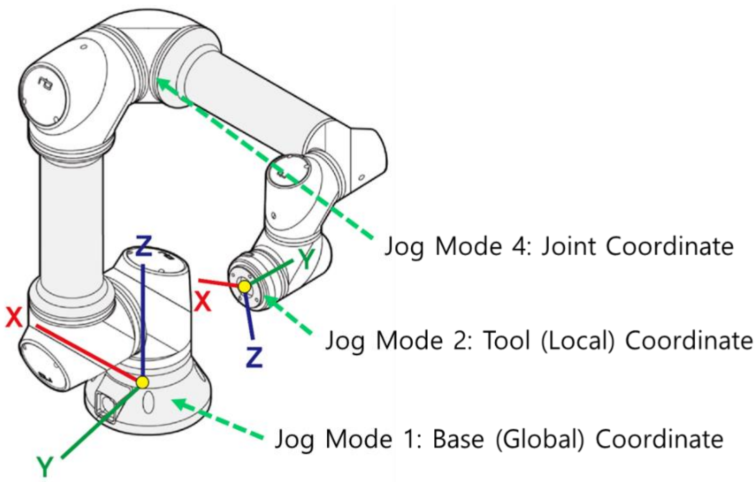
Warning:

1) ‘Direct-Teaching’ can be used only when the robot is initialized and activated.
2) The load value in ‘Setup-Tool’ should be set prior to using ‘Direct-Teaching’ when a tool is installed at the tool flange. Without a proper value of the load, ‘Direct-Teaching’ may not work properly.
3) In ‘Setup-Interface’, the sensitivity of joint reaction can be adjusted.
4) Please ensure that the robot is not moving before using ‘Direct-Teaching’.

■ Jogging


There are four modes of jogging.

-  Mode 1: TCP Movement in the Cartesian coordinate system with respect to the base (global) frame.
-  Mode 2: TCP Movement in the Cartesian coordinate system with respect to the tool (local) frame.
-  Mode 3: TCP Movement in the Cartesian coordinate system with respect to the user coordinate frame.
-  Mode 4: Angular joint movement.

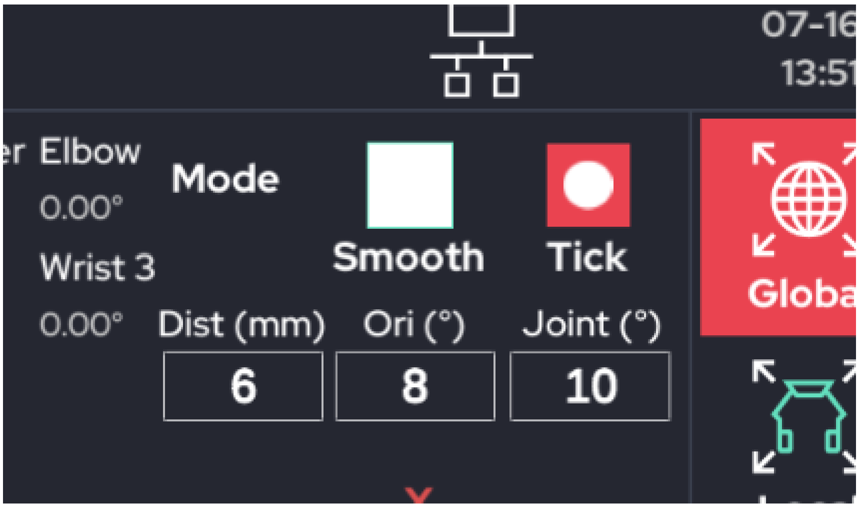


There two ways to control jogging:

- Smooth: Use for continuous motion of the robot. When the '+' or the '–' button is pressed and held, the robot moves continuously until the button is released.
- Tick: Use for discontinuous motion of the robot. The robot will move a specific amount as defined by the user each button click.

※ The control method for jogging can be selected via a toggle button located in the top right in 'Make' screen.

※ In 'Setup-Interface', a user can specify the amount of movement for each 'Tick' button press. Or it can be directly changed in pop-up window as below.

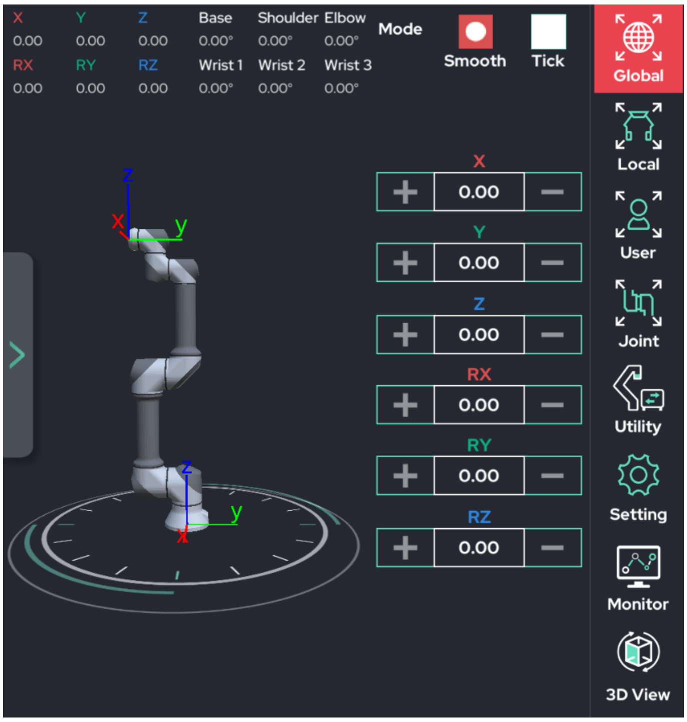



## Warning

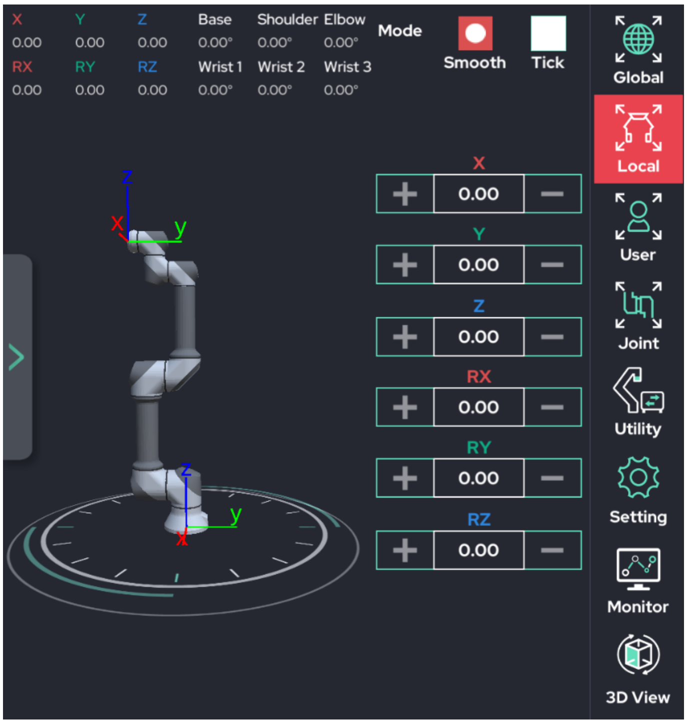
Warning:

1) Make sure that there are no obstacles or people in the robot's workspace before the use of jogging.
2) It is highly recommended to use the 'Safety Slider' feature in 'Setup-Interface'. This feature is activated as a factory default.

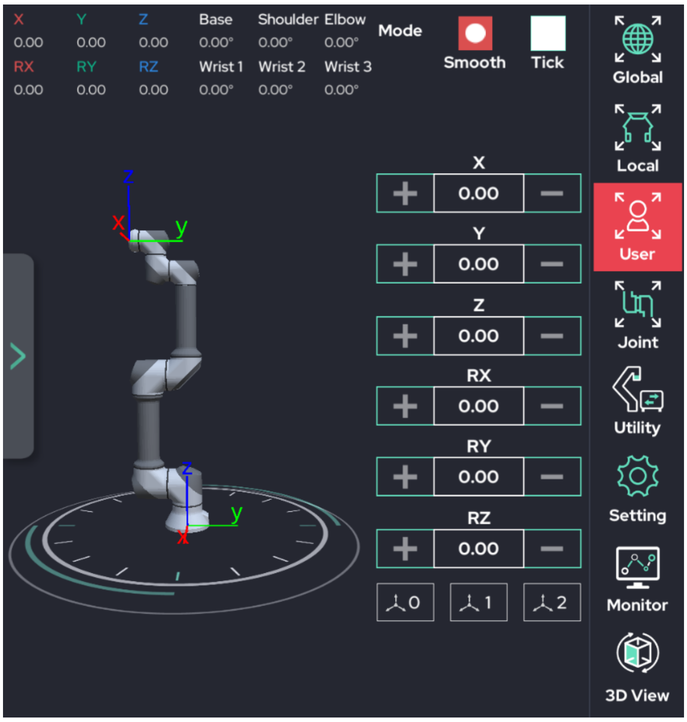
<Jog Mode 1: TCP jog w/ Global coordinate>



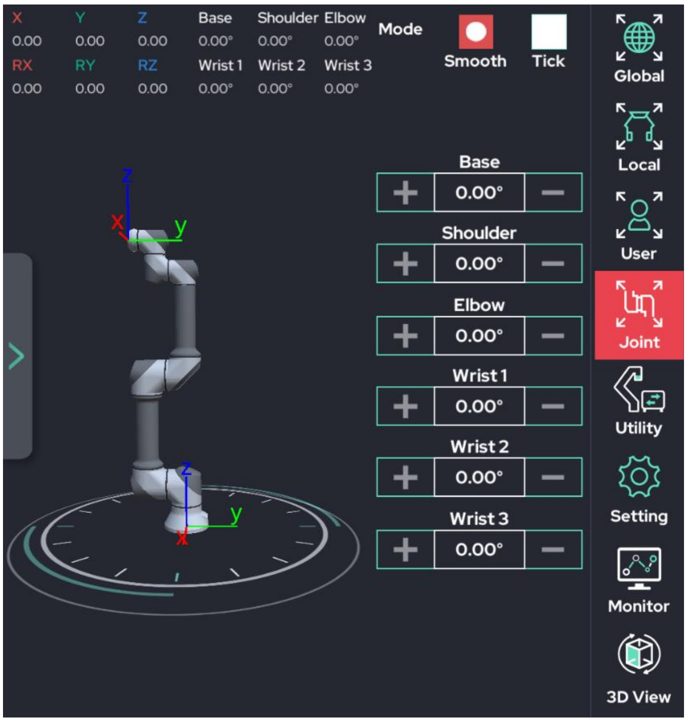

< Jog Mode 2: TCP jog w/ Local coordinate>



< Jog Mode 3: TCP jog w/ User coordinate>

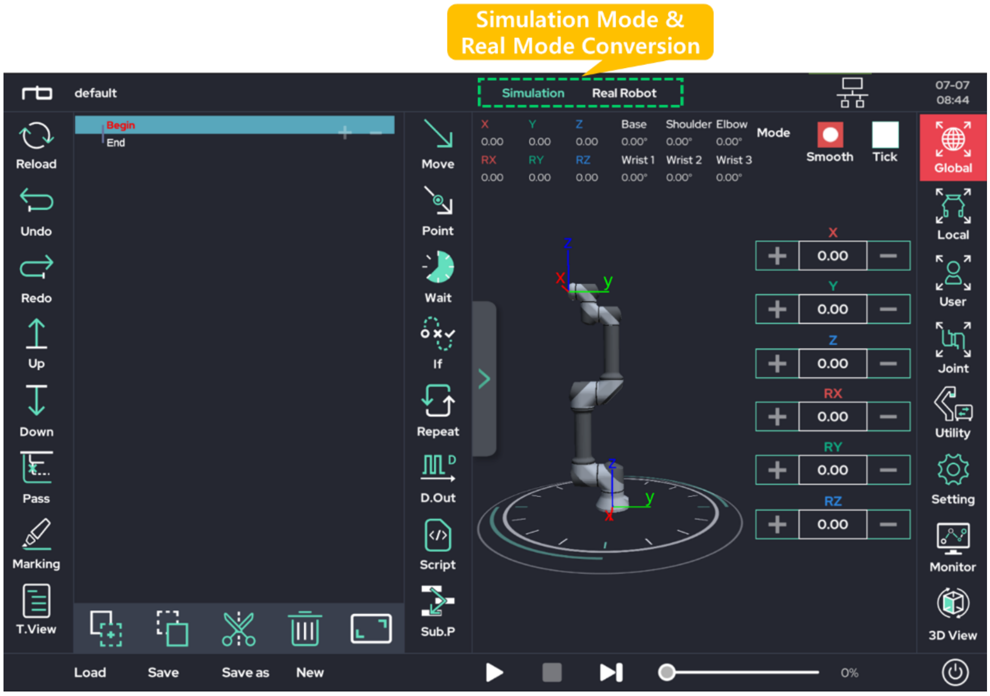



< Jog Mode 4: Joint jog w/ joint coordinates>



■ Real Robot and Simulation Modes

Two Modes are available for testing the robot's movement.




- Simulation Mode:

  Allows the user to virtually move the robot arm on the UI screen without moving the actual robot.

  It is recommended to run simulation mode first for safety reasons before teaching a new motion.

- Real Robot Mode:

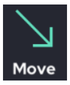
  Drives the real robot as displayed on the UI screen.

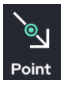
**Warning**

Warning:

1) Real Robot mode is only available when the robot is connected and activated.
2) Simulation mode only requires the provided tablet and the control box. It does not require the robot arm.
3) When using Real Robot mode, please make sure that the nearby environment is clear & safe before operating, as the robot will move.

■ Teaching Robot Movement

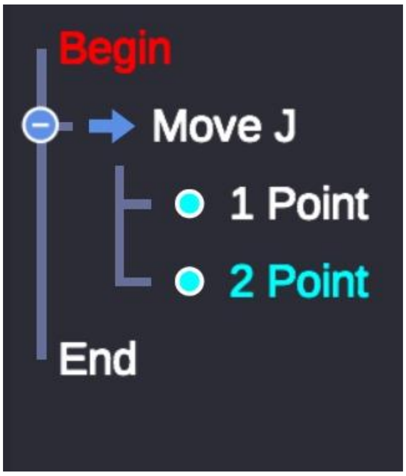
The basic robot teaching functions are 

Move and 

Point. Both icons are on the top bar when using the Make screen.

- Move: Defines motion property. Generates a movement command for the robot arm. Requires points to be defined.
- Point: A sub-function of Move. Defines a destination position for each movement.

After using the Move and Point functions in an empty program, the script field in the UI will look like the following.



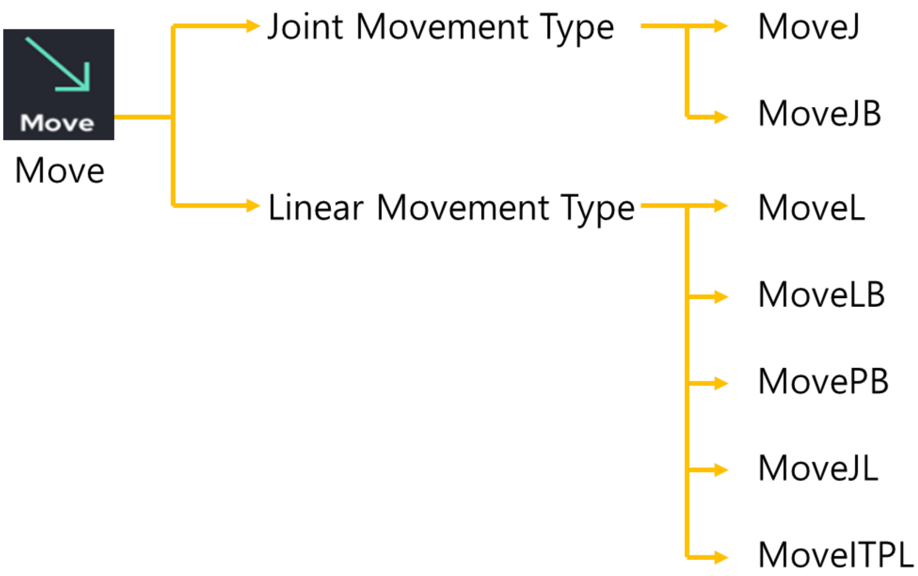

← motion property

← 1st destination

← 2nd destination

Details on each of the Move and Point functions follow on the pages below.

■ Move Function




**Move** sets the robot arm's motion properties. The two primary types of movements are **Joint** and **Linear**. These types are further broken down into commands, as shown in the figure above.

## ■ Joint Movement Commands

The Joint Movement Commands generate movement by setting the angular value of each individual joint (in degrees).

### ▷ MoveJ (Move Joint) :

Sets each joint angle to the values contained within the target **Point**. Note: The movement speeds for all joints are slowed relative to the joint that requires the most movement time.

### ▷ MoveJB (Move Joint Blend) :

Starting from the initial arm configuration, the arm will move smoothly between each **Point** without stopping by using the Move J method.

## ■ Linear Movement Commands

The Linear Movement Commands generate movement by setting the position of the TCP in the Cartesian coordinate system. These commands use Cartesian coordinates (x,y,z coordinate values and rotations) as the target values for the movement.

### ▷ MoveL (Move Linear) :

Moves the TCP linearly (using x, y, and z) from the current position to the position contained within the target **Point** (in mm). Will also rotate the TCP (using Rx, Ry, and Rz) based on the orientation contained within the target **Point** (in degrees).

### ▷ MoveLB (Move Linear Blend) :

Starting from the initial arm configuration, the arm will move smoothly between each **Point** without stopping by using the Move L method. This method will generate an arc-shaped path.

For each **Point**, the user must specify a Blend Radius. This Blend Radius determines how far away the TCP will be from the **Point** when moving along the path.

If the Blend Radius is set to 0, the path will be the same as only using the Move L method.

The Blend Radius has a maximum value, which is half of the distance between the initial **Point** and the destination **Point**. This ensures that the arm will maintain a blended movement.

Move LB has two modes, Constant and Intended.

- Constant mode maintains the first **Point's** TCP orientation (Rx, Ry, and Rz) during movement, only changing the tool's position (x, y, and z) through the movements.
- Intended mode changes both the orientation and position the TCP as the arm moves.

### ▷ MovePB (Move Point Blend) :

MovePB is similar to MoveLB, but it is more universally available. For each **Point**, the user can set the blend amount in either distance or percentage (%). The speed can also be set separately for each point.

▷ MoveJL (Move J with Linear Input) :

Like MoveL, the Cartesian value of the target point is used as input. However, instead of going straight to the point, it uses MoveJ's method. When the Cartesian coordinate system input is received, it is converted into the target joint angle through inverse kinematics and inputted again to MoveJ.

▷ MoveITPL (Move Interpolation) :

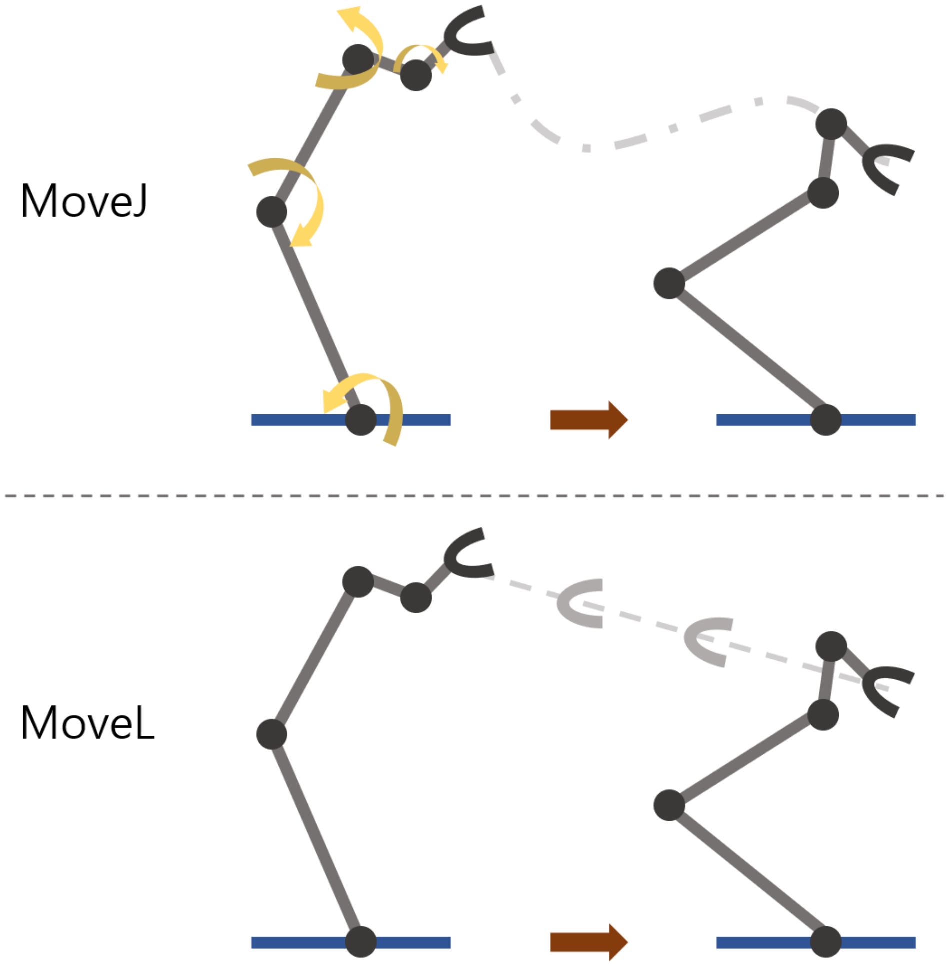
Starting from the starting point (the current position), move smoothly between the points without any stops using the Move L method.

MoveLB or PB blends across (blend) each waypoint, but MoveITPL moves along the trajectory exactly past each waypoint. So, there is no separate blend setting.

MoveITPL has two modes. Constant mode is to move the tool orientation while maintaining the starting point value. Intended mode is to change the orientation of each tool.

The speed can be set separately for each intermediate waypoint.

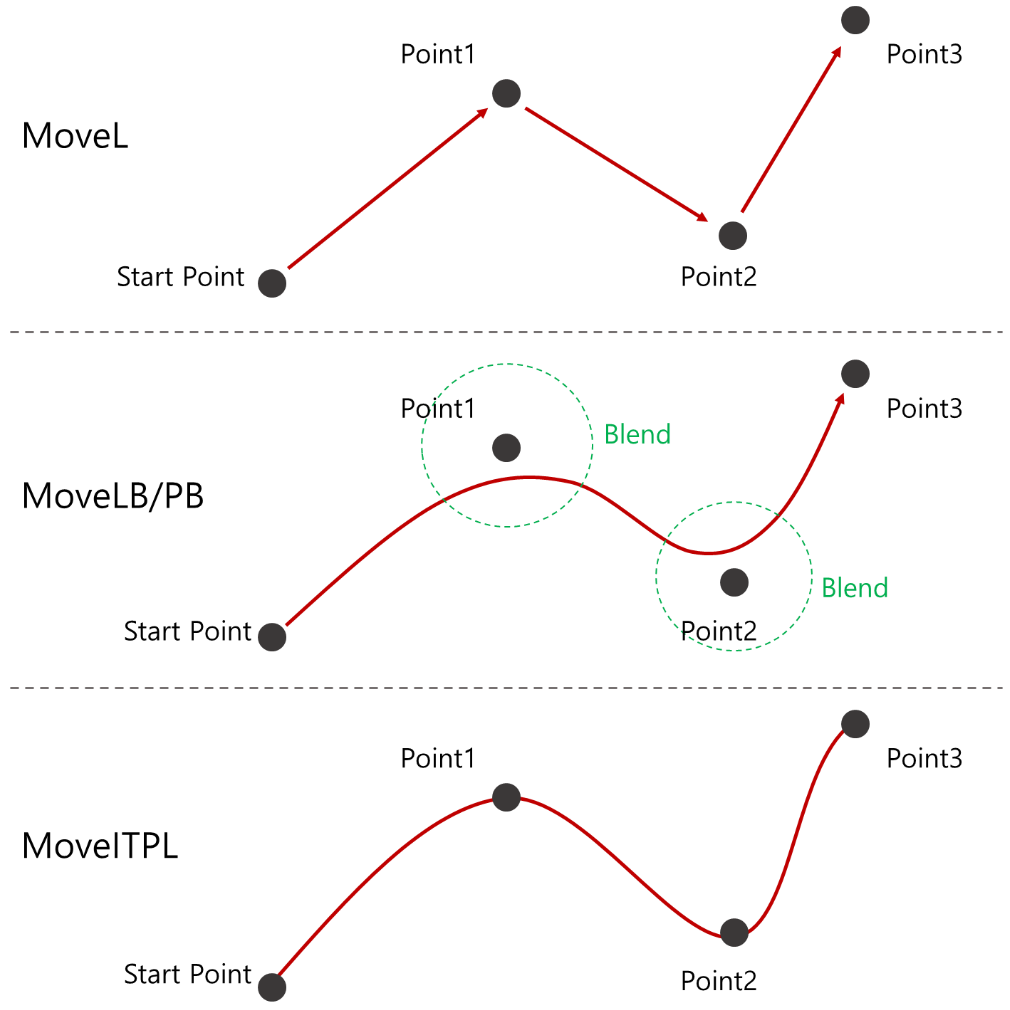
■ Difference between **MoveJ** and **MoveL**




MoveJ does not consider the movement trajectory of the terminal (TCP). It is an operation that only uses the joint angle information of the starting point and the joint angle of the target point. The driving speed of other joints are adjusted to the joints that require the most driving time.

MoveL is a mode that uses inverse kinematics to move the trajectory of the terminal (TCP) linearly from the starting point to the target point. 6 Cartesian coordinate values (x, y, z, Rx, Ry, Rz) are the inputs for the target point value.

■ Difference between **MoveL**, **MoveLB/PB**, and **MoveITPL**




MoveL moves in a straight, linear path between the start and destination points. The arm will arrive at each sequential arrival Point, stop, and then continue to the next Point.

MoveLB/PB starts at the initial Point, uses each intermediate Point as a waypoint, then stops at the final Point. The arm will not stop at the specified waypoints. Instead, it will arc around each point according to the blend distance, and then continue without stopping.

MoveITPL, the points other than the arrival point move to the waypoint, creating a trajectory that passes exactly through the waypoint. The trajectory is created without stopping and a separate speed setting is possible for each waypoint.

**Warning**

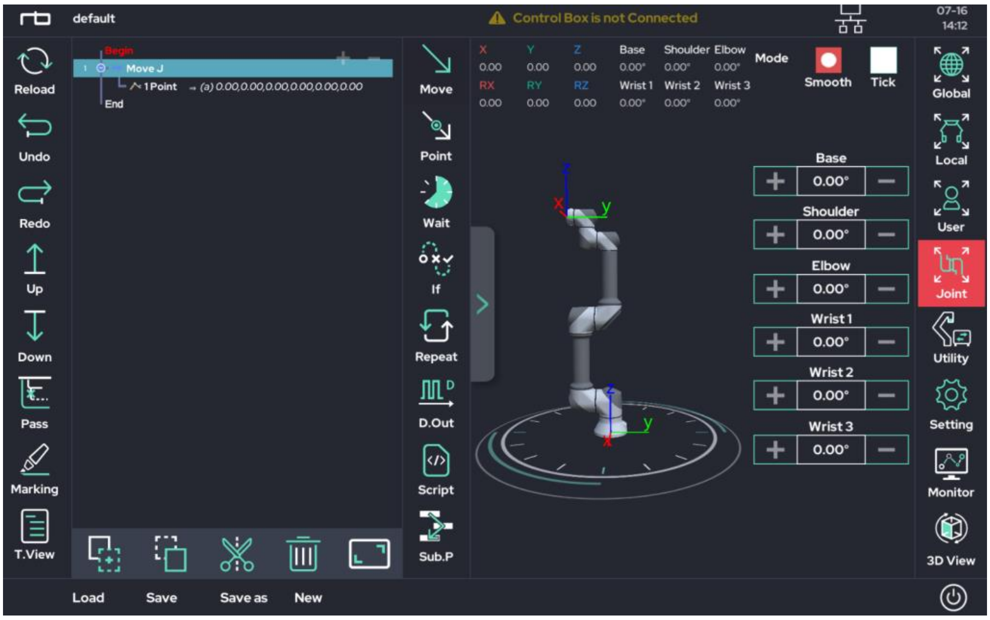
Warning:

1) The five linear motion commands (MoveL, MoveLB, MovePB, MoveJL, MoveITPL) move the robot using inverse kinematics calculations. Therefore, movement may be limited in singularity positions where inverse kinematics calculations are not possible.

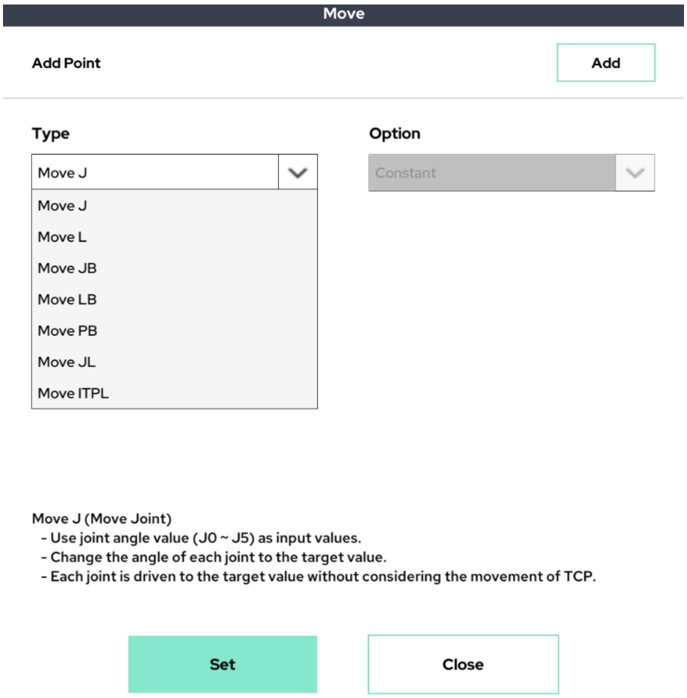
2) Certain joints may move faster or be restricted in motion while in the dead zone of the robot. Further information about dead zones can be found in Section 1.7.

■ Changing Move Function Commands

When the Move Function is used for the first time in a program, the program tree will be created as shown below. By default, the Move function is set to MoveJ.

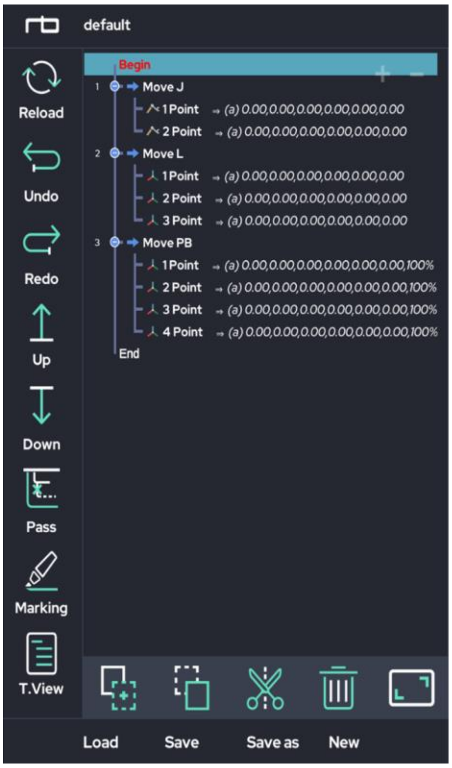



Click MoveJ in order to change the Move command type. A popup will appear as shown below.




Select the desired movement type and click close to change the movement type.

An example of a teaching program is shown below.




Program Start
  Movement (Property: Move Joint)
    - Destination 1 : Target Value is Joint Angle
    - Destination 2 : Target Value is Joint Angle
  Movement (Property : Move Linear)
    - Destination 1 : Target Value is Coordinate Value
    - Destination 2 : Target Value is Coordinate Value
    - Destination 3 : Target Value is Coordinate Value
  Movement (Property : Move Point Blend)
    - ViaPoint 1 : Target Value is Coordinate Value
    - ViaPoint 2 : Target Value is Coordinate Value
    - ViaPoint 3 : Target Value is Coordinate Value
    - Destination : Target Value is Coordinate Value
Program End

### MoveJ, MoveJB

The arm moves to the joint angle configuration contained within each Point. Each angle value is relative to the base position.

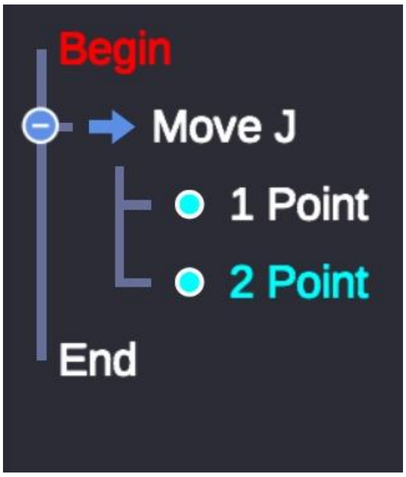
※ Since the robot arm consists of six joints, the MoveJ and MoveJB functions will move all six joints based on the configuration contained within each Point.

### MoveL, MoveLB, MovePB, MoveJL, MoveITPL

The arm moves relative to or directly to a target TCP position contained within each Point. Each Point determines a target location within the Cartesian coordinate system for the TCP to pass through.

※ Since the Cartesian coordinate system consists of six values (x, y, z, Rx, Ry, Rz), all six values will need to be set as subitems of MoveL, MoveLB, MovePB, MoveJL, and MoveITPL.

■ Point Function




← Motion property

← 1st Destination

← 2nd Destination

As explained earlier, the Point function is a sub-function of the Move function. Move specifies the properties of the motion, whereas Point is responsible for setting the target position.

Note: In the Point function, the target value will vary depending on the command type of the Move function.

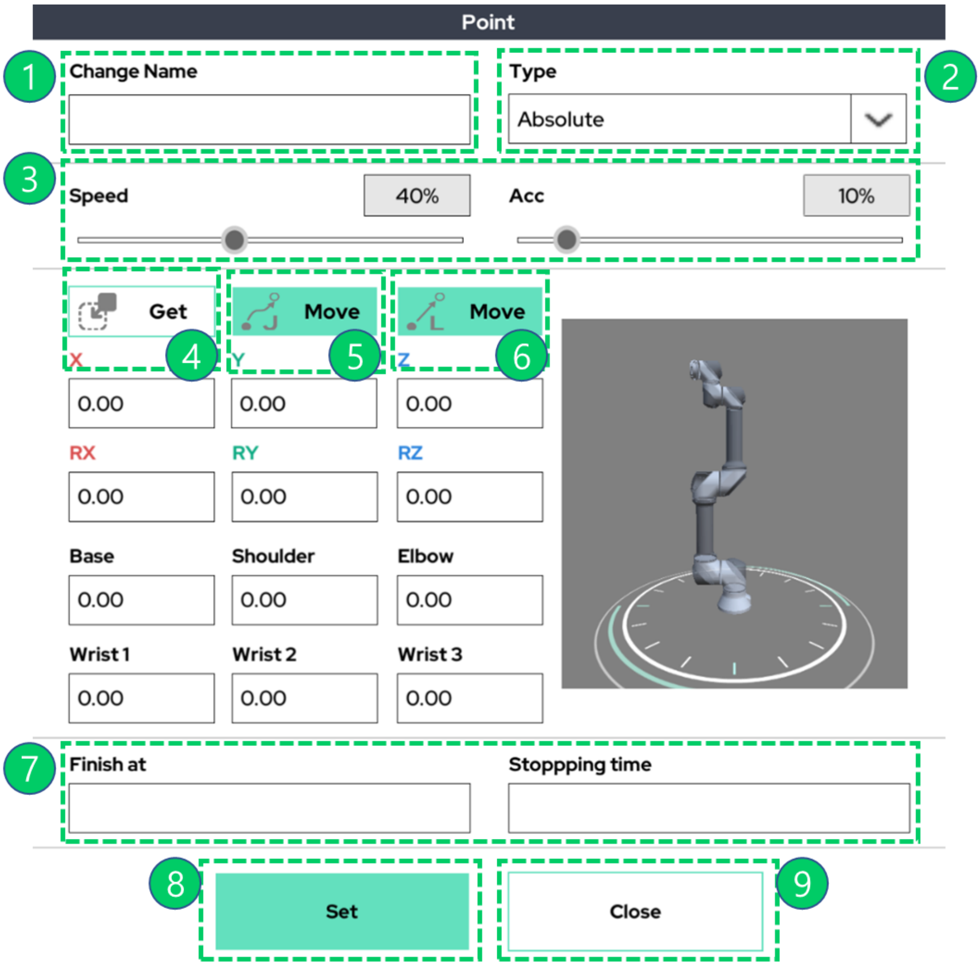
▶ Joint Movement Type(MoveJ, MoveJB) Point :

Contains the target joint angle values (in degrees) for all six joints

▶ Linear Movement Type(MoveL, MoveLB, MovePB, MoveJL, MoveITPL) Point :

Contains the target destination point (in Cartesian coordinates) for the TCP.

When a user taps on a Point in the program tree, the Point function popup window will appear. The window contains the following fields:

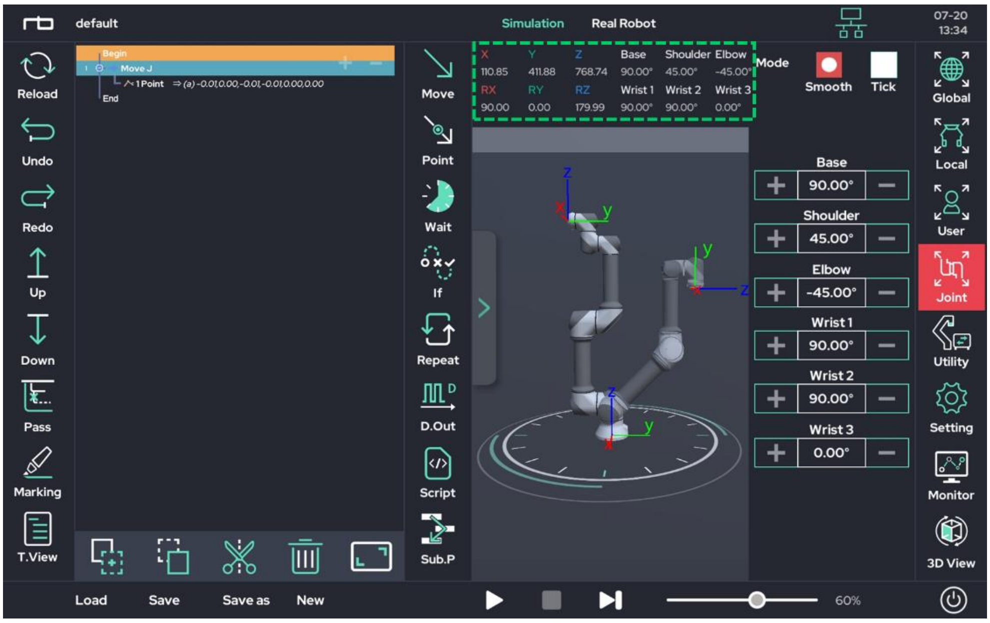



Each area is described in the table below.

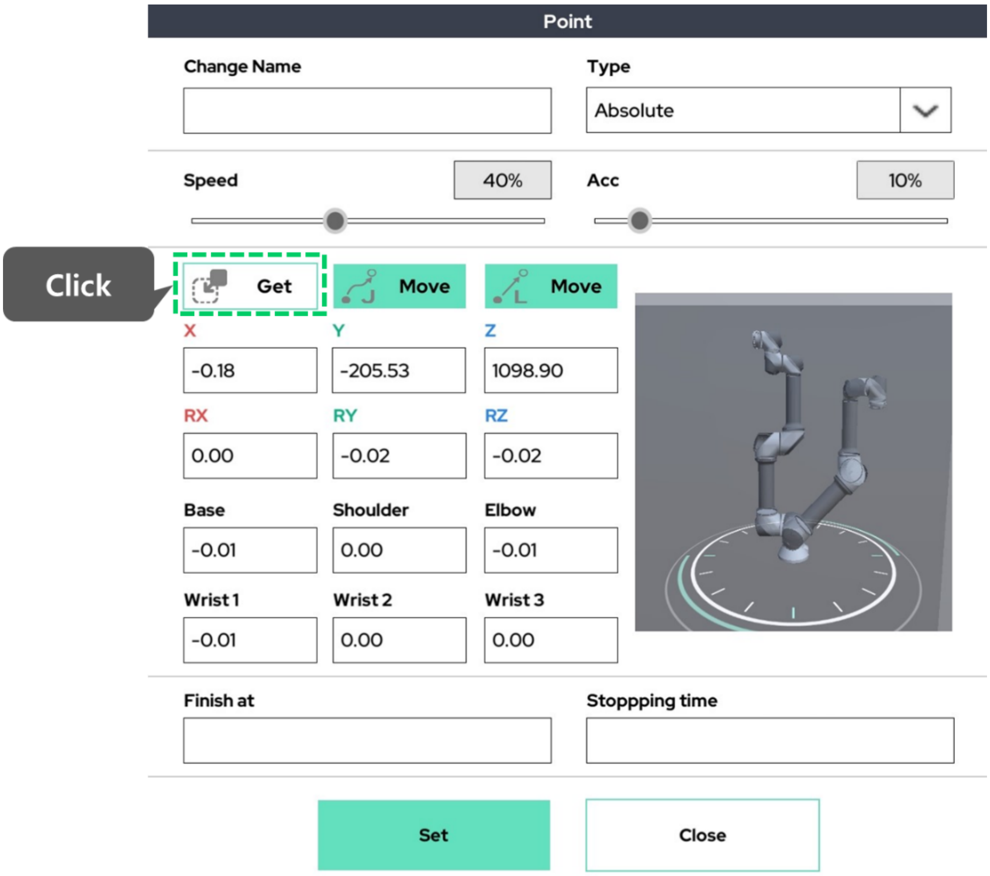
| Section | Description |
|---|---|
| ① | Sets the name of the point (not required). After setting the name, the location information of the point can be used as a variable later. |
| ② | Allows a user to select the setting type of the point function.<br>● The Joint Move has three setting options.<br>● The Linear Move has four setting options.<br>● The default type when creating a Point is 'Absolute'. |
| ③ | Sets the speed and acceleration of arm movements to the location |
| ④ | Updates the Point information with the current robot position.<br>● After moving the robot to the desired position/posture, press Get to store the information.<br>● To save the Point at the current position/posture, press the Set button (Section 7).<br>Depending on the Point type (Section 2), the Get button may or may not supported. |
| ⑤ | Moves the arm to the specified Point.<br>● Must hold down the button to move the arm to the saved position. Note: the movement is a joint movement type.<br>● When the movement is completed, a pop-up message will be shown.<br>Depending on the Point type (Section 2), button may or may not supported. |
| ⑥ | Moves the arm to the specified Point.<br>● Must hold down the button to move the arm to the saved position. Note: the movement is a linear movement type.<br>● When the movement is completed, a pop-up message will be shown.<br>Depending on the Point type (Section 2), button may or may not supported. |
| ⑦ | Specify an escape condition (Finish At) and an escape time (Stopping Time) for the action. Not a required input.<br>● If the input is left blank, the operation will end normally after reaching the target point.<br>● Once the escape condition is satisfied, the operation stops according to the escape time and continues to the next action.<br>The minimum escape time is 0 seconds. |
| ⑧ | Saves the changed settings. |
| ⑨ | Closes the Settings window. Will not save user input without pressing the Set button (Section 7). |

※ An example using the Get function (Section 4) is shown below.

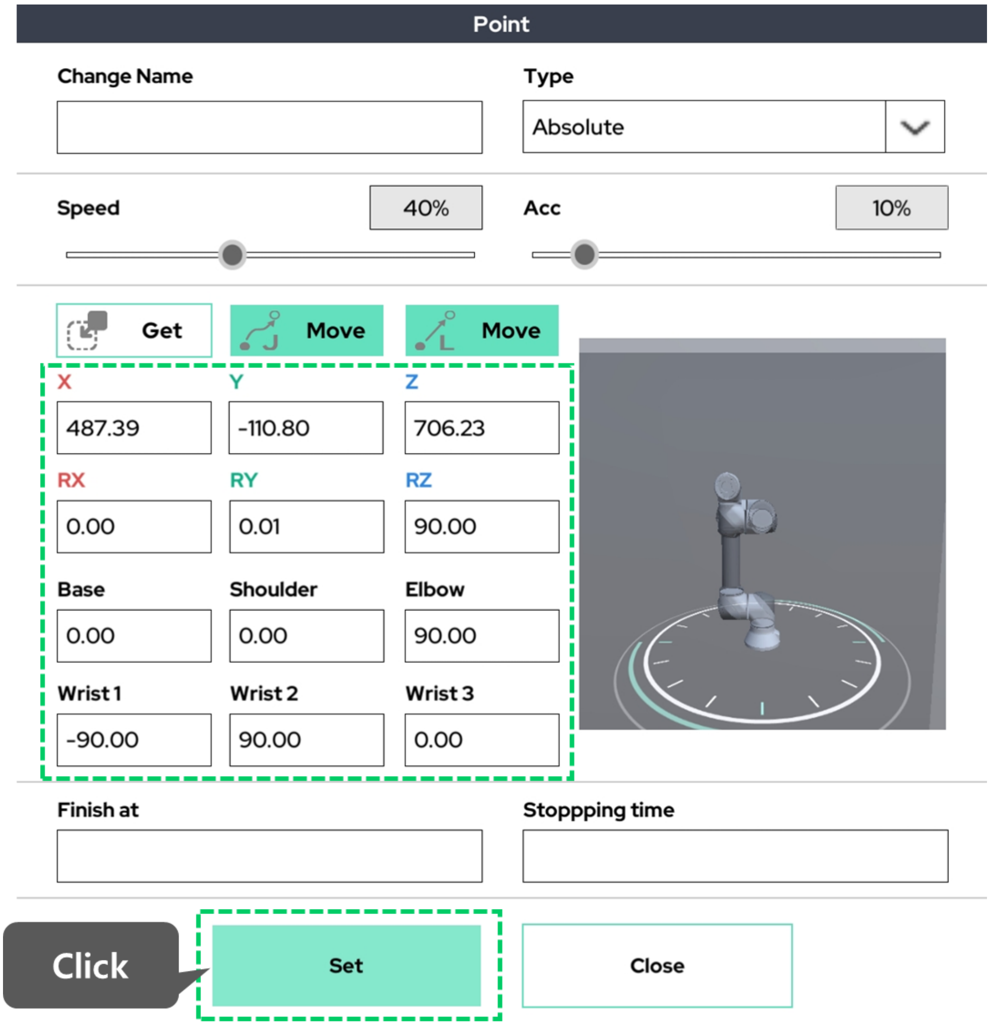
1. Use the jog / direct teach function to move to the desired posture / position



2. Get current posture / location information by pressing Get button



3. Save after confirming reflection

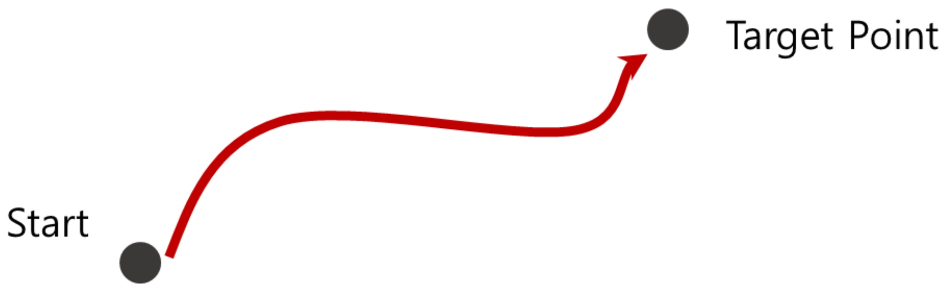


※ An example using the Finish at/Stopping time option (Section 6) is shown below.

■ When not using the Finish at function

(If left blank)

End of motion after arrival to original set target point, execute next command

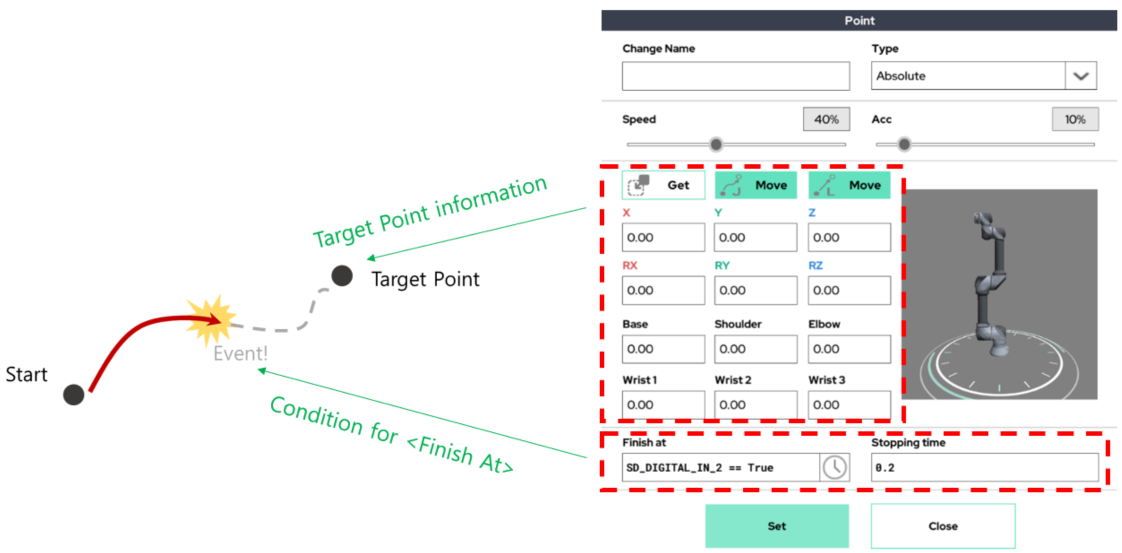



■ When using the Finish at function

(When entering a specific conditional expression)

Even if the target point is not reached, the operation is terminated when the Finish at condition occurs and the next command is executed.

If condition does not occur during operation, execute the next command after reaching the target point normally.




The following setting options exist for each type of move function.

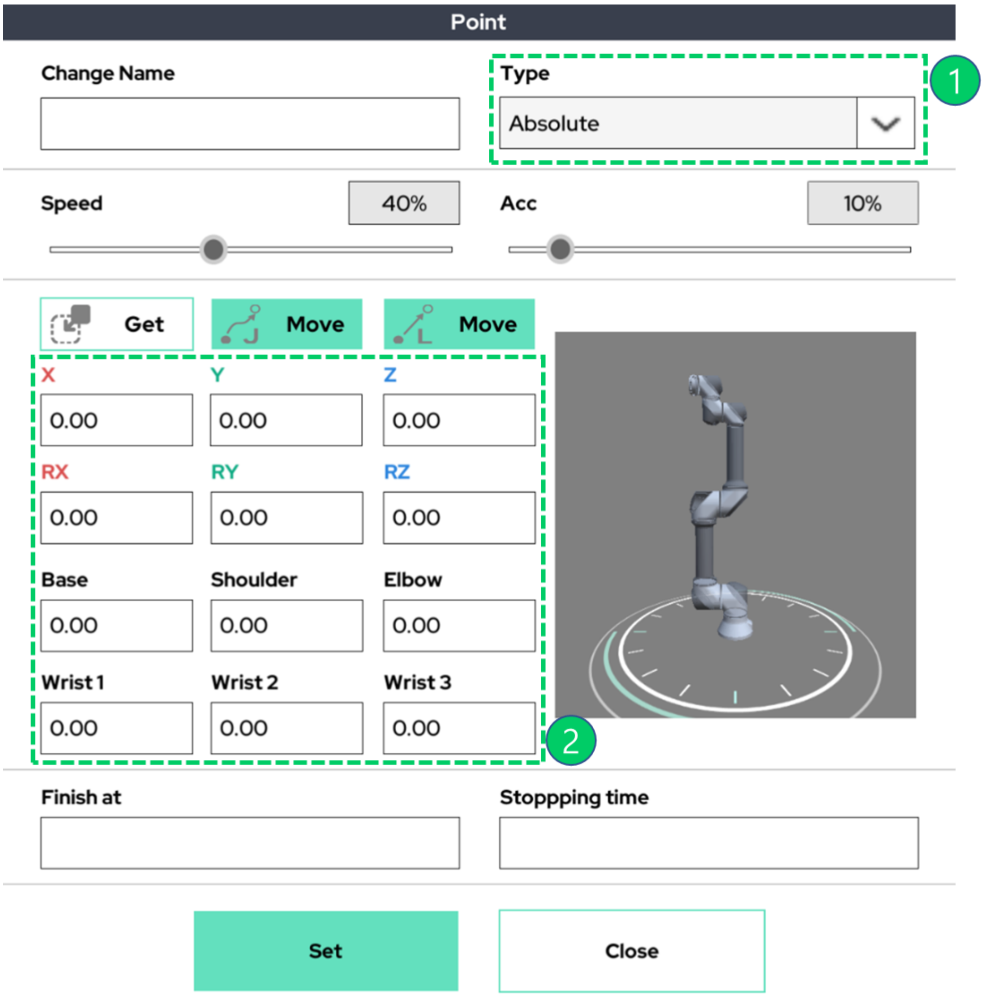
| Joint Move Type's sub Point | | |
|---|---|---|
| Option | Absolute | >Sets the Points for MoveJ by using fixed, user defined joint angles<br><br>>Requires the user to set the desired posture/joint angle configuration through the Get function. |
| | Variable | >Sets the Points for MoveJ by using one of several methods.<br><br>>Allows the user to set the desired posture/joint angle configuration through the Get function.<br><br>>The user can also change a joint angle by setting it to a variable or a mathematical operation. |
| | Relative | >Sets the Points for MoveJ by changing the joint angles relative to the previous angle position.<br><br>>If a joint movement is set to zero, then that joint will not move. If all are set to zero, then the robot will not move.<br><br>>The user can also change a joint angle by entering a variable or mathematical operation. |

>> continue

| Linear Move Type's sub Point | | |
|---|---|---|
| Option | Absolute | >Sets the Points for MoveL by using fixed, user defined Cartesian coordinate values<br><br>>After moving the robot's TCP, Cartesian coordinate values through the Get function can be set<br><br>>Note: The default Cartesian coordinate system for the Absolute Point Type is the base coordinate system of the robot arm (manufacturer's default coordinate system). |
| | Variable | >Sets the Points for MoveL by using one of several methods. target Cartesian coordinate value.<br><br>>Allows the user to set the Points for MoveL by using fixed, user defined Cartesian coordinate values<br><br>>The user can also change the TCP Point by setting it to a variable or a mathematical operation. |
| | Relative | >Sets the Points for MoveL by setting the relative distance / offset from the previous Point.<br><br>>The user can also choose a user defined Reference Point from which to move. The default value is PT_LAST_TCP, which indicates the last arrival point.<br><br>>In Reference Frame, the user can specify which coordinate system use for relative movement. The default value is Frame_Base, which represents the base coordinate system of the robot arm. The user can choose changes to the user coordinate system or the tool's local coordinate system.<br><br>>In addition, the user can set a point by using a variable or a mathematical operation. |
| | User Coordinate | >Similar to Variable, but sets a target point based on a user-defined coordinate system.<br><br>>Allows the user to select the user coordinate system as a reference by setting the Reference Frame.<br><br>>Select the desired reference coordinate system and use the Get function to automatically enter the robot's pose / position information based on the selected coordinate system.<br><br>>For example, if the user's coordinate system 0 is selected and 0 is entered in all Cartesian coordinate values, TCP moves to the origin of the user coordinate system.<br><br>>In addition, the user can set a point by using a variable or a mathematical operation. |

The figure below shows each different type of Point as it displays in the UI.

▶ Joint Type – Absolute point




① Absolute Option point

② The robot's posture/angle value is saved through Get button.

▶ Joint Type – Variable point


① Variable Option point.

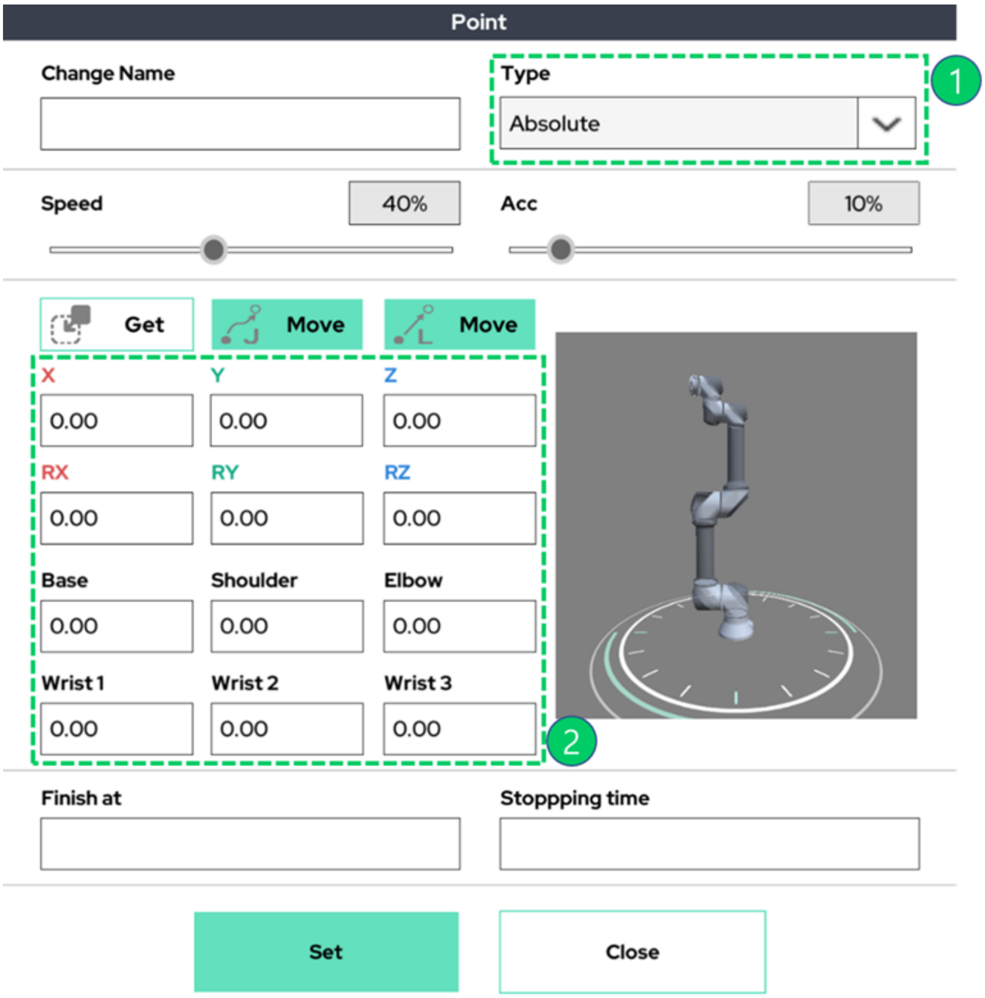
② Allows the user to enter the joint angle for the target posture or enter the parameterized information as an equation.

▶ Joint Type – Relative point


① Relative Option point

② Allows the user to enter how much each joint should move relative to the previous joint angle. All angles are in degrees. In addition, it allows the user to enter parameterized information or formulas.

▶ Linear Type – Absolute point




① Absolute Option point

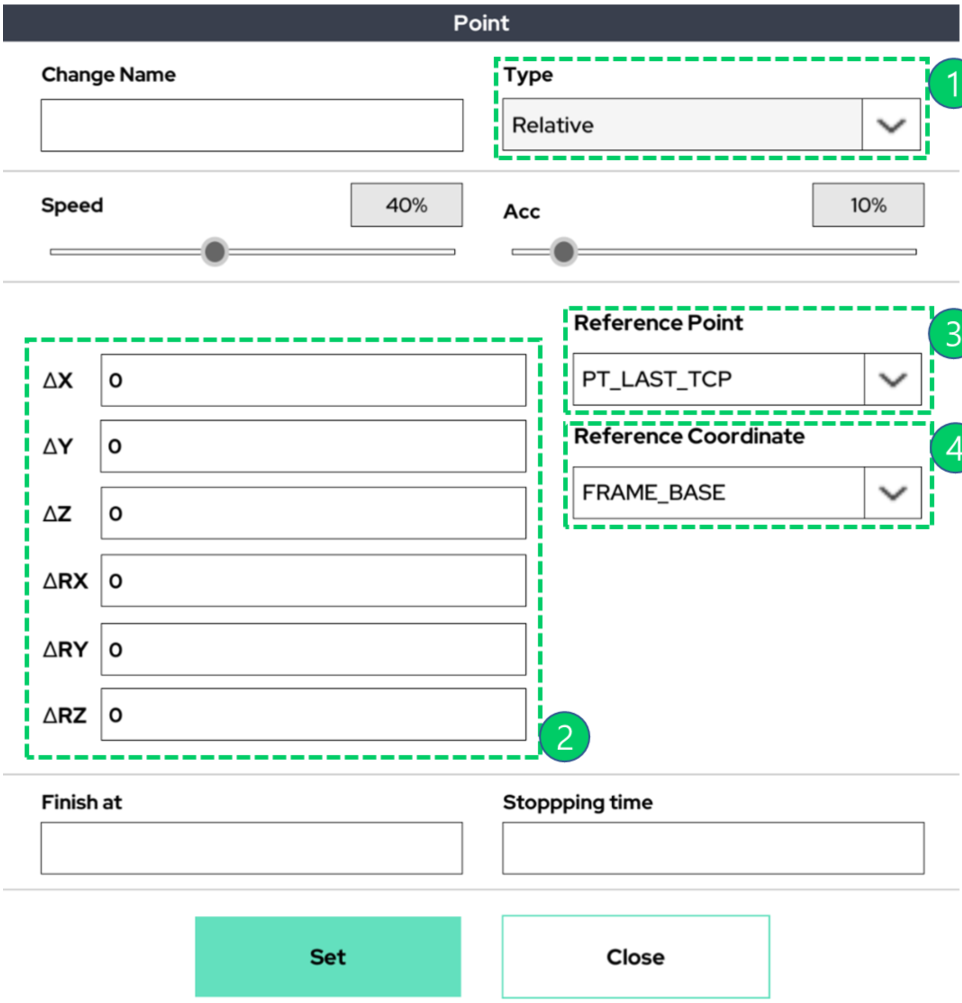
② Allows the user to save a posture/position by using the Get/Save button. The reference coordinate system of the Cartesian coordinate system value is the robot base coordinate system.

▶ Linear Type – Variable point


① Variable Option point

② Allows the user to enter the target Cartesian coordinate values. The user can also enter parameterized information as formulas. The reference coordinate system of the set Cartesian coordinate values is the base coordinate system of the robot arm.

▶ Linear Type – Relative point



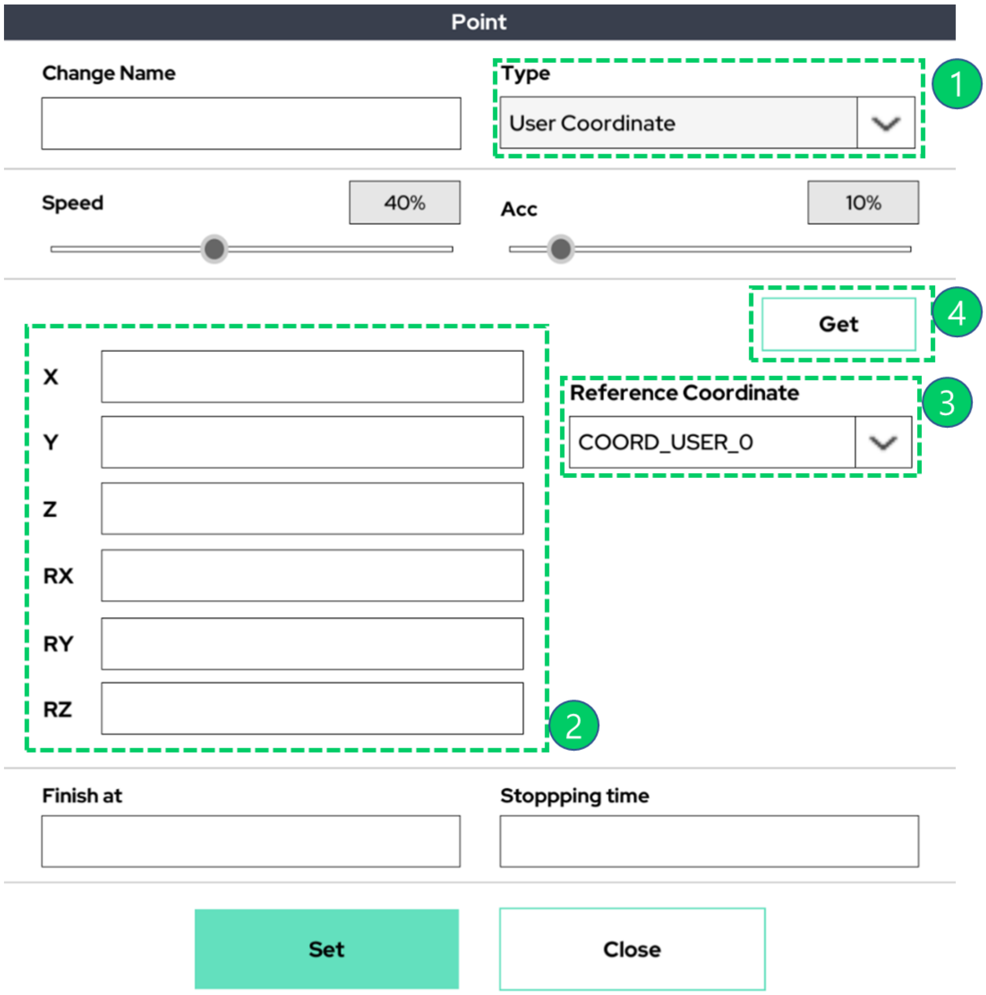

① Relative Option point

② Requires the user to enter the distance/angle offset relative to the reference point. Also allows users to enter variable information.

③ Allows a user to select a user defined point from which to move. The default value is PT_LAST_TCP, which indicates the last arrival point.

④ Chooses a coordinate system to specify relative movement. The default value is Frame_Base, which represents the base coordinate system of the robot arm. The user is also able to choose the user coordinate system or the tool's local coordinate system.

▶ Linear Type – User Coordinate point




① User Coordinate Option point.

② The User Coordinate Option is similar to Variable, but it allows the user to set the target point based on a previously defined user coordinate system. Users can also enter variable information.

③ Selection box for the user coordinate system that the user would like to use as a reference.

④ The Get button will load in the robot's current posture/position information based on the selected coordinate system.

**Warning**

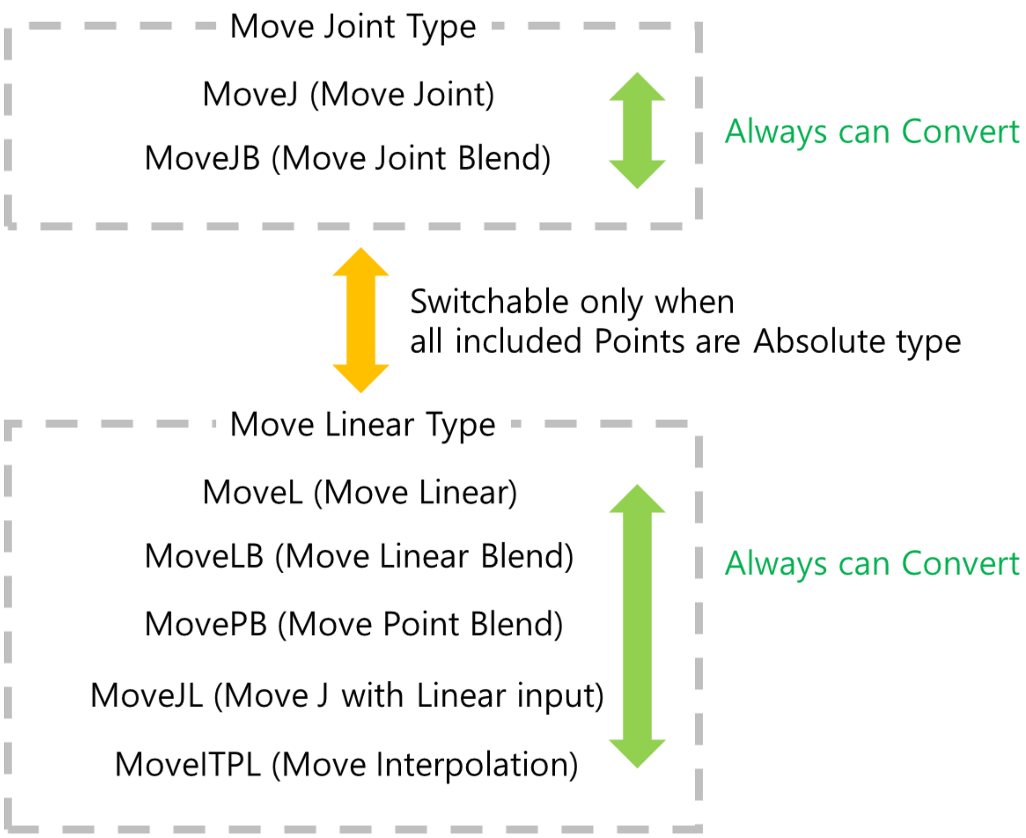
Warning:

1) A user coordinate system can be set through the Coordinate menu in the Setup screen or by using the Setting function in the Make screen.
2) Up to 3 user coordinate systems can be set and used.
3) The factory default user coordinate system is the same coordinate system as the robot base coordinate system.

■ Changing Movement Properties

The following conditions apply when changing the action properties (type of move) of a configured action.

- Switching in the same series can be done without any restrictions.
- Switching to another types (Move Joint types-> Move Linear types / Move Linear types-> Move joint types), can be done only when the type (option) of Point function is used as Absolute.

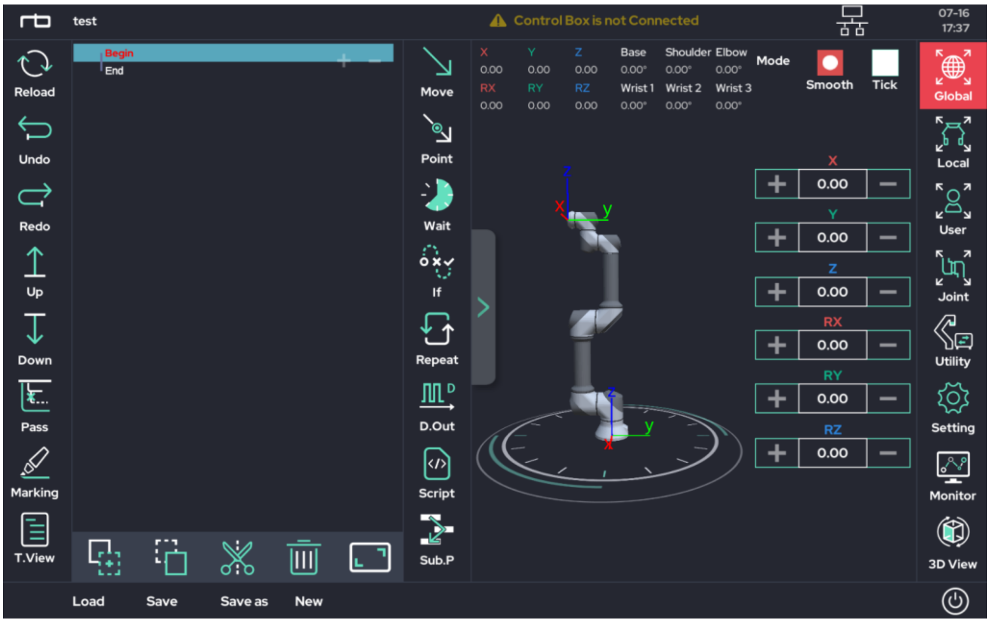


■ Example of Basic Program Creation

The following is an example of creating and running a simple program based on the above Move and Point functions.

[Step 1]

Create a new project. In this case, the name of the project is ‘test’.



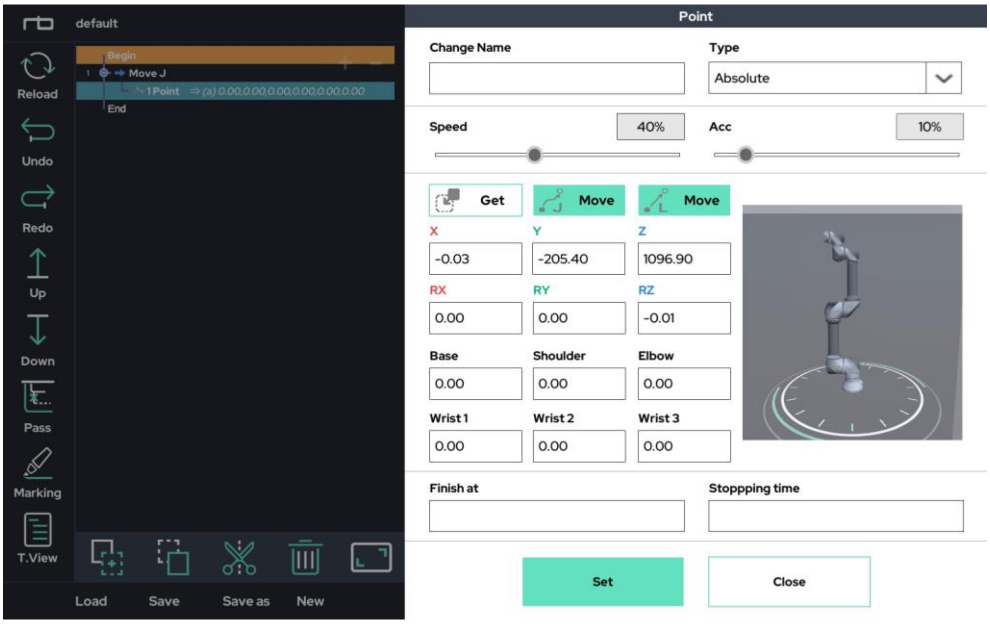

[Step 2]

Click the Move function Move to add a Move command to the program tree. The default command will be MoveJ. A Point function will also be added to the tree as shown below.

[Step 3]

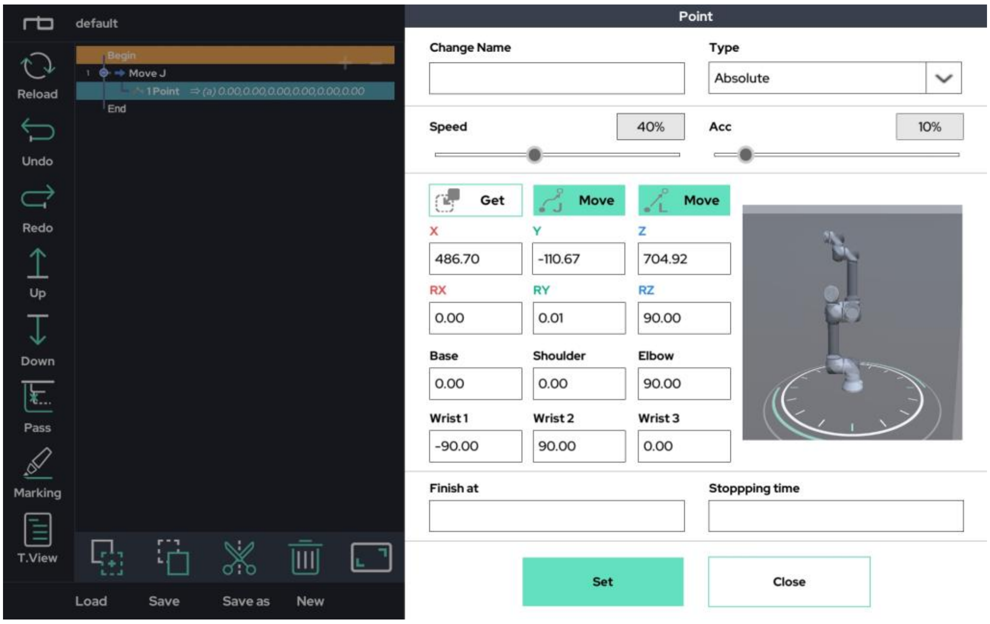
Using the Jog button, move the robot to its intended position. In this example, the robot was moved to the following joint angle: [Base:0', Shoulder:0', Elbow:90', Wrist1:-90', Wrist2:90', Wrist3:0'].

Click on a Point in the program tree to display the Point setting popup window as shown below.



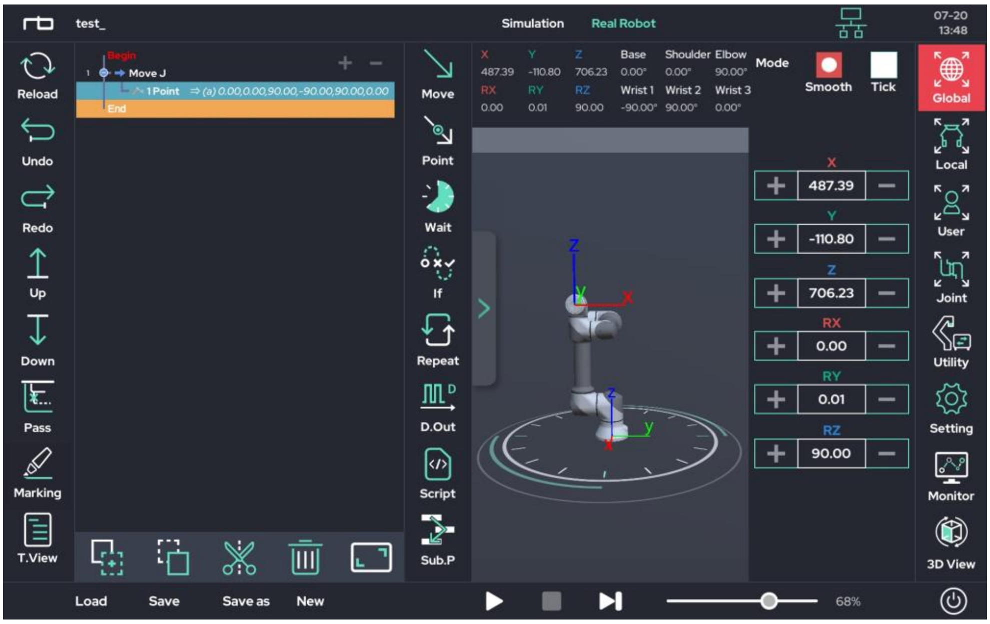

[Step 4]

In the Point popup window, click the Get button to update the fields with the current robot posture/angles. Press Set to save this Point.



[Step 5]

After saving the point, the UI will look as follows.



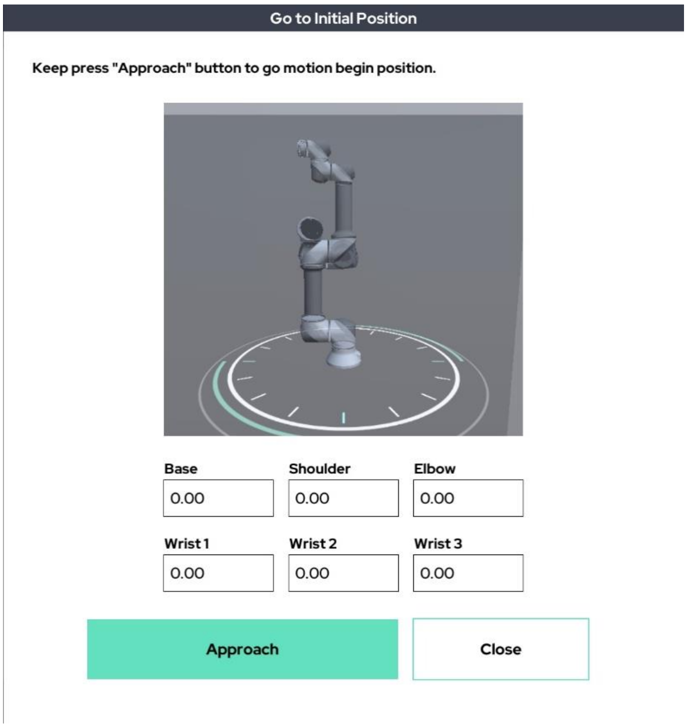

[Step 6]

Repeat steps 1 – 4 several times to teach the robot the desired motion. Our completed example program will look like the following.

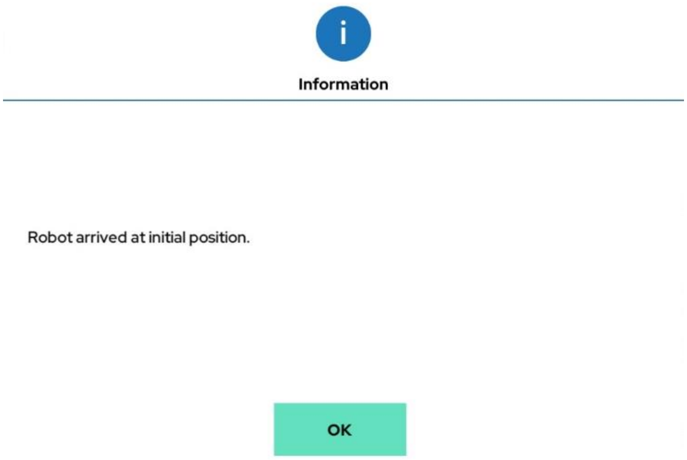
[Step 7]

After the program is finished, run it on the work screen by pressing the play (▷) button. To run the movements using the simulation arm, use the Simulation mode. To run the movements using the real robot arm, use Real Robot mode.

After clicking the play button (▷), the robot will move to its initial position as shown below.

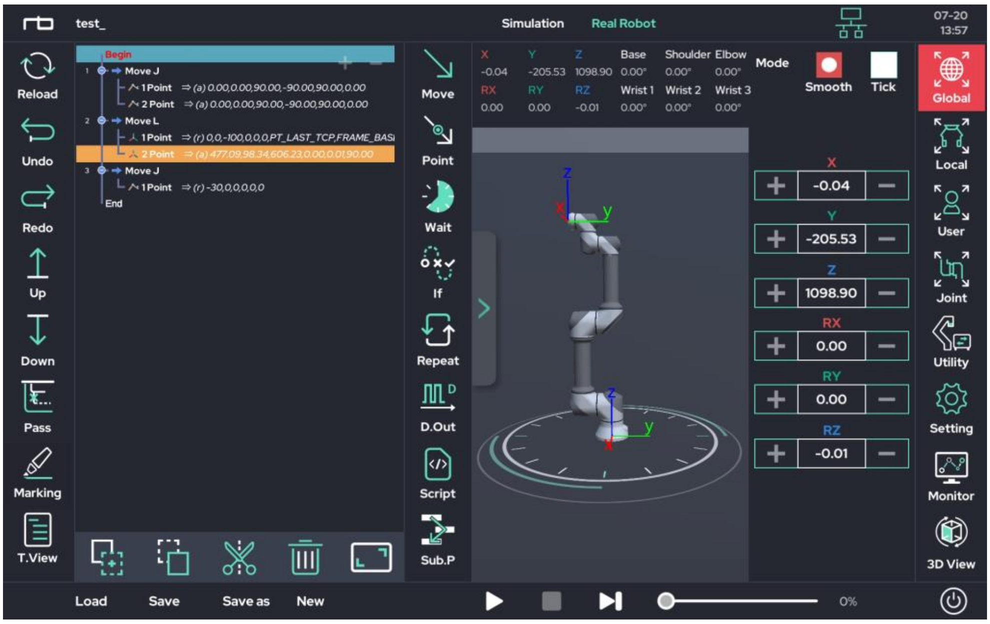



By holding down the Approach button, the robot arm will move to the initial position for the program. Once the robot reaches its starting position, a pop-up message will confirm to the user that the robot has reached its starting position.

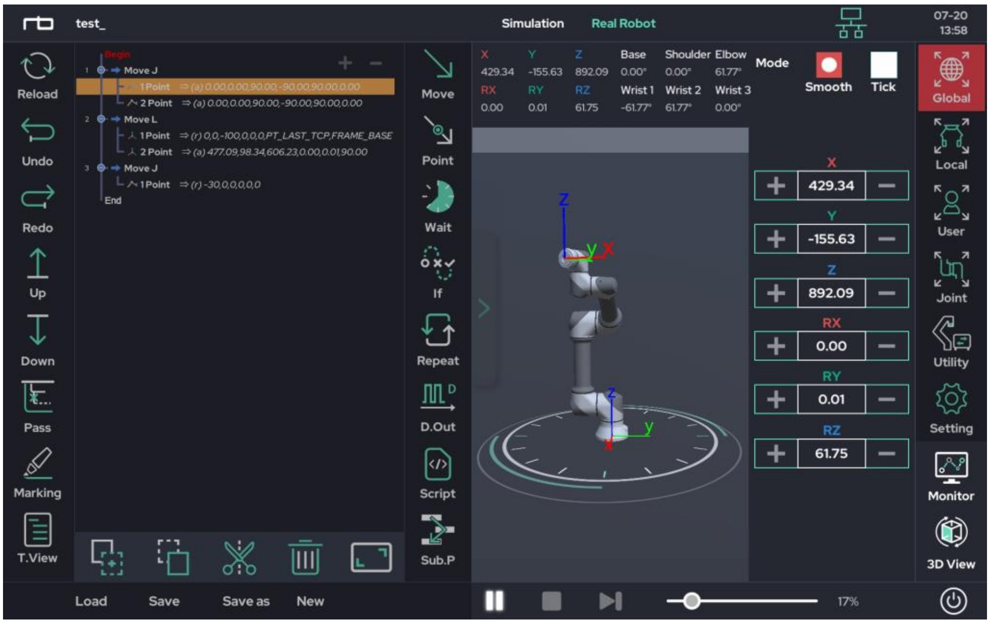


[Step 8]

After receiving the popup in Step 7, the program is ready to run. Click the play button at the bottom again to run the program.




The image below shows the program running.




## Warning

Warning:

1) The Point that the robot is current moving towards will be displayed as yellow in the program tree.

- Initial Movement Position

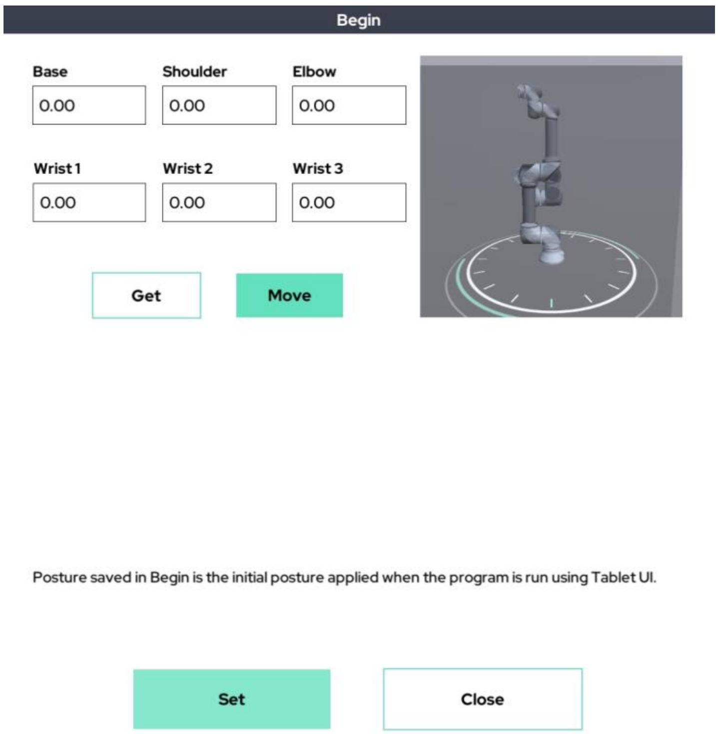
The initial position can be modified in the Begin section of the program. Before running a program that contains movement, the robot must return to the initial position.

The initial position can be changed by the following way.

1. Move the robot to the desired starting position using either the Jog or teaching button

2. Click Begin in the Program Tree to open the Begin menu

3. Click the Get button to record the current posture, then click the Set button to save the position




**Warning**

Warning:

1) When the program is first created, the default starting angles will all be set to zero.

■ Collision detection during operation

The RB Series has two built-in collision detection functions:

- External Collision Detection (Environment-Collision Detection)
- Internal Collision Detection (Self-Collision Detection)

< External Collision >

< Internal Collision >

● External Collision (Environment Collision Detection)

-Detects unplanned external collisions

-Detects unexpected collisions with the environment, including people

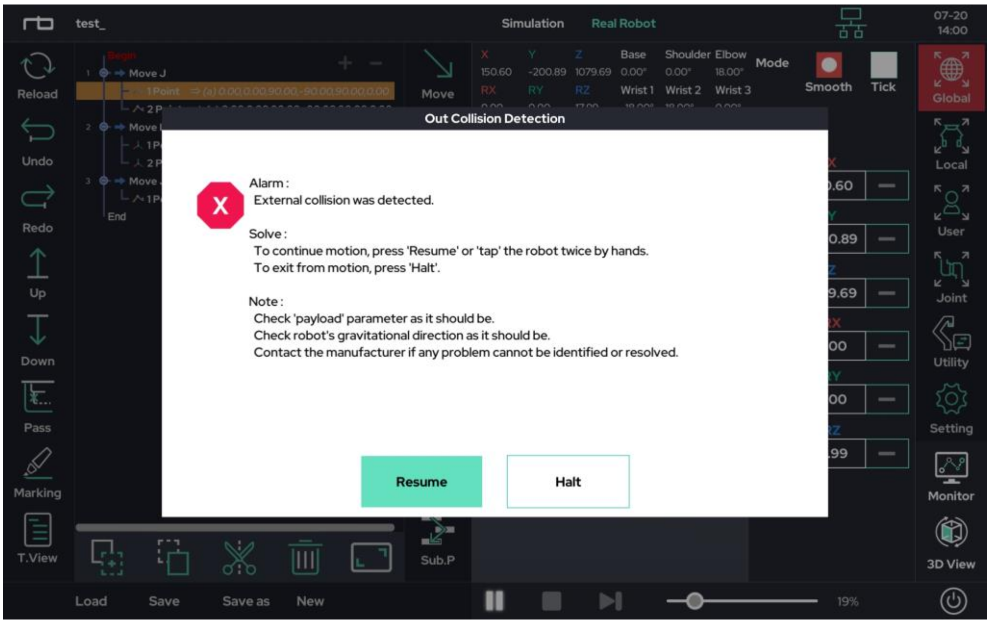
-Collision sensitivity can be changed in the Setup

-The user can change the collision sensitivity in real time while the program is running through the Set function.

-For accurate collision detection, the load / center of mass of the tool should be set accurately

-When operating with high sensitivity collision detection setting, a regular motion could be recognized as a collision due to the sudden acceleration / deceleration of the robot.

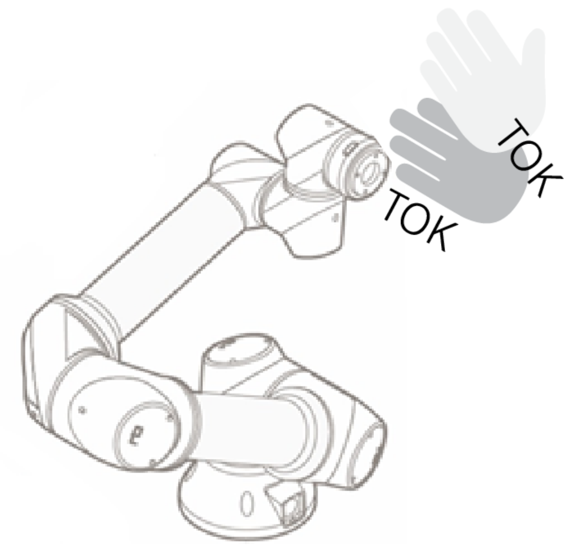
If the robot arm detects an external collision while in real mode, the following pop-up will appear.




To continue, choose one of the two options:

- Resume: Checks the status and continues robot operation
- Halt: Exits the program

Alternatively, tap (hit) on the robot arm twice to continue the operation. This will perform the same function as the Resume button.



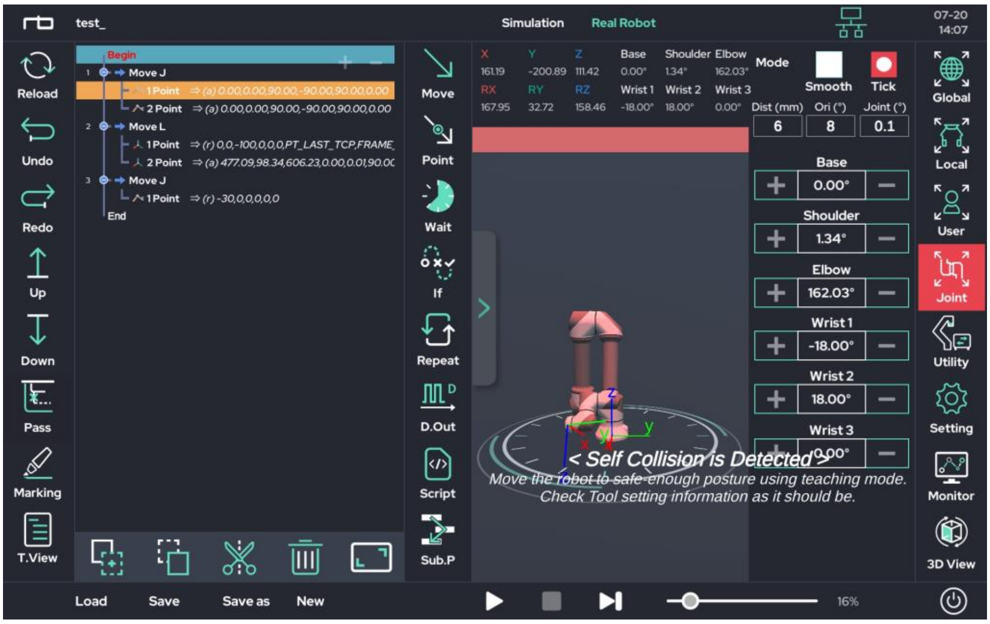
● Internal Collision Detection

- Occurs when the robot predicts that it will collide with itself.

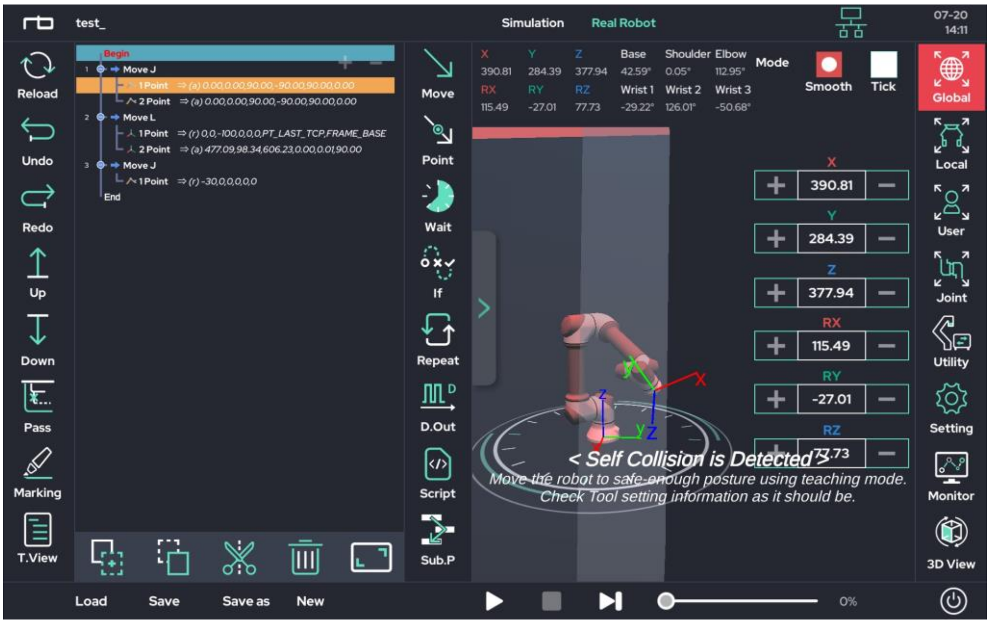
- If the robot extends beyond the preset Workspace limits, it will stop by itself. The setup for the surrounding environment area is done in the Setup screen.

- Users can also set a virtual box for collision detection. This will cause the robot to stop itself if either the virtual box is expected to collide with itself or it goes out of the Workspace. The virtual box is configured in the Setup-Tool.

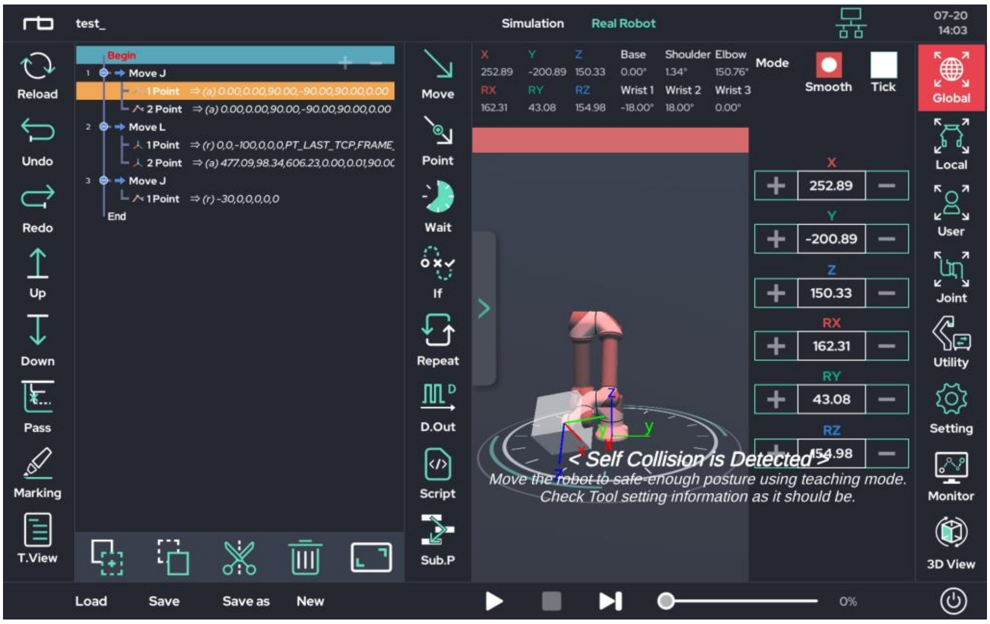
The image below shows a situation where the user caused the robot to crash into itself. Just before colliding into itself, the robot will stop, prompting the UI to display a warning in red.



The image below shows a situation where the robot is about to leave the user-defined Workspace. Just before leaving the Workspace, the robot will stop, prompting the UI to display a warning in red.




The image below shows a situation where the virtual collision box set up by the user detects / predicts a collision. The robot will stop, prompting the UI to display a warning in red.




If the robot stops during operation in real mode, please move the robot arm to a safe position before continuing work.

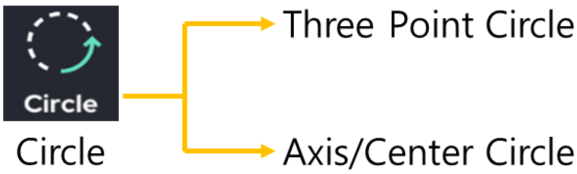
## 7.4 Teaching Icons and Description

In the previous section 6.3, only the basic teaching functions (Move and Point functions) are described. This section is dedicated to the other teaching functions.

■ Circle Function :

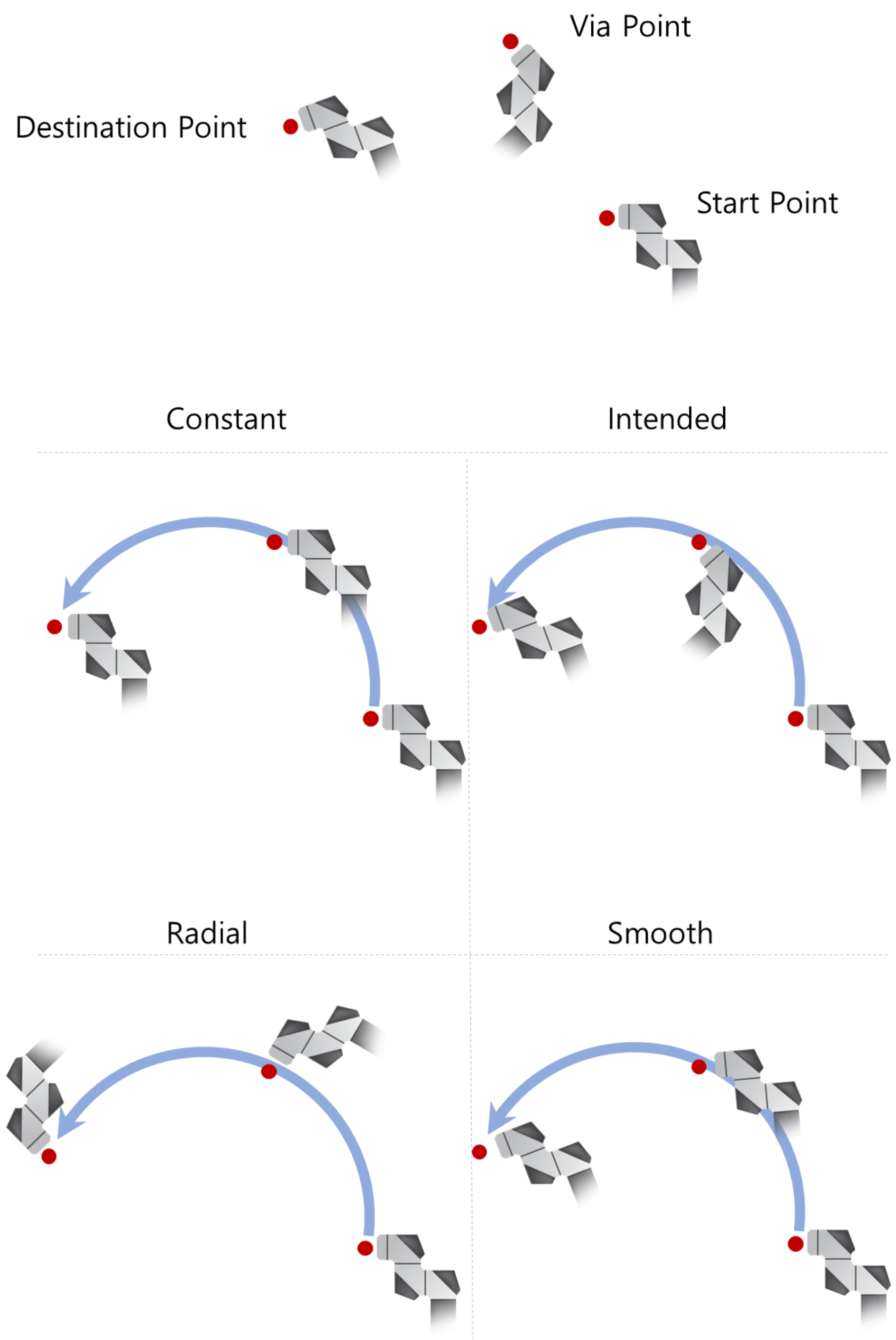

The Circle Function provides a movement method for circular motion.

There are two Circle methods: Three Point and Axis/Center.




- The Three Point method allows a user to draw an arc between three Points. The method requires the user to provide two Points: the middle Point and the end Point. The initial Point will be the most recent position that the robot is in.
- The Axis/Center method allows the user to draw a circle around a center Point. The method requires the user to provide the center Point and the axis around which the robot will draw the circle. The radius of the Axis/Center method is determined by distance between the robot's most recent position and the center Point.

The Circle function offers four Orientation Options: Constant, Intended, Smooth, and Radial.




Constant: Maintains the initial TCP orientation (Rx, Ry, and Rz) of the TCP through the movements.

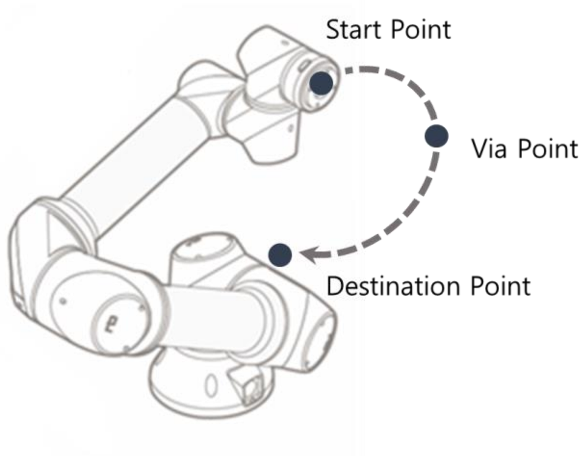
Intended: The TCP rotation set by the user is followed.

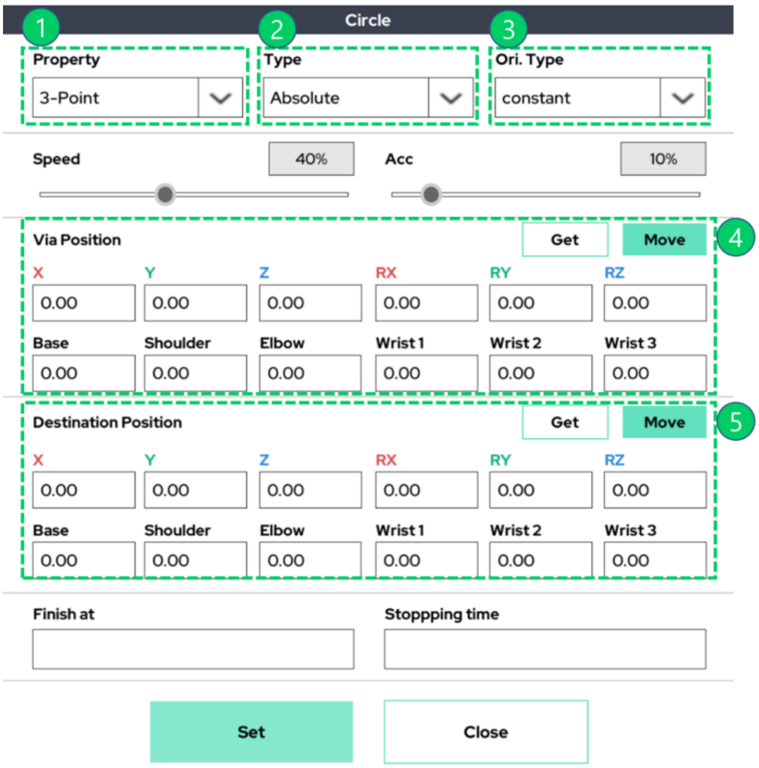
Radial: Rotates the TCP orientation with respect to the center point of rotation.

Smooth: The turn changes immediately from the start point to the destination point. The rotation information of the waypoint is ignored.

### Three Point Circle Type

The Three Point Circle method draws an arc connecting three points: the starting point, the intermediate waypoint, and the arrival point.




① Circular Motion type selection (3-point setting type)

② Point type (Absolute / Variable / Relative / UserCoord.)

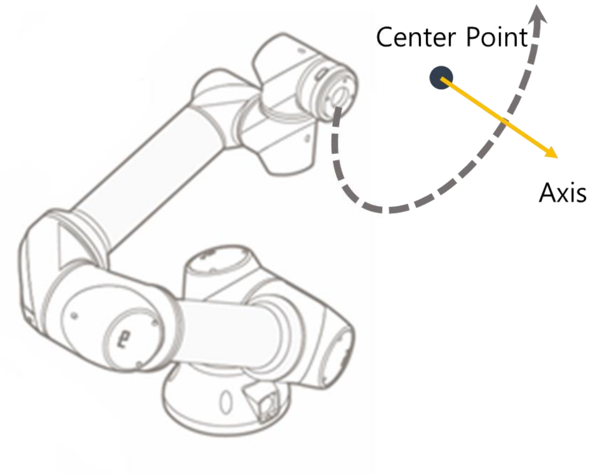
③ Orientation option (Constant / Radial / Intended / Smooth)

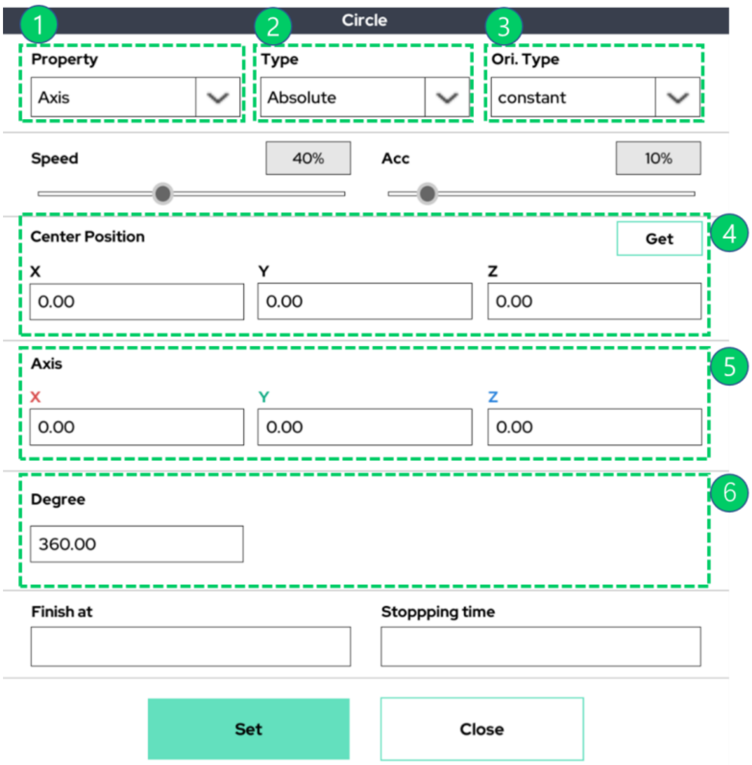
④ Via Point information

⑤ Destination Point information

## Axis/Center Circle Type

Set the center point for the circular motion, the axis of rotation, and the angle to rotate.




① Circular motion selection (axis / center setting type)

② Point type (Absolute / Variable / Relative / UserCoord.)

③ Orientation option (Constant / Radial / Intended / Smooth)

④ Center point information

⑤ Axis information

⑥ Rotation angle information

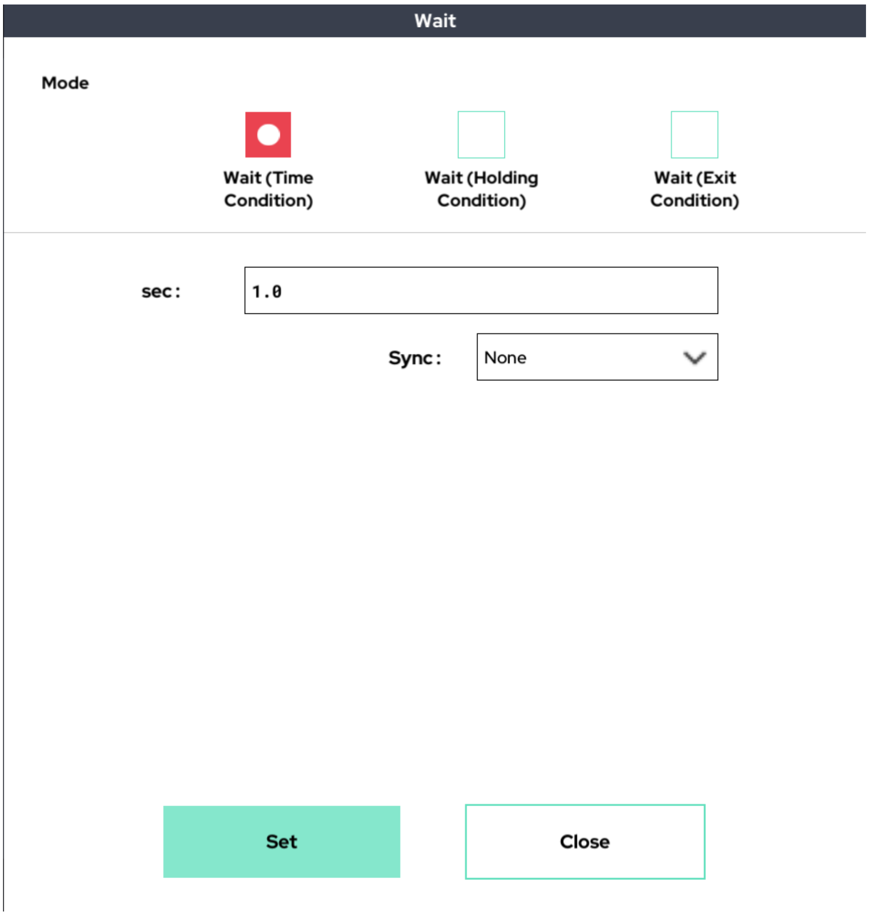
■ Wait Function :


Waits for either a specified condition or a specific amount of time.

There are three modes:

1) Wait for a specified amount of time
2) Wait while a condition is true.
3) Wait until the condition evaluates to true.

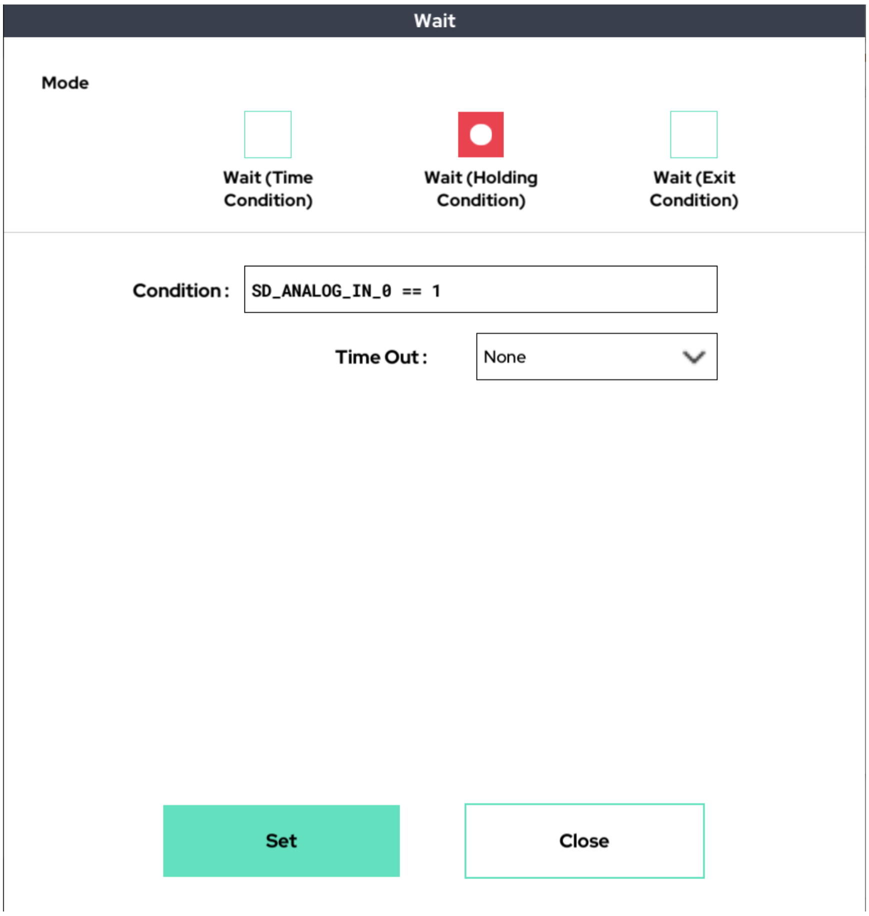
1) Time Condition




Ex) waits for specified amount of time (i.e. 3.0 seconds), then executes the next command

When using 'sync speed control bar' function in Sync, the waiting time is adjusted in inverse proportion to the speed control bar value.

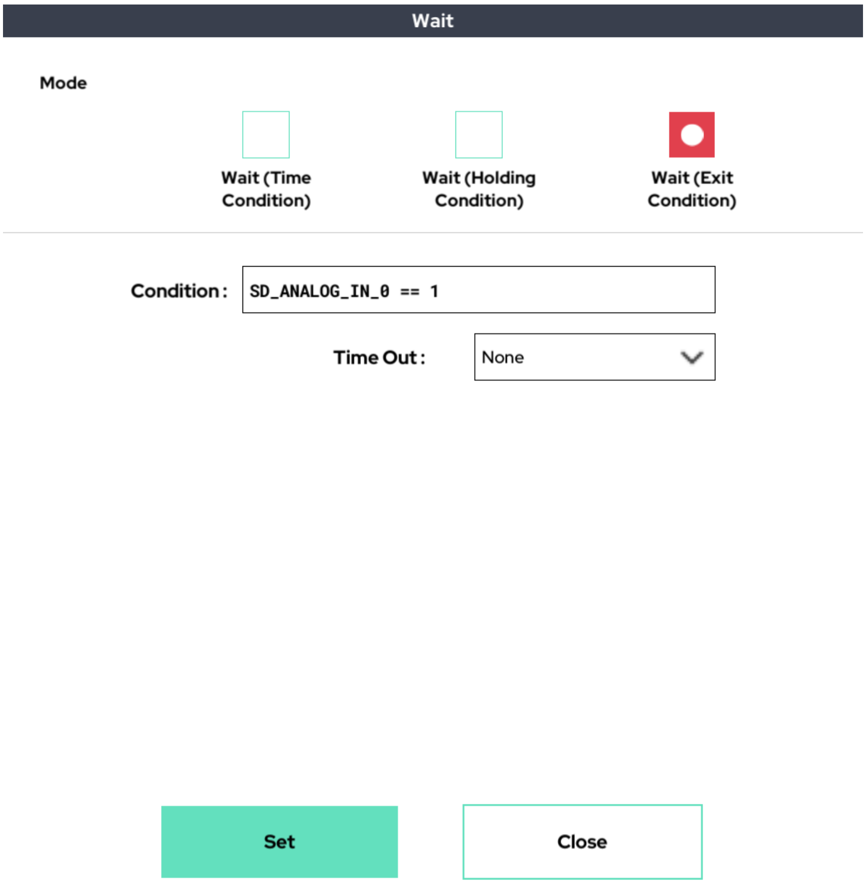
2) Holding Condition




Ex) if the condition is true, the function waits indefinitely

The Time Out function is a function to prevent the condition from continuing to wait until it becomes False in a situation where it cannot be False. Escape the wait after the written time has elapsed.

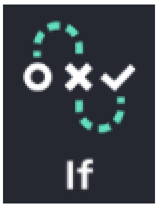
3) Exit Condition




Ex) If the condition is true, the process exits the wait function and then executes the next task.

The Time Out function is a function to prevent the condition from continuing to wait until it becomes True in a situation where it cannot be True. Escape the wait after the written time has elapsed.

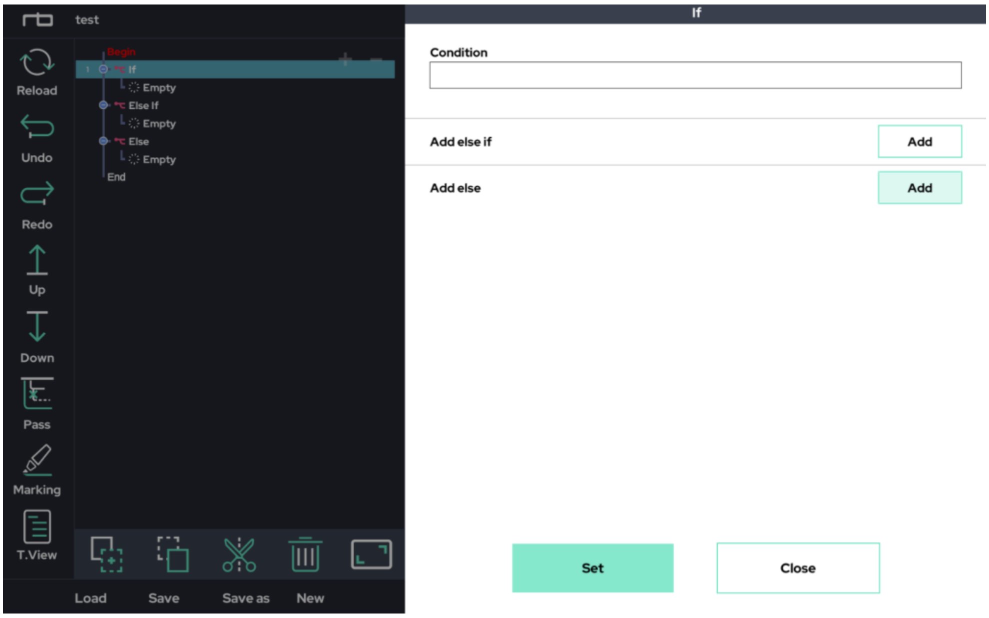
■ If Function :




The If Function allows the users to insert a conditional ‘if’ statement.

Depending on the conditions, branches can be set up so that the robot can perform different commands. Users can set the If / else if / else statement.

After adding the If function to the program tree and clicking the added If function, the following popup window appears. Users can enter the conditional statement they would like to use in the If statement.

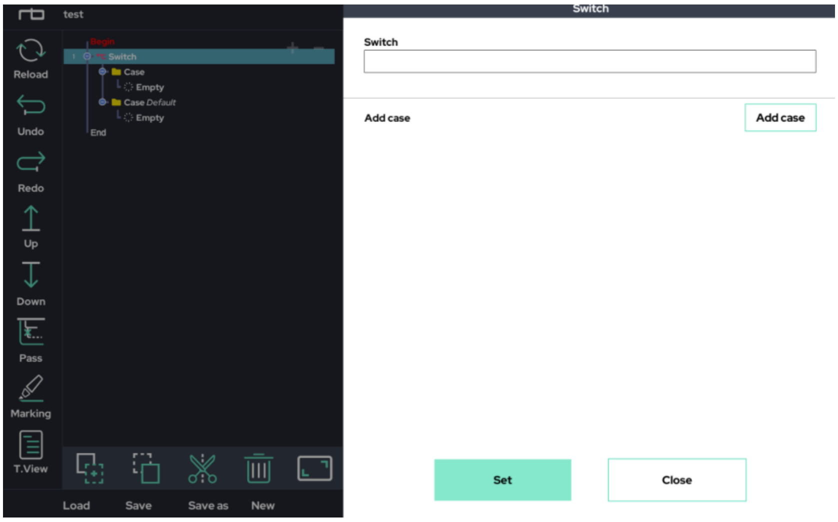



Else if (+ Add else if) or Else functionality (+ Add else) can be created along with branch of conditional statements.

■ Switch Function :

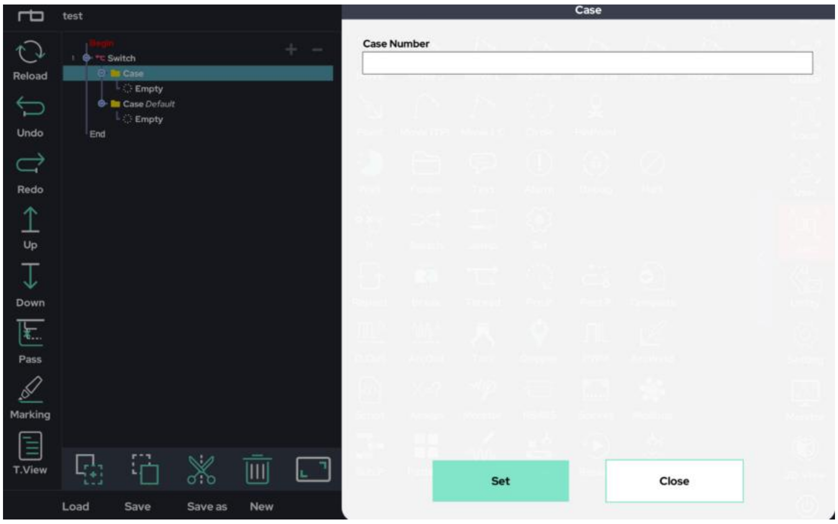

Switch statement. Depending on the conditions, branches can be set up so that the robot can perform different commands. Switch / case statements are available.

The following popup window appears by clicking the added switch function in the program tree. Users than can enter the criteria arguments for the Switch statement to work.

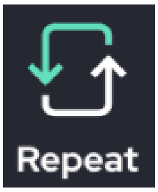



When first creating a Switch statement, 'default' will automatically be created. Additional case statements can then be added using the (+ Add case) button.

After clicking the (+ Add case) button, the following window will appear. Enter the conditional argument in the field, then press the Set button to save.



■ Repeat Function :




Repeats the nested program by the specified condition. There are three modes – these modes look similar to those within the Wait function:

1) repeat a specified number of times.

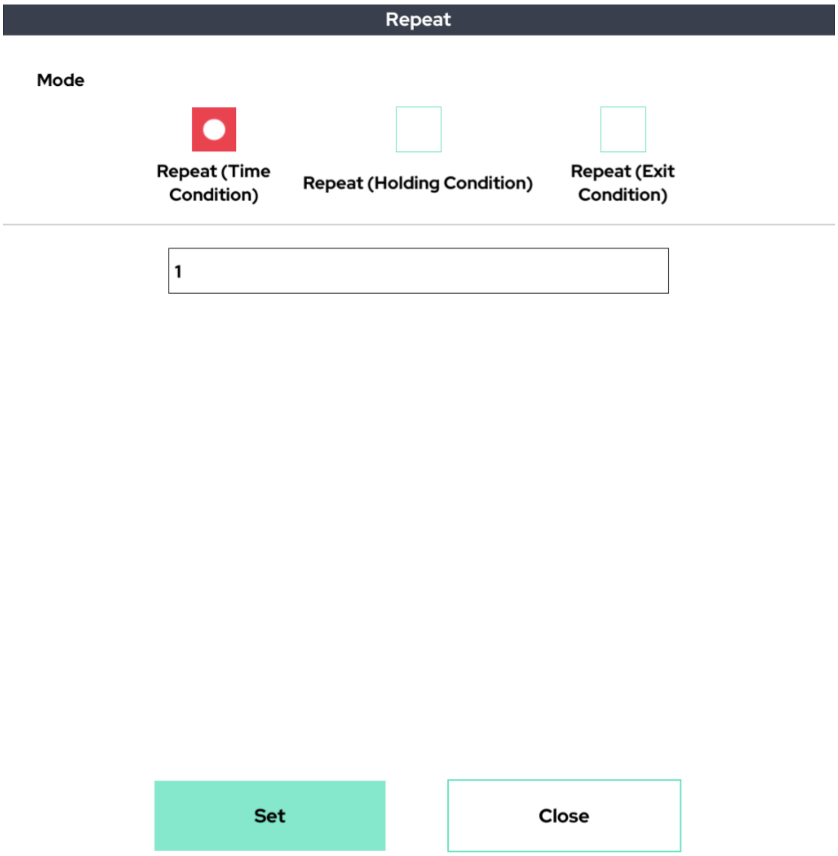
Note: if a user sets this value to 0, it will repeat indefinitely

2) repeat while the specified condition is true

3) repeat while the specified condition is not true.

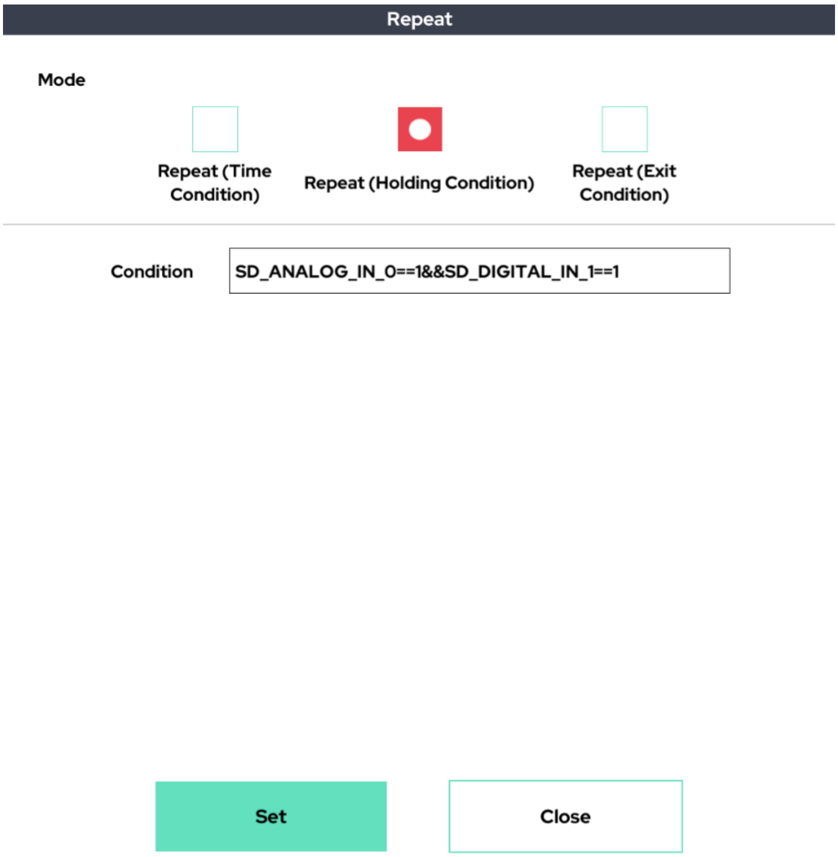
After clicking the Repeat button, a popup menu containing the three modes will appear. Once it opens, select and use the desired function.

1) Time Condition



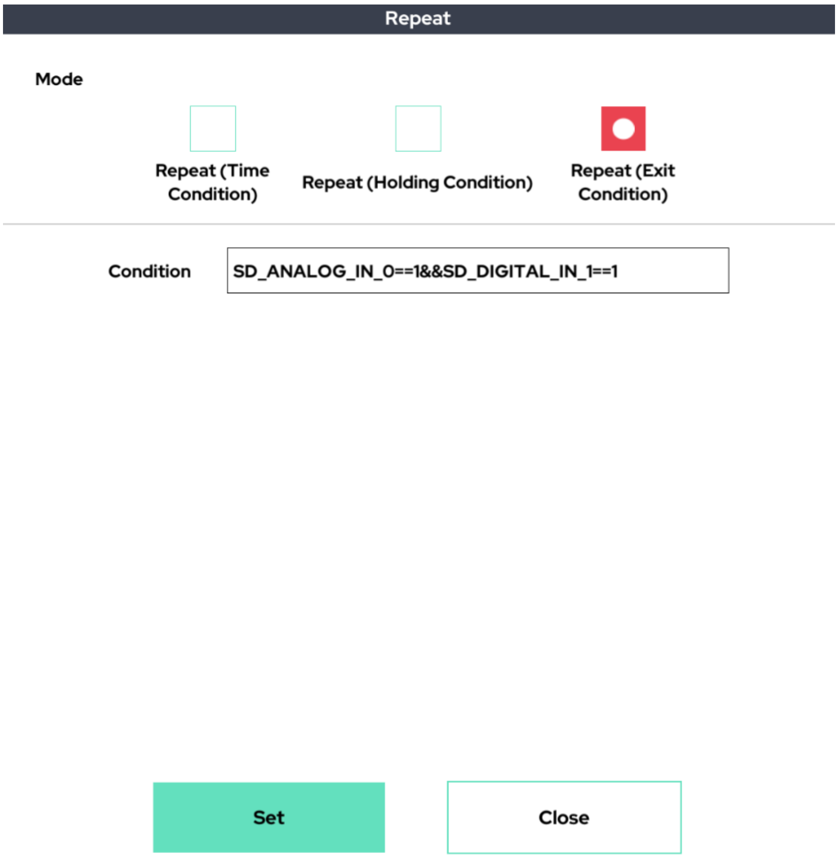

Ex) The above example will repeat a subprogram 1 times.

2) Holding Condition



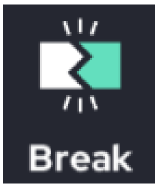

Ex) While ANALOG_IN_0 and DIGITAL_IN_1 both evaluate to 1, the subprogram will repeat. The subprogram will continue to repeat until at least one of the two values changes.

### 3) Exit Condition



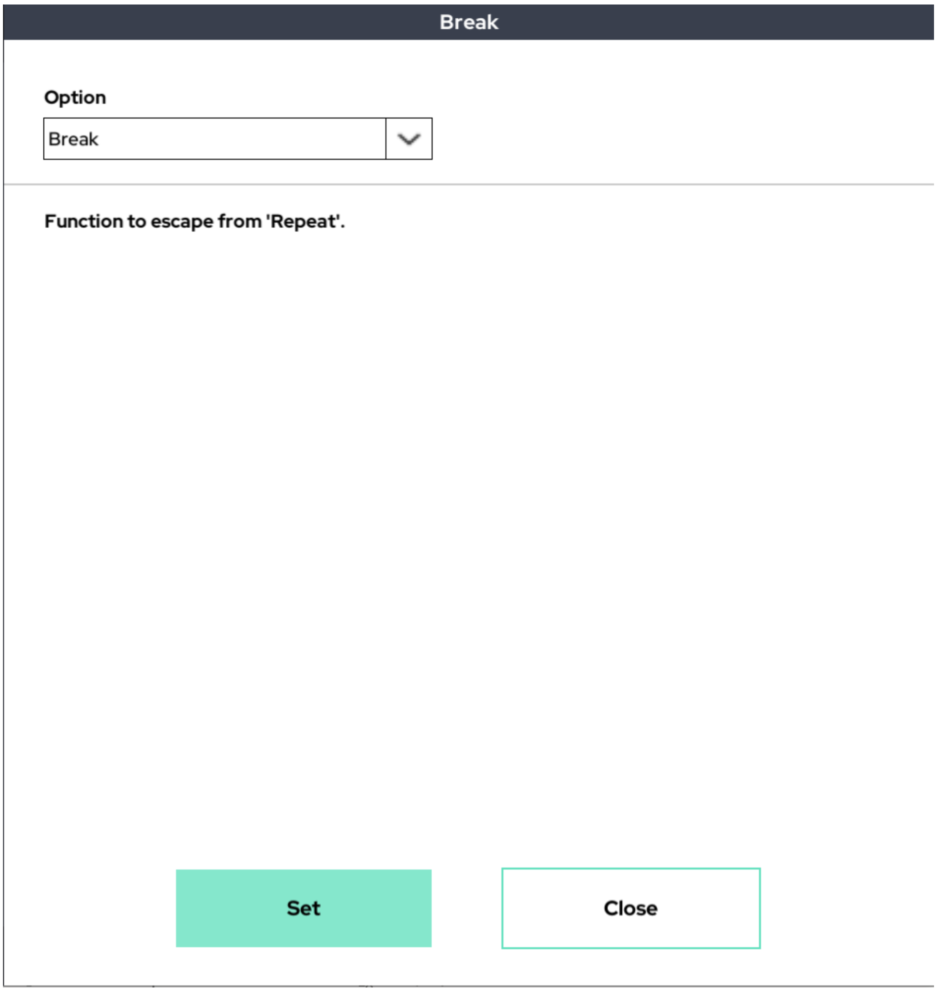

Ex) The subprogram will repeat until both ANALOG_IN_0 and DIGITAL_IN_1 evaluate to 1. The subprogram will continue to repeat until both values become 1.

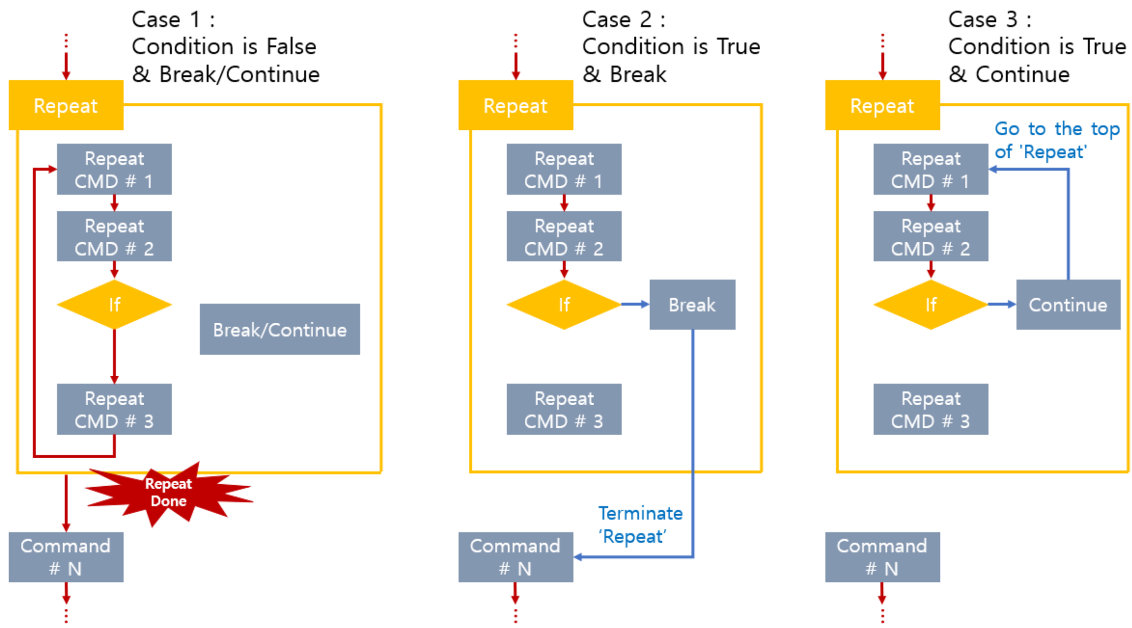
■ Break Function :




This is a function to forcibly terminate the Repeat (break) or move to the top of the Repeat (continue). Even if the Repeat condition determines that the subprogram should continue, the Break function can be used to escape the Repeat. The Continue function is used into the Repeat function, and when used, it moves to the top of the Repeat without executing the subprogram

It can only be used as a subitem of the Repeat function – it cannot affect any other part of the program.

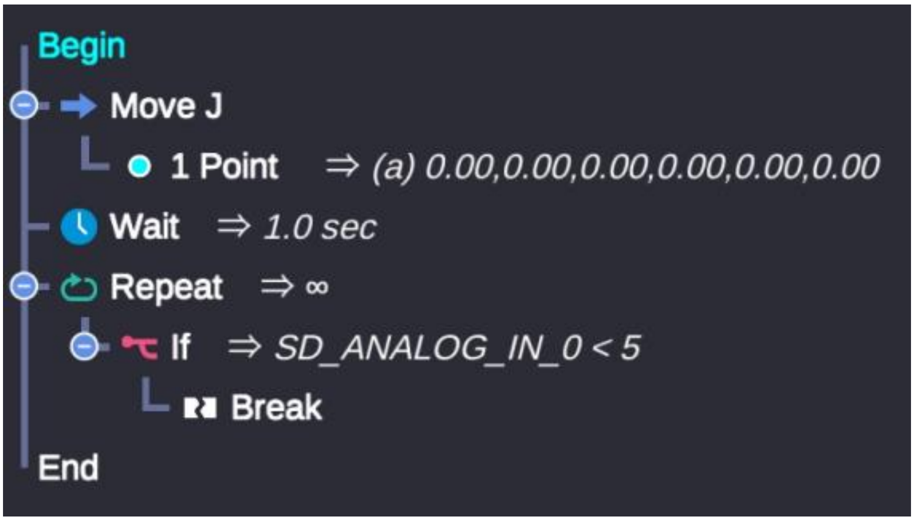





If a repeat break and continue are used, it will behave as shown in the figure above.

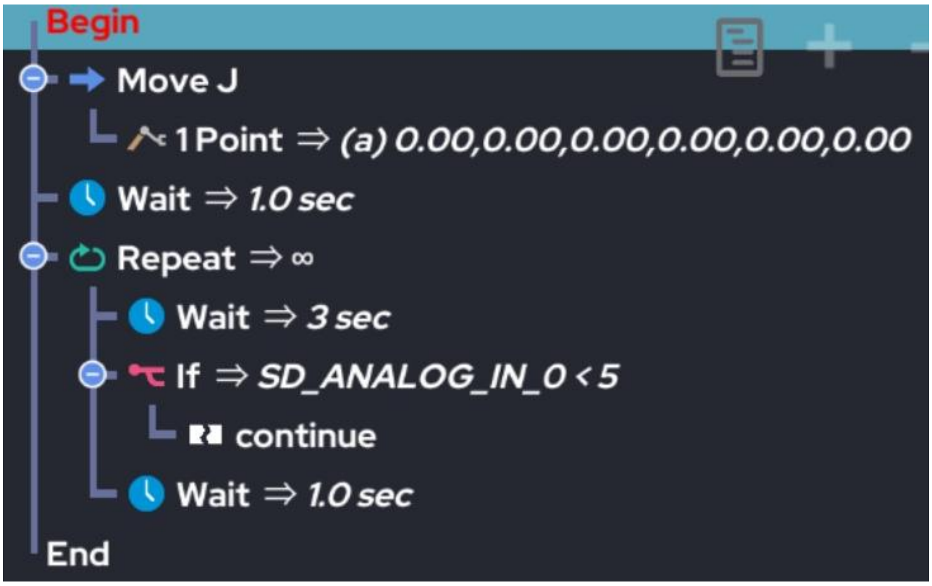
∗ Example of break function

In the example below, there is an infinite loop that contains an If function. If the condition (SD_ANALOG_IN_0 < 5) ever evaluates to true, the subprogram escapes the loop and executes the next command (in this case, **End**). If the condition never occurs, the loop will repeat indefinitely.




∗ Example of continue function

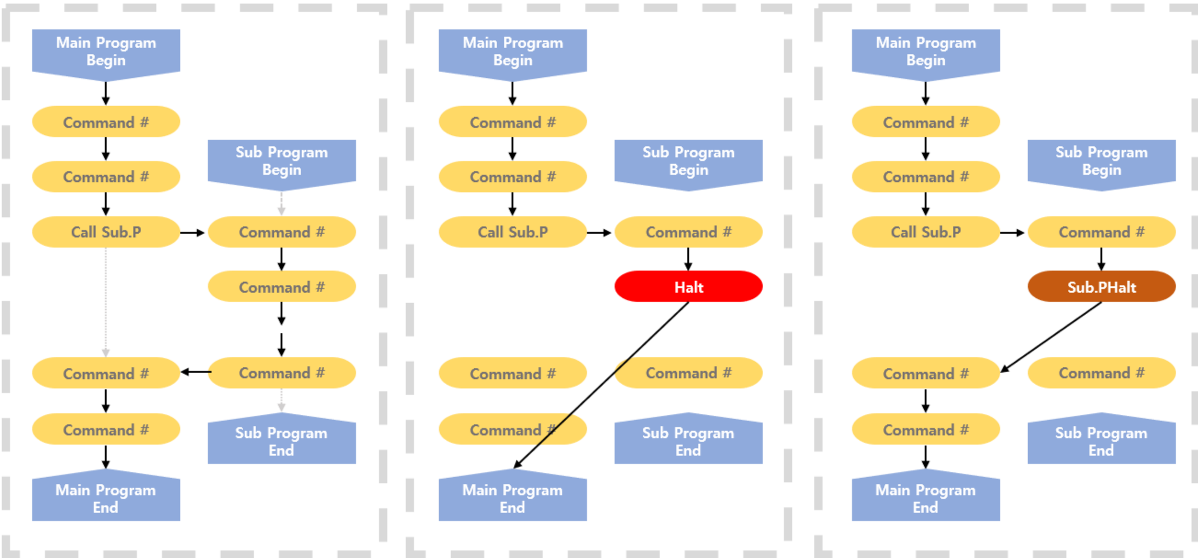
The example below has an infinite loop with an if function. If the condition 'SD_ANALOG_IN_0 < 5' is true, the command at the bottom of the Continue function is not executed and the command at the top of the loop is executed (in the example, wait 3 seconds).



■ Halt Function :


Terminates the program.

Halt is divided into Halt and Sub.P Halt. Halt is a function that terminates the main program regardless of whether it is executed in the main program or sub-program. Sub.P Halt must be used within the sub program, and the moment it is executed, the sub program ends and returns to the main program. Please refer to the diagram below.

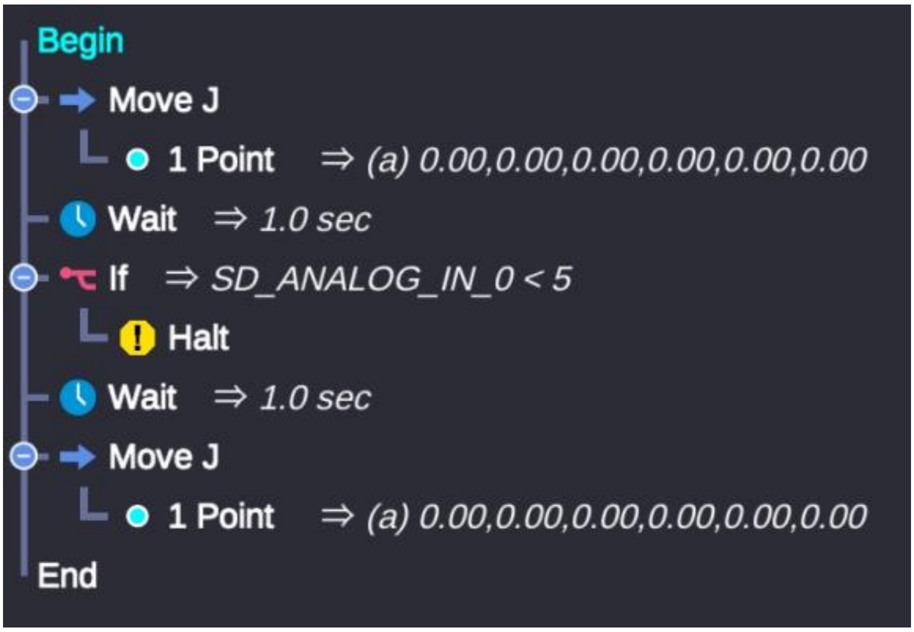



common case

during the subprogram When Halt is called

during the subprogram When Sub.P Halt is called

In the example below, the program will check the If function and call the Halt function if the condition is true. If the condition is true, the program will terminate and will not execute the next commands.

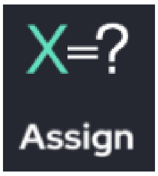



## Warning

Warning:

1) When the Halt function is executed, the main program will terminate – this includes any additional Thread functions.

■ Assign Function :

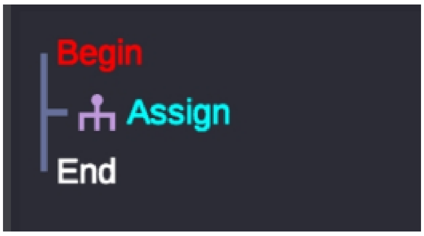



Declare and designate the value of a variable. Variables can be changed through the program to allow for greater flexibility with conditionals.

A variable can be one of the 4 following types:

- Variable Type: Saves a single numerical (float) variable.
- Array Type: Saves multiple values in a list. The maximum length of the array is 10.
- Point Type: Saves position information (saves x, y, z, Rx, Ry, Rz).
- String Type: Saves a string (alphabetic and numerical characters – e.g. "ASDF1234")

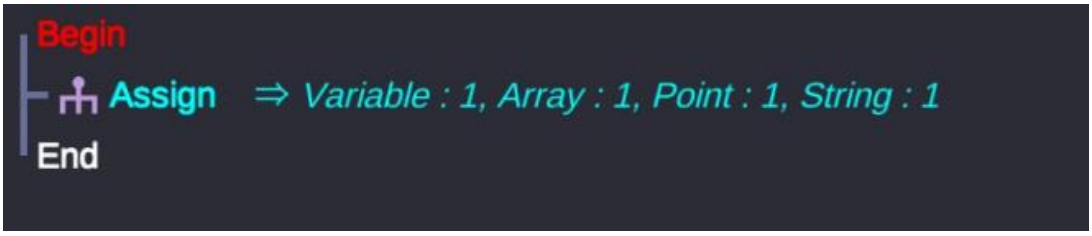
When the Assign function is added to the program tree, it will look as shown below.



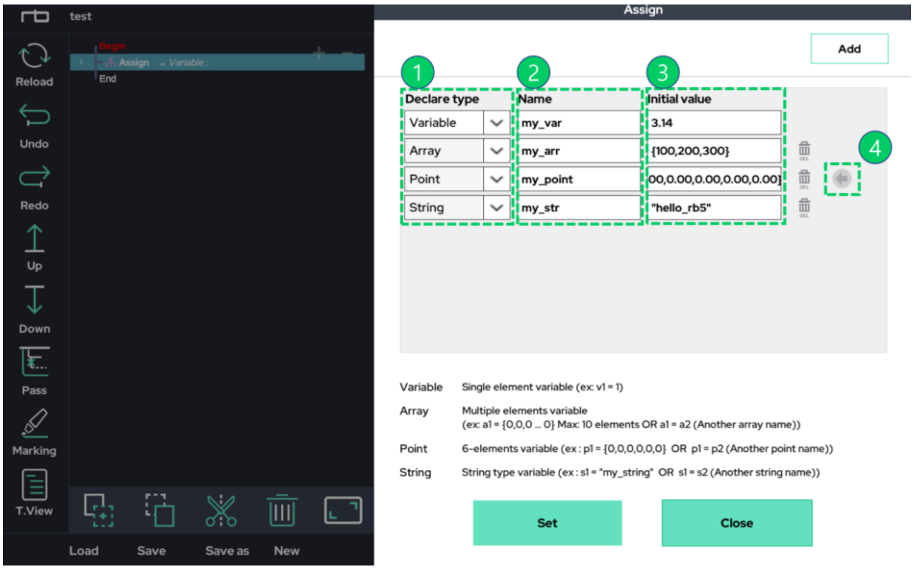

To assign a variable, click on Assign and a popup will appear. Then, the variable can then be assigned within the popup. Multiple variables can be declared by clicking the Add button. To save the variable, click on the Set button.

If a declaration is made, the variable name and initial value will be displayed on the tree as shown below

If multiple declarations are made, the program tree will show how many variables of each type were declared.




An example popup window of the Assign function is shown below. Note, the below window shows 4 declarations.




Each part of the popup encircled in green dotted lines are explained below:

1) Declares the type of variable (Variable, Array, Point, String).

2) Sets the name of the variable

3) Sets the initial value during the declaration.

For the Variable Type, the initial value is set as a single number (e.g. 1).

For the Array type, place initial values within curly braces (e.g., {100, 200, 300}).

For the Point type, use curly braces around the initial values, which will be in the form of an array of six lengths, (e.g., {300, 300, 300, 0, 90, 0}).

For the String type, put use quotations around the string for the initial value (e.g., “hello_rb5”).

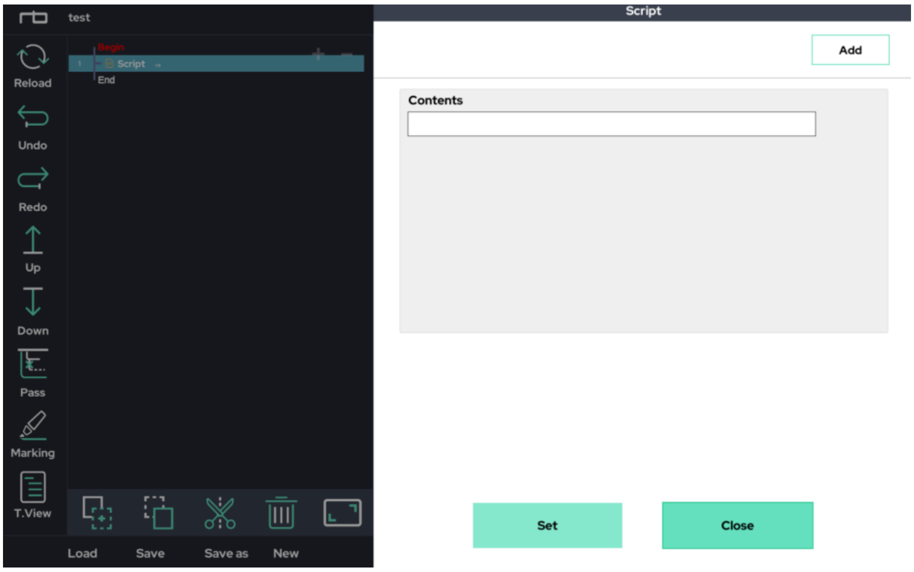
4) Button for the Point type.

The six coordinates (x, y, z, Rx, Ry, Rz) of the current robot configuration are imported as initial values.

■ Script Function :

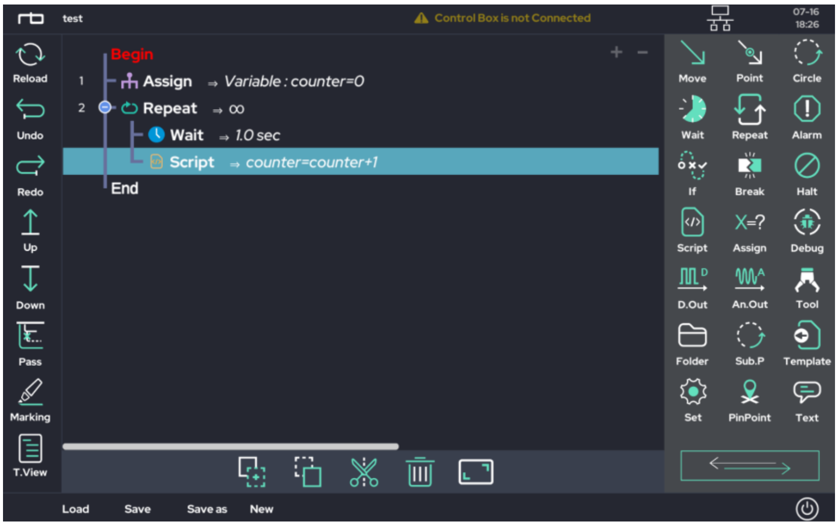

Allows the user to write custom scripts. These scripts allow for custom operations / calculations. The Script Function also allows for functions such as variable substitution and assignment.

Add the script function to the program tree and click the added script function. The following popup window will appear.




From here, the user can enter a custom script. If the user wants to execute multiple lines, click the Add button at the bottom of the popup window.

The following example is a program that uses the Repeat function to repeat once every second. After each second, the Script function increases the variable called ***counter*** by 1. This will repeat indefinitely, as the loop is set to continue an infinite number of times.

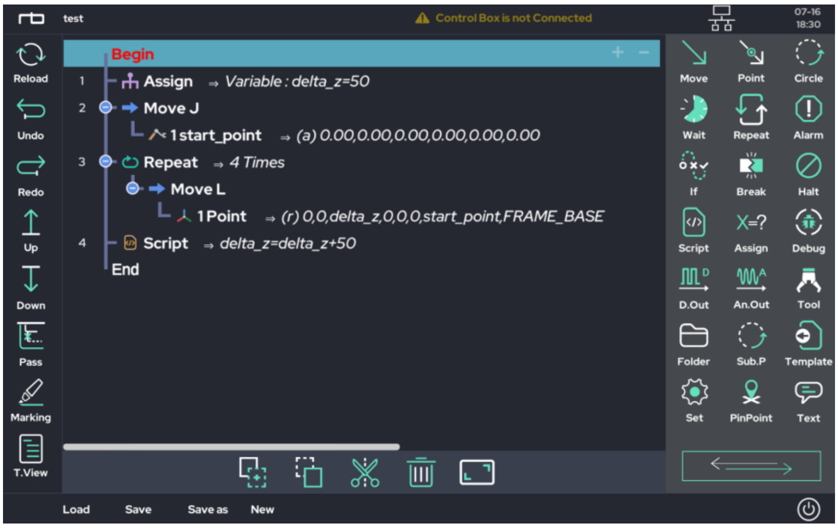



In the example below, the variable ***delta_z*** is set to 50. The MoveJ function is used to move to a specific pose named ***start_point*** (using the Point naming feature).

Once MoveJ moves to start_point, the Repeat function is set to repeat its subitems four times.

- MoveL uses the Relative Point function to move vertically in the z direction by ***delta_z*** (50 mm) from ***start_point***. (See the relative point function of the linear movement series of the point function.)
- At the end of the loop, ***delta_z*** is increased by 50 using the Script function.

To summarize, the robot moves to the first position with MoveJ, saves the position as start_point, and then repeats 4 times with Repeat Function and moves up by 50mm each loop using the MoveL function.



## Warning

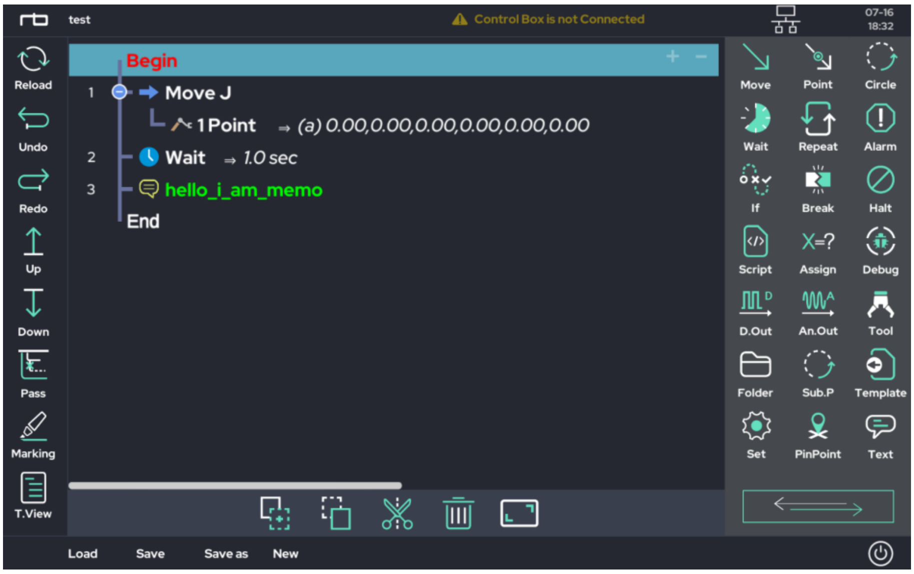
Warning:

1) The script function is an area where the user can freely write and execute a script.
2) If the users write a script that doesn't match the syntax, the program may malfunction or stop. Be mindful and use the proper syntax when using this feature.

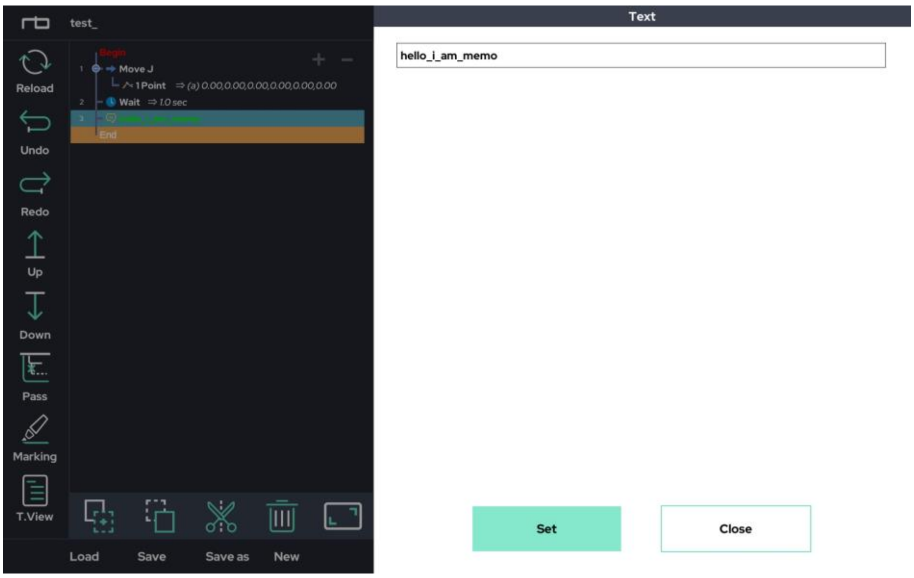
■ Text Function :


The Text Function allows users to make notes/comments in the program list tree.

The text function is displayed as green text in the program tree and does not affect the functionality of the program. Click the Text icon to add it to the Program Tree.



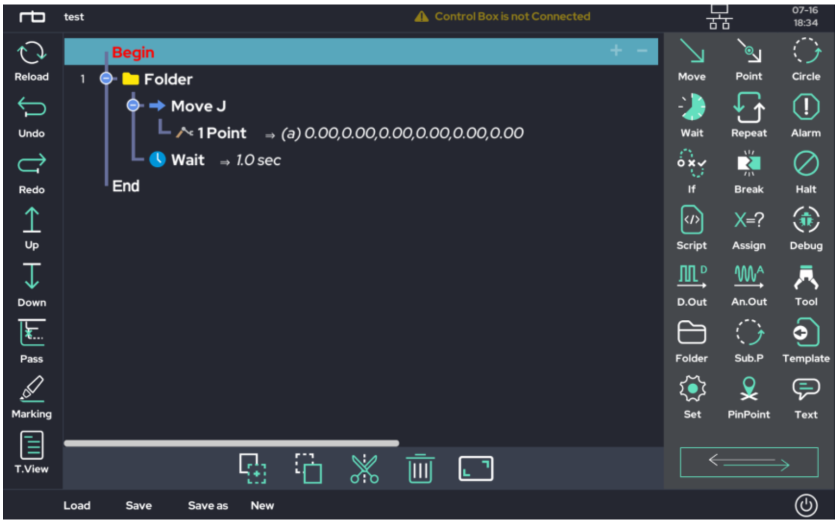

Users can add messages by clicking on the new Text line in the program tree. Notes can be added by adding text to the popup. Press Set to save.



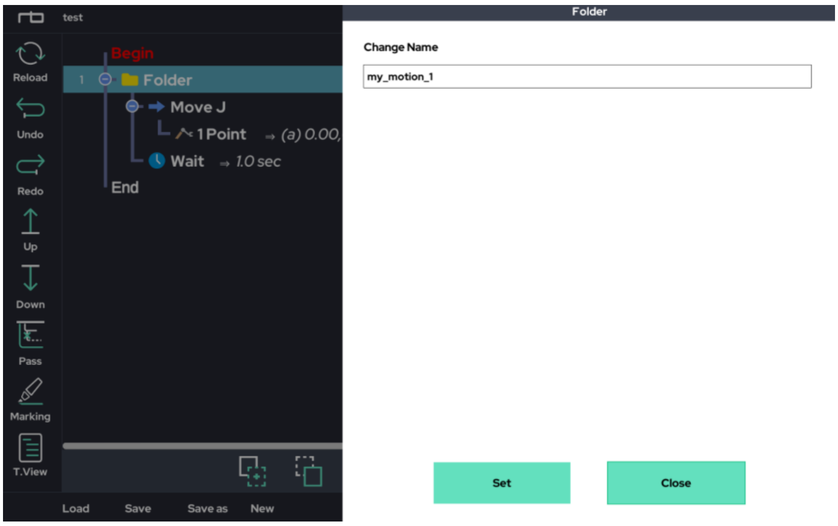
■ Folder Function :


The Folder function helps to organize commands and manage them as modules. Each Folder can contain commands as sub-items, helping with the flow of the program. Each folder can then be renamed to help provide details to the flow of the program.

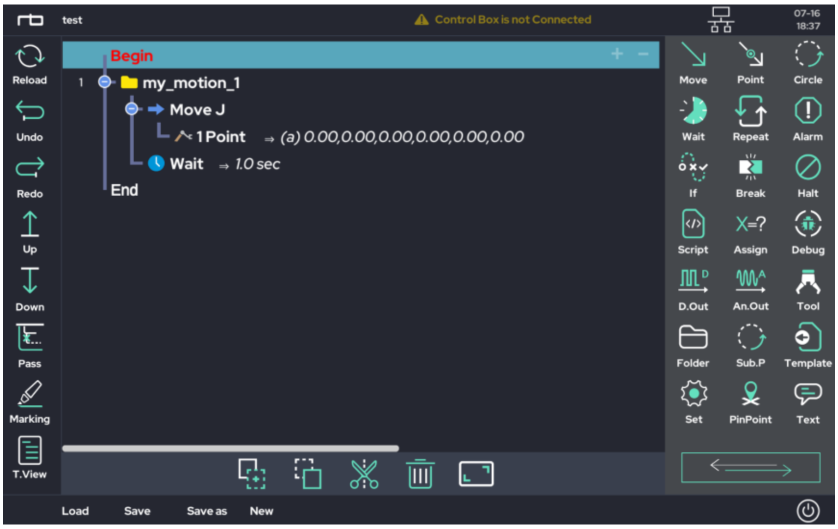
By clicking the Folder icon, it will be added to the program tree. Commands can then be added, as shown below.




To rename the folder, click on the new Folder in the program tree. A popup will appear for the user to change the name. Press Set to save the new name.




The program tree will now show the folder with its new name.

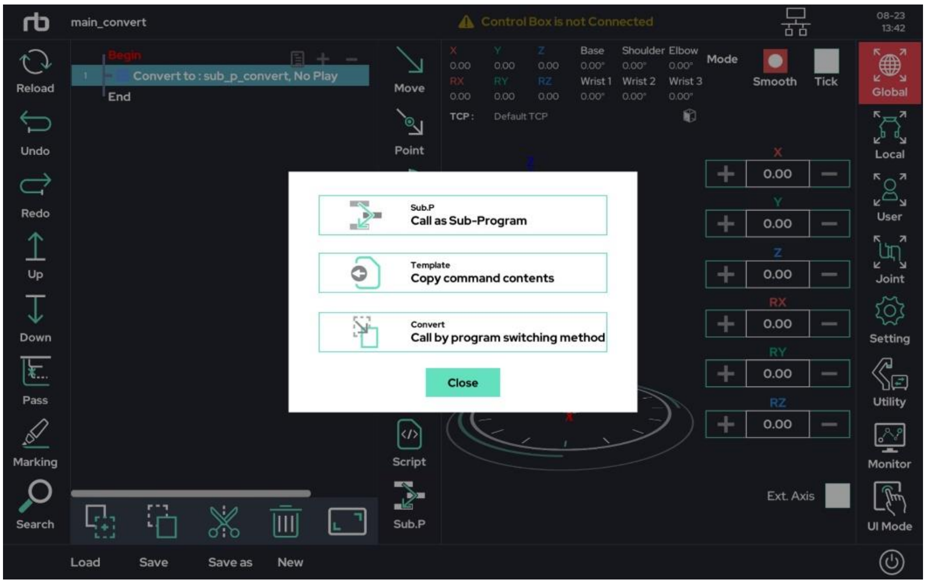



Like the Text function, it does not affect the function of the program. This function only helps to manage the flow of the program by allowing for module creation.

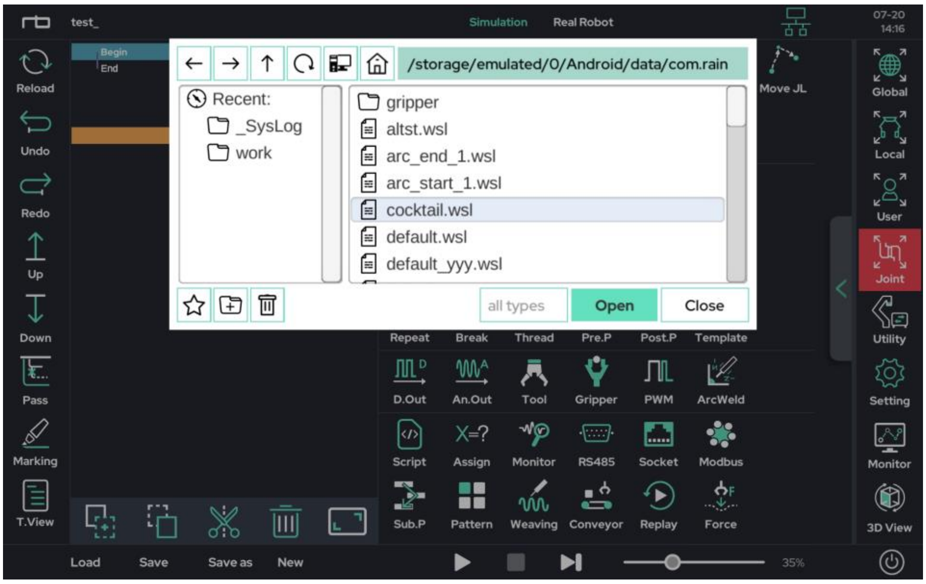
■ Sub.P(Sub Program) Function :


Allows users to insert other program files into the current project. These other program files are made in advance and accessed through the file explorer window.

If you click the Sub.P icon in the program, the following pop-up window appears, and at this time, click 'Sub.P'.

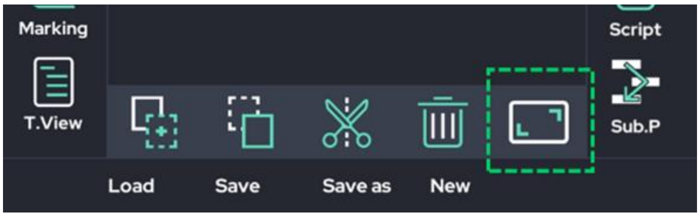


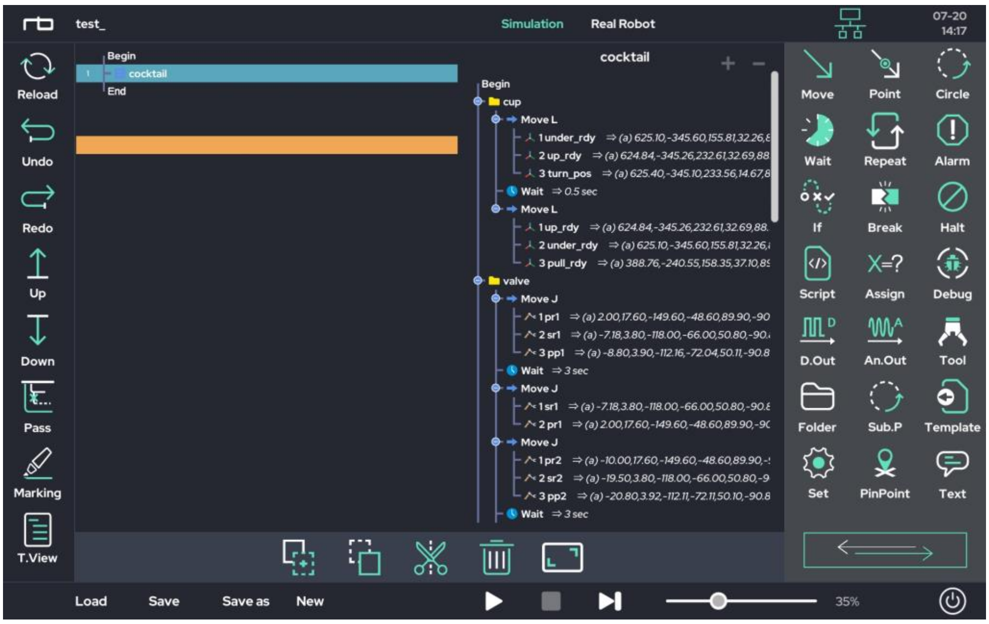

If you click 'Sub.P', a popup window with file explorer function appears as shown below.



Through the file explorer window, a user can view other projects created on the tablet PC. To add another file as a sub program, select the desired project and click the Open button.

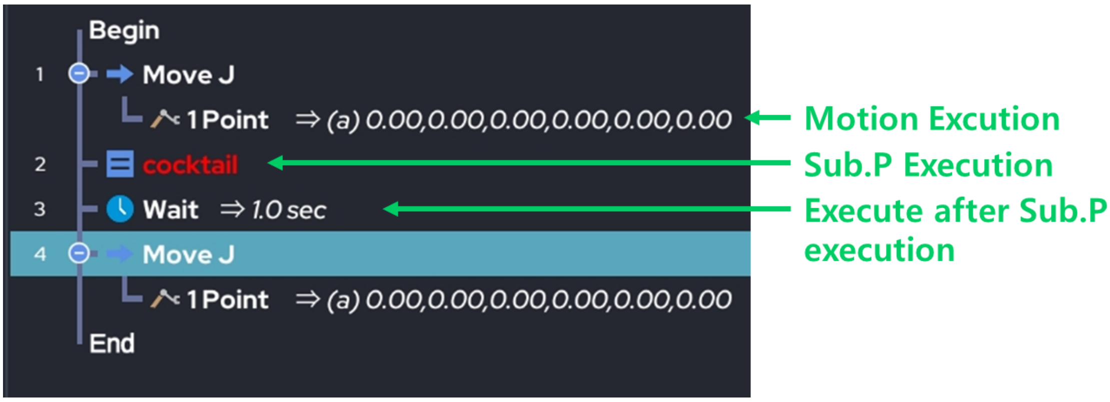

In the above example, a subprogram named **cocktail** has been inserted into this project. To see the contents of the subprogram, expand the program tree viewer (shown below in the green dotted lines) and click on the loaded subprogram. The current project is displayed on the left side, and the loaded project contents are displayed on the right side.



A subprogram is executed sequentially along with other programs.

If other commands are placed after a subprogram, they will be executed after the subprogram finishes.




## Warning

Warning:

1) The contents of a subprogram called by the Sub.P function can be seen by the user, but they cannot be modified. If modifications are required, the project must be opened separately.

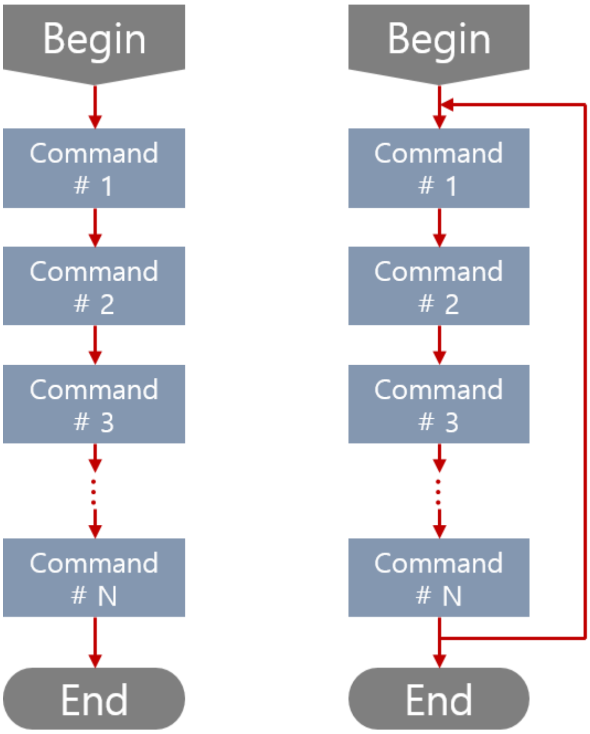
2) The Sub.P function can be called up to 10 levels deep. It is not recommended to use recursion with the Sub.P function.

■ Pre.P(Pre Program) Function :


The Pre-Program function is a dedicated Folder placed at the beginning of the program. *The Pre-Program folder will execute its contents only once.*

- Pre-Program will **not** have an effect on a program in Make mode, since the program will exit when it finishes executing.
- Pre-Program will have an effect on a program in Play mode, since the program is on repeat.

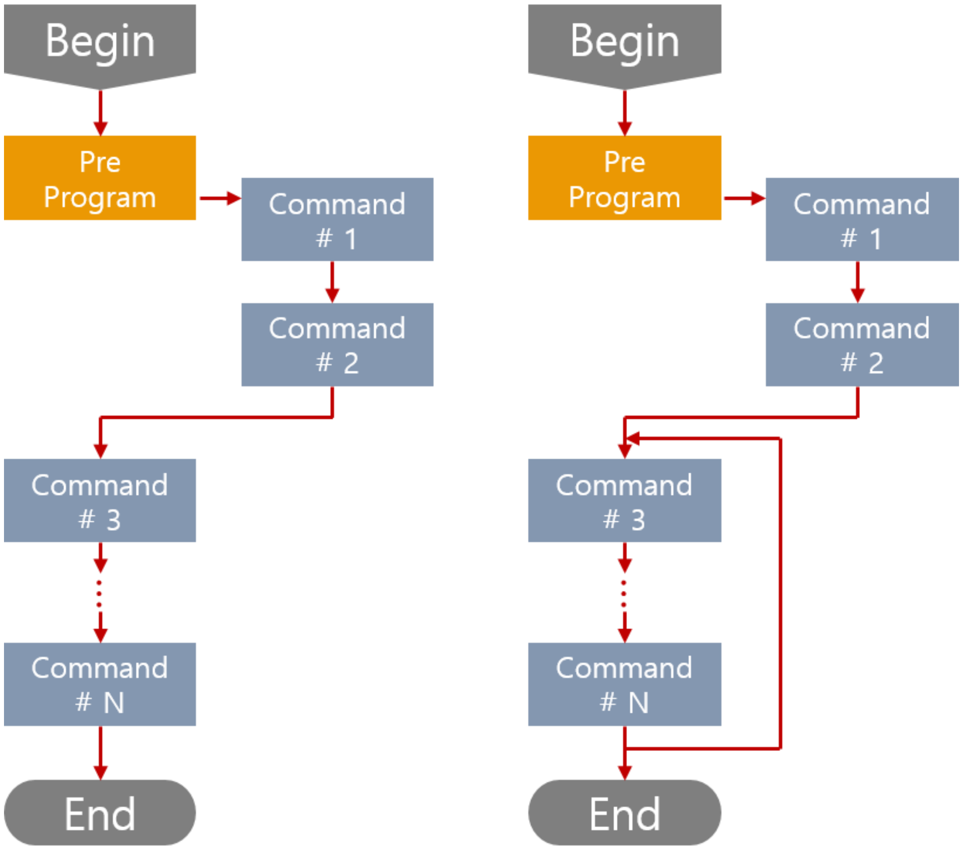
The figure below shows the general command flow when the Pre.P function is not used.




<Run in Make> <Run in Play>

The left column shows the flow of a program being executed in the Make screen, whereas the right column shows the same program being executed in the Play screen. In Make, the program between Begin and End runs once. In Play, the program between Begin and End runs indefinitely.

The figure below shows the program instruction execution flow when the Pre.P function is used.



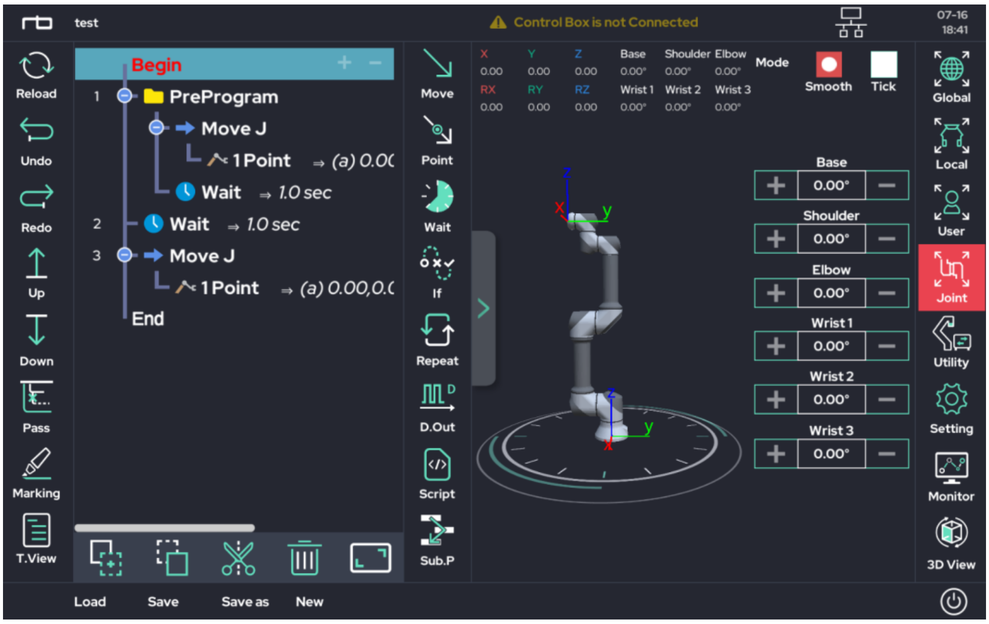

<Run in Make> <Run in Play>

In the Make screen, commands between Begin and End are executed in sequence, regardless of the use of the Pre.P Function.

In the Play screen, the program repeats between Begin and End, but the commands contained within the Pre-Program Folder is executed only once.

Pre.P function is useful for running Functions that need to be performed once, such as variable declarations, and communication connections.

The figure below shows the Pre.P (Pre-Program) Function used in an actual project. The Pre.P function must be directly after the Begin line, as it runs before the rest of the program. Users cannot copy the Pre.P Folder and paste it elsewhere.



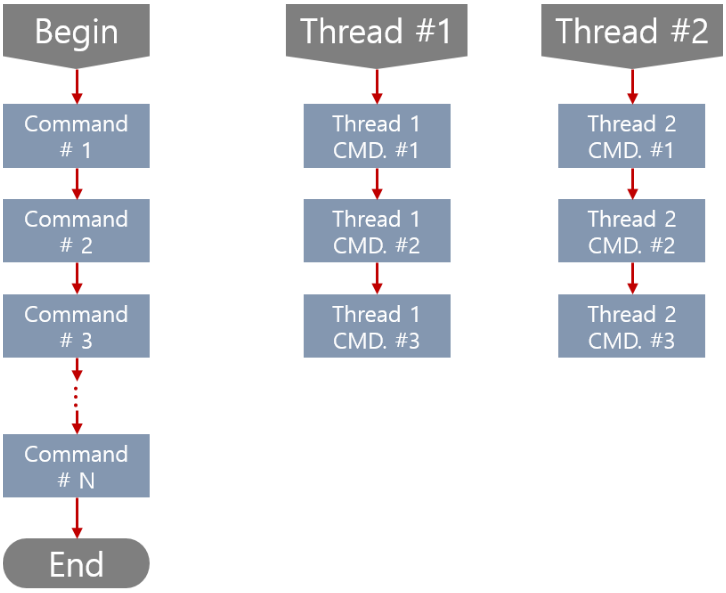

**Warning**

Warning:

1) If the Pre.P function is used in a project called through the Sub.P function in the main program, the Pre.P function applies only to the main program.

■ Thread Function :


Will create a separate program tree called "Thread." This program will run in parallel (at the same time) with the main program. However, the thread program tree is limited to using functions that do NOT control robot operation. In other words, the user cannot put a Move, Point, or Circle function in the thread program tree.

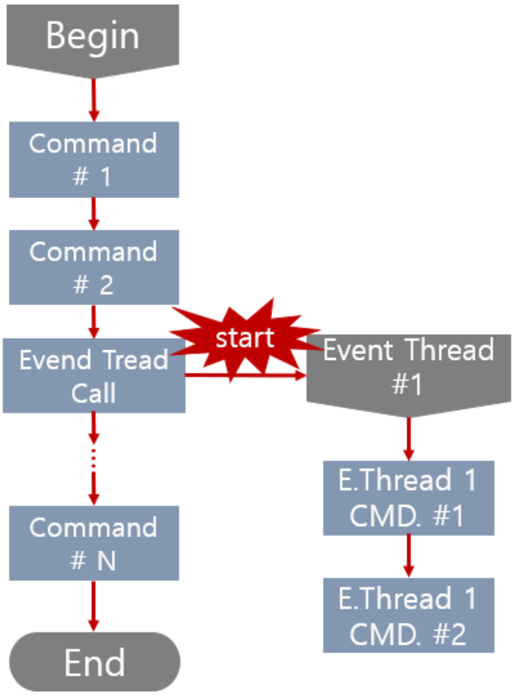



As shown above, the Thread Function is configured in parallel with the main program

- Threads do not repeat automatically and will end when the main program ends – even if the thread has not completed. To implement a Repeat Function, highlight a command within the Thread program tree and press the Repeat icon.
- To implement a thread that repeats every second, use the Thread icon, use the Repeat Function within that thread, then place a one second Wait Function within the Repeat.
- The Thread Function will support only up to 3 different threads
- Thread functionality works only in the current running program. If a subprogram called through a Sub.P function uses a thread, it will not work properly.

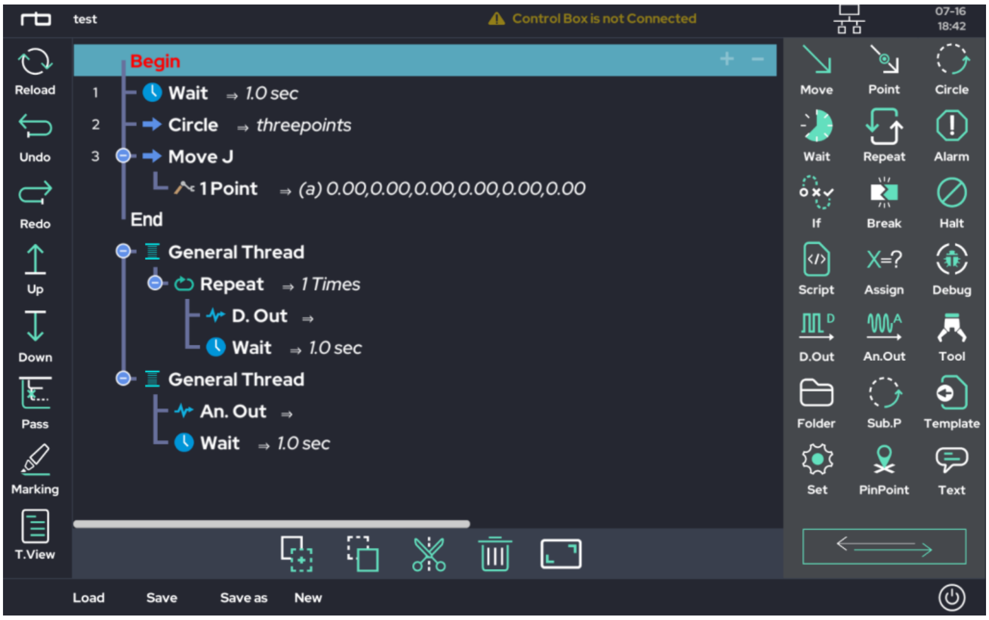
Thread types are as follows:

- General Thread : It stops with the user's intentional pause, alarm, collision detection, etc.
- Non-Stop Thread : It does not stop except for collision detection.
- Non-Stop Thread2 : It doesn't stop until the program Halt.
- Event General Thread : This is a General thread executed by the event thread call function in the main program.
- Event Non-Stop Thread : It is a non-stop thread that is executed by the event thread call function in the main program.




The figure below is an example of how the Thread function can be inserted into an actual project. In the example below, two threads are inserted.

As shown in the figure above, the event thread starts running when the event thread call function is used in the main program.




## Warning

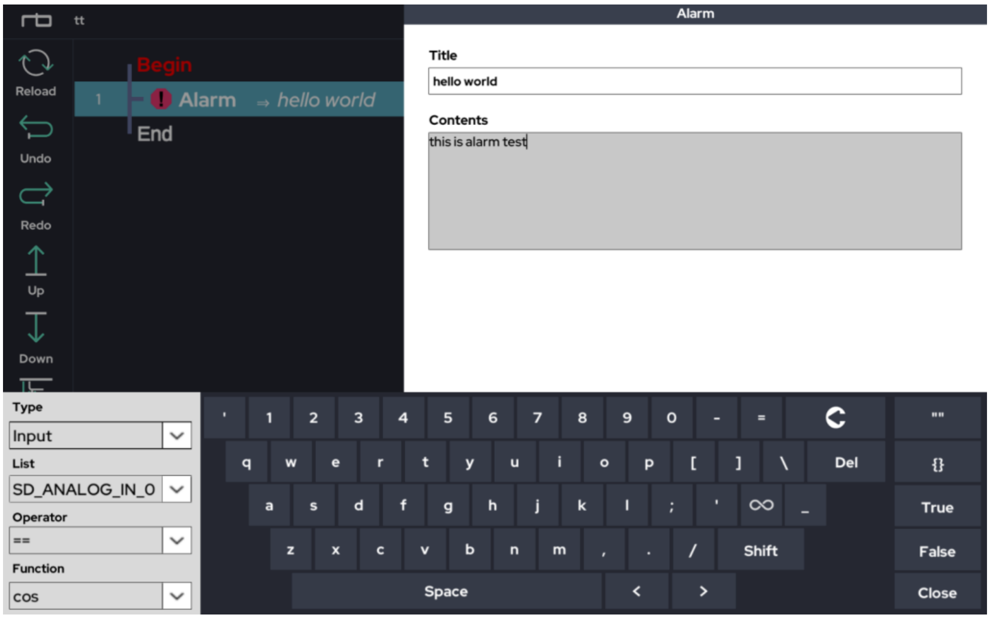
Warning:

1) For the stability of the program, the use of threads is not recommended within any program called by Sub.P.
2) Commands such as Move or Circle Functions cannot be placed within a thread.
3) When using Pause or Alarm function, both main program and thread are paused.
4) When the main program exits, the thread will also exit – even if the thread has not yet finished executing.

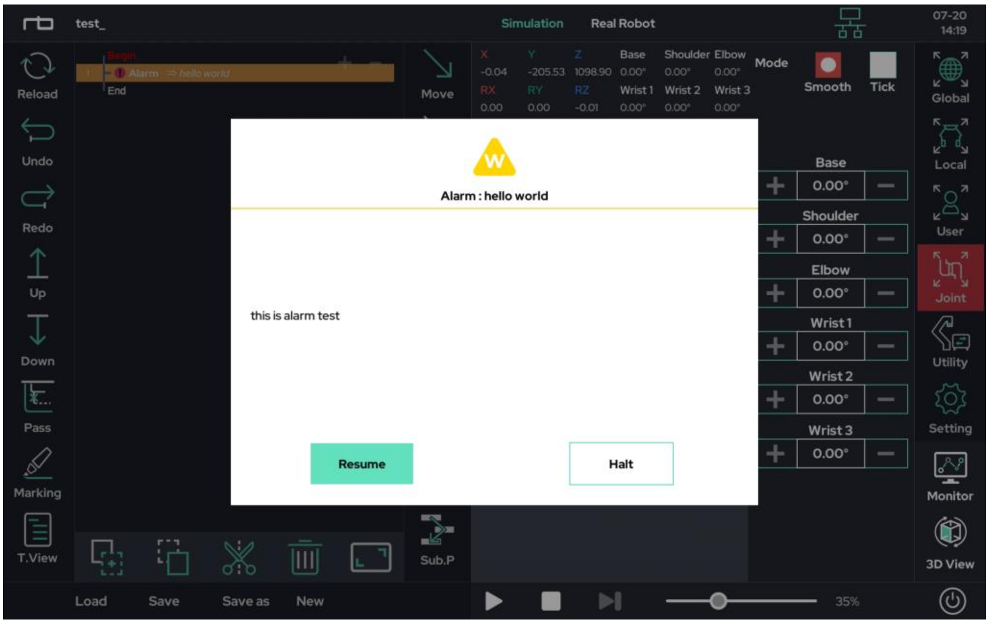
■ Alarm Function：


Places a popup message within the flow of the program. The message will disrupt the execution of the program, prompting user confirmation to continue or stop the program.

After clicking the Alarm icon, an Alarm will be placed within the program tree. Click the new Alarm to display the setting window as shown below.




Enter the title and content of the alarm window. The title will appear at the top of the popup, and the content will provide more in-depth information about the alarm. The below image is an example of a user-made Alarm.




To better control the flow of the program, the user can either Resume or Halt the program's execution from the pop-up.

- Resume: Continue to the next command
- Halt: Terminate the program

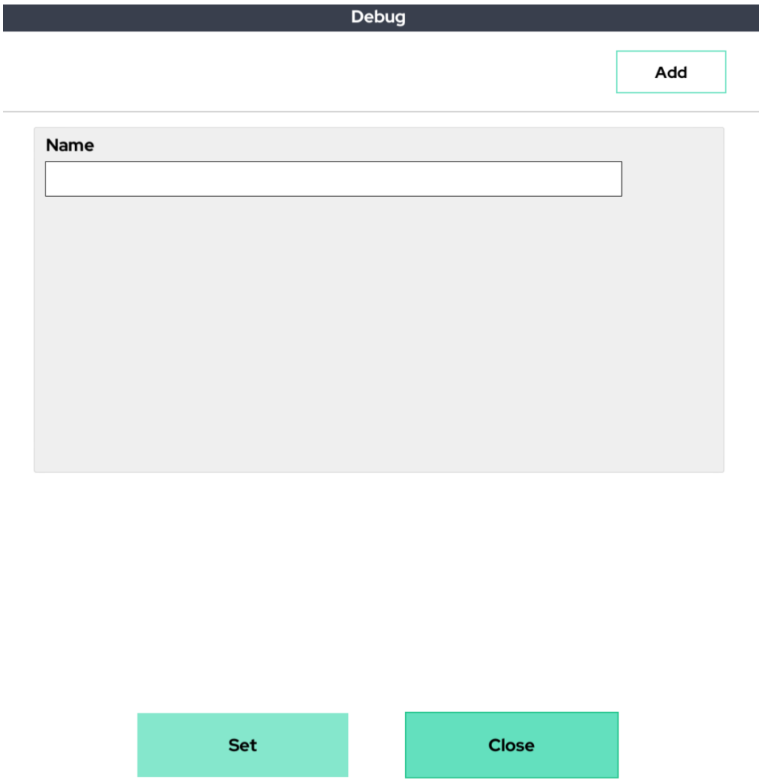
Pressing the Resume button in the pop-up window will resume the program, whereas pressing the Halt button will stop program at this point.

When using the alarm function, both the main program and thread programs are paused and at the same time.

■ Debug Function :


Function for debugging internal values. Users can make a pop-up display the value of a variable or internal parameter, similar to an Alarm.

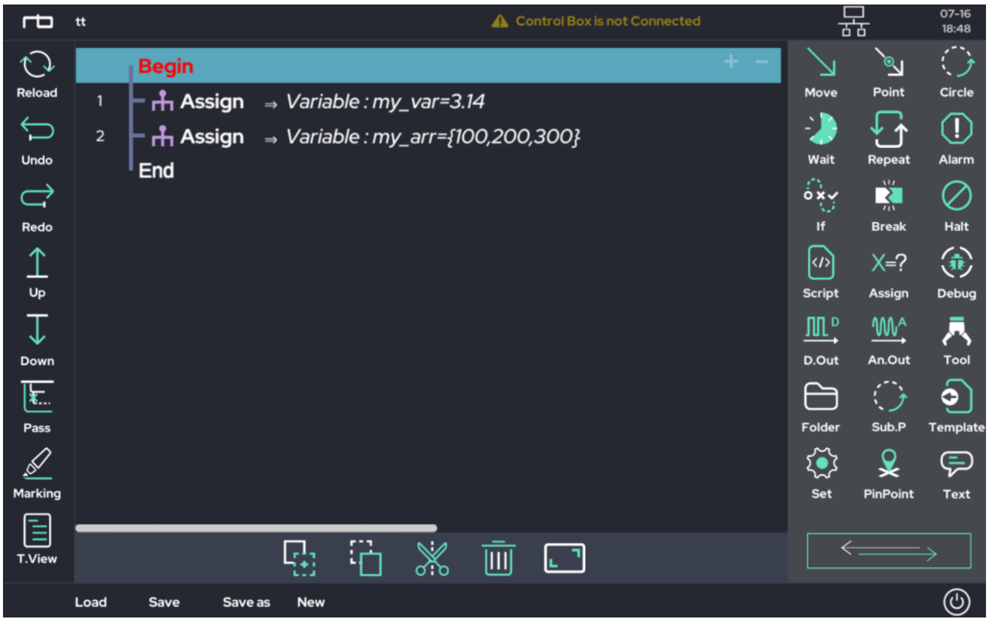
Debugging is for observing internal variables. It is mainly used to check the value of variables used in the program during program teaching / development.



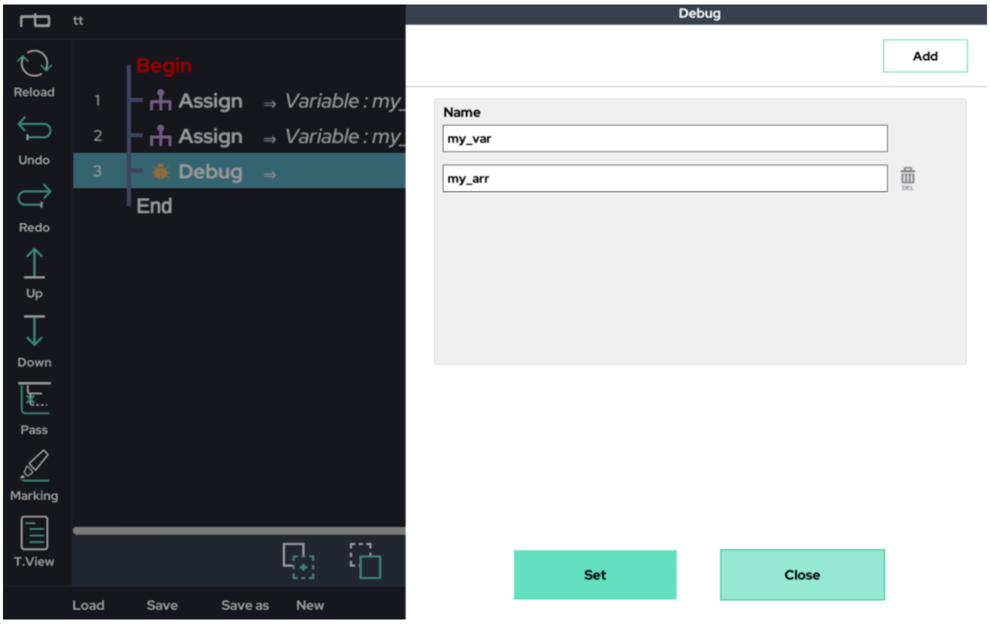

After adding the debug function to the program tree, click Debug to see the popup window as above. Enter the variable name in the Name field to view how variables change. To observe several variables within the same popup, press the (Add) button to add another variable.

The follow is an example on using Debug.

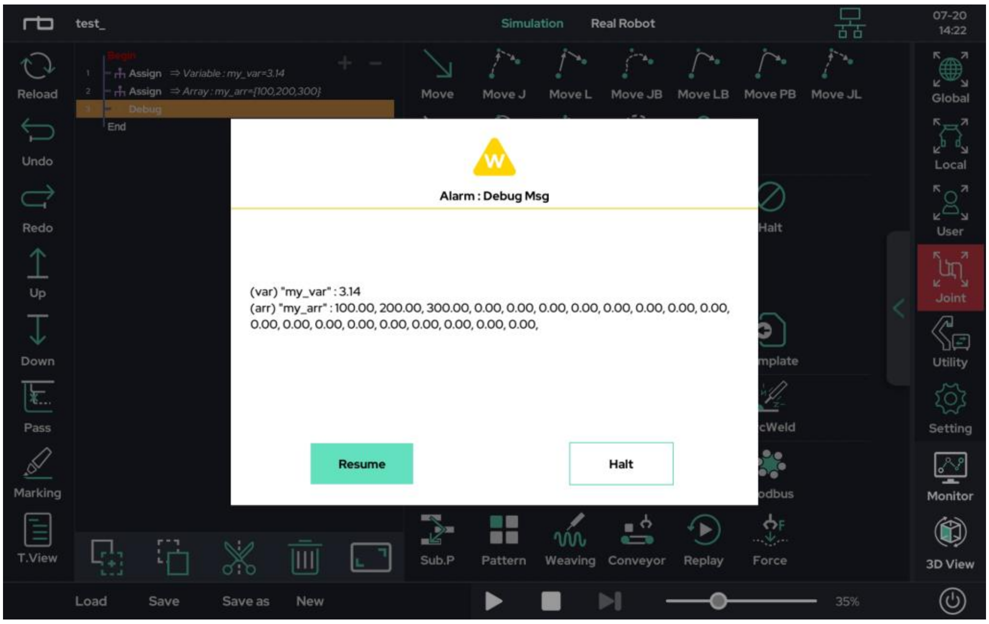
Declare one variable type variable (my_var = 3.14) and one array type variable (my_arr = {100,200,300}) using the Assign function as shown below.




Add a Debug Function below it. Set the variables in the Debug window to observe the two previously declared variables as shown below.



Once the setting is complete, run the program (the tablet PC and the control box must be connected before execution), and the following pop-up window will appear when the Debug command is executed. The pop-up will allow the user to observe the specified variable values.




- Resume: Continues to the next command.
- Halt: Terminates the program.

■ Set Function：


The Set function allows users to temporarily change parameter settings, regardless of the default values contained within the Setup menu. While the settings in the Setup menu are applied as defaults to all projects, the Set function allows users to temporarily override these parameters.

The various parameters that you change on the Setup screen are applied as default values for all projects that use that control box. If you need to use certain parameters separately for a particular project, you can manage parameter settings by project by adding the Set function to the top of the project (for example, Pre.P. sub).

The Set function is a temporary setting, not a permanent setting. When a new Set function is called for the same parameter setting, the parameter is reflected based on the new Set function.

When the program ends, the parameter settings will return to the default values as defined within the Setup menu.

The parameters that can be changed via the Set function are as follows:

- Time
- Collision Threshold
- Tool Payload
- Linear Move Offset
- Inbox check mode
- TCP Position
- Tool Collision Box
- Global Workspace
- Inbox size
- Collision Check On/off

- Overall speed multiplier
- Overall acceleration multiplier
- Serial communication configuration
- Fixed Velocity / Acceleration
- Spiral circle mode
- UI speed control bar
- Stop mode after collision detection
- User coordinate center shift
- Program flow after collision detection
- Disable Box D.out
- XYZ Projection
- Orientation Align
- User Coordinate Config
- XYZ Shift
- XYZ Shift2
- Vibration sensor
- Digital Input Simulation
- Program Flow Control
- High acceleration Mode
- Motion Time Constraints
- High Sensitivity Coll.Detect
- Micro offset value
- User Coordinate Shift 6D
- User Coordinate Auto Alignment
- Timer Setting
- No-Arc Move speed

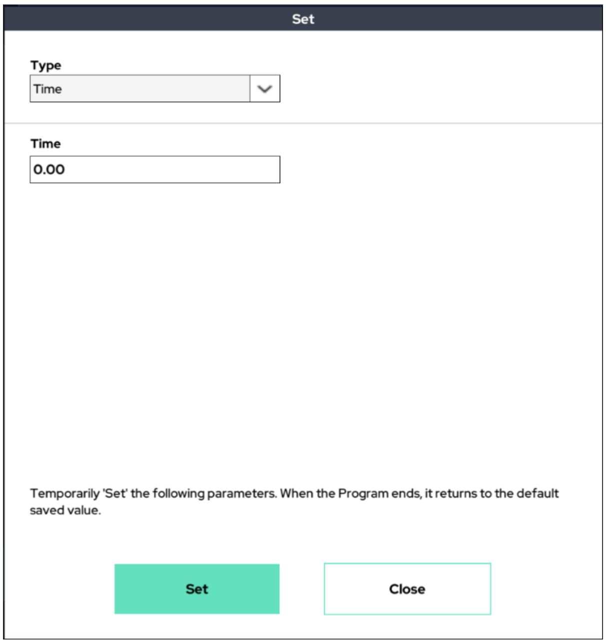
**Warning**

Warning:

1) The value set in the Set function is a temporary value. When the program exits, it automatically returns to the default values set from the Setup Menu.

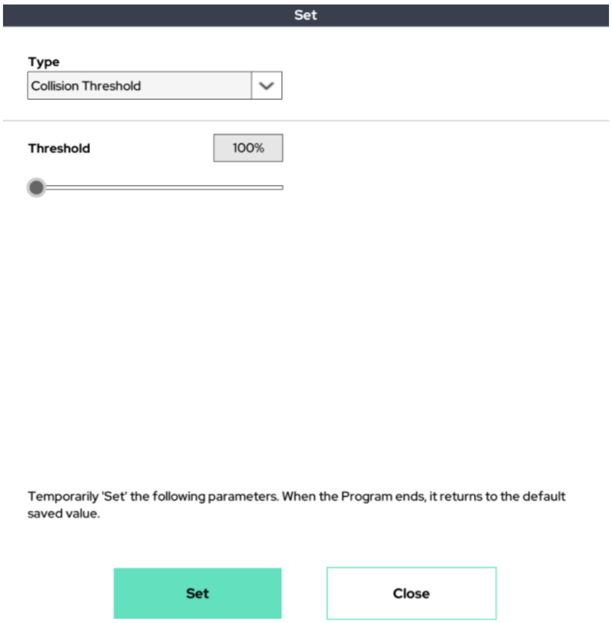
2) The functions provided by the Set function allow you to change the setting value to another value in the middle of the program flow.
   For example, you can use Set's 'Collision On / Off' feature to selectively turn on/off collision detection in the middle of a program flow.

## Set Function: Time



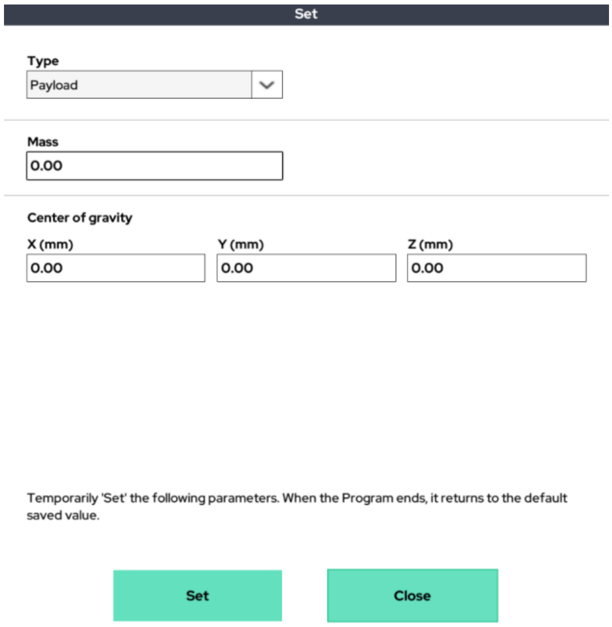

Starts the timer and sets the initial value. Starting with the value entered, the value of the timer increases.

## Set Function: Collision Threshold Change




Temporarily sets the collision detection sensitivity. The lower the value, the more sensitive the robot is to collision. This has the same functionality as the Collision Threshold option within the Setup Menu.

## Set Function: Tool Payload

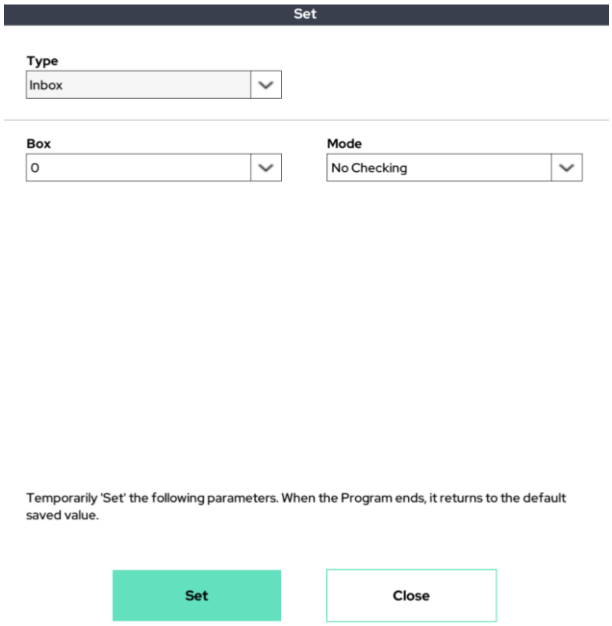



Temporarily set the tool's weight and center of gravity. This has the same functionality as the Payload option within the Setup Menu.

## Set Function: Linear Move Offset


Gives a slight offset relative to the base coordinate system. This function allows users to temporarily set an offset of up to 20 mm.

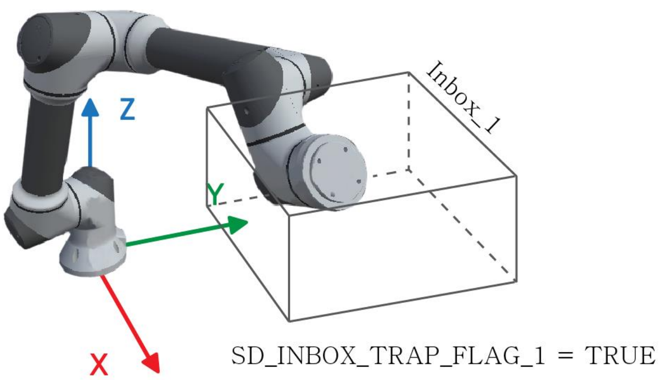
## Set Function: Inbox Check mode




Allows the user to enable the Inbox Checking feature. The Inbox Checking feature checks whether a certain part of the robot is in a predefined area (either in the Setup screen or the using the Set function). The parts of the robot that can be checked are as follows.

- Is the center of the tool flange inside the specified area?
- Is the TCP inside the specified area?
- Is any part of the gripper (tool box) in the specified area?
- All the above

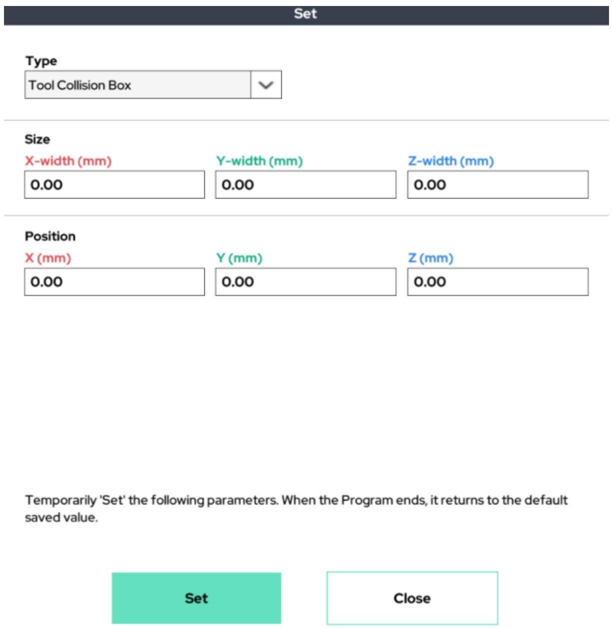
The size and position of the box can be set in the Inbox screen using Setup mode (or through the Set function). After enabling the Inbox Checking feature, the user can use the value via Script, If, or some similar function. Under the "Type" box, choose "Shared Data." Then under the "List" box, use either the SD_INBOX_TRAP_FLAG_0 (not in the box) or SD_INBOX_TRAP_FLAG_1 (in the box) variables.



Set Function: TCP Position

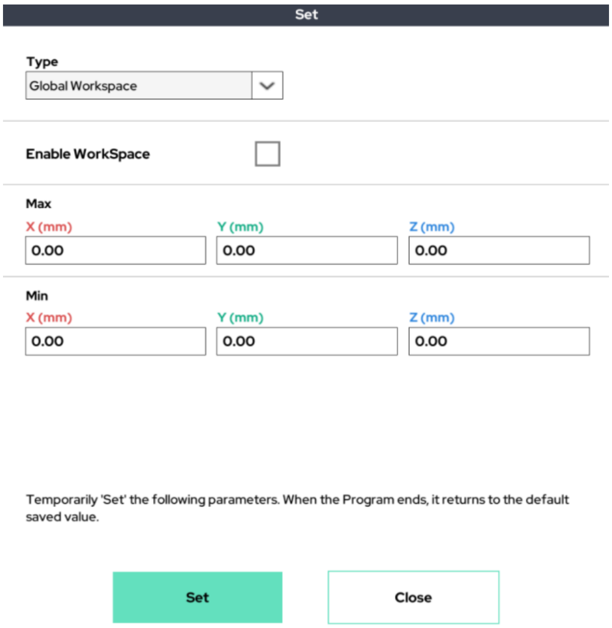

Temporarily set a relative offset of the tool's TCP position. Note: This will change the X, Y, and Z used for Global TCP calculations. It has the same functionality as the End Effector menu in Setup-Tool.

Set Function: Tool Collision Box



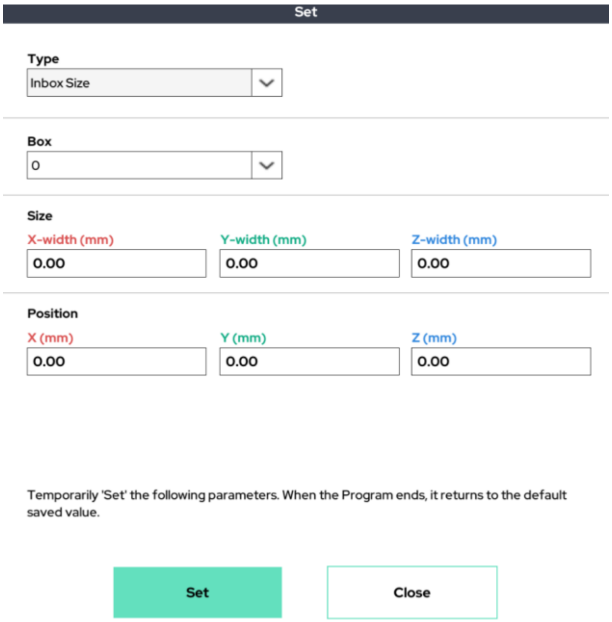

Temporarily set the size and position of a virtual box surrounding the gripper for self-collision prevention. The size and position of the virtual box will be relative to the TCP Position. It has the same functionality as Tool Setting for Collision Check in Setup-Tool.

Set Function: Global Workspace



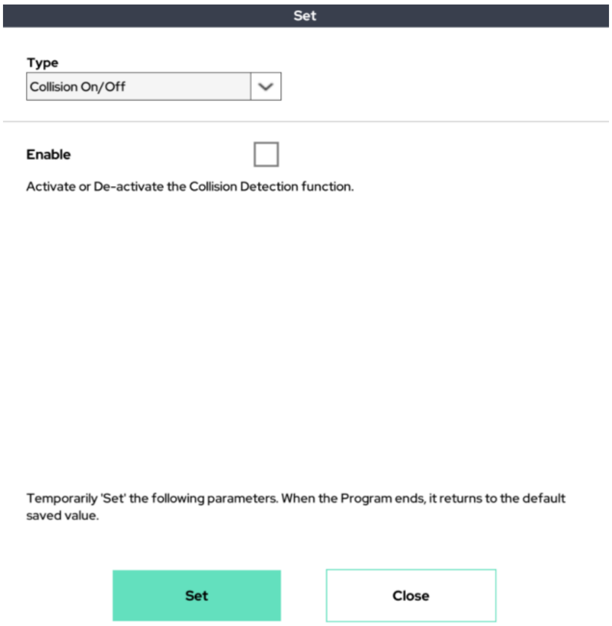

Temporarily set the limits of the workspace for collision prevention. It has the same functionality as the Workspace Limits menu in Setup-Cobot.

Set Function: Inbox Size



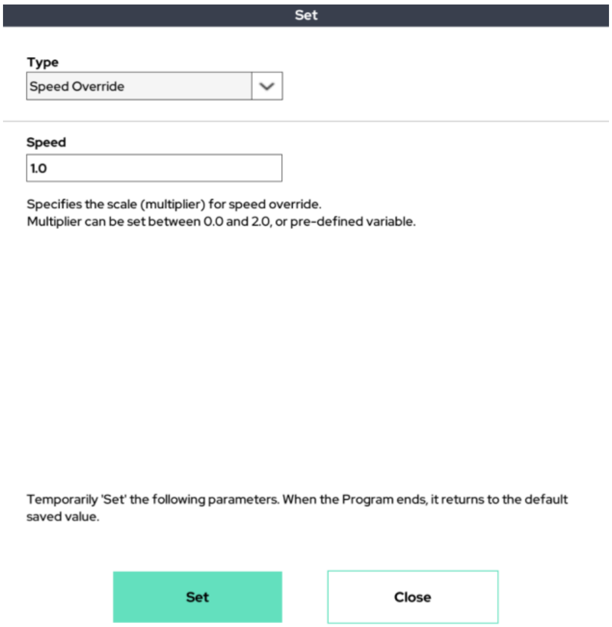

Temporarily set the position and size of the Inbox. It has the same functionality as the Inbox settings in Setup-Inbox.

Set Function: Collision Detection On/Off

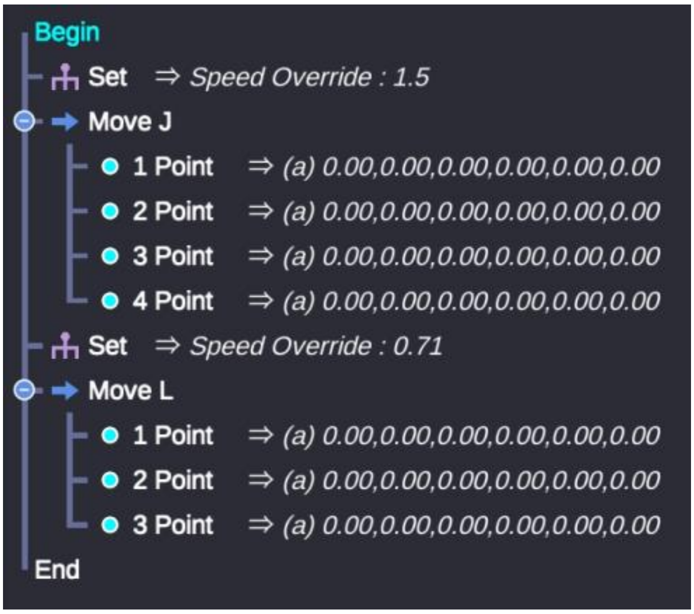



Temporarily sets the use of external collision detection mode. It has the same functionality as the Enable Collision box in Setup-Cobot.

Set Function: Speed Override (speed multiplier)

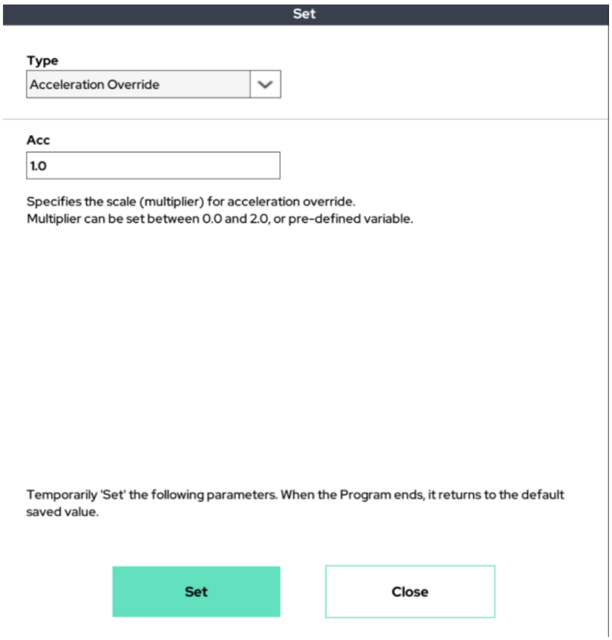



Allows the user to temporarily to change the base scaled speed used by the Move and Point functions. Users can either enter a value between 0 and 2.0, or a predefined variable.



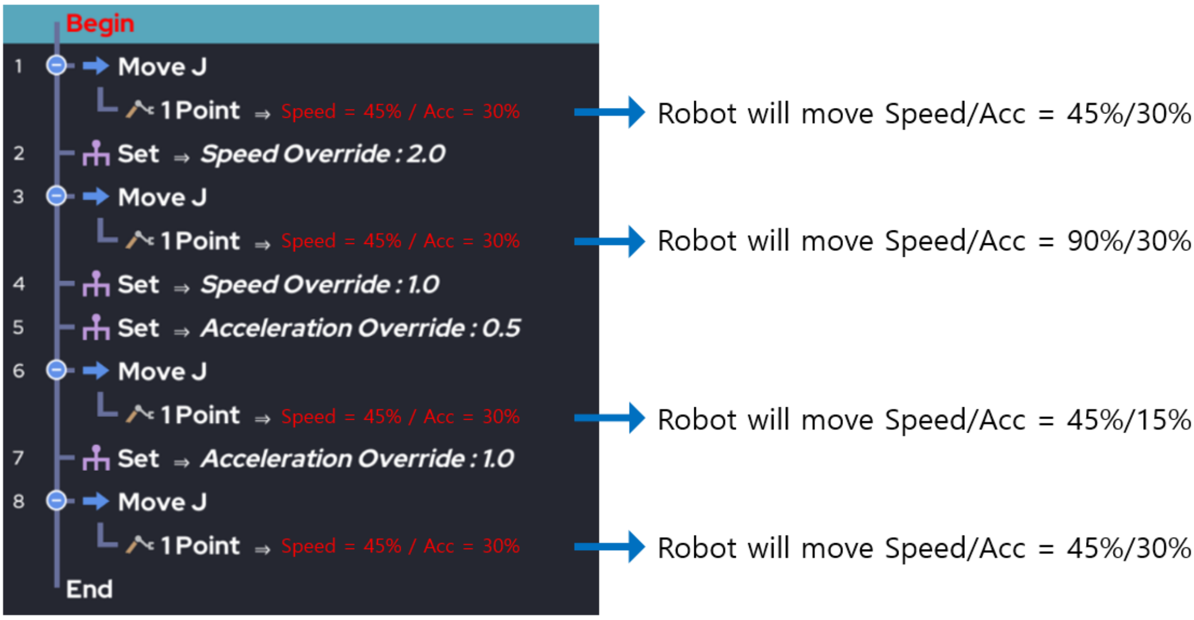

In the example above, the base speed for Move J is overwritten to be 1.5 times the normal speed, whereas the base speed for Move L is overwritten to be 0.71 times the normal speed.

Set Function: Acceleration Override (acceleration multiplier)

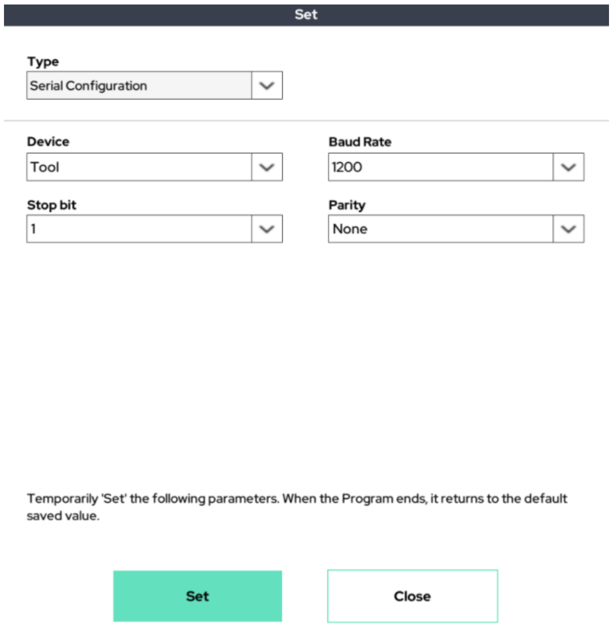



Allows the user to temporarily to change the base scaled acceleration used by the Move and Point functions. Users can either enter a value between 0 and 2.0, or a predefined variable.

Through the code below, you can see how the speed and acceleration change when Speed Override and Acceleration Override are used.

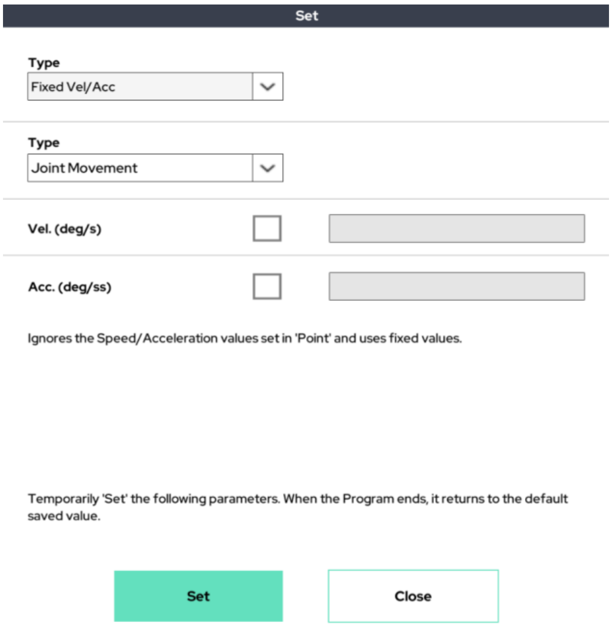


Set Function: Serial Communication Configuration




The baud rate and stop bit / parity of the serial communication are temporarily set. It has the same meaning as set in Setup-Serial.

Set Function: Fixed Velocity/Acceleration



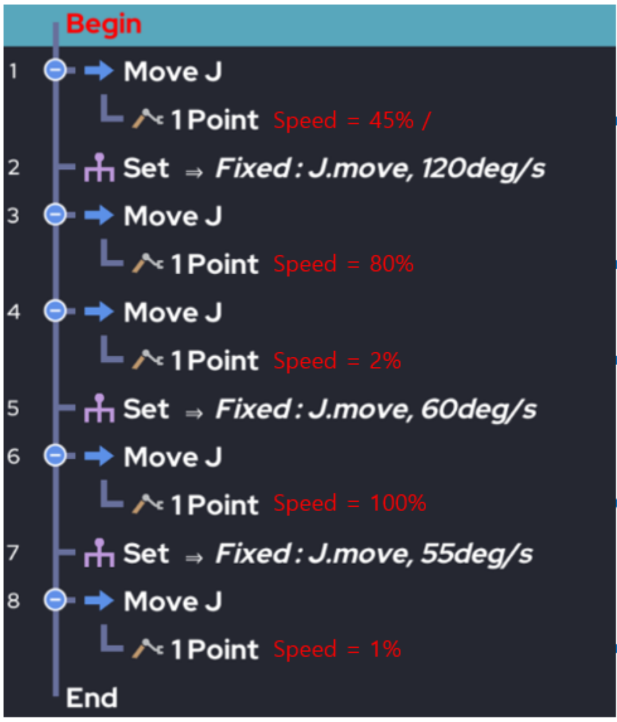

This function is used when you want to use a fixed value, ignoring the set speed / acceleration for each Move point. There are two sub options: Joint Movement and Linear Movement.

The velocity (deg / s) and acceleration (deg / s ^ 2) set in the Joint Movement affect the movement speed and acceleration of the Joint movement types MoveJ and MoveJB.

The velocity (mm / s) and acceleration (mm / s ^ 2) set in Linear Movement affect the movement speed and acceleration of the linear movement types MoveL, MoveLB, MovePB, MoveJL, MoveITPL and Circle.

If you do not want to force speed / acceleration through this function, clear the check box. In this case, it follows the speed / acceleration value set for each point during operation.

Ex) If you need to keep a certain speed and acceleration during operation, you can use this Set function as in the code below.




Robot will move Speed 45 %

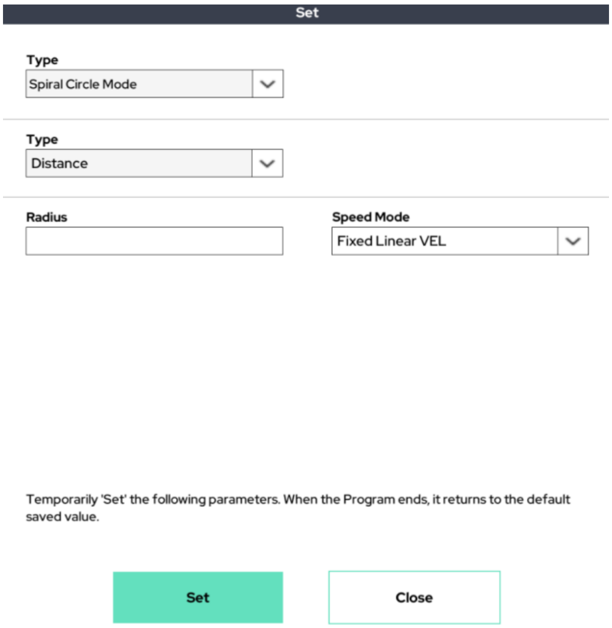
Robot will move 120 deg/s

Robot will move 120 deg/s

Robot will move 60 deg/s

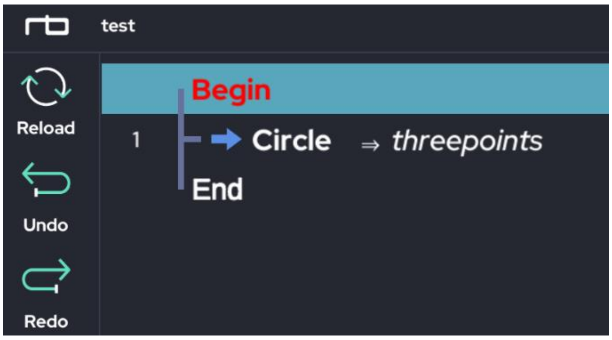
Robot will move 55 deg/s

Set Function: Spiral Circle Mode

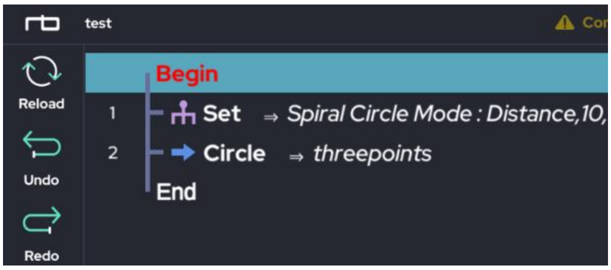



This function is used to change the circular motion into spiral motion. Draw a circle / arc when using the Circle function. If Set-Spiral mode is used over the Circle function, the existing circle / arc will be changed to spiral motion. Therefore, to implement spiral motion, this function should be inserted above the Circle function.

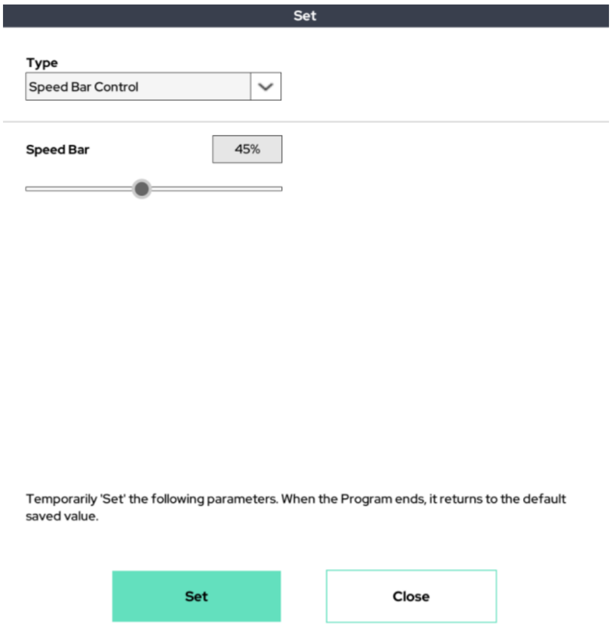
Ex 1) Only Circle is used: Create a general circle / arc trajectory




Ex 2) Set-SpiralMode + Circle: Spiral trajectory

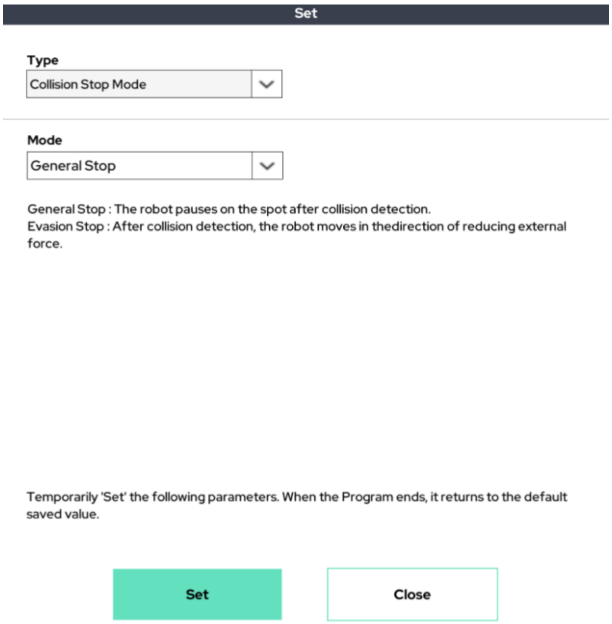


Set Function: Speed Bar Control




The speed control bar (bottom right) of the UI can be adjusted with the program. You can change the UI speed control bar by using this function in the desired section.

Set Function: Collision Stop Mode



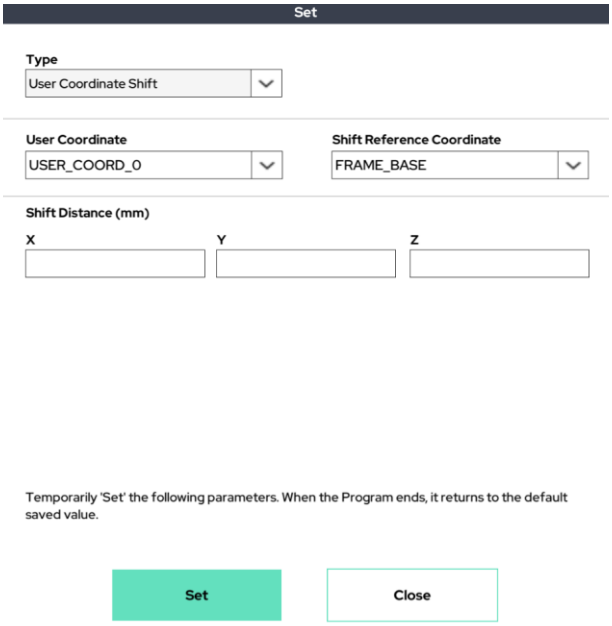

Select the robot's motion type when after detecting an external collision. There are two options.

General Stop: After the collision is detected, the trajectory movement is paused on the spot.

Evasion Stop: After the collision is detected, the robot moves a small amount away from the external force, then pauses the trajectory movement.

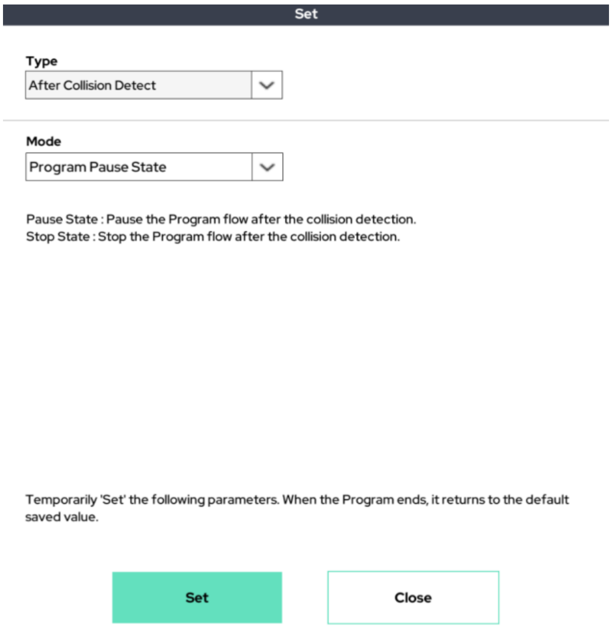
It has the same meaning as Setup-Cobot's "Action after Collision".

Set Function: User Coordinate Shift




This function is to move the origin of user coordinate system temporarily. You can set the user coordinate system number and shift distance you want to shift and choose which coordinate system to shift the shift distance.

Set Function: After Collision Detection




The program flow can be selected after external collision detection.

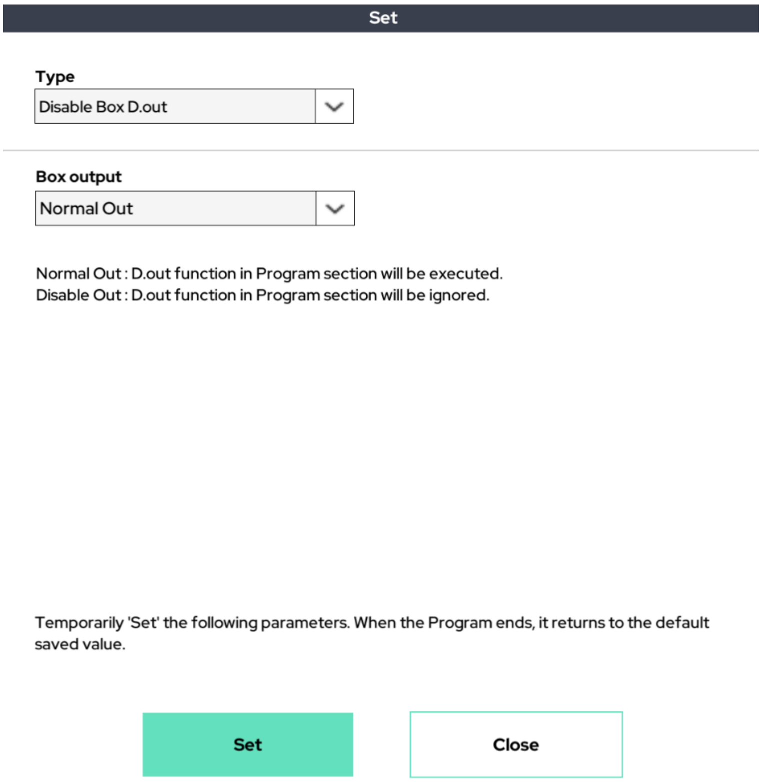
Our default setting is to pause the program after detecting an external collision. After detecting a collision, a collision detection alert pops up and the program and threads are paused.

If you want to terminate the program after collision detection, you can use this function to select the option as Stop state.

Pause State: Program flow is paused after external collision detection.

Stop State: Program flow stops after external collision detection.

Set Function: Disable Box D.out

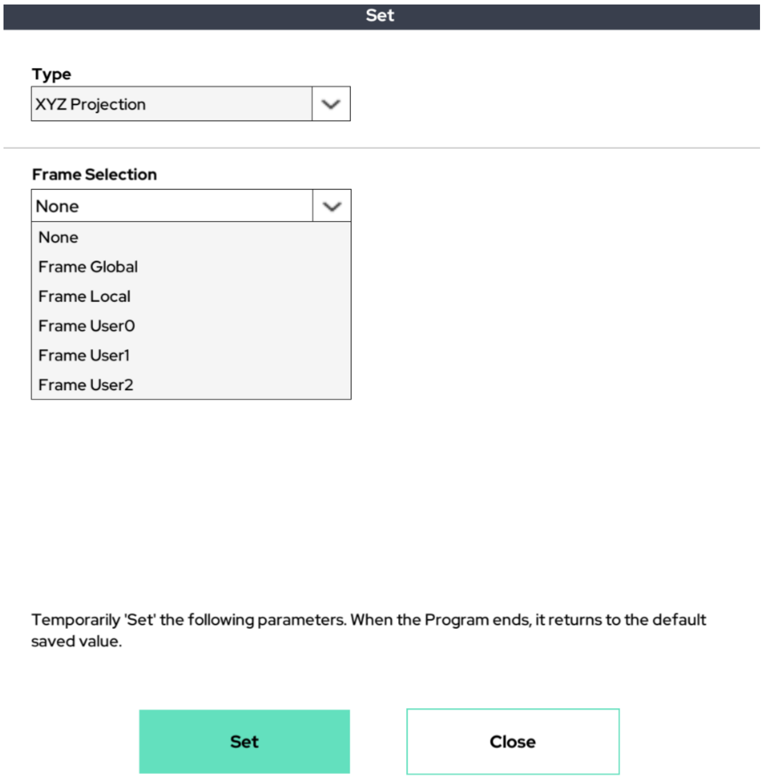



This function temporarily disables the digital output of the control box.

Even if the digital output command inserted in the program is not erased, this set command can be used to ignore the digital output command in a specific section.

It can be used for development testing, etc., and by selecting an option, the output can be deactivated/activated according to the program section.

Set Function: XYZ Projection

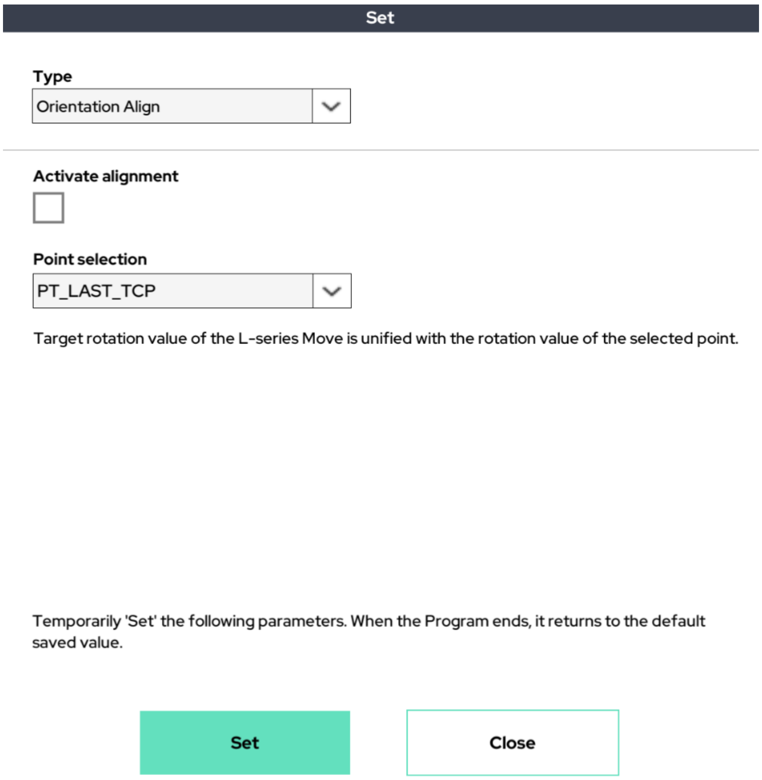



This is a function to fix the target position coordinate value of L series movement (eg MoveL. MovePB, Circle etc). If you select the value to be fixed and the reference coordinate system, the position coordinate value of the target point or set point is fixed to the value of the selected axis of the selected coordinate system.

For example, if the base coordinate system (Global) is selected as the coordinate system and Z Projection 100mm is selected/written, the Z height of all moving target values/set coordinate values is applied collectively as 100mm.

This function is also a set function, which can be activated/deactivated for each section of the program. If you want to disable it, select None in the coordinate system.

Set Function: Orientation Align

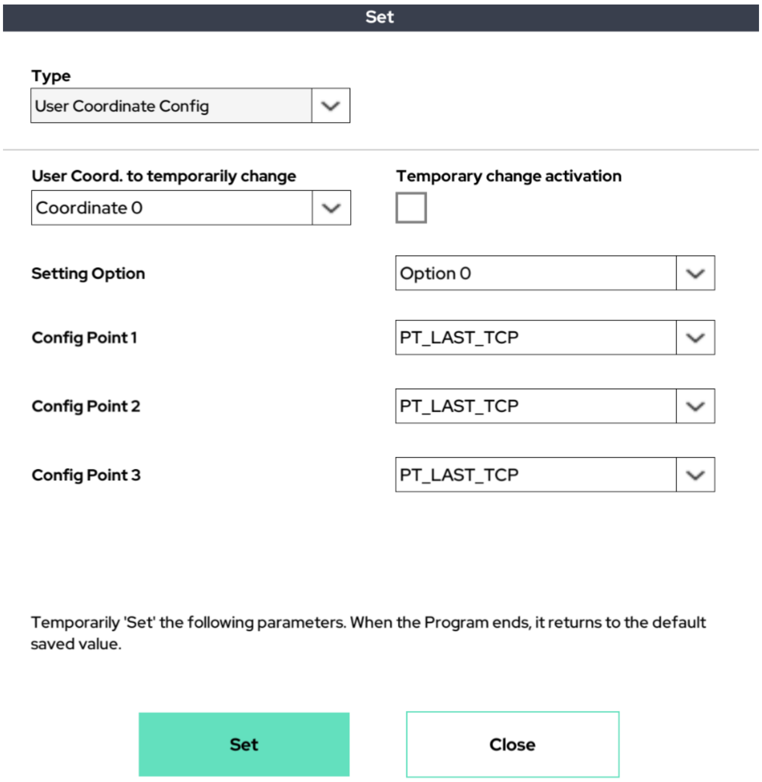



This function is to fix the target rotation coordinate value of L series movement (eg MoveL. MovePB, Circle etc).

Fix the rotation of L series motions with the rotation value of the selected Point.

As a sub-function of the Set function, this function can be turned on or off depending on the program section. This function can be used when you want to uniformly rotate the TCP rotation at a time.

Set Function: User Coordinate Config

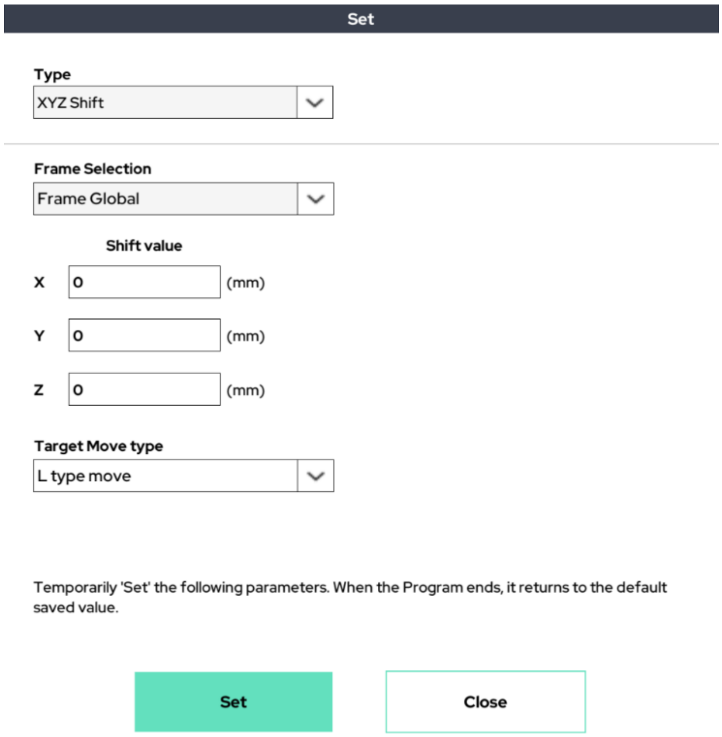



This function allows you to temporarily change the user coordinate system settings.

By selecting three points in the middle of the program flow, the user coordinate system setting can be arbitrarily changed in the middle of the program.

Because it is a sub-function of Set, the user coordinate system setting returns to the default value when the program ends.

## Set Function: XYZ Shift

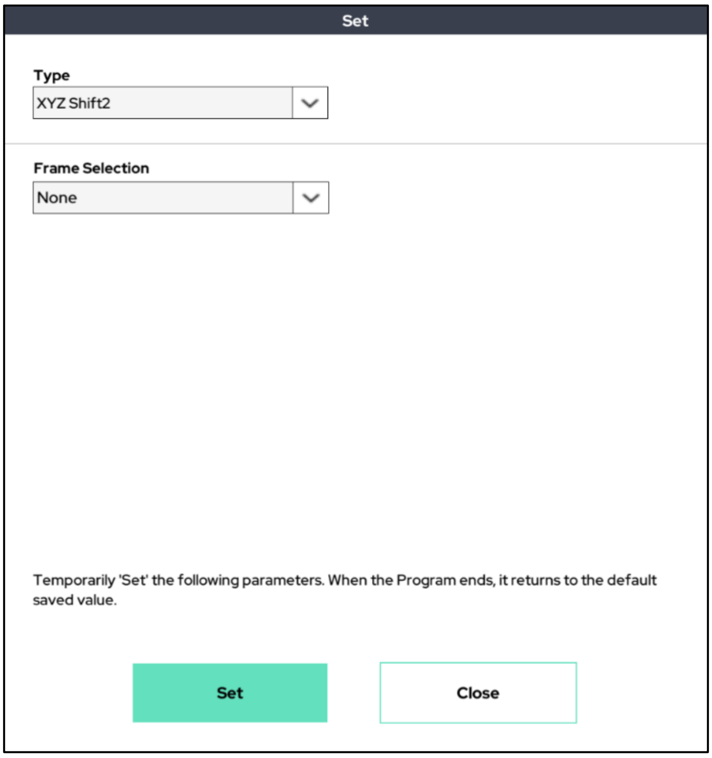



This function allows you to temporarily shift the target point.

User can select a base/tool/user coordinate Config and enter shift values from the target point.

At this time, select whether to apply this shift only to L type or to both L type and J type.

Set Function: XYZ Shift2



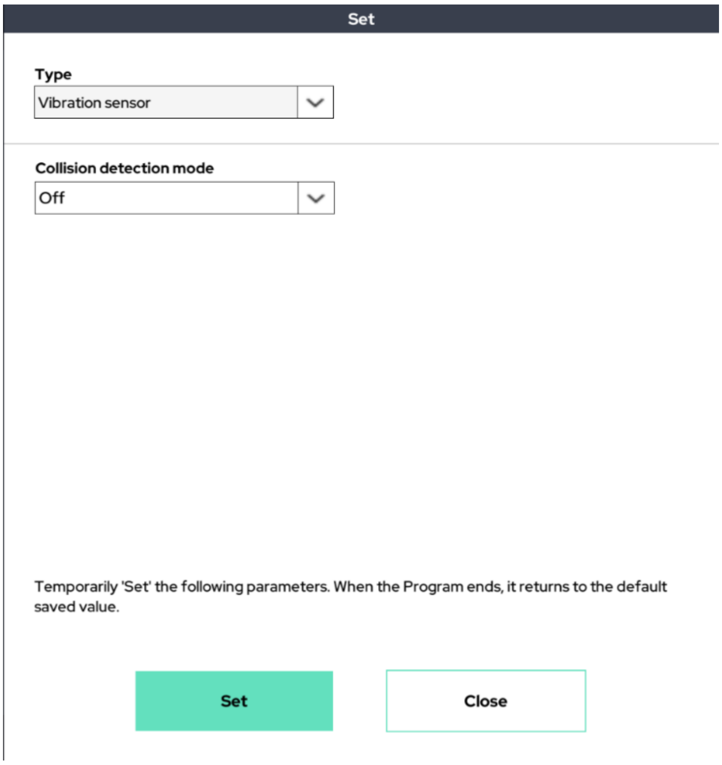

This function allows you to temporarily shift the target point.

User can select a base/tool/user coordinate Config and enter shift values from the target point.

At this point, this shift is only applicable to L series operation, and both the XYZ position value and the rotation value can be entered.

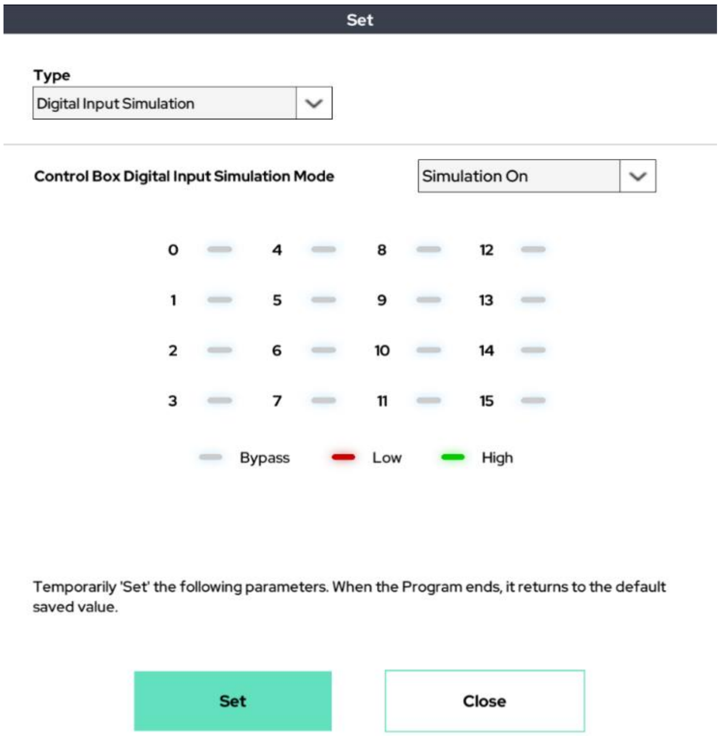
At this time, select whether to apply this shift only to L type or to both L type and J type.

Set Function: Vibration sensor




This function allows you to temporarily exclude collision detection by vibration during collision detection.

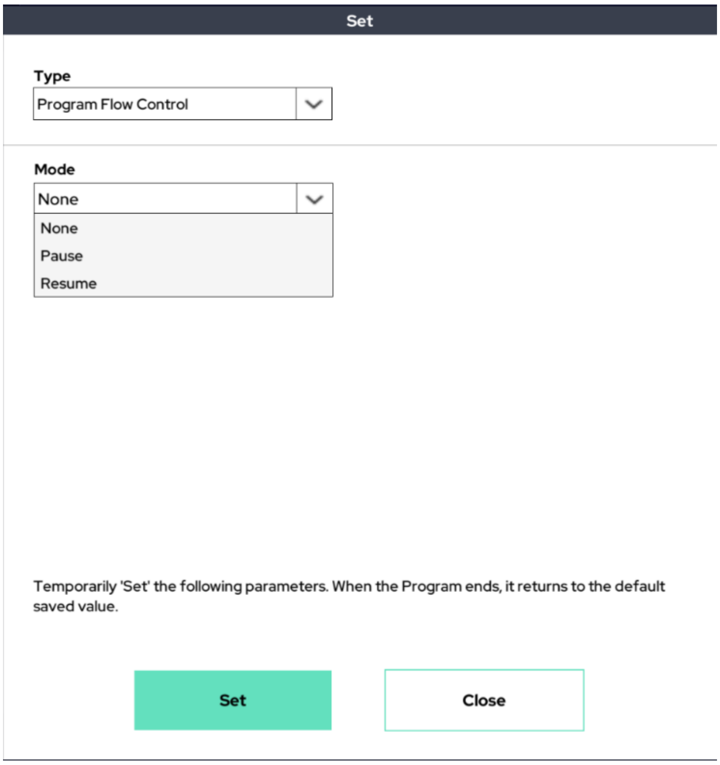
Set Function: Digital Input Simulation




This function allows you to simulate Digital input signal.

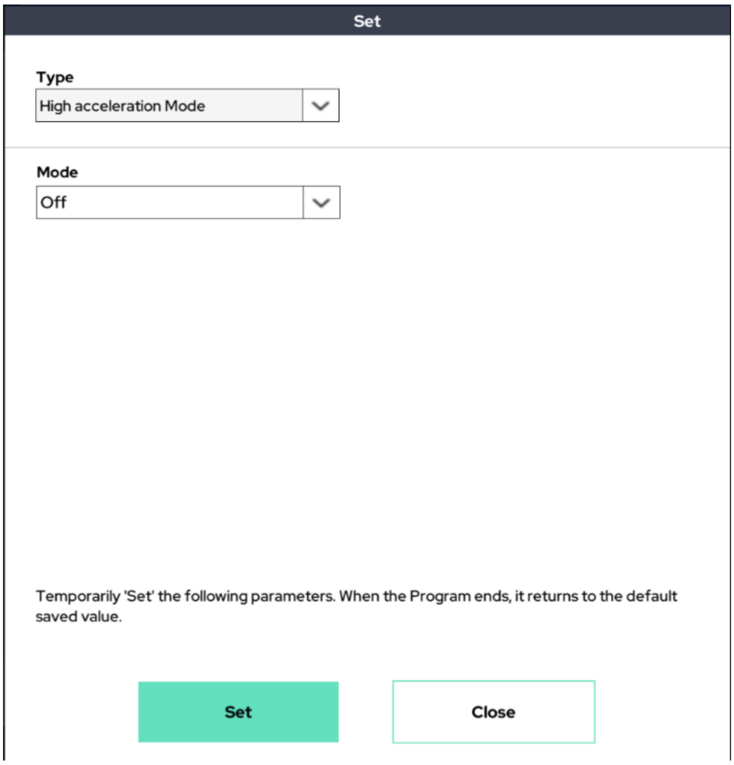
Create the desired input by setting the state of the port to which you want to input.

Set Function: Program Flow Control




This function allows you to pause and restart without using alarms and I/O when a program is running.

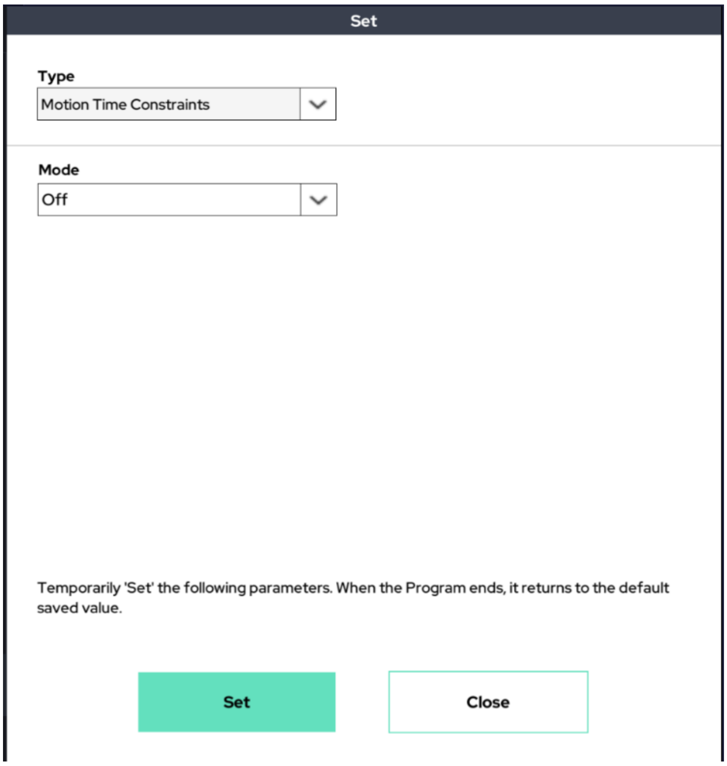
Set Function: High acceleration Mode




High acceleration mode reduces the time the robot's operating speed reaches the desired operating speed through changes in the reduction/acceleration profile.

This function allows you to pause and restart without using alarms and I/O when a program is running.

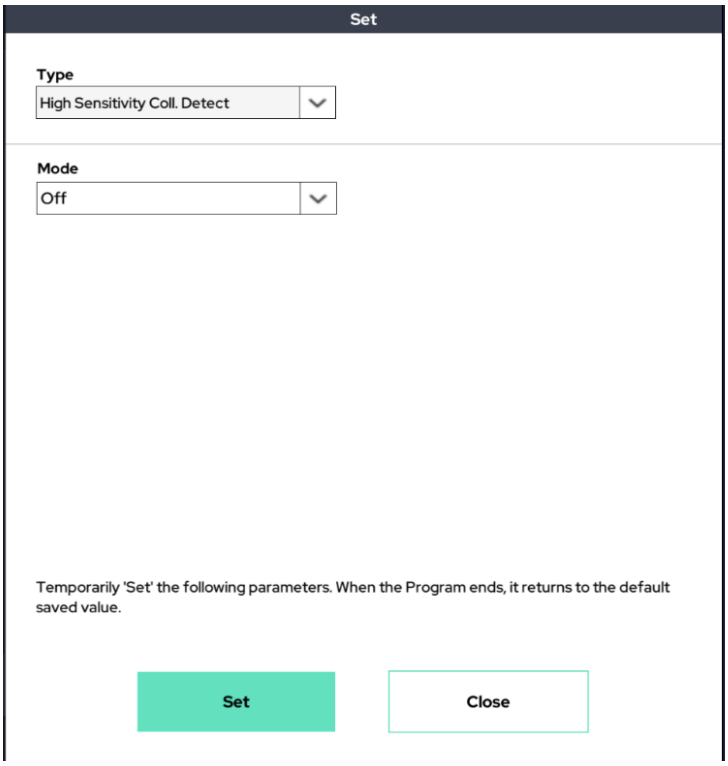
Set Function: Motion Time Constraints




Motion Time Constraint is a function that constrains the time taken to move a point to point by the time entered. At this time, it is possible to increase time but not to reduce it.

This function allows you to pause and restart without using alarms and I/O when a program is running.

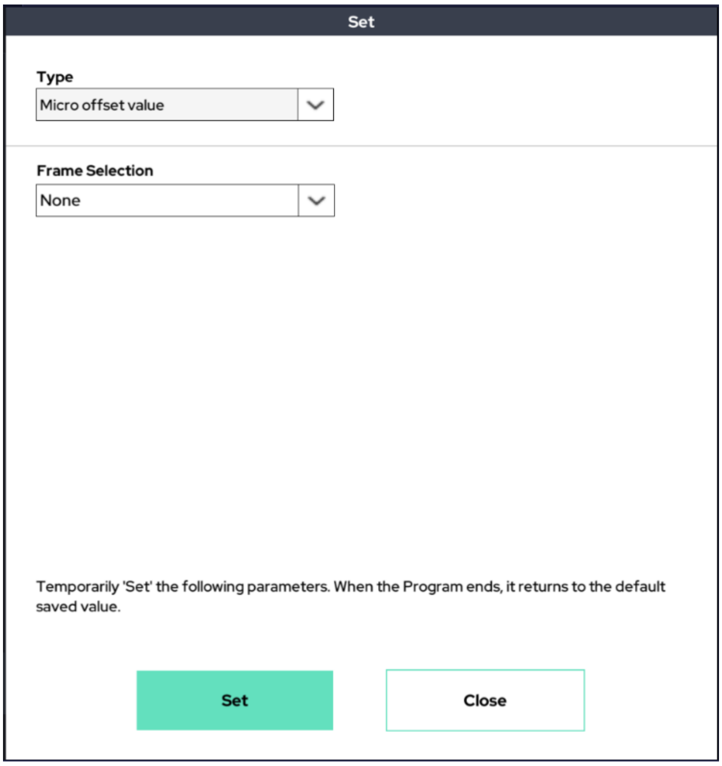
Set Function: High Sensitivity Coll. Detect




High sensitivity Coll.Detect allows the detection of collision to be 30% more sensitive than the existing sensitivity. In Setup, the sensitivity that made collision detection the most sensitive is also 30% more sensitive than 0%.

This function allows you to pause and restart without using alarms and I/O when a program is running.

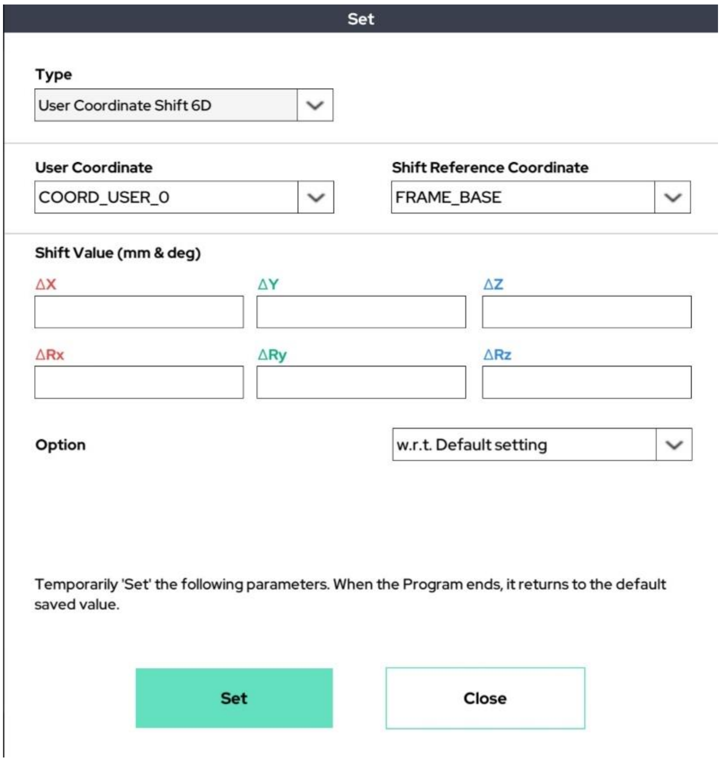
Set Function: Micro offset value




User can give a slight offset based on the desired coordinate system. This function enables temporary offset settings of up to 20 mm.

This function allows you to pause and restart without using alarms and I/O when a program is running.

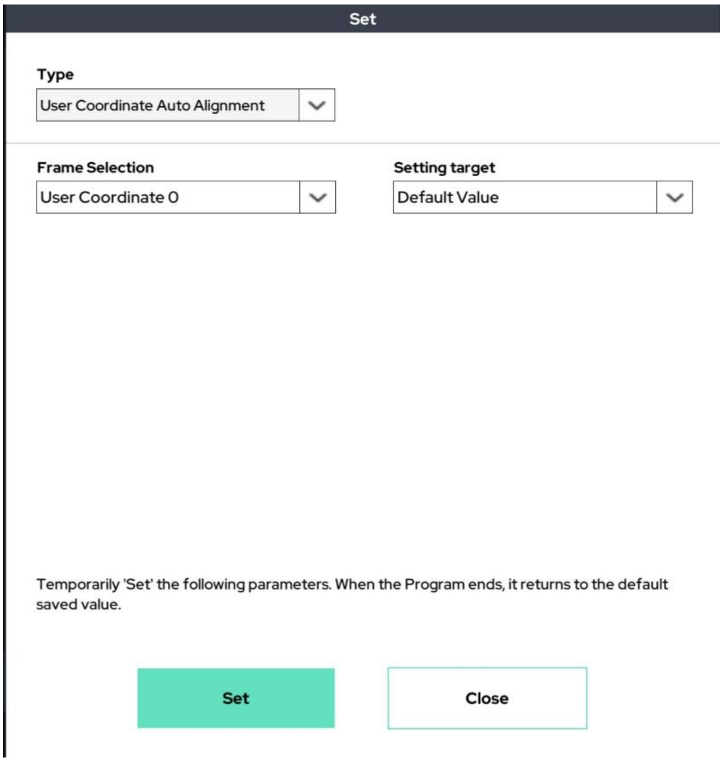
Set Function: User Coordinate Shift 6D




The user can temporarily shift the user coordinate. This function allows the user to temporarily change the position, rotation of the user's coordinate.

This function allows you to pause and restart without using alarms and I/O when a program is running.

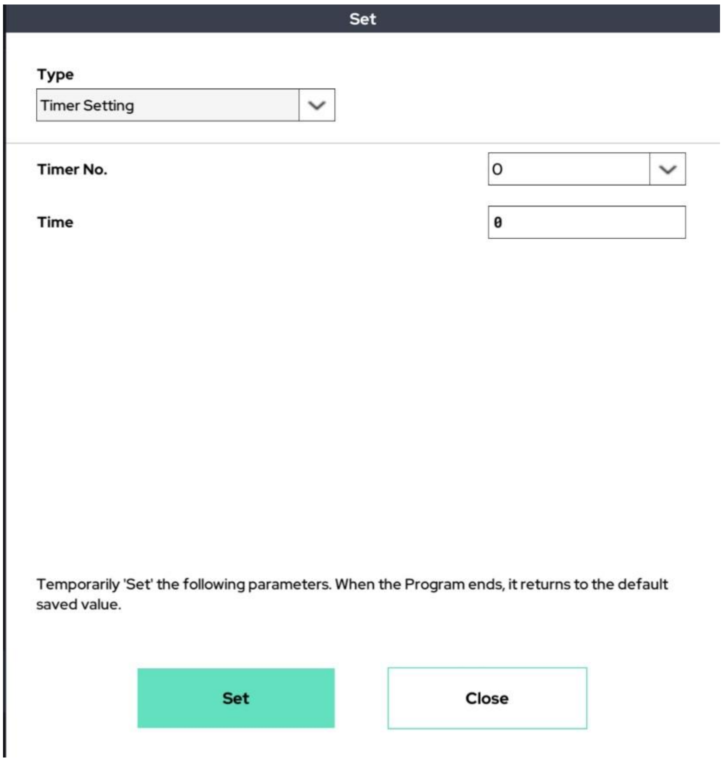
Set Function: User Coordinate Auto Alignment




This function allows the user to change the user coordinate to the last TCP frame. It is also possible to return to the default user coordinate.

This function allows you to pause and restart without using alarms and I/O when a program is running.

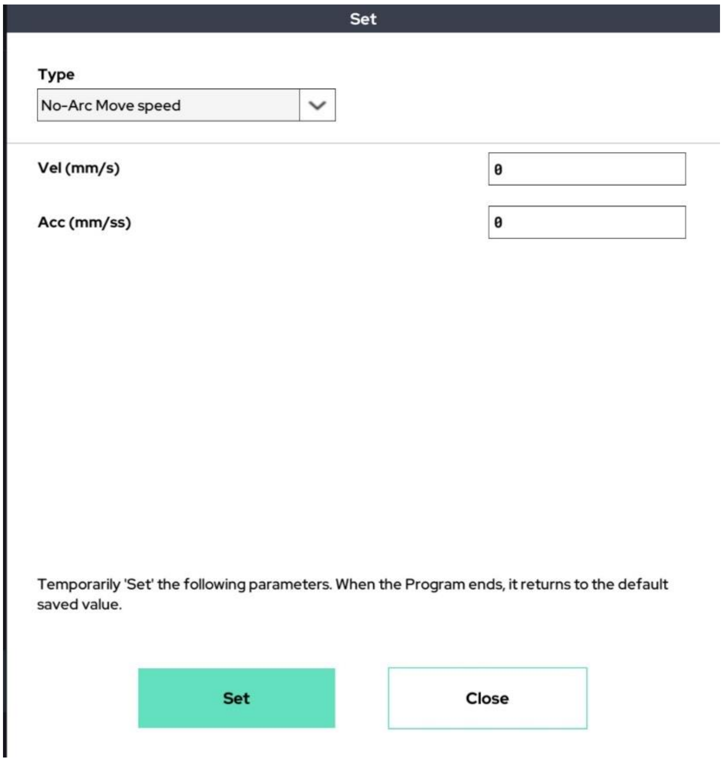
Set Function: Timer Setting




User can select the timer their want to use and set the initial value of the timer. The timer starts from the initial value set by the user.

This function allows you to pause and restart without using alarms and I/O when a program is running.

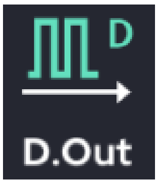
Set Function: No-Arc Move speed




Set the move speed of the robot in the no-arc state where welding is not performed.

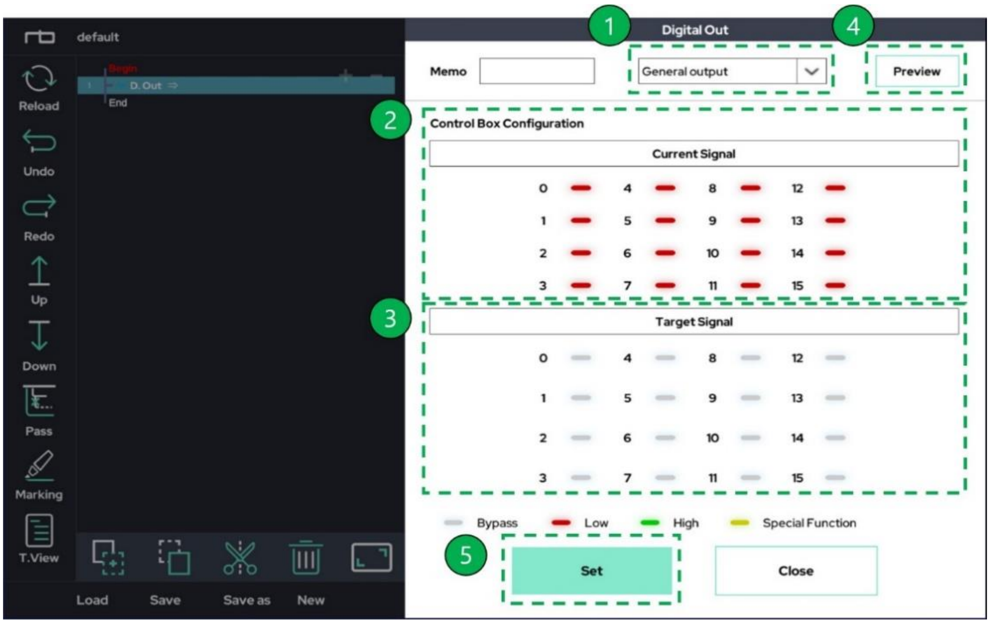
This function allows you to pause and restart without using alarms and I/O when a program is running.

■ D.Out (Digital out) Function :




Allows the user to set the digital output of the control box. The user can set the digital output signal of whichever port (0 ~ 15) they would like. Each port has three possible settings: high signal, low signal, and bypass.

After adding the D.Out function to the program, click on D.Out in the program tree to have the following pop-up window appear.




① Selection the detailed features available in the D.out function.

② Shows the status of the current Digital Out output from the control box.

③ Allows the user to set their desired setting for a port (0 ~ 15). The three setting toggles are Bypass, Low and High.

Bypass: Maintains the previous output signal state (gray).

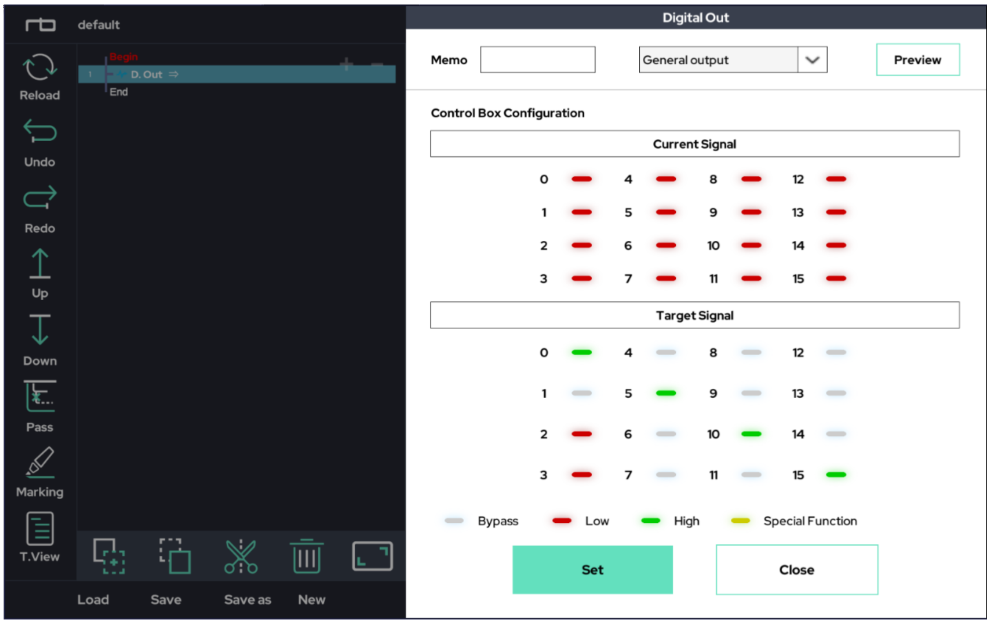
Low: Sets the output signal to the low (0) level (red).

High: Sets the output signal to the high (1) level (green).

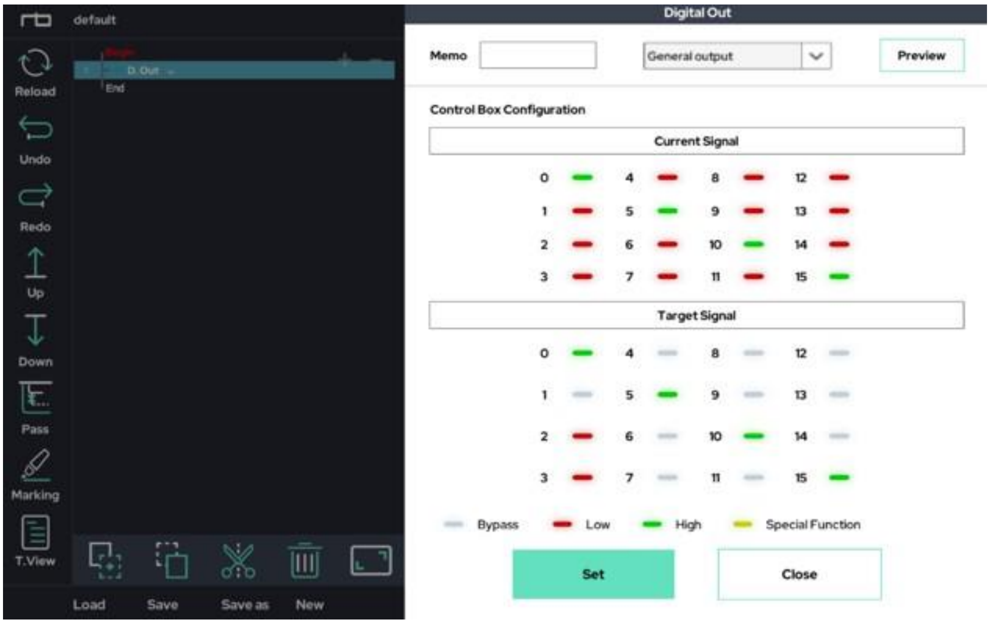
④ Allows the user to review the settings selected within the target signal menu. A further explanation is shown below.

⑤ Saves the settings specified within target signal menu.

Digital Out： General output



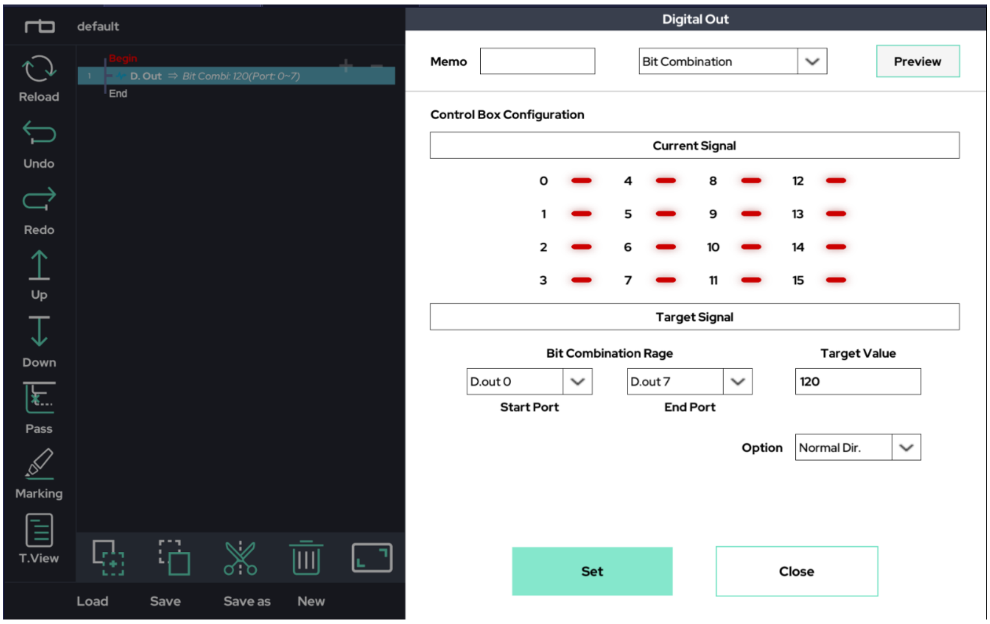

With the control box connected to the teaching pendant, set the Target Signal menu as shown above (to the right). Then, press the Preview button.



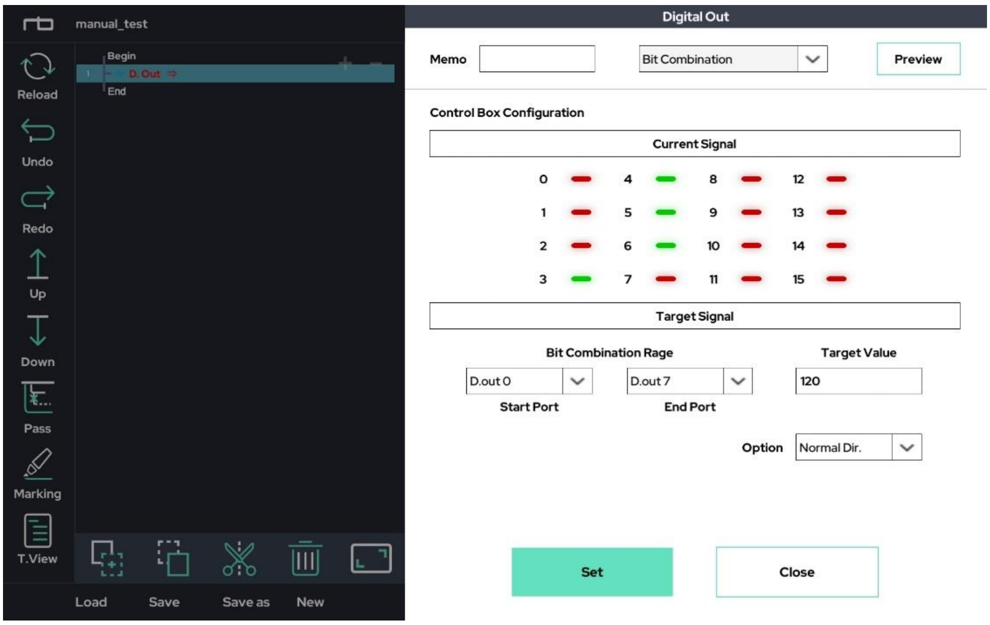

As shown above, the Current Signal menu will change to match the settings that the user has put in the Target Signal menu.

## Digital Out : Bit Combination

User can export the Digital output as a bit combination by selecting the start port and end port to use and entering the desired value in Target Value.



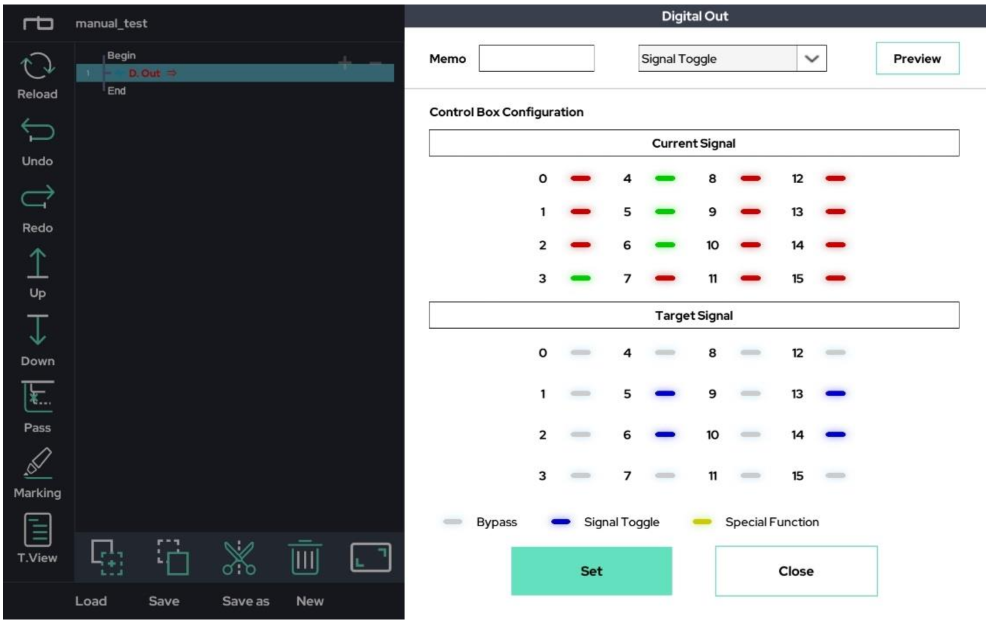

With the control box connected to the teaching pendant, set the Target Signal menu as shown above (to the right). Then, press the Preview button.



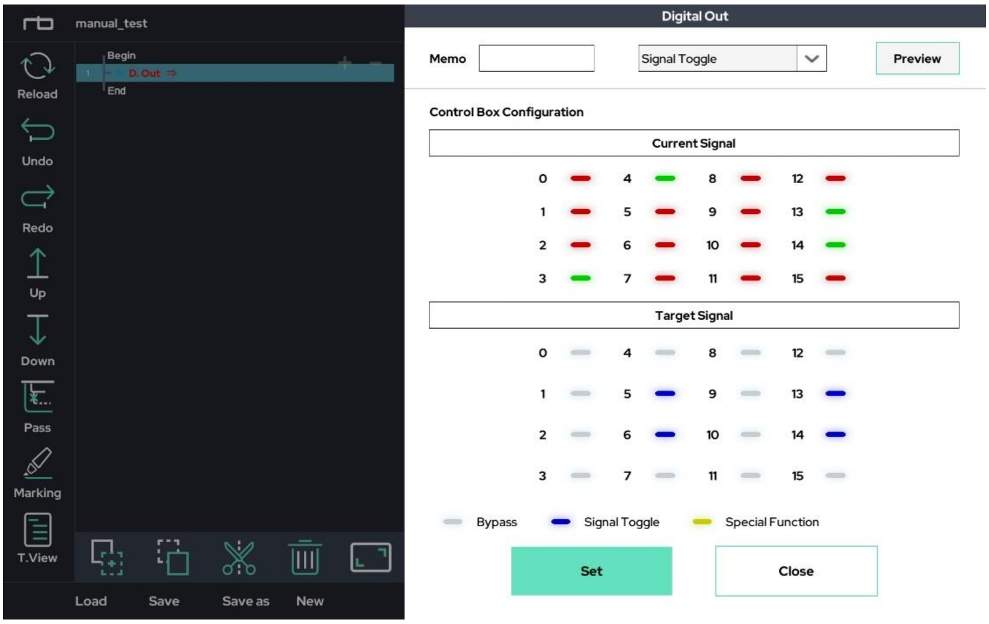

As shown above, the Current Signal menu will change to match the settings that the user has put in the Target Signal menu.

## Digital Out： Signal Toggle

To output by toggling a signal.



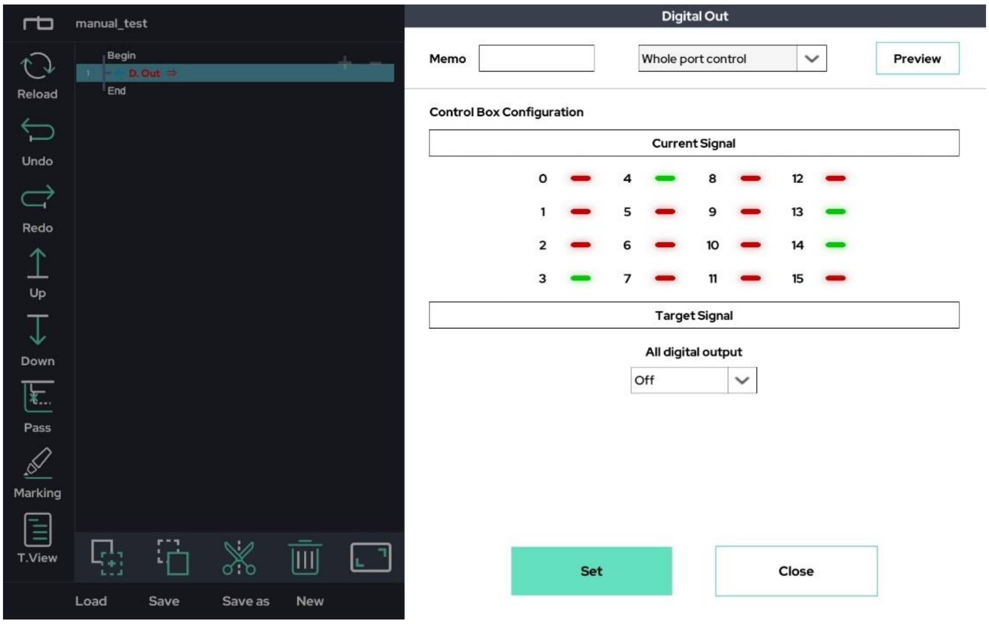

With the control box connected to the teaching pendant, set the Target Signal(the toggle signal represent blue) menu as shown above (to the right). Then, press the Preview button.



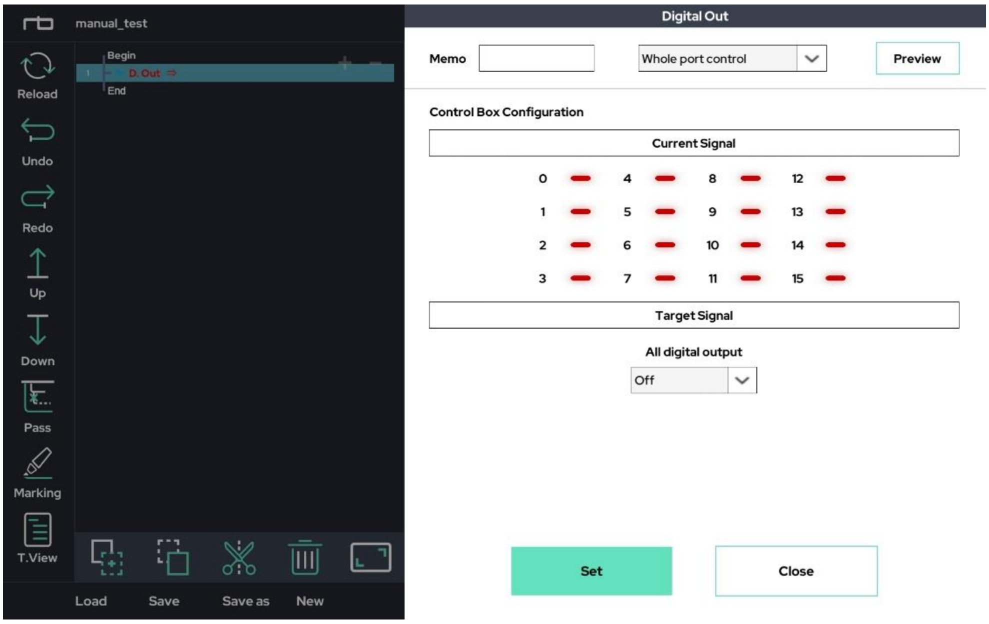

As shown above, the Current Signal menu will change to match the settings that the user has put in the Target Signal menu.

Digital Out： Whole port control

Control in all port signal at one time



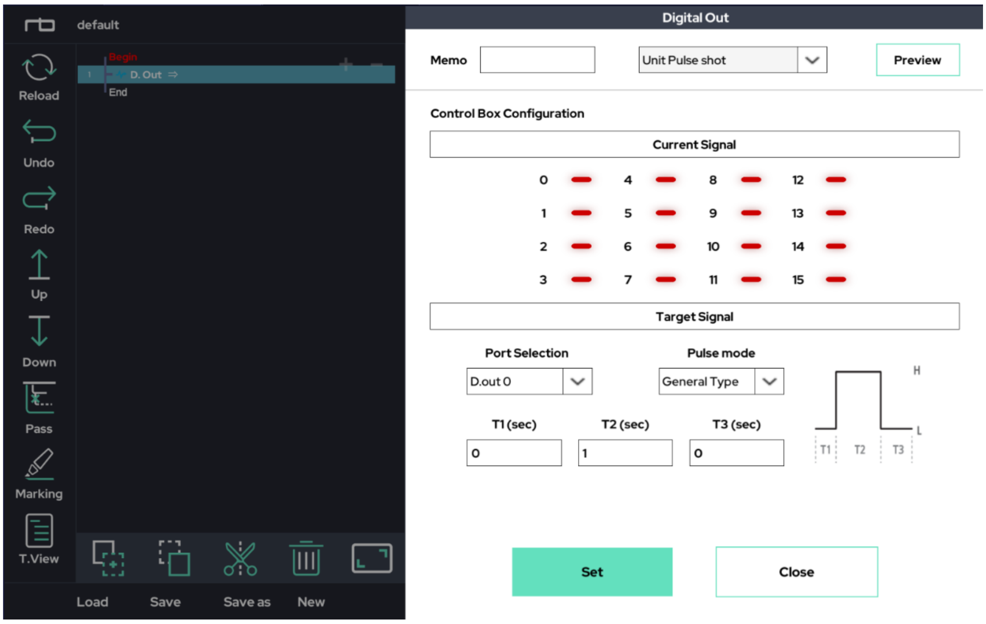

With the control box connected to the teaching pendant, set the Target Signal menu as shown above (to the right). Then, press the Preview button.



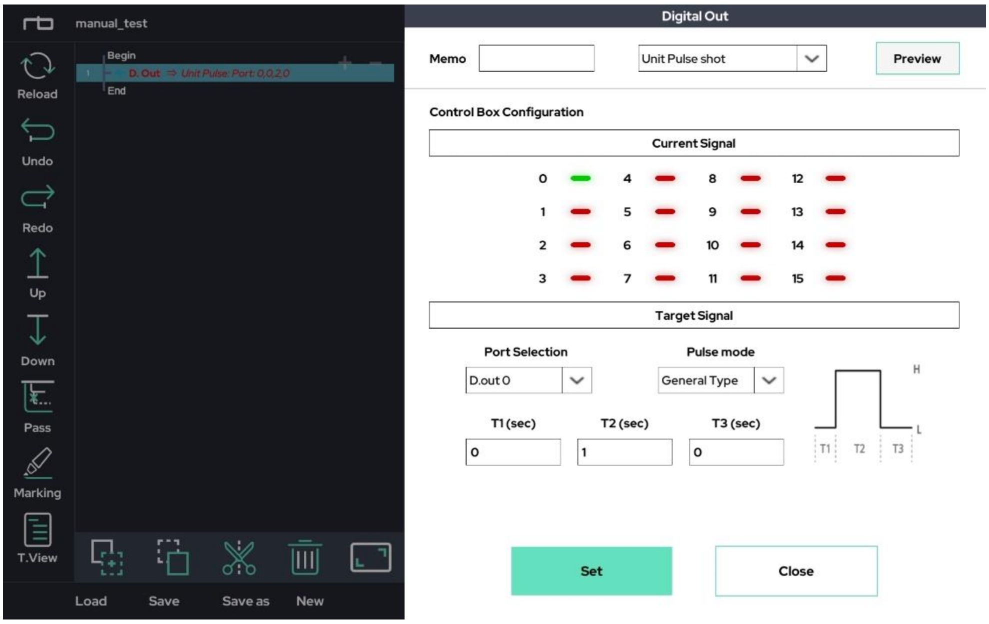

As shown above, the Current Signal menu will change to match the settings that the user has put in the Target Signal menu.

### Digital Out： Unit Pulse shot

Select the port you want to use and enter the time between 0 and 2 seconds for T1 and T3 to output a unit pulse signal for the time you entered.




With the control box connected to the teaching pendant, set the Target Signal menu as shown above (to the right). Then, press the Preview button.

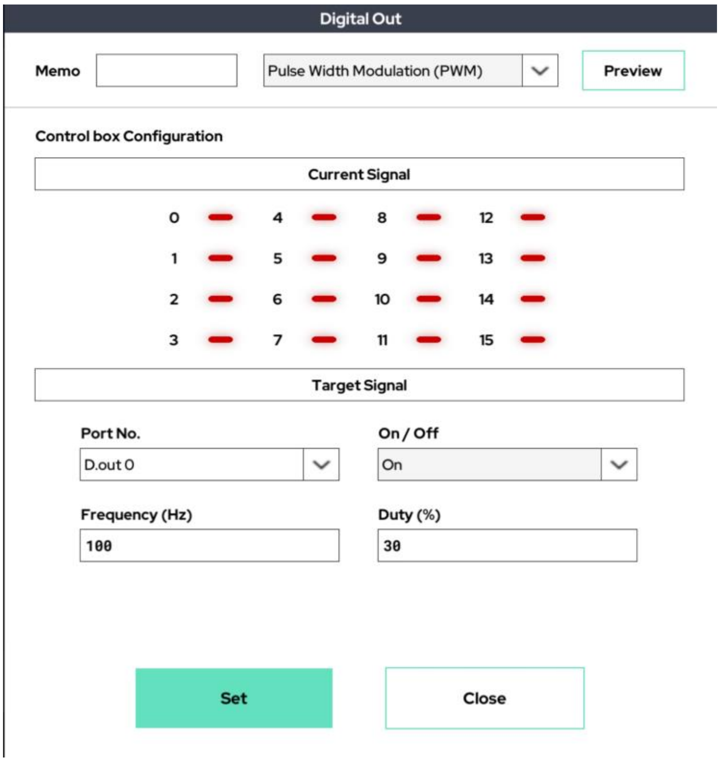


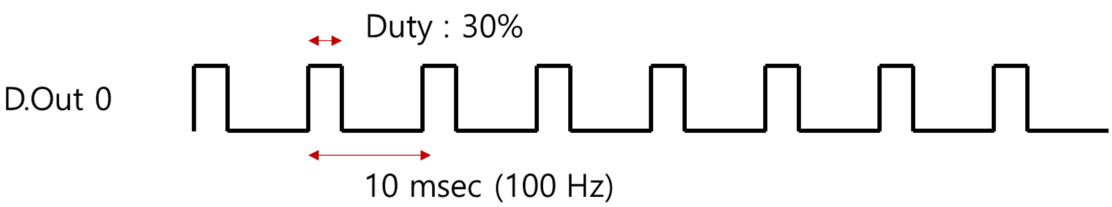

As shown above, the Current Signal menu will change to match the settings that the user has put in the Target Signal menu.

## Digital Out : Pulse Width Modulation(PWM)

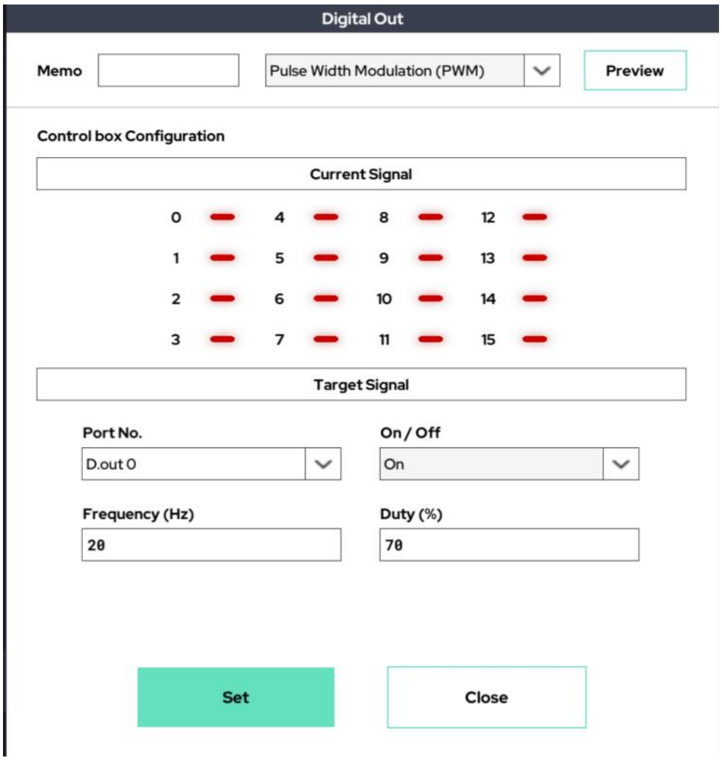
The user can use the PWM (Pulse Width Modulation) Function to set the frequency and duty ratio of a PWM pulse, then send that signal through digital output port.

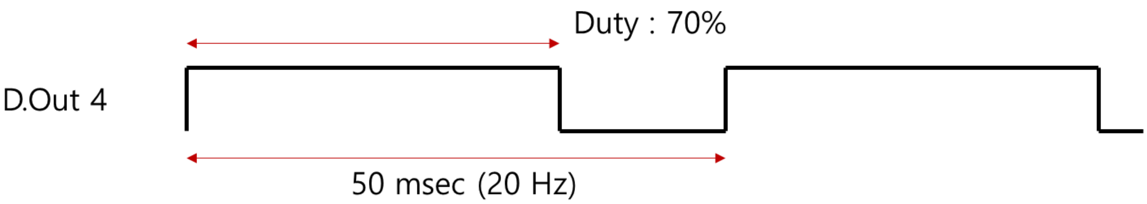
Example 1)



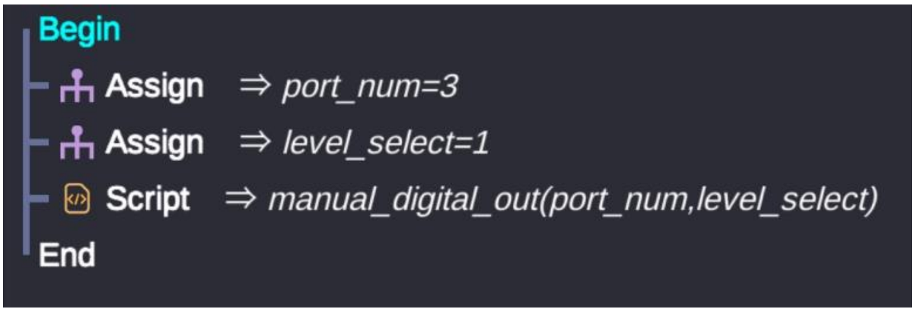
Example 2)




In addition to using the D.out function, users can create a command to export digital output using the Script function as shown below.

※ Function: manual_digital_out (port number, output level)

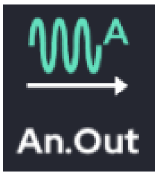



**Warning**

Warning:

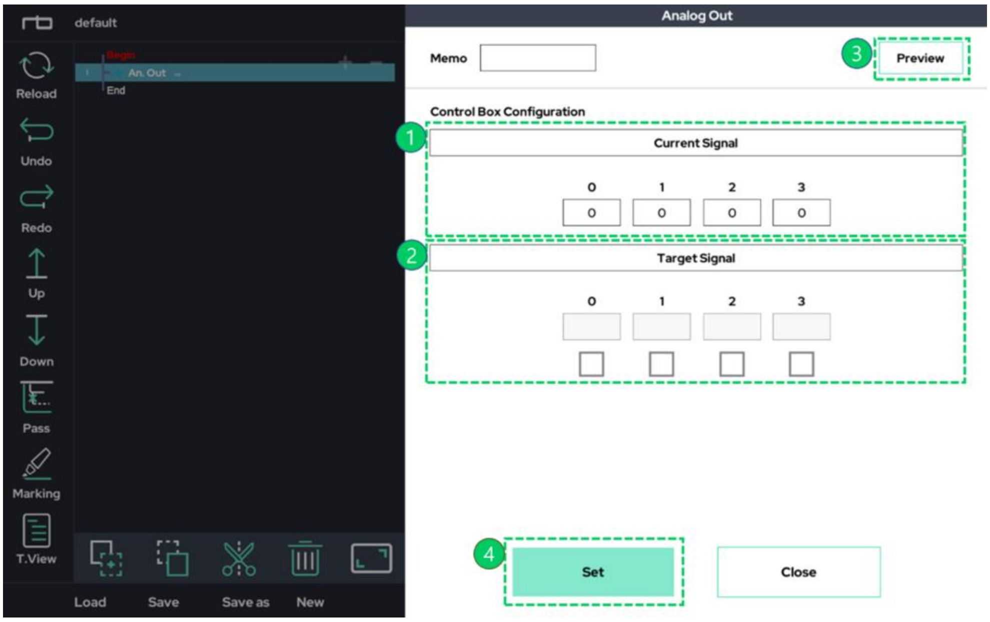
1) If a special function is assigned to a specific digital output port in Setup-I/O, that port is not available through the D.out function.
2) If a special function is assigned to a specific digital output port, it will be indicated in color yellow.
3) If you want to leave a comment about the D.out function you set, you can use the memo function at the top right of the popup window.
4) Before using the digital output, please fully understand the electrical properties of the digital output port provided by the manufacturer.

■ An.out(Analog out) Function :




The Analog Out Function controls the analog output of the control box. Outputs the selected voltage through the target (0 ~ 3) analog ports. Each port can output a voltage range of 0 ~ 10V.

After adding An.Out to the program, click on An.out in the program tree to open the following popup window.

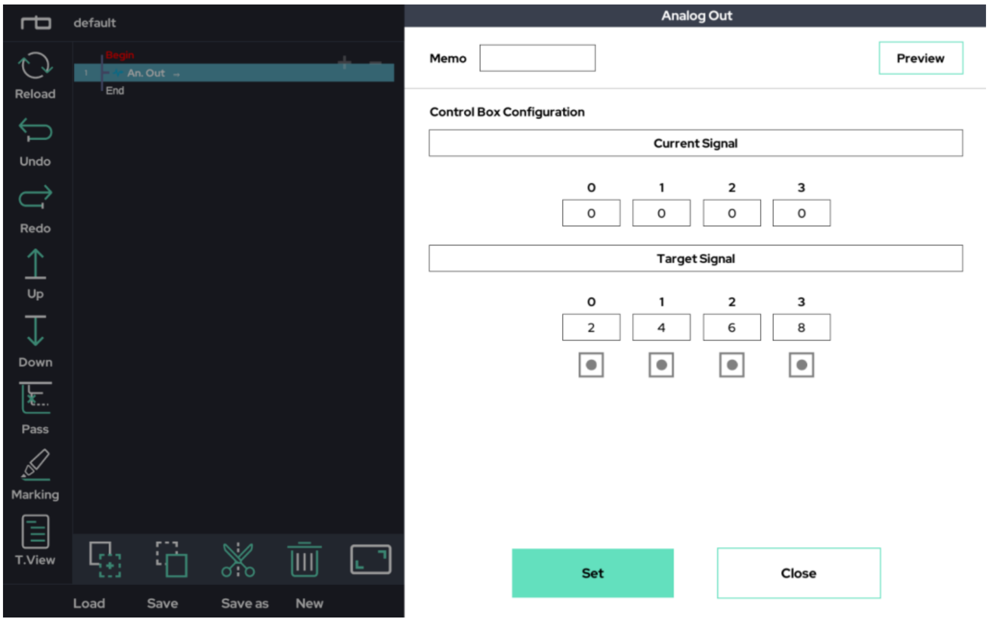



① Shows the status of the current Analog Out output from the control box.

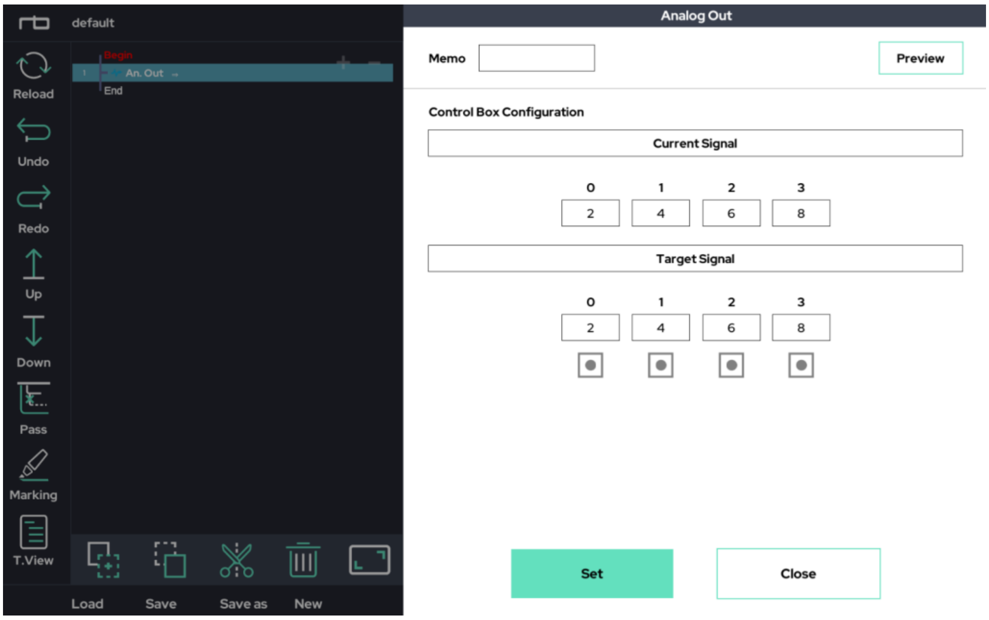
② Allows the user to enter their desired voltage setting. If the check box is empty, it is set to maintain the existing voltage output. To set the output, check the box, then enter the desired voltage (0 ~ 10V).

③ Allows the user to preview the settings selected within the target signal menu. A further explanation is shown below.

④ Saves the settings specified within target signal menu.



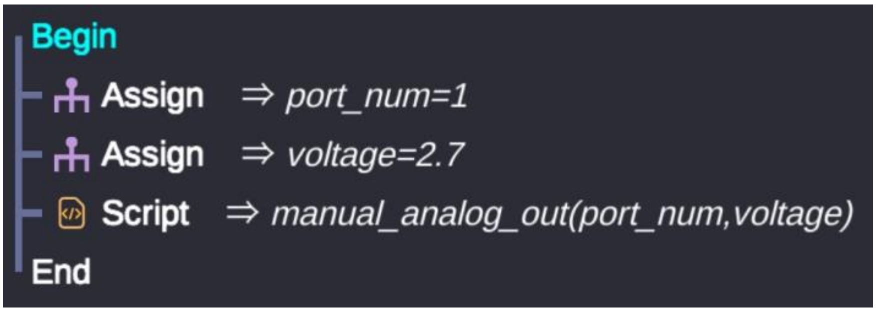

With the control box connected to the teaching pendant, set the Target Signal menu as shown above (to the right). Then, press the Preview button.




As shown above, the Current Signal menu will change to match the settings that the user has put in the Target Signal menu.

In addition to using the An.out function, users can create a command to export analog output using the Script function as shown below.

※ Script Function: manual_analog_out (port number, output voltage)

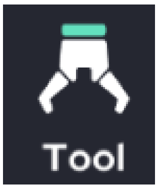



**Warning**

Warning:

1) If you want to leave a comment about the A.out function you set, you can use the memo function at the top right of the popup window.
2) Before using the analog output, please fully understand the electrical properties of the analog output port provided by the manufacturer.

■ Tool Out Function :




The tool flange has two digital outputs. Signals from two digital outputs can be specified. In addition, the level of voltage to be output from the tool flange (0V or 12V or 24V) can be adjusted together.

Click the Tool Icon to add it to the program. Click on Tool in the program tree to have the following pop-up window appear.

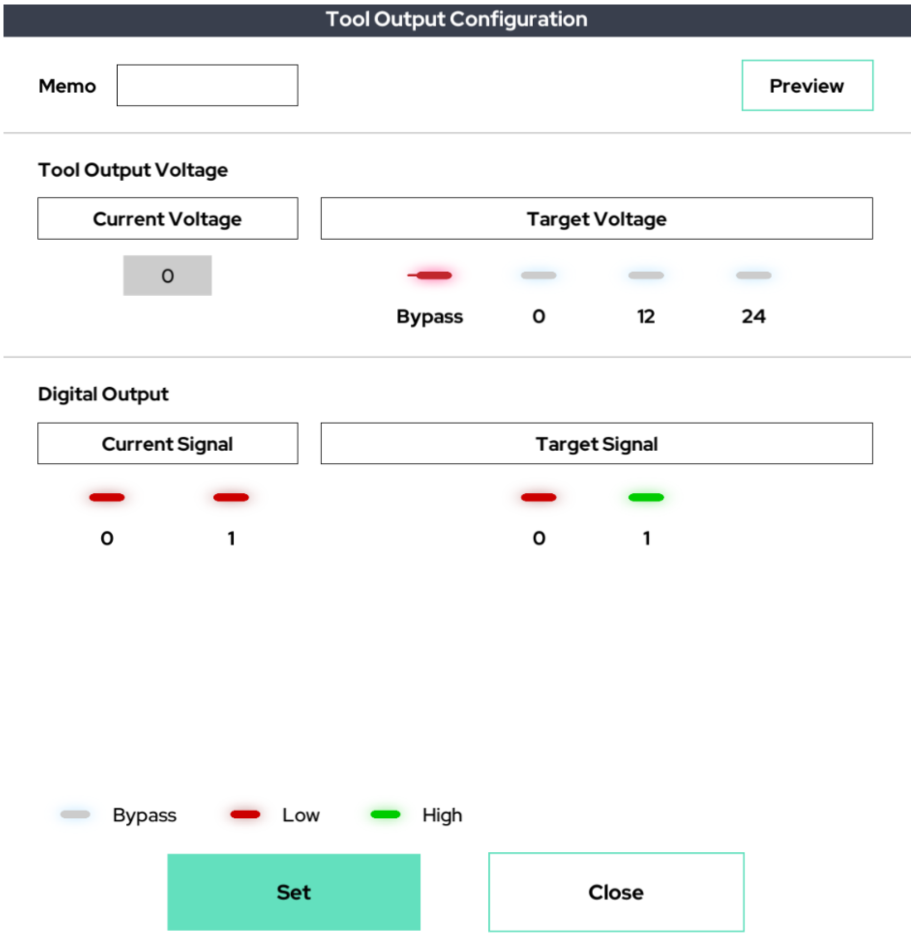

① Shows the current status of the tool flange output at the end of the robot.

② Sets desired voltage and digital output.

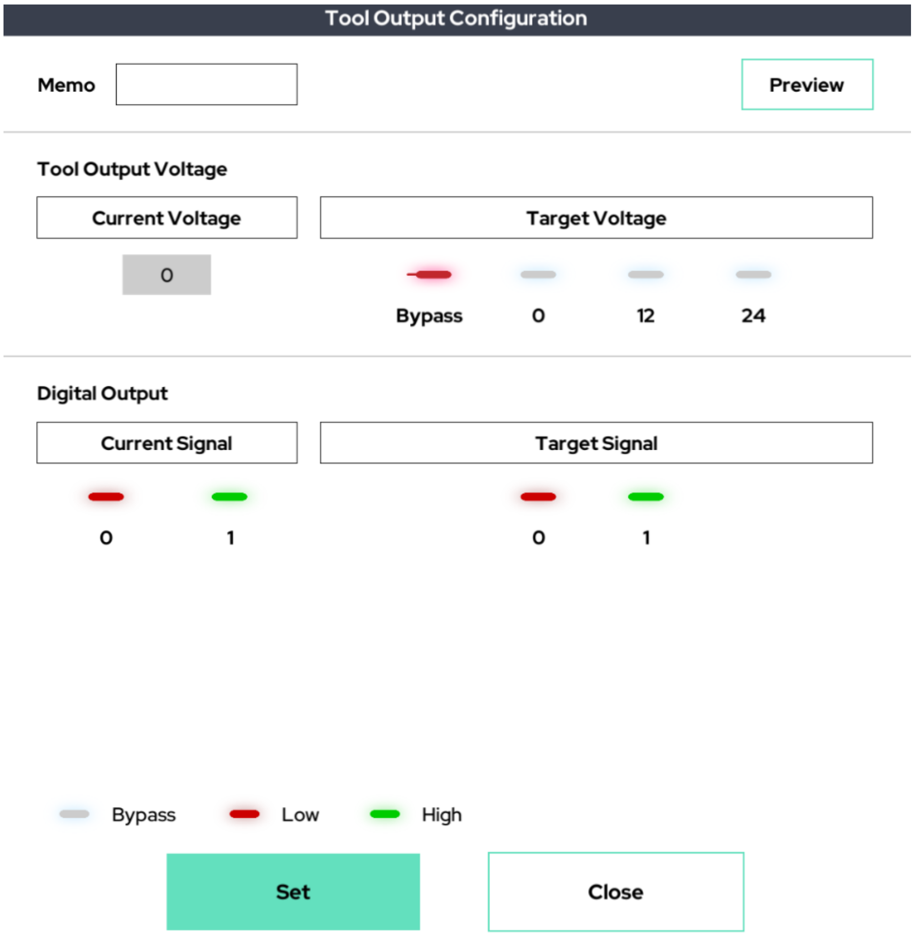
- The output voltage can be selected between 0V, 12V, and 24V. There is also an option to Bypass.
- The digital output can be toggled between Bypass, Low, and High.

③ Allows the user to preview the settings selected within the target signal menu. A further explanation is shown below.

④ Saves the settings specified within target signal menu.




With the control box connected to the teaching pendant, and after activating the robot, set the Target Signal menu as shown above. Then press the Preview button to preview the tool flange output signal.




As shown above, the Current Signal menu will change to match the settings that the user has put in the Target Signal menu.

**Warning**

Warning:

1) The user can add a comment about the Tool.out function by using the memo function at the top right of the popup window.

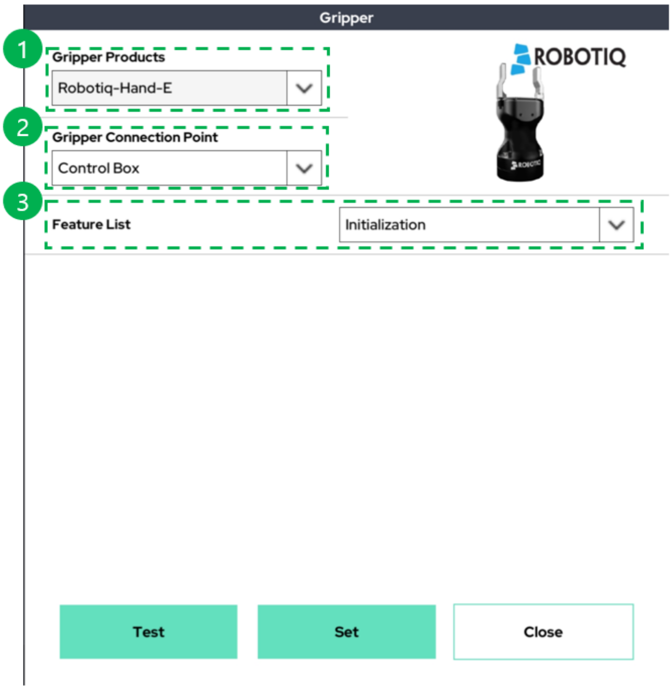
2) Before using the tool flange output, please fully understand the electrical properties of the port provided by the manufacturer.

■ Gripper Function :


This is a dedicated function for the gripper dedicated to cooperative robots. It is possible to conveniently test and insert into the program and use of cooperative robot grippers from various companies such as Robotiq's grippers. It is not a simple I/O method, but it is a function that helps users to use a gripper that is cumberso

me to write by using serial communication such as RS485 or using CRC.

Add the gripper function to the program tree and click the added Gripper as below.




1. Select the gripper product.
2. Select gripper connection point(Control Box, Tool Flange).
3. Select the function to be used as the gripper.

**Warning**

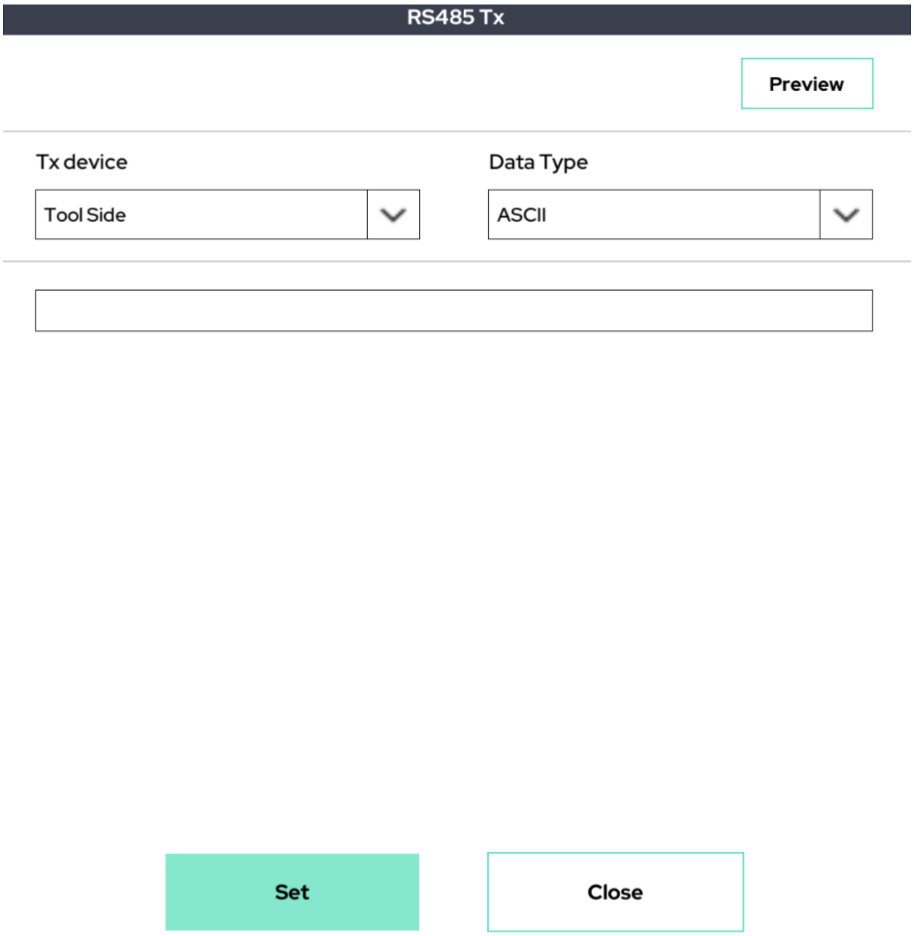
Warning: The product list provided in the gripper function will be updated through user request.

■ RS485 Function :

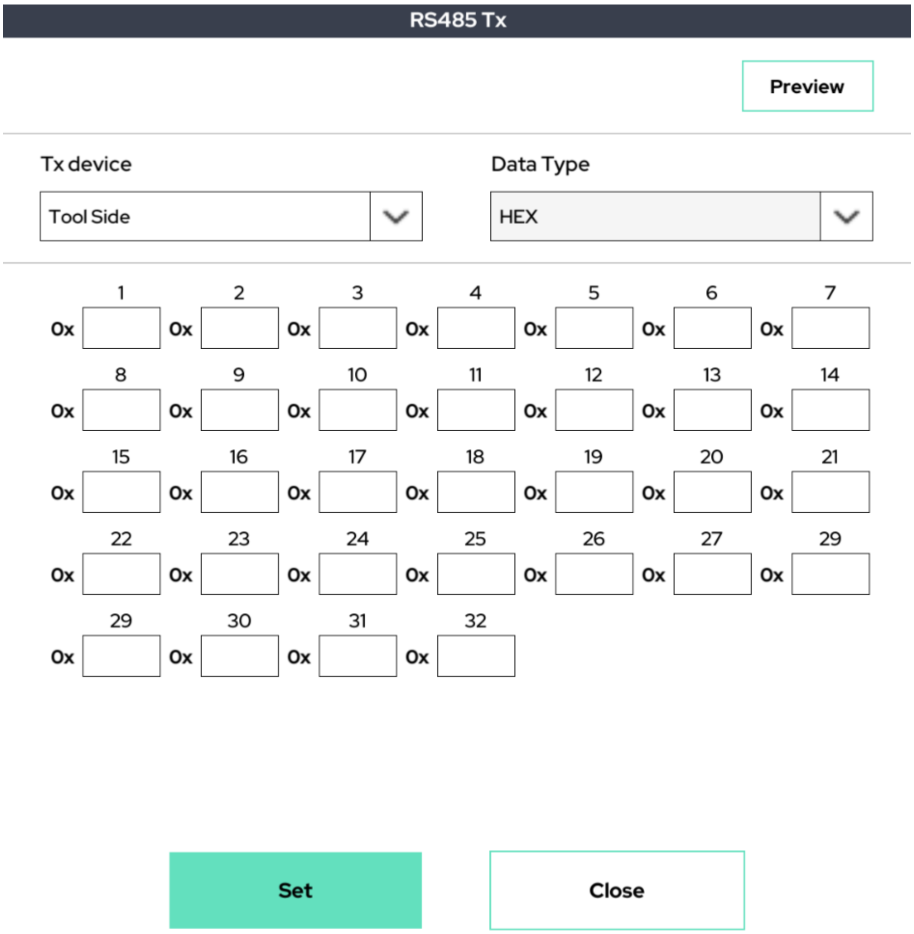

This function allows the user to set the RS485/232 output for the tool flange or the control Box. Users can output in ASCII mode, or in HEX mode.

The UI Tablet (Teaching pendant) only supports UI485 Tx.

The configuration can be previewed through the Preview button on the right side of the popup window.




ASCII mode

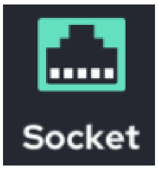



HEX mode

Baud rate and other protocols (Parity bit, Stop bit) for use in Serial-Communication can be set in Setup-serial menu. Alternatively, the user can use the Set-Serial_Configuration option at the top of the project.

To use serial communication on the box side, plug a commercially available USB-Serial (RS232 / 422/485) device into the USB port.

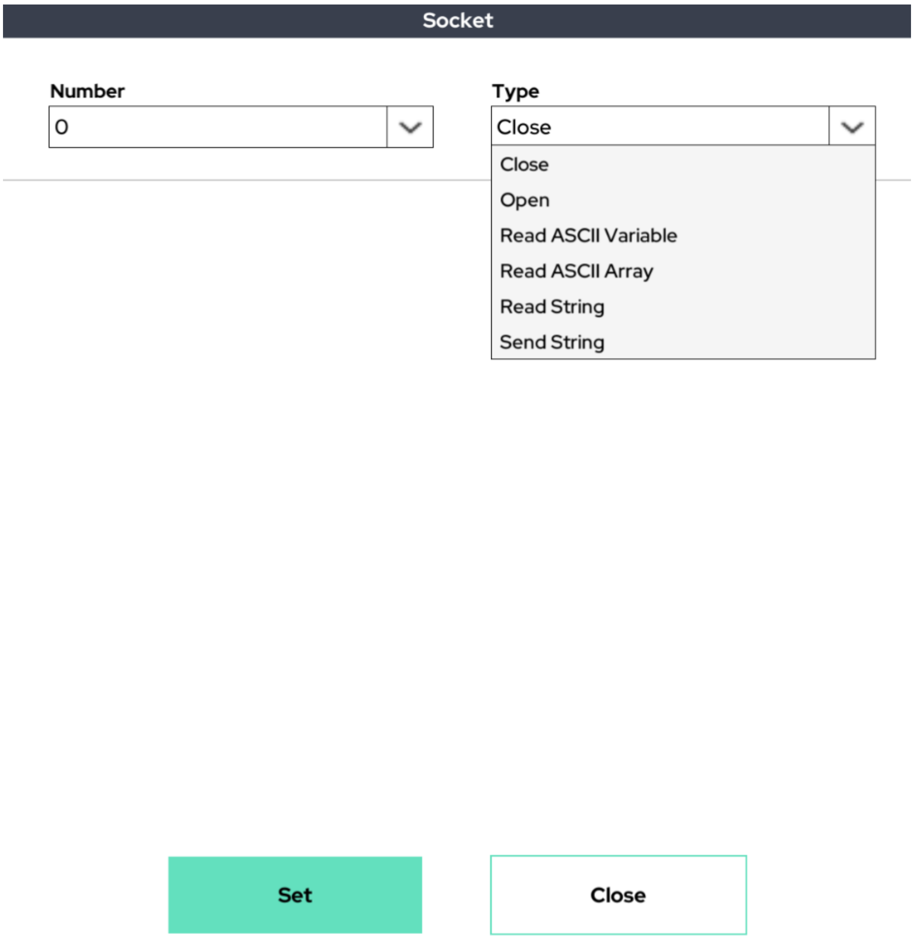
■ Socket Function :




The Socket Function allows for socket communication. It provides the user the ability to open sockets to connect, send request messages, and retrieve data to/from specific server. Socket communication can be connected to at most 5 separate servers.

The Socket Function uses the IP settings as defined in the Setup screen. A user that would like to change the IP settings can go to the Setup-System screen.

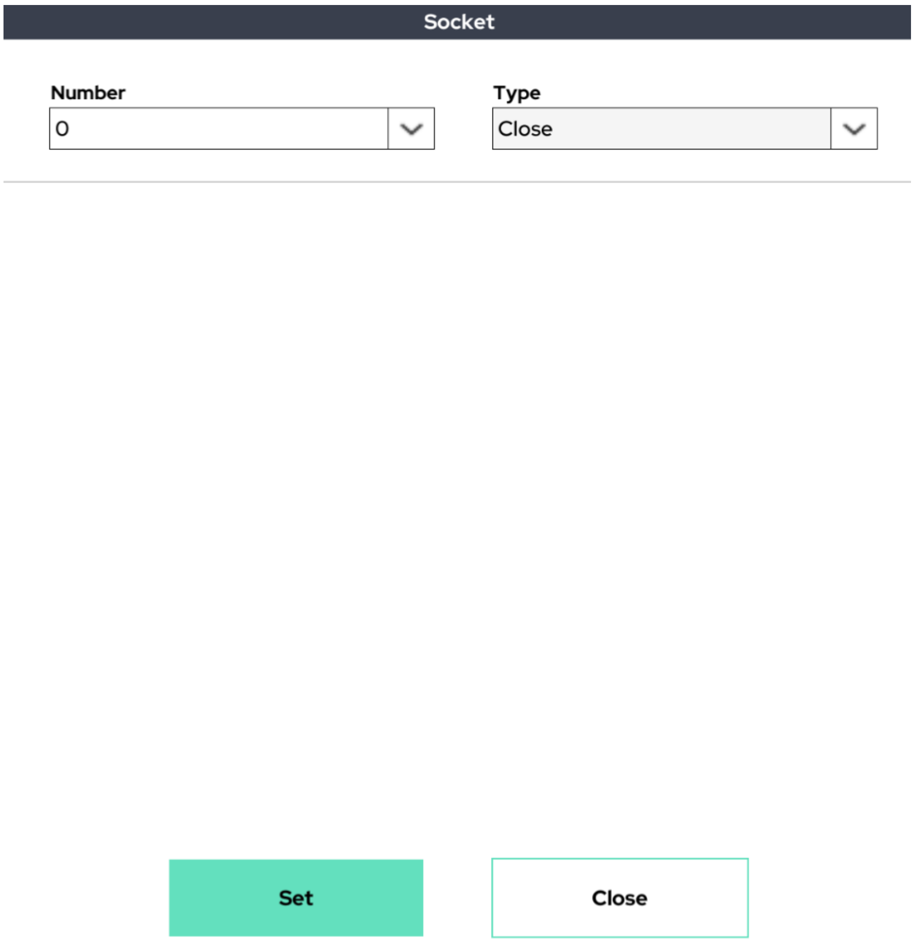
The Socket Function provides six different options as follows.




- Close: Closes the socket.
- Open: Opens socket and connects with server.
- Read ASCII Variable: Reads a value sent from the server. The user will need to choose a variable to be overwritten with the received value.

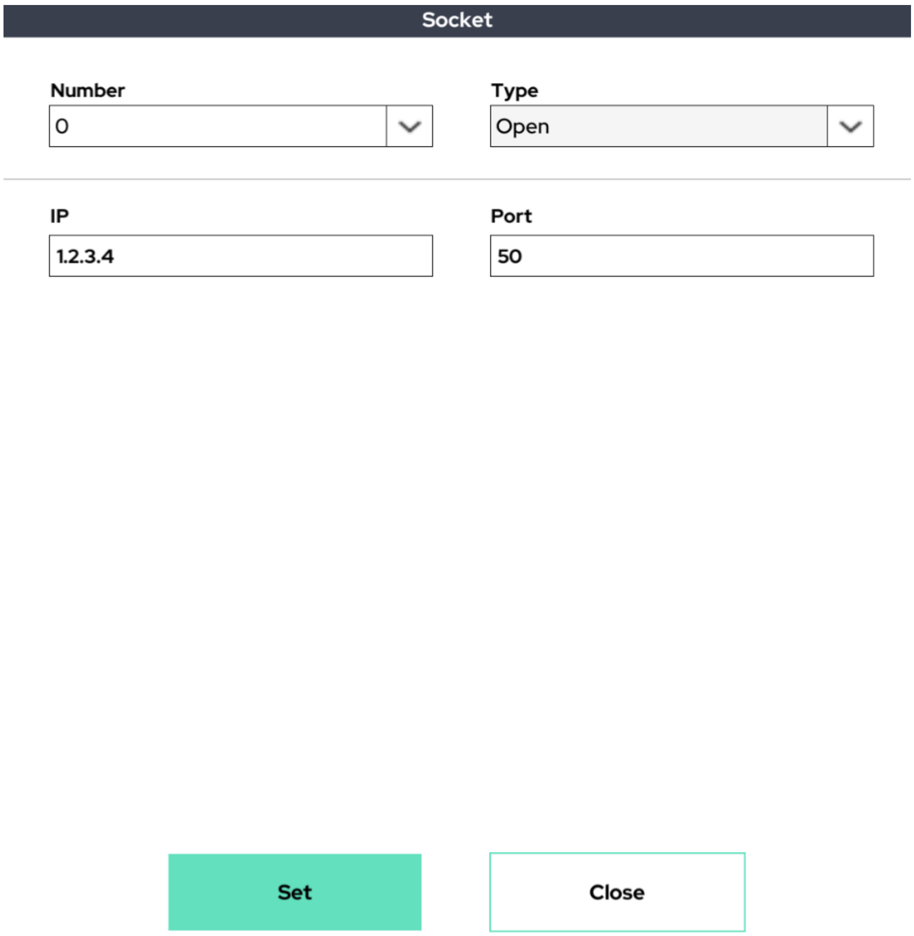
- Read ASCII Array: Reads an array sent from the server and puts it into an array type.
- Read String: Reads a string from the server and puts it into a string type.
- Send String: Send the specified string to the server.

Socket Function: Close



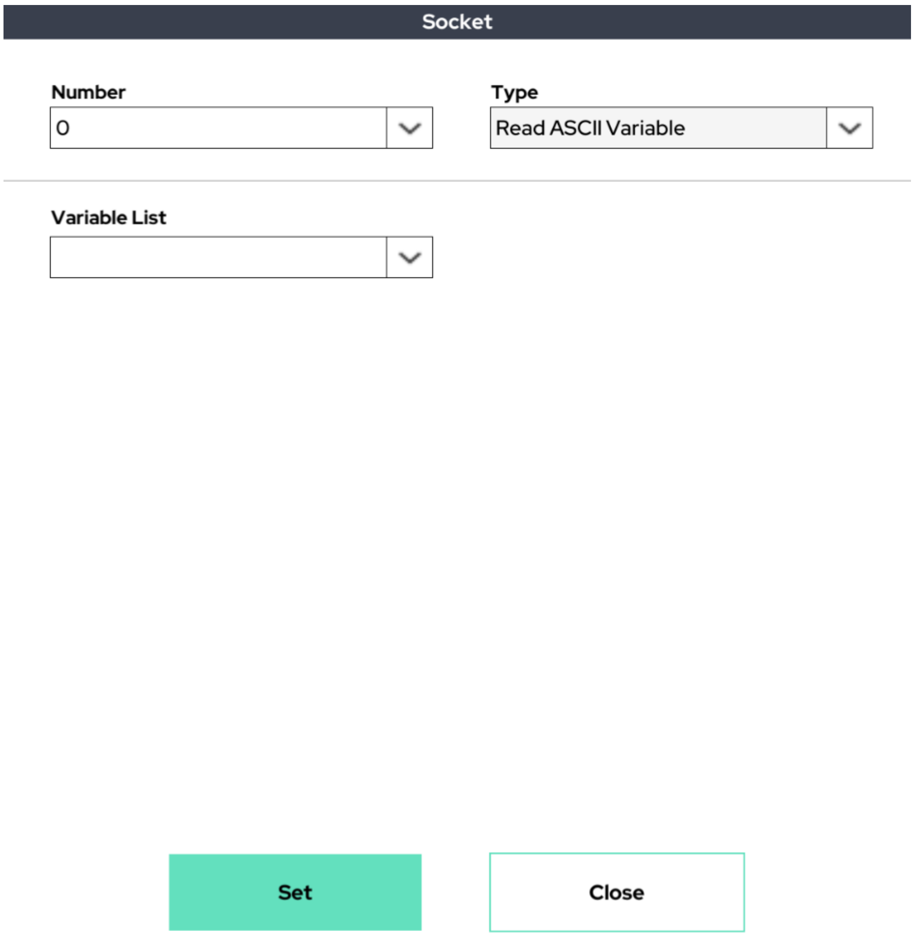

This option closes the selected socket (0 ~ 4).

Socket Function: Open




Opens the selected socket (0 ~ 4) and connects to the partner server. This option requires the user to set the IP address and port number of the server they would like to connect to.

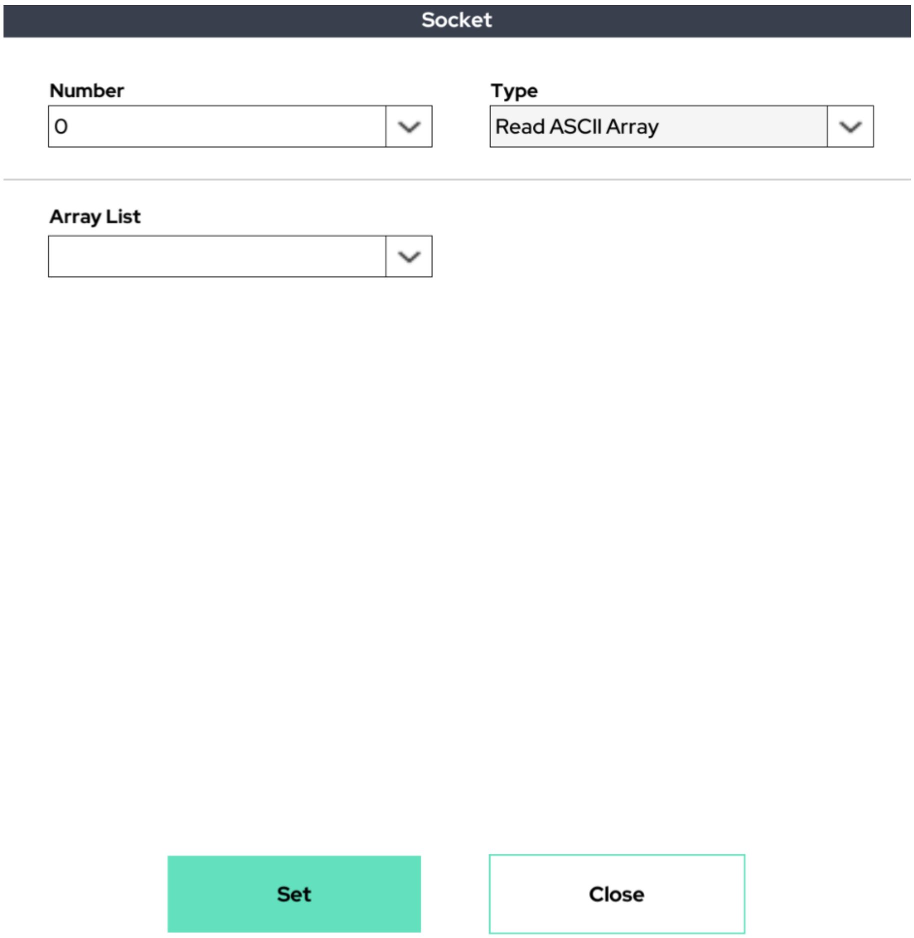
Socket Function: Read ASCII Variable




Allows the user to select one predefined variable (from the Assign Function) and overwrite the value of that variable with a value received from the server.

*(Note: specific rules apply. These rules can be found at the end of the Socket Function section.)*

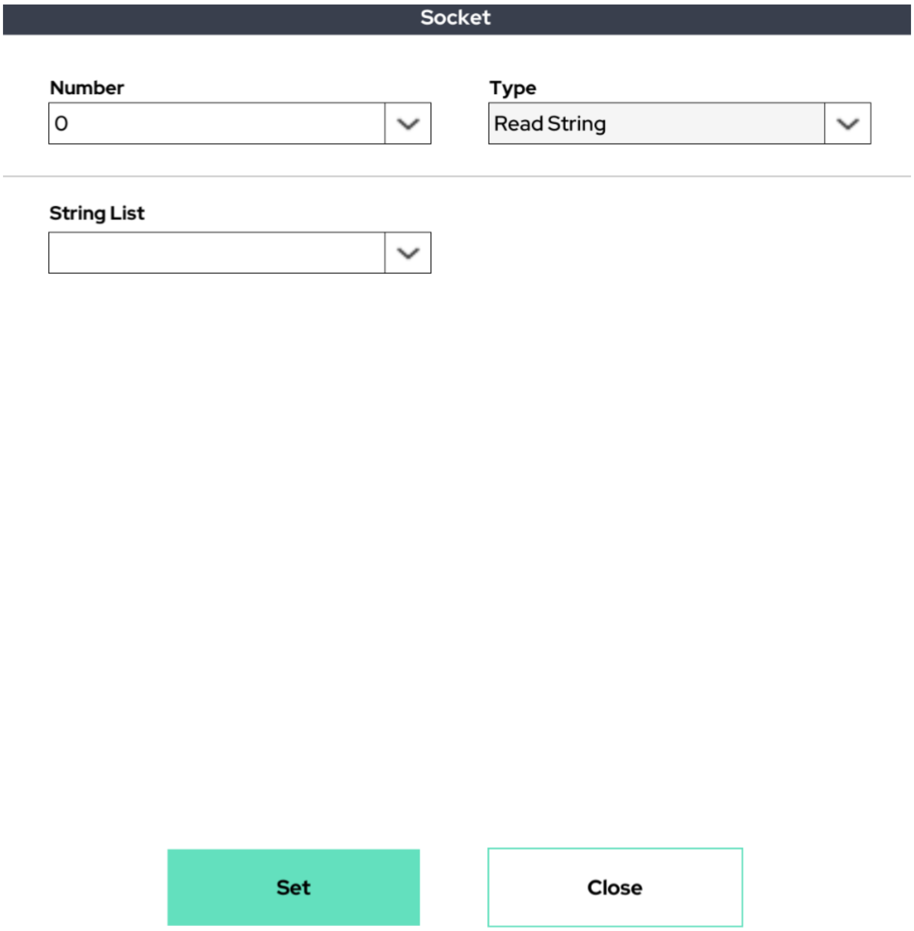
Socket Function: Read ASCII Array




Allows the user to select one predefined array (from the Assign Function) and overwrite the values contained within that array with the values of an array sent by the server.

*(Note: specific rules apply. These rules can be found at the end of the Socket Function section.)*

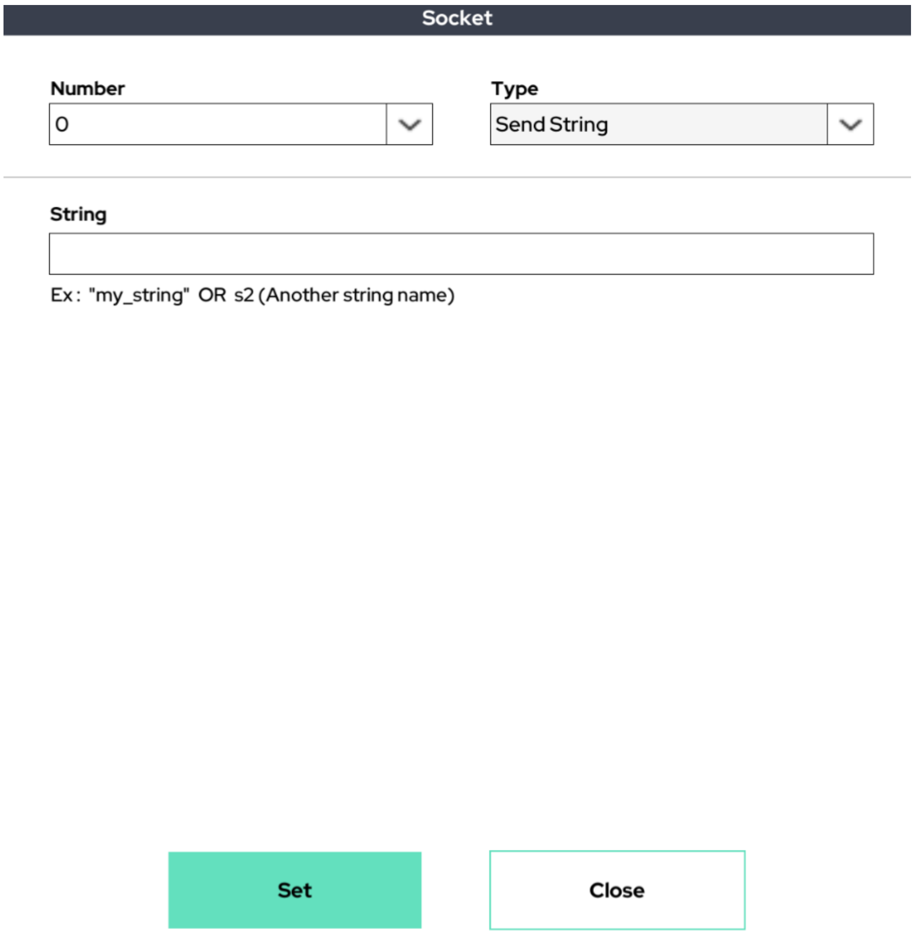
Socket Function: Read String




This is the function to put the ASCII string received through Socket communication into the selected string variable.

*(Note: specific rules apply. These rules can be found at the end of the Socket Function section.)*

## Socket Function: Send String




Allows the user to send a specific string to the server. Users can enter a string directly in the field or send a predefined string type variable.

## Warning

Warning:

**The syntax that needs to be followed:**

In order to use the Read ASCII Variable, ASCII Array, and String options provided by the robot manufacturer, the data format received from the server MUST follow the following format. If a special communication grammar/syntax is required, please consult with the manufacturer.

Read ASCII Variable

When receiving a value from the server, the value must be sent as a numerical value. (i.e. the numerical value hasn't to be contained within quotation marks)

(e.g. 123, 4567)

Read ASCII Array

When receiving an array from the server, the array hasn't to be contained within a string. In this case, there must be curly braces, and commas must be present between each number value.

(e.g. {100,200,300}, {400,500,600,700})

Read String

When receiving a string from the server, the string must be inside quotes.

(e.g. "this_is_string_from_server")

**Internal variables to help socket communication:**

The RB Series comes with built in variables for users to check information regarding the status of the sockets, as well as the data coming through those sockets. The internal variables are shown below. They can be accessed using the Script Function, or some similar function (i.e. If) that allows a user to access variables. The variables can be found in the List drop down after selecting Shared Data from the Type drop down menu.

SD_SOCK_IS_OPEN_ # (where # denotes the socket number 0 ~ 4)

A provided variable that stores whether the socket is open or connected to the server.

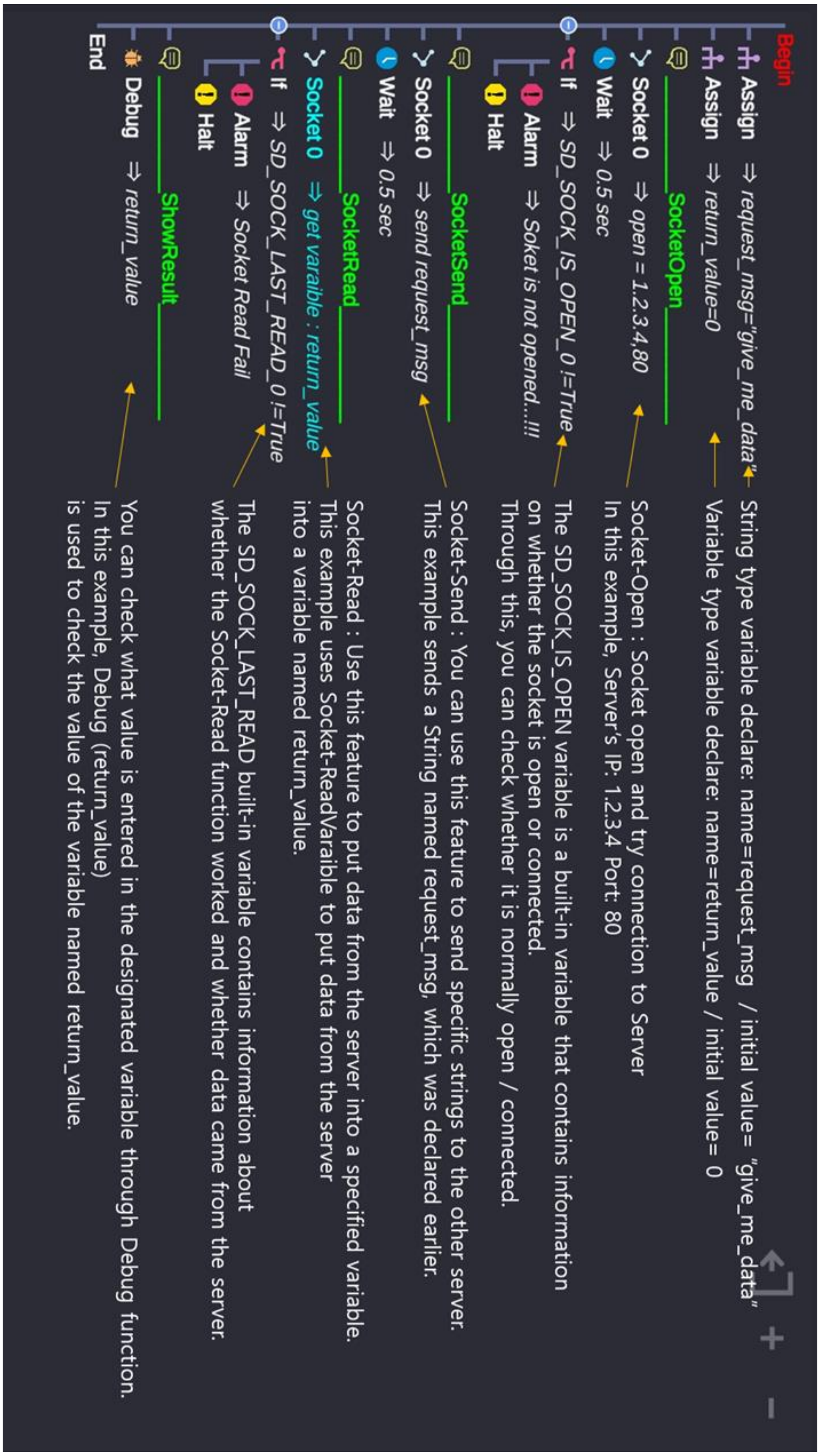
After a user opens a socket using the Open option in the Socket function, the user can check if the socket is connected by using If (SD_SOCK_IS_OPEN_#)..

SD_SOCK_LAST_READ_# (where # denotes the socket number 0 ~ 4)

A provided variable that stores the last character that was sent via the socket. It can be used to check whether the Read function executing normally.

For example, after using the ReadAsciiVariable option in the Socket function, users can check if the last Read function performed normally by using If (SD_SOCK_LAST_READ_0). This variable will have a value of zero if no data came from the server.

The figure below shows an example of the Socket Function.



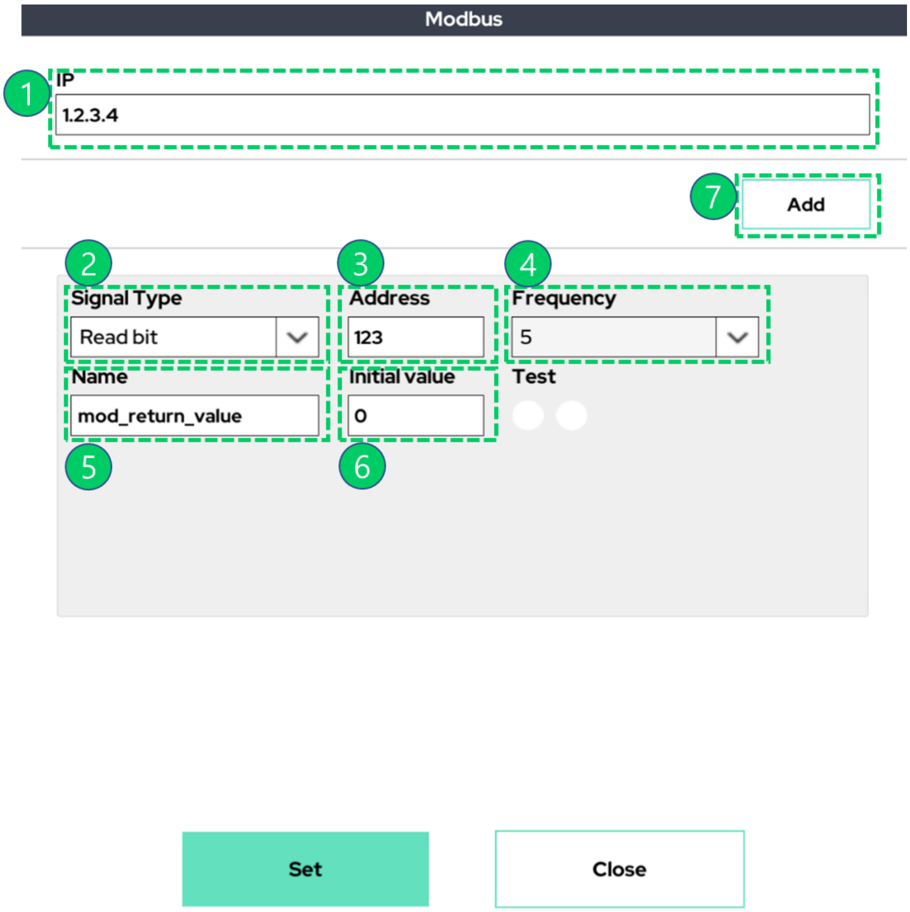
■ Modbus TCP(Client) Function :

Provides the ability to request and receive data from a specific IP / address. Data request frequency and format can be specified.

The port number for Modbus TCP is fixed at 502 (Modbus standard).

The protocols and formats associated with Modbus TCP servers are listed in the Appendix.

Note: The Modbus TCP client function must be added at the top of the program under Pre.P.




① Input the IP address of the server.

② Select the signal type (Read bit (1bit), Read word (16bit), Write bit (1bit), Write word (16bit)).

③ Input the address of the endpoint connection on the server.

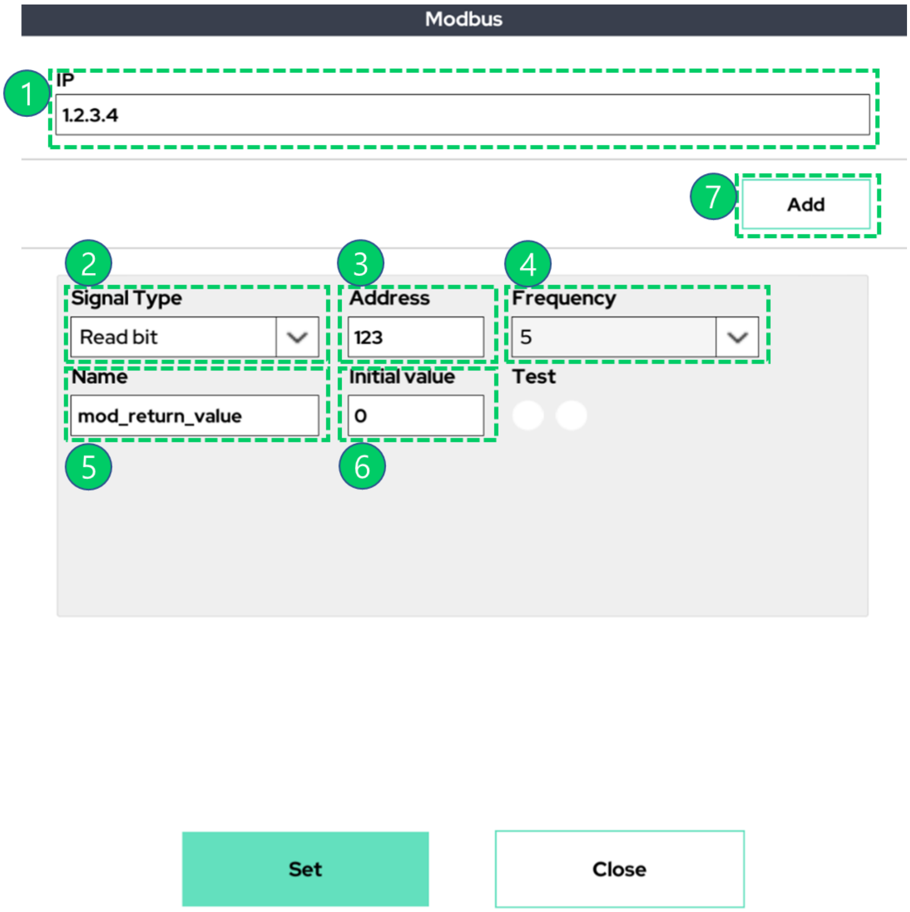
④ Select the frequency of read / write requests per minute (Hz).

⑤ If using a Read method, contains the variable name to save the read value. If using a Write method, set to the variable name to output.

⑥ Initial value of the variable set in step 5.

⑦ Button to add the signal.

Below is an example of the Modbus Function settings




### Example 1 Interpretation)

Reads a word of information (16 bits) at address 123 from the server (IP: 1.2.3.4). Stores the data in a variable named mod_return_value. Will read information at a rate of 5 times per second (5 Hz).

### Example 2 Interpretation)

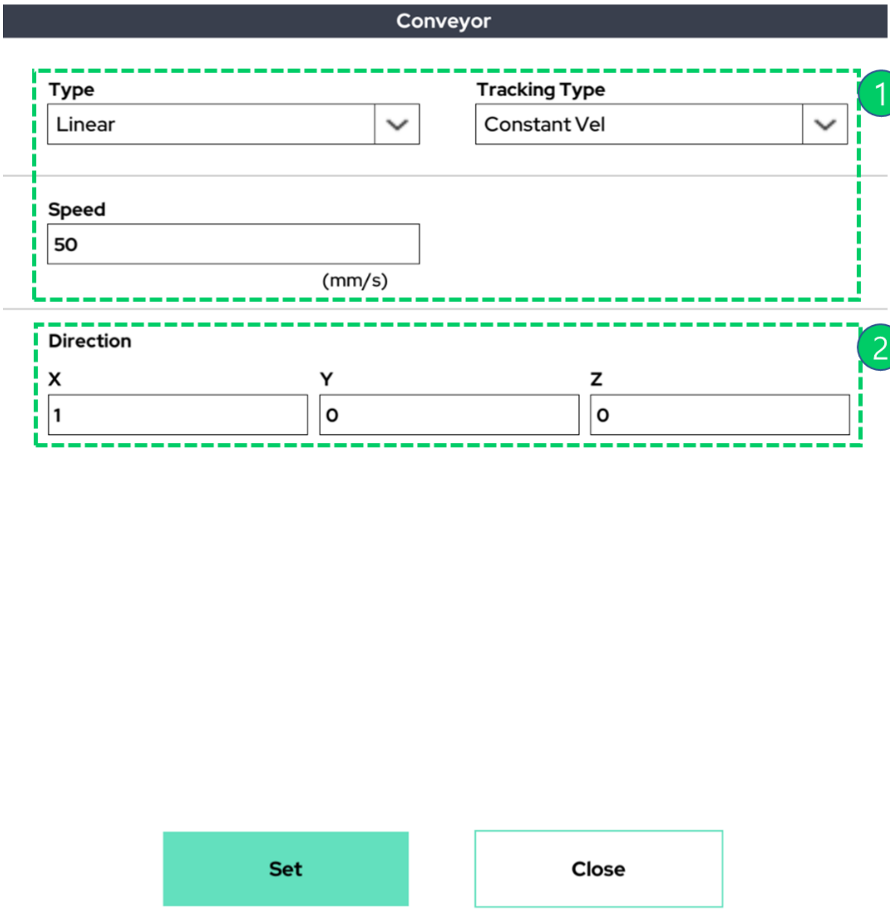
Writes the bit value (1 bit) stored in the variable mod_write_bit to address 456 on the server (IP: 1.2.3.4). Will write the data at a rate of 50 times per second (50 Hz).

■ Conveyor Function :


Allows the user to use the robot as a conveyor by generating movement at a consistent speed in a specified direction. The user can also place their own desired movement into the conveyor flow by using the MoveL, MoveLB, or Circle functions.

Note: Joint movement (MoveJ, MoveJB, etc.) cannot be used as a subitem of Conveyor. Only MoveL, MoveLB, MovePB, MoveITPL, Circle are supported.

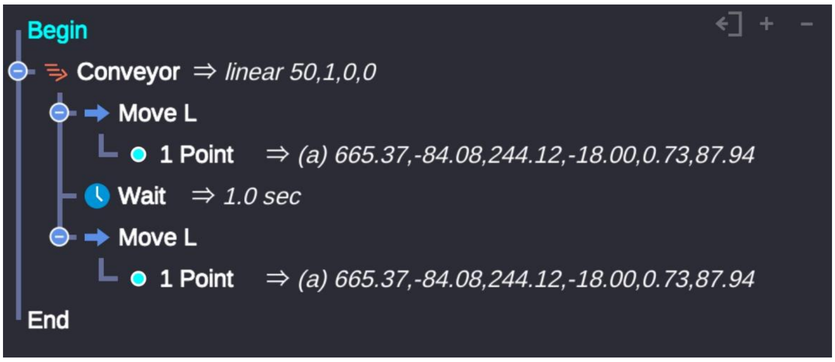
Add the conveyor function to the program tree and click the function to see the options.




① Set the move type and speed of the conveyor.

② Set the direction for the conveyor movement (x, y, z value is based on robot arm base coordinate system). The robot will move at the specified speed in the specified direction until the conveyor movement ends.

An example program tree using the Conveyor Function will look as follows:

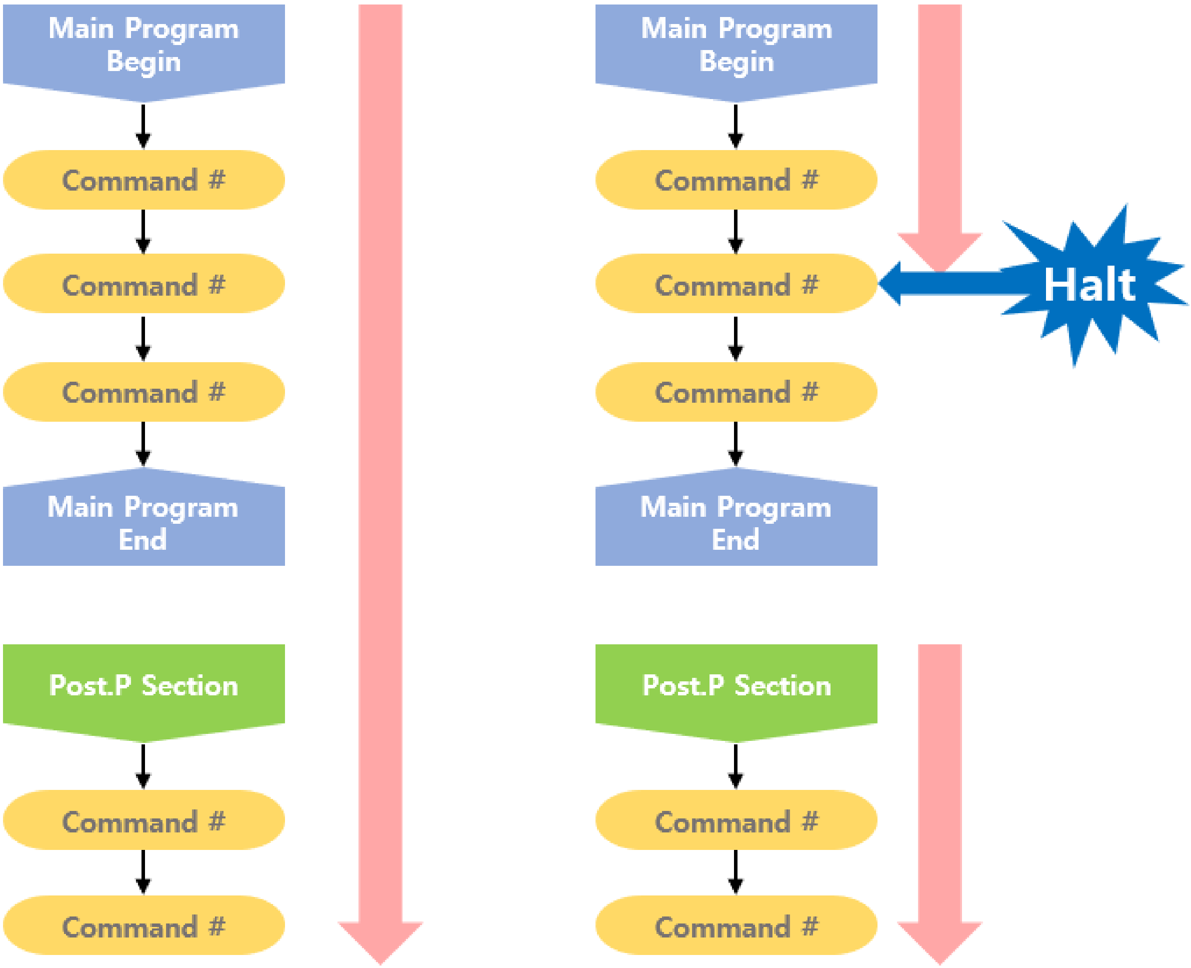


■ Post.P(Post Program) Function :


The Post.P Function allows the user to insert a command that will be executed after the program has completed.

The instructions declared within the Post.P Function are executed sequentially after the program ends.

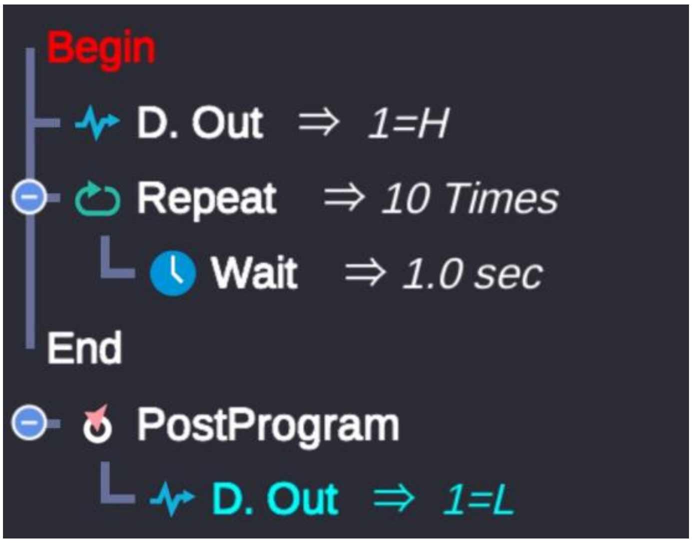
The execution of Post.P proceeds as shown in the diagram below.




Example 1)

At the beginning of the program the D.out function sends a High signal to port 1. However, the program did not send a Low signal before the end of the program.

By using the PostP. Function, when the program ends, port 1 will automatically send a Low signal.




As in the example above, the Post.P function can be used for safety functions.

Example 2)

In the below example, the PostP. Function is used to test whether the program ends normally. If the program ends normally, the warning lamp (connected to D.out No. 0) will not be turned on. If the program ends abnormally, the warning lamp will be turned on.


The SD_IS_INTENDED_STOP flag in this example is a system internal variable and is always initialized to 0 (false) when the program starts. If the program stops normally by user's intention, this variable will be 1 (true). If the program stops abnormally for various reasons, this variable remains false.

Any intended End signal by the user, such as pressing the UI "end" button, receiving an I / O stop signal, ending by other communication, etc., will be determined to be a normal end. (SD_IS_INTENDED_STOP = true)

If the program exits due to singularity access or exits due to command syntax problems, it is determined that the shutdown is not user intended. (SD_IS_INTENDED_STOP = false)

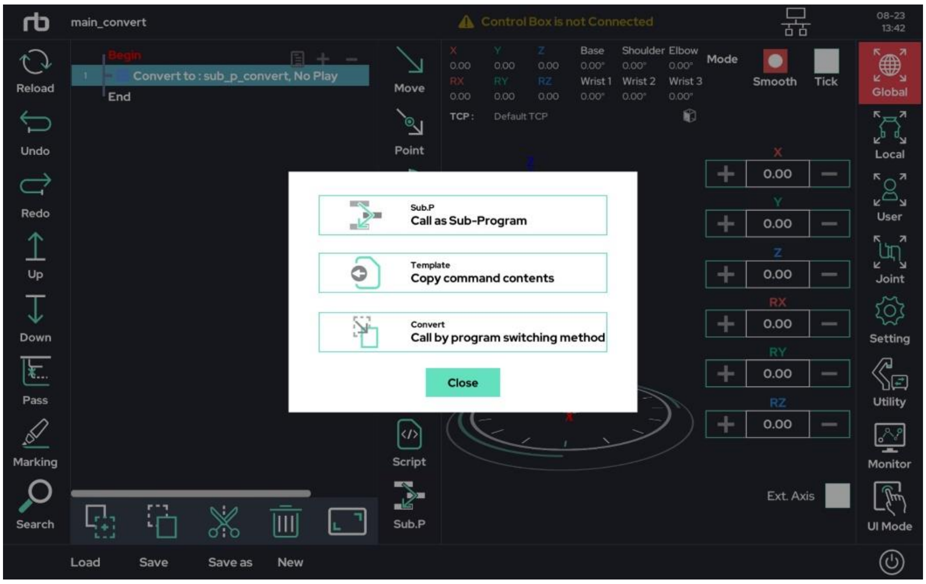
- The functions defined in the Post Program will be performed even if the program did not terminate normally (e.g. when users press Halt in an Alarm popup).
- Commands related to the movement of the robot arm, such as MoveJ and MoveL, cannot be used within Post.P
- Post.P works only within a top-level program. If a subprogram invokes the Post.P function, the Post.P portion of the subprogram will not be executed.

■ Template Function :

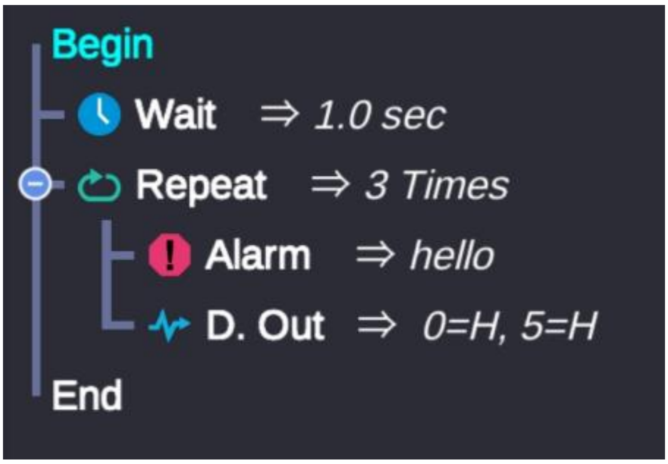

This function inserts another pre-made program file (teaching file) into the current document in a modifiable form.

The Template function is similar to the Sub.P function. However, any file that is loaded by the Template function can be modified in the current program.

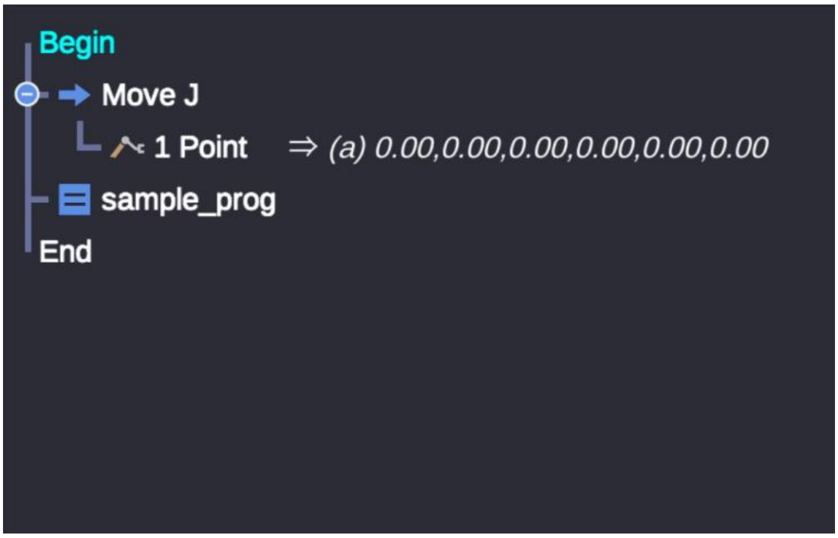
If you click the Sub.P icon in the program, the following pop-up window appears, and at this time, click 'Template'.



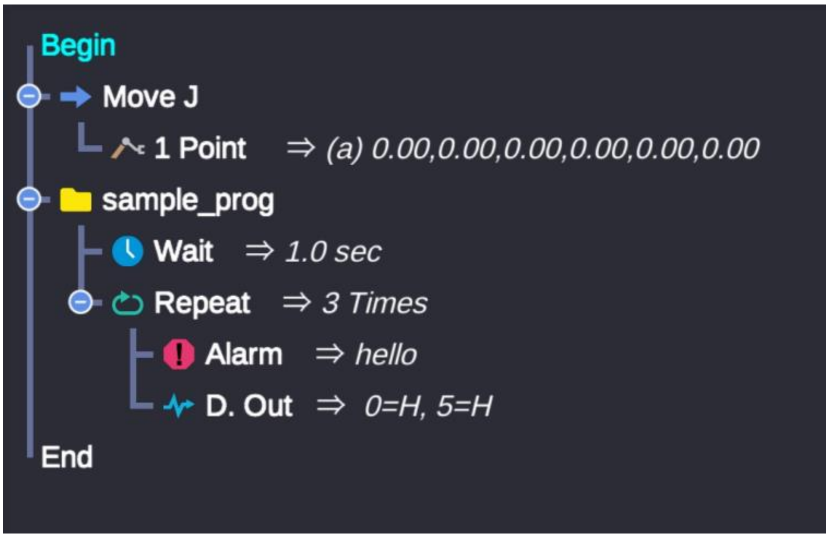

Assume that a project named "sample_prog" has been created as shown below.



example 1) sample_prog is called by Sub.P




example 2) sample_prog is called by Template



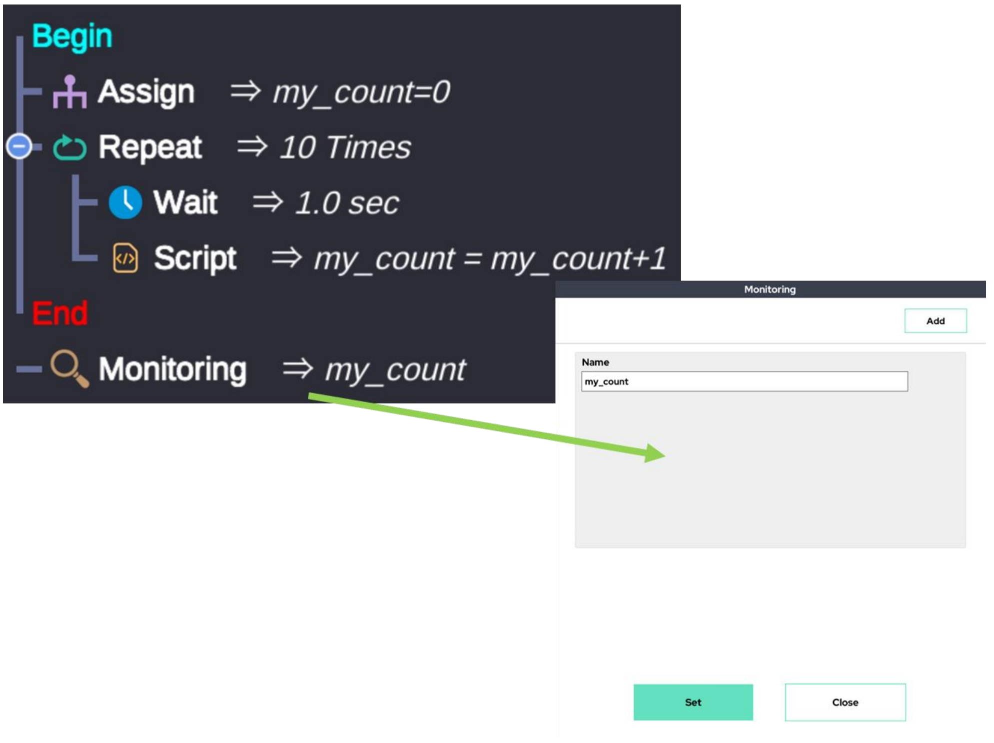

If the file is loaded into Sub.P as shown in Ex.1), the project will execute, but it is impossible to modify the file in the current program. In addition, when the loaded subproject is changed, the operation of the parent program is also changed.

If the file is imported by the Template function as shown in Ex2), it is loaded in a form that can be modified in the current program. Once copied to the template, the contents of the copied subprogram are not changed even if the original is modified.

■ Monitor Function :


This function is used to select variables (single variables, arrays, point variables, etc.) that the user wants to observe in real time while the program is running.

Variables declared in the Monitor function can be viewed by clicking the monitor icon on the right side of the Make / Play page.

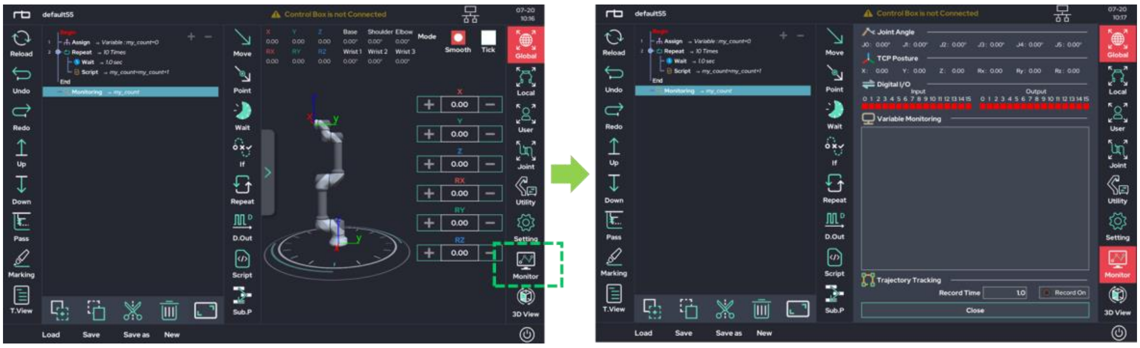



In the program example above, a variable named 'my_count' is declared. The Repeat function increments 'my_count' by 1 every second.

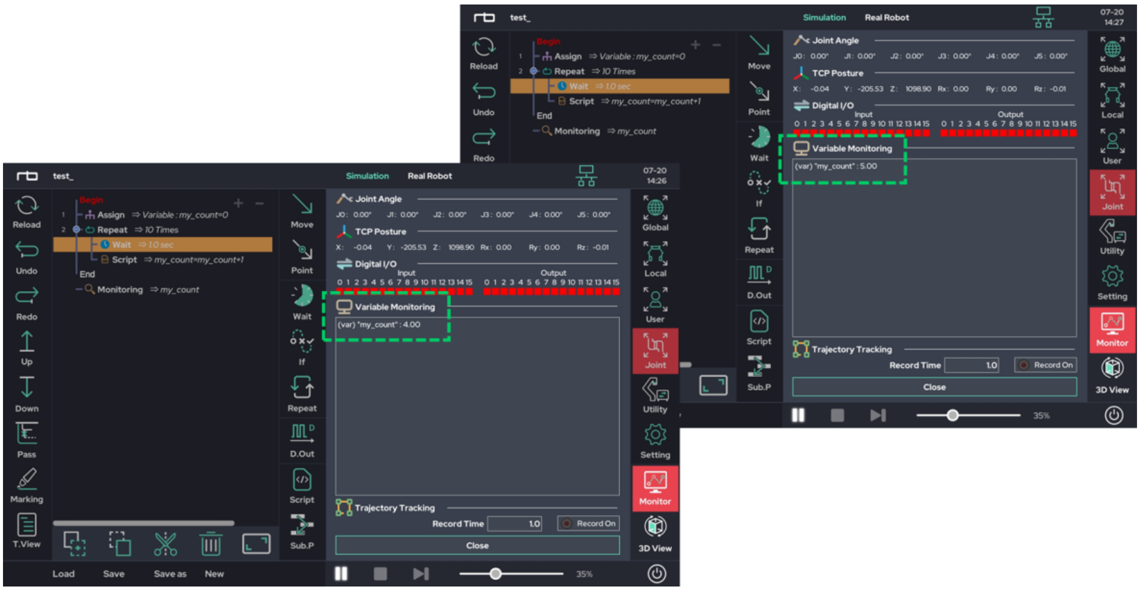
By using the Monitor function, the user can select the 'my_count' variable as the object to observe.

As shown in the above image, in the Monitoring window, the user can enter the name of the variable to be observed.

If the user wants to observe the value of the monitored variable, they can click the Monitor icon on the right side of the screen.



After that, if the user presses the play (▷) button, they can observe the value of 'my_count' increasing every second.



■ Pattern Function :


This function allows the user to define repetitive behavior.

By defining information about the operation space, and by defining which actions to be performed at each location, the user can set the robot to perform the same action at every point in space.

The user can implement palletizing through this function.

There are three sub settings.

- ■ Pattern Property:

  Define the target space for the repetitive motion.

  The property supports various shapes such as straight line, plane, 3D cube, and arbitrary point.

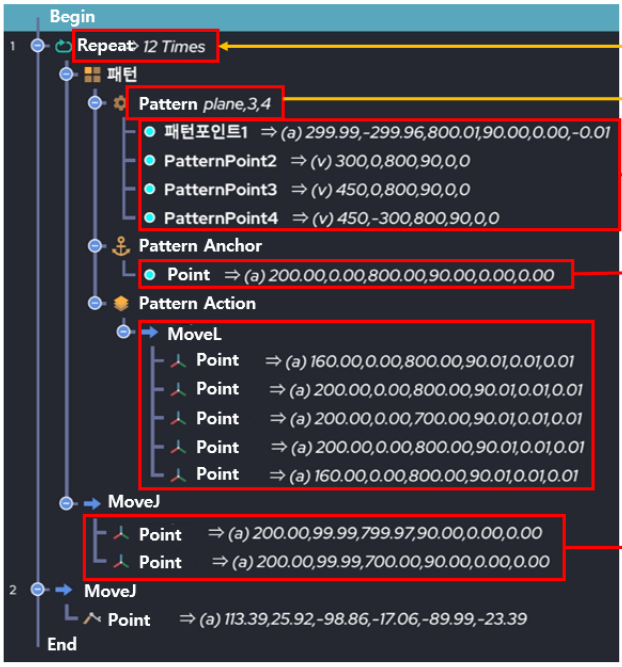
- ■ Pattern Anchor:

  The Reference point of the action defined in the Pattern Action.

- ■ Pattern Action:

  This setting defines the motion relative to the reference point set in the Pattern Anchor. The defined relative behavior is repeated at every pattern point set in the Pattern Property.

The following is an example of Pattern function.



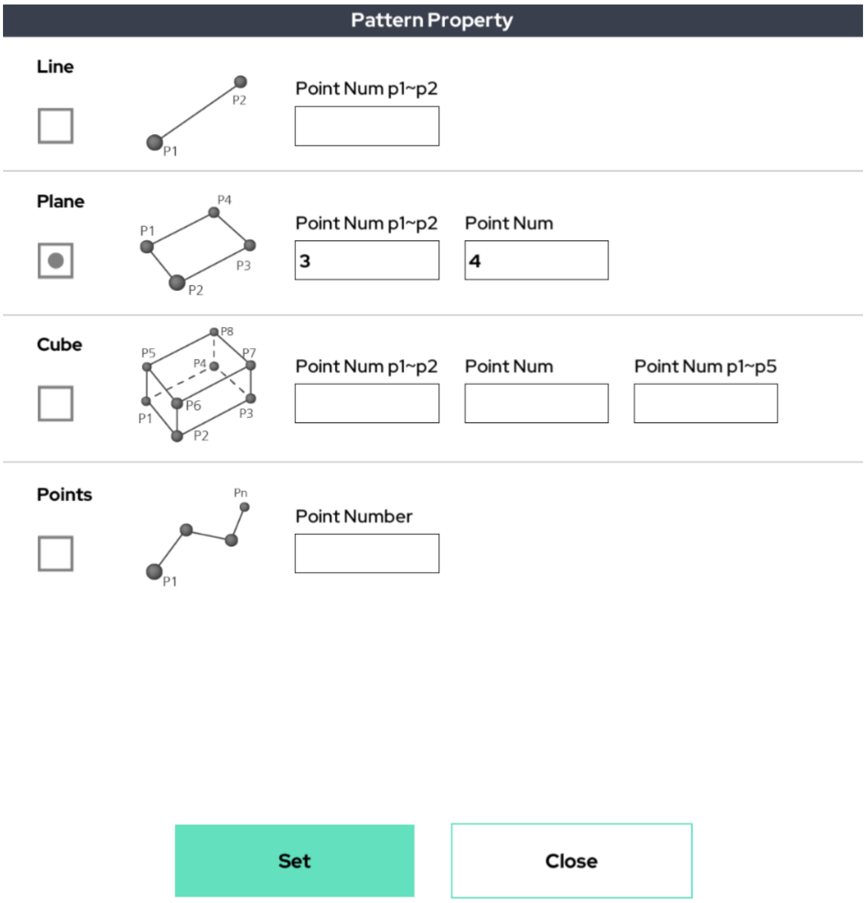

Since we set that there are 3x4 points in the plane pattern, 12 repetitions are required to do one cycle.

Set the 4 corner points that define the plane pattern

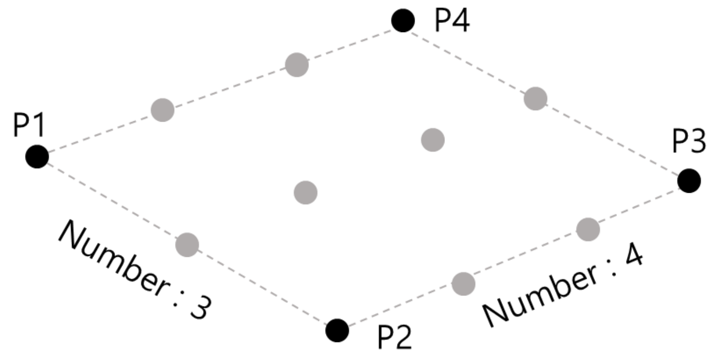
Set Pattern Anchor Point
(Normally this point sets a point equal to one of the corner points.)

Set the action other than the action to be performed in the pattern.

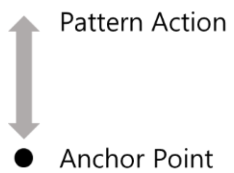
Step 1) Set the Pattern Property as shown below.



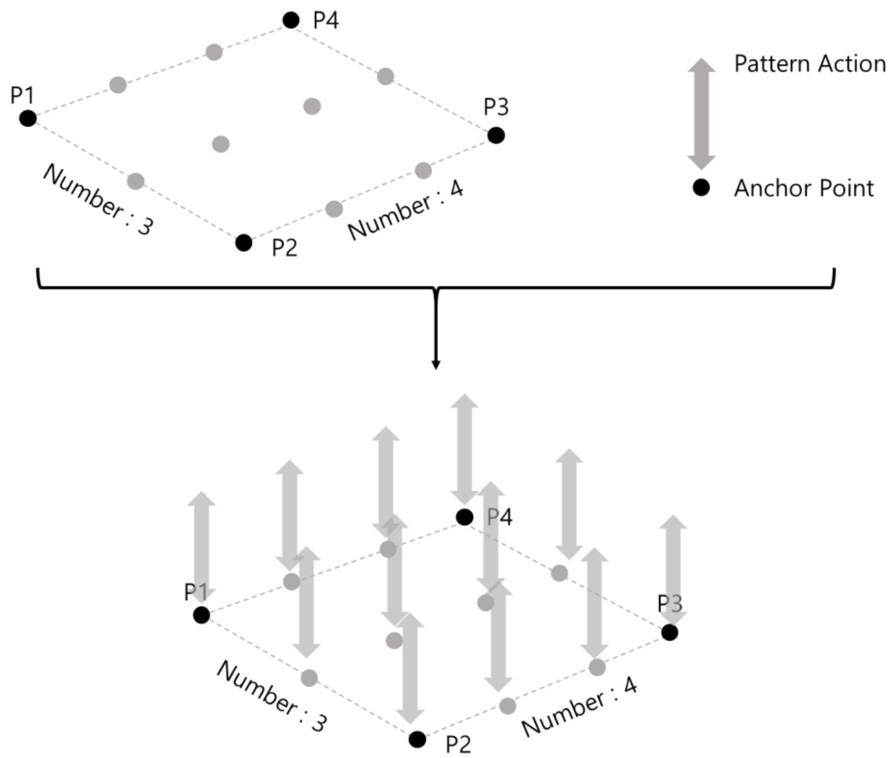

With the above settings, the following repeat points are formed in space.



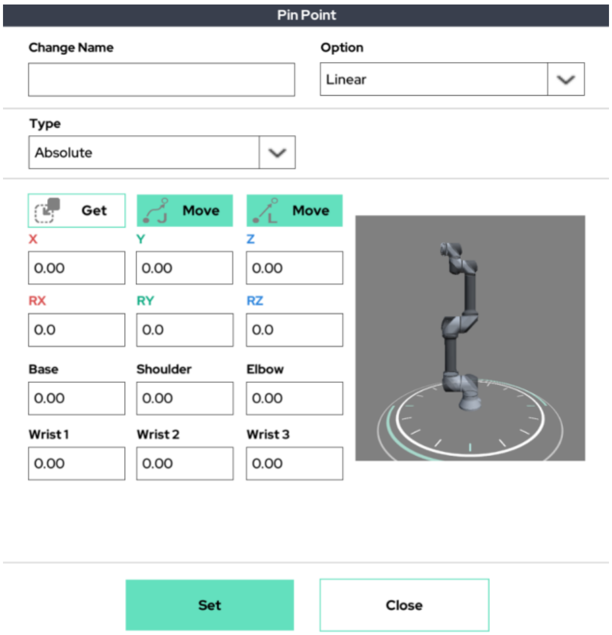

Step 2) Using the Pattern Anchor and the Pattern Action, define the relative movement as below




Step 3) f



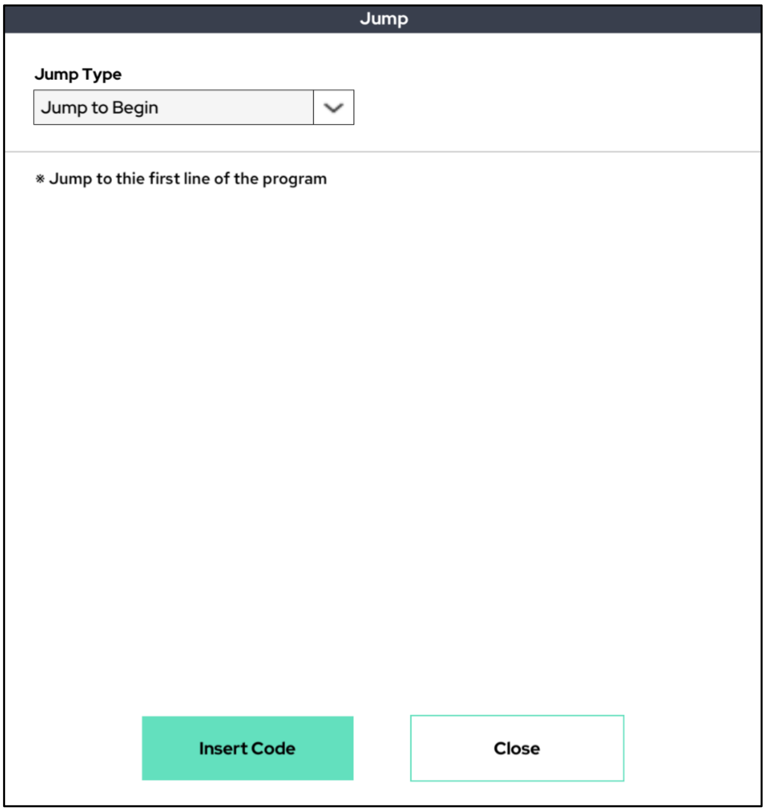
■ Pinpoint 기능 :




This is a special function for storing posture information only. This is a function to save information of a specific posture/position as a Point variable. If you create a PinPoint while teaching a specific posture and give it a PinPoint name, the posture information is converted into a Point variable.

The information saved as Point variable can be used in other operation commands/settings.

■ Jump 기능 :


This function allows you to discontinuously control the program flow. You can change the program flow through several sub-options.

Option) Jump to Begin

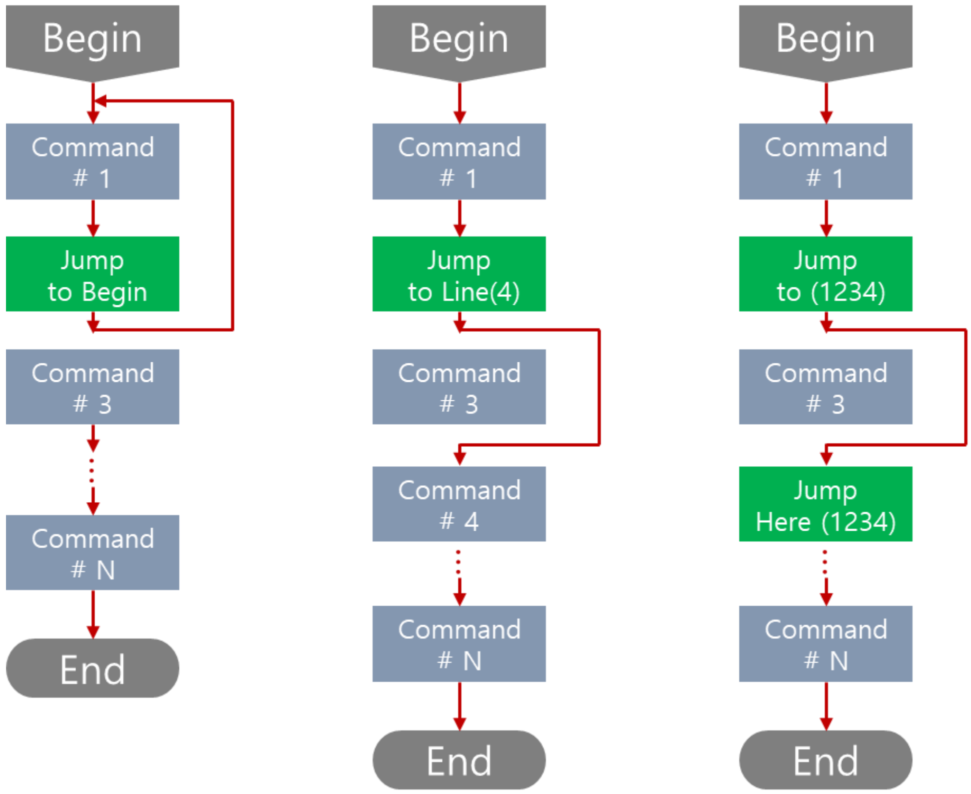
Move the program flow to the first line.

Option) Jump to Line

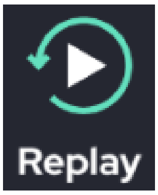
Move program flow to a specific line number.

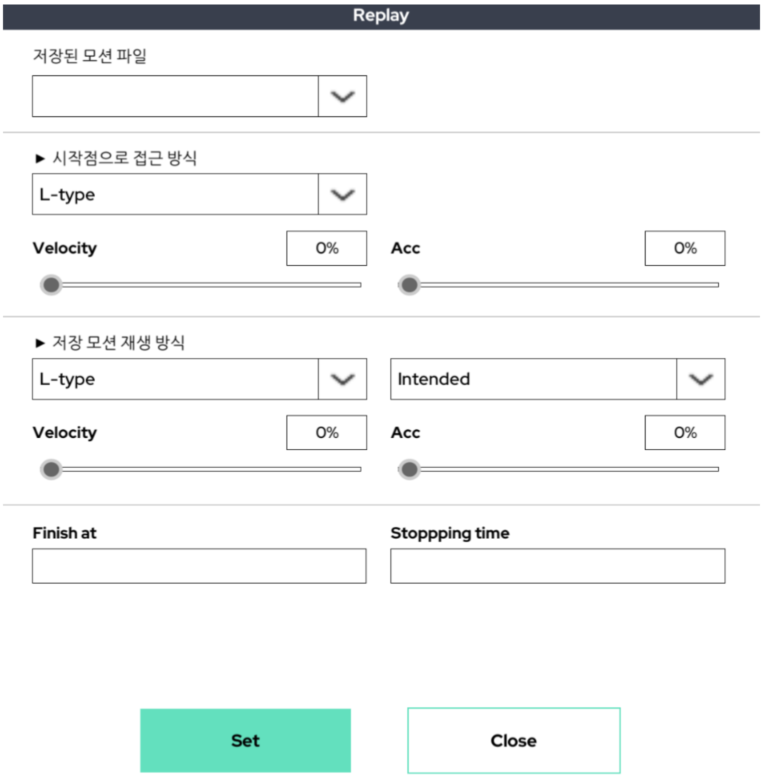
Option) Jump To / Here

Move program flow to a specific address value. JumpTo calls the address value you want to move, and JumpHere writes the address value.


Begin
Command
# 1
Jump
to Begin
Command
# 3
Command
# N
End
Begin
Command
# 1
Jump
to Line(4)
Command
# 3
Command
# 4
Command
# N
End
Begin
Command
# 1
Jump
to (1234)
Command
# 3
Jump
Here (1234)
Command
# N
End

■ Replay 기능 :

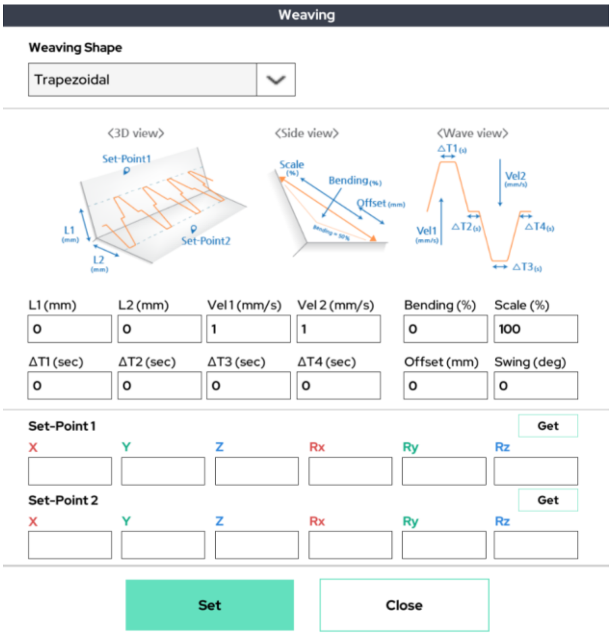


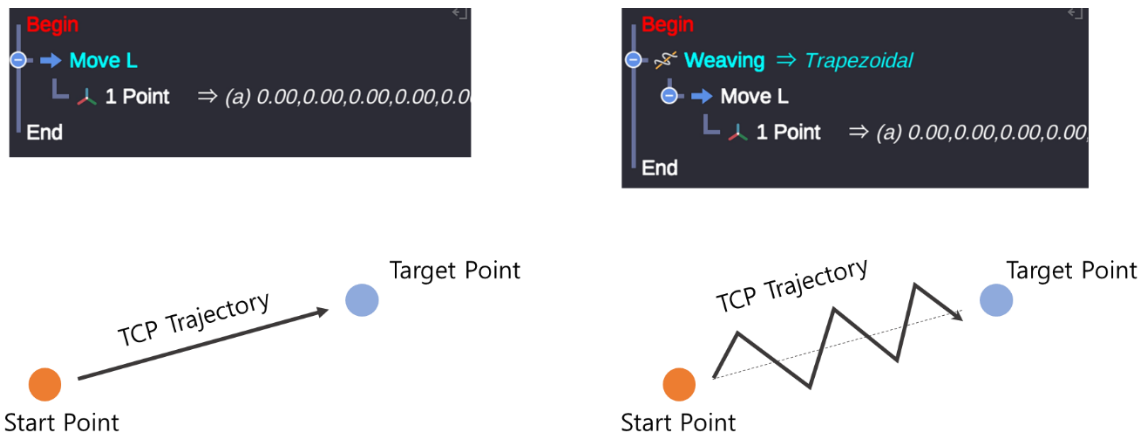

This function is to play the recorded teaching motion. Motion recording is performed in the settings of the Make page. If you select the name and motion speed/property of the recorded motion, the recorded motion is played again.

■ Weaving 기능 :


It is a special function for welding weaving. TCP trajectories are automatically changed to set the weaving actions included under the weaving function. Simply select and enter the desired weaving shape and weaving options.

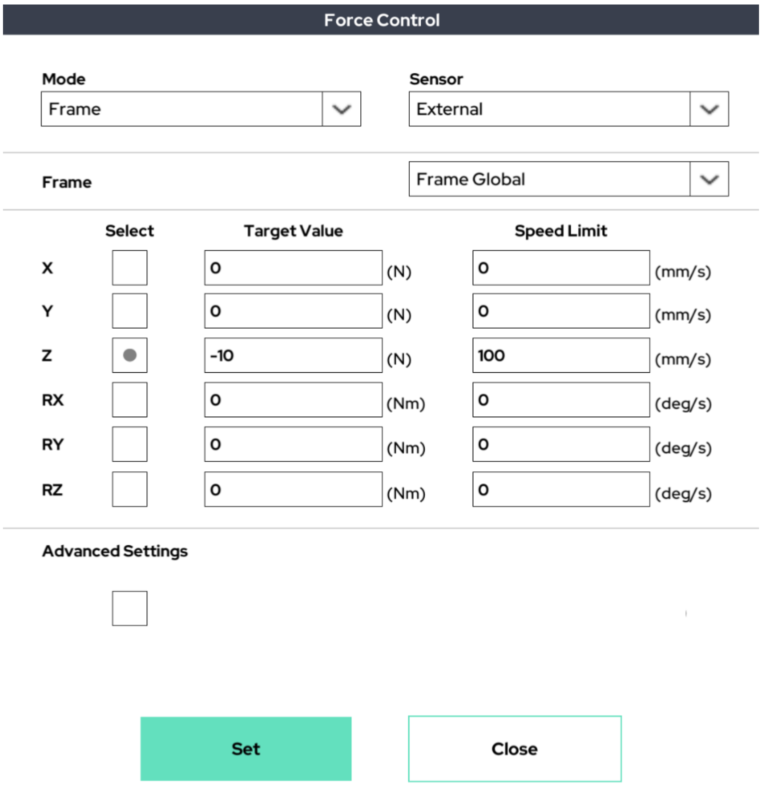



The left side of the figure below is for normal operation only. If this motion is put as a sub-item of weaving, TCP trajectory reflecting the weaving trajectory is drawn (in the example on the right, in the case of triangle wave weaving).



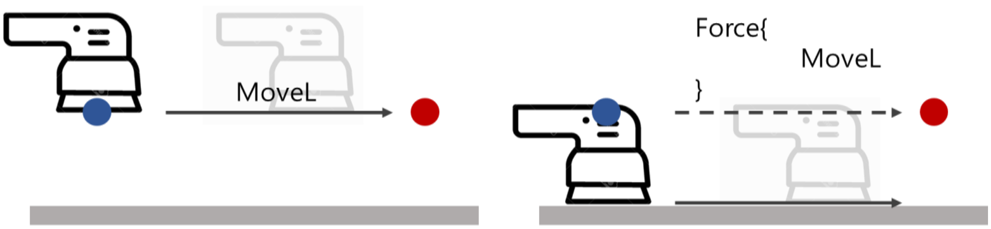
■ Force 기능 :


This is a function for force control. The movements below the force control function automatically change the trajectory to give the set force.



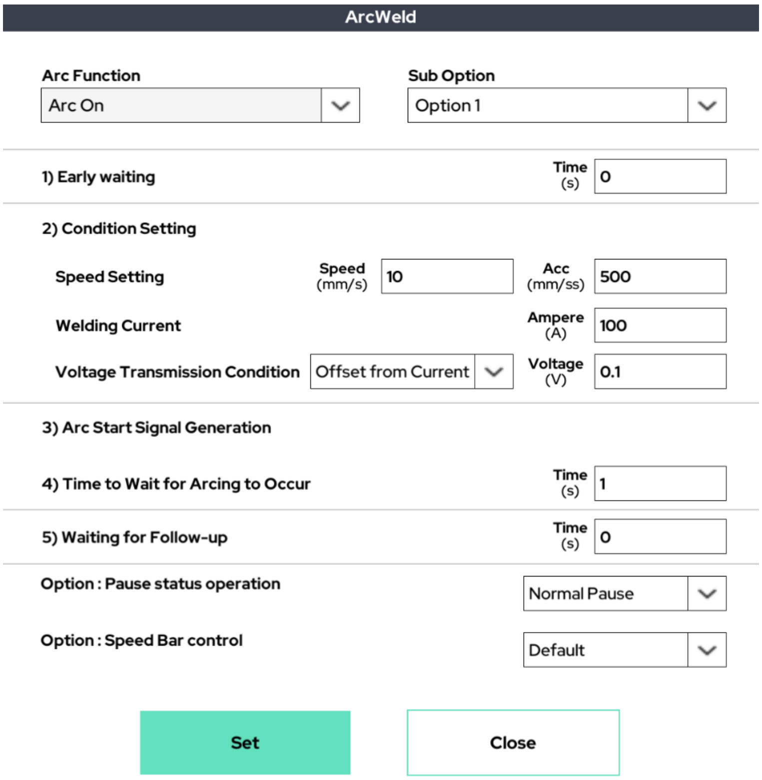

Select and input the desired force control mode, the sensor to be used for force control, and the force control target value.

The left side of the figure below is for normal operation only. The motion starts in the air above the plane and ends in the air. If you put this action as a sub-item of force control as it is, it will change to the action of pressing the ground with a certain force (when setting the force control to the ground).




<Before applying force control> <After applying force control>

■ Arcweld Function :

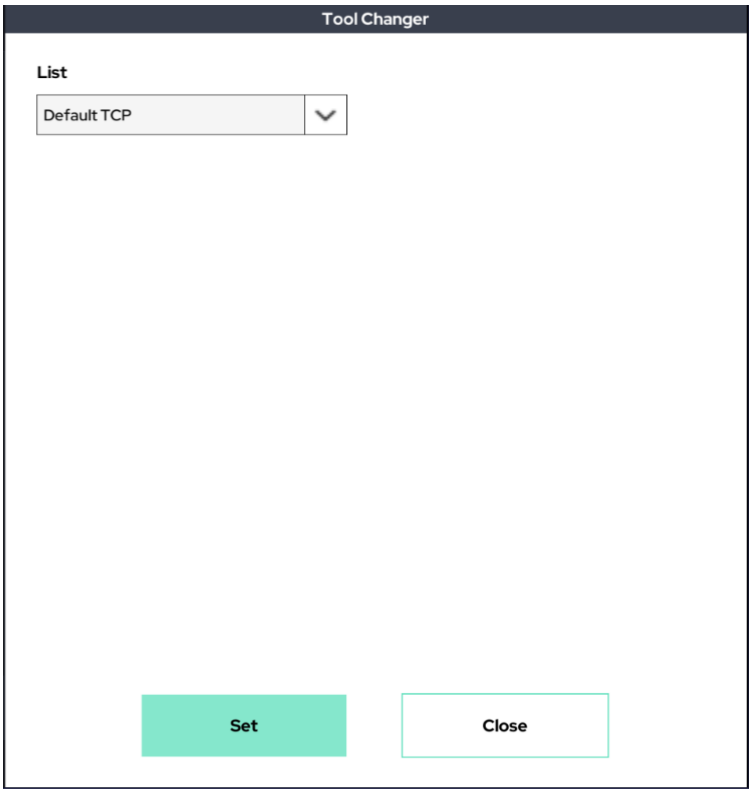




This is a special function for arc welding. A special macro function designed to quickly enable implementable functions, such as Wait / D.out.

To use this function, the Device field on the Setup page must precede setting the parameters and connection information for the welder.

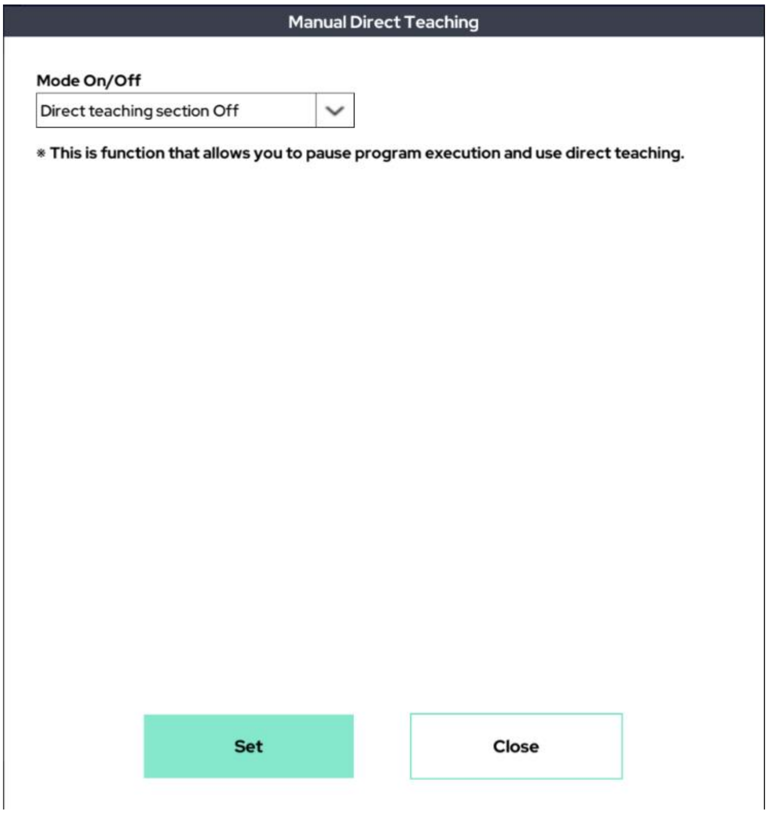
As illustrated above, this feature allows quick and easy insertion of weld speed/weld current / voltage settings / safety signal processing options into the program to be used for welding.

■ TCP Set Function :

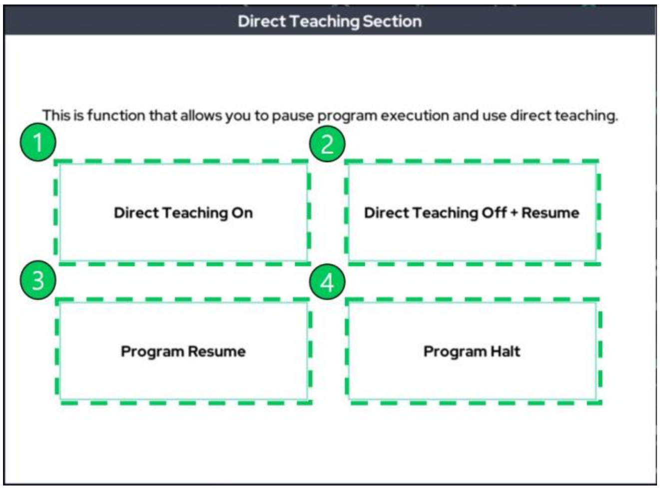




The ability to change the TCP value during program execution with the TCP value pre-saved in Setup' Tool List. It does not change again until the TCP value is replaced or the program is shut down.

■ Manual Direct Teaching Function :


A feature that enables direct teaching during program execution. When mode On, the program pauses when the manual direct teaching command is executed and a pop-up window as shown below appears on the screen.




You can select four features in the pop-up window.

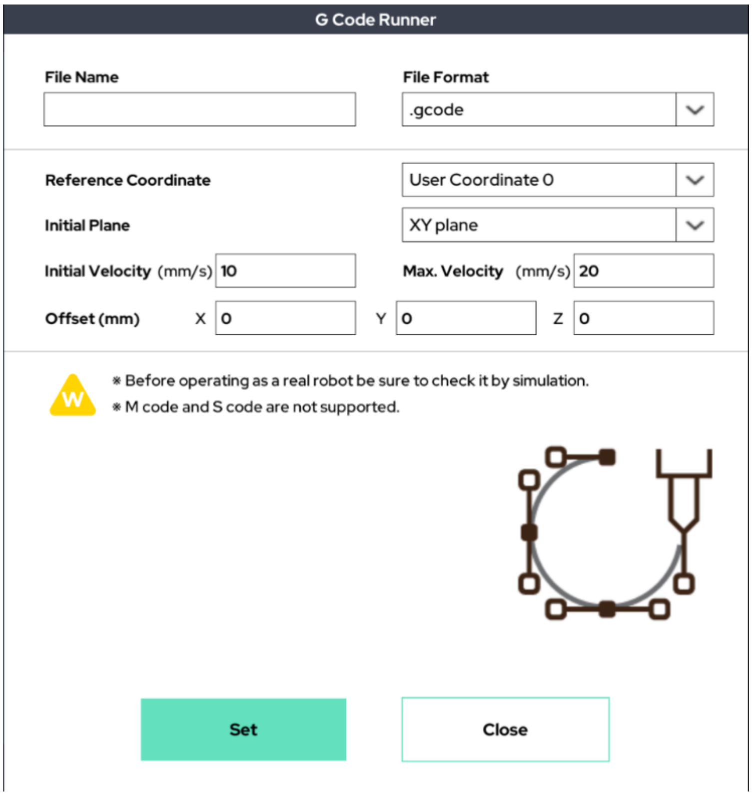
① Use the direct teaching feature while the program is paused.

② If you used the direct teaching feature in ①, turn off the direct contact function and resume the program.

③ Ignore the manual operation and resume the program.

④ Exit the program.

■ G Code Function :


This function allows the robot to move to the path stored in the G code. The G code file must be stored in a folder at the specified path (₩Tablet₩Android₩data₩com.rainbow.cobot₩files₩work) in advance to be available.

Enter the name of the G code file that user saved in File Name. The plane in which the robot moves can then specify the xy, yz, and zx planes of the user-specified coordinate system as the starting planes.

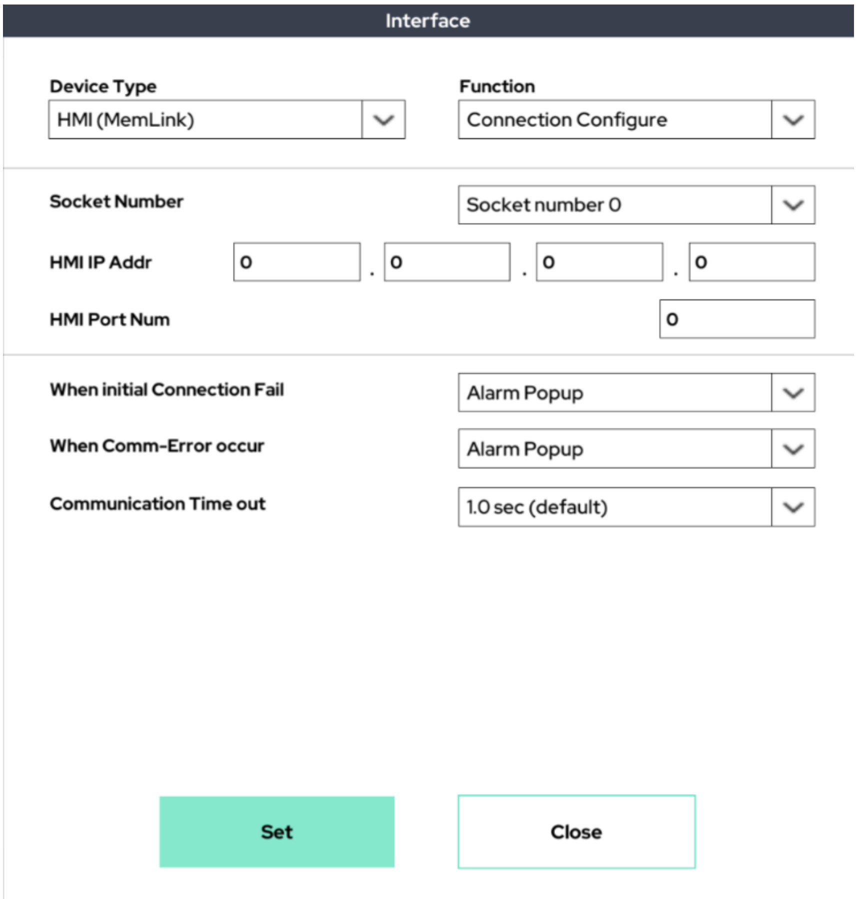
■ Interface Function :


The interface function is for connecting external devices such as PLC, HMI, and PC with the control box. The list of external devices that can be used using the interface is as follows.

- HMI(MemLink) - Proface, TOP
- PLC(MC Protocol) – Mitsubishi PLC
- Mulic Player
- PLC(XGT Protocol) - LS Electric PLC
- CSV File
- Pickit
- Modbus Client(Interrupt)

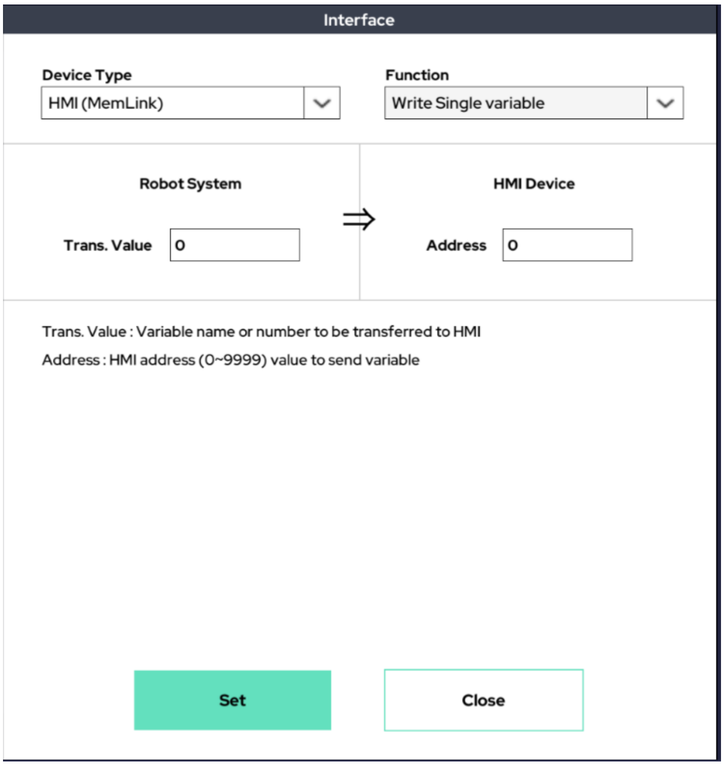
Because each external device has different detailed features available, you should refer to the following information.

HMI(MemLink)-Connection Configure



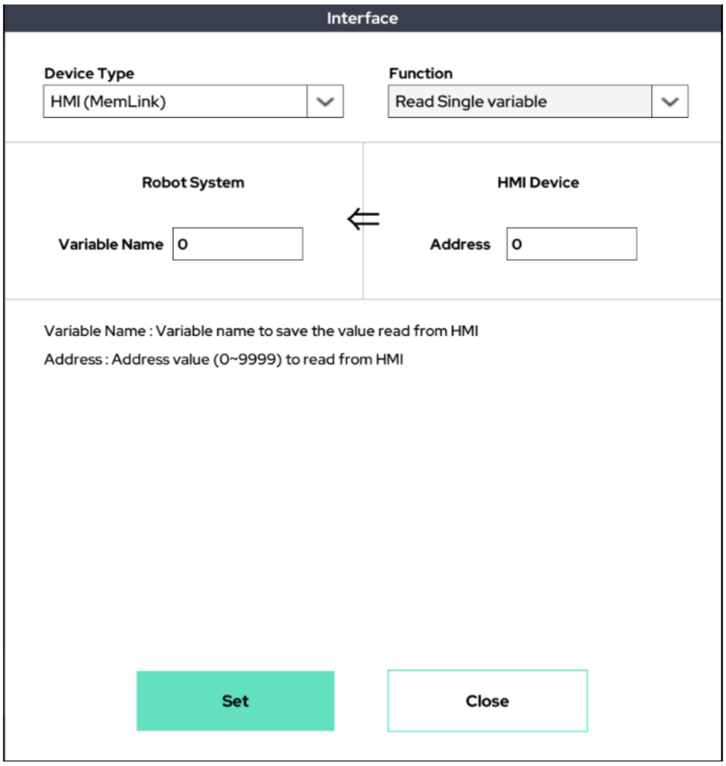

A function that connects communications between the HMI and the RB system. User will enter the socket number, IP address, and port. User can also decide whether to turn on or ignore alarm pop-up in the event of a connection failure or communication error and set a communication timeout time.

HMI(MemLink)-Write Single variable



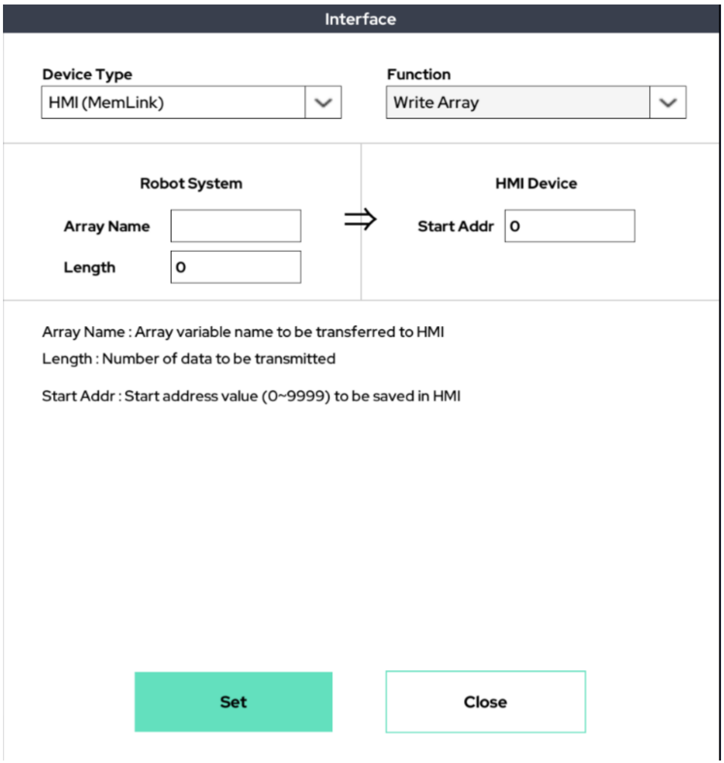

The ability to enter values for one address of HMI. Enter a number or variable name for the transfer value.

HMI(MemLink)-Read Single variable



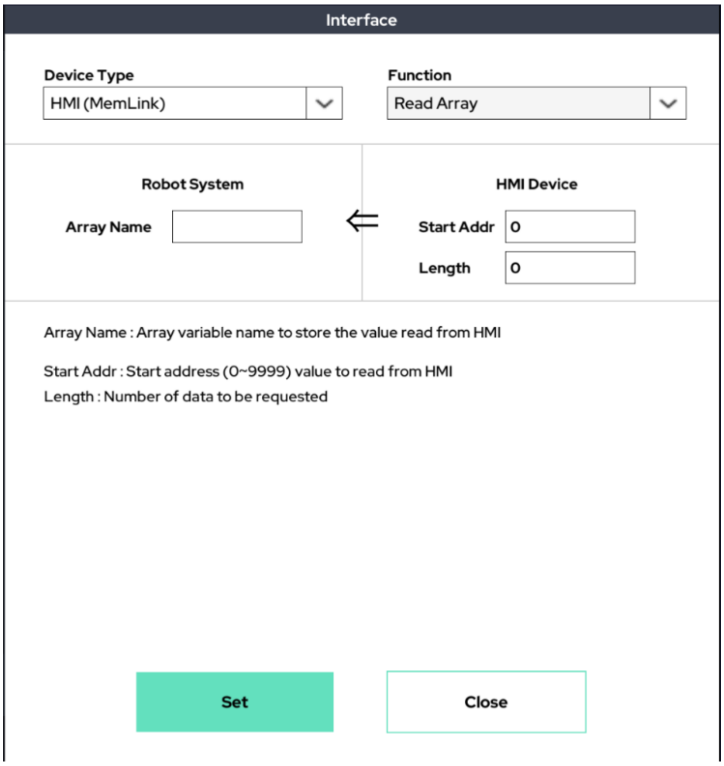

The ability to read values from one address in HMI. The read values are stored in the variable you specify (Variable).

HMI(MemLink)-Write Array



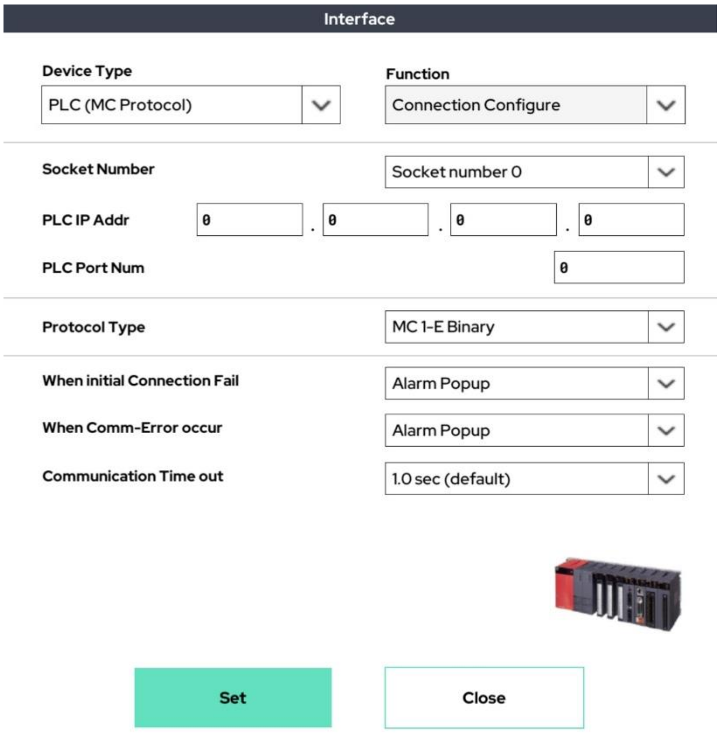

The ability to enter numbers from the starting address of the HMI to the specified number of addresses. The pre-declared array must be written to Array Name and should not exceed the maximum length of the array, 20.

HMI(MemLink)-Read Array



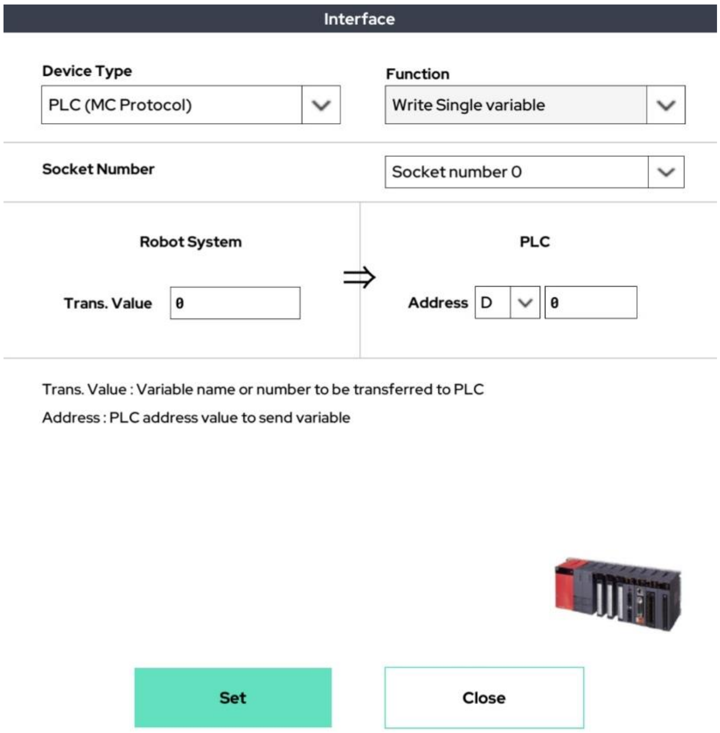

The ability to read data from the starting address of HMI to the specified number of addresses. The pre-declared array must be written to Array Name and should not exceed the maximum length of the array, 20.

PLC(MC Protocol)-Connection Configure



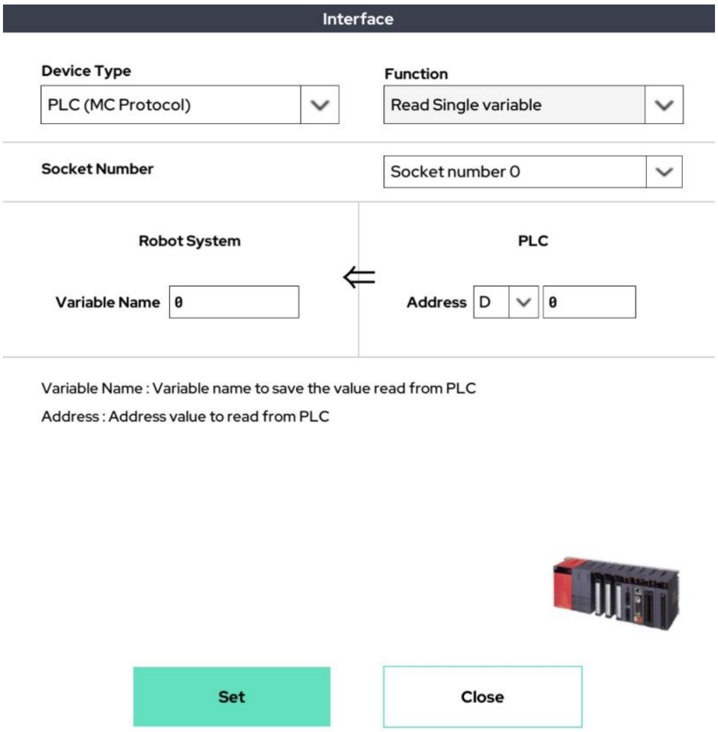

A function that connects communications between the Mitsubishi PLC and the RB system. User will enter the socket number, IP address, and port. User can also decide whether to turn on or ignore alarm pop-up in the event of a connection failure or communication error and set a communication timeout time.

PLC(MC Protocol)-Write Single variable



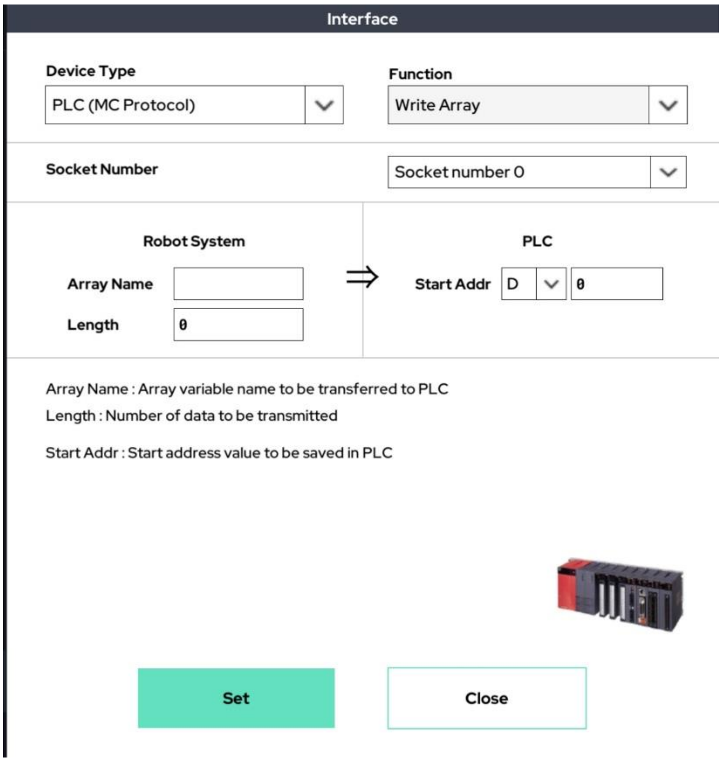

The ability to enter values for one address of PLC. Enter a number or variable name for the transfer value.

PLC(MC Protocol)- Read Single variable



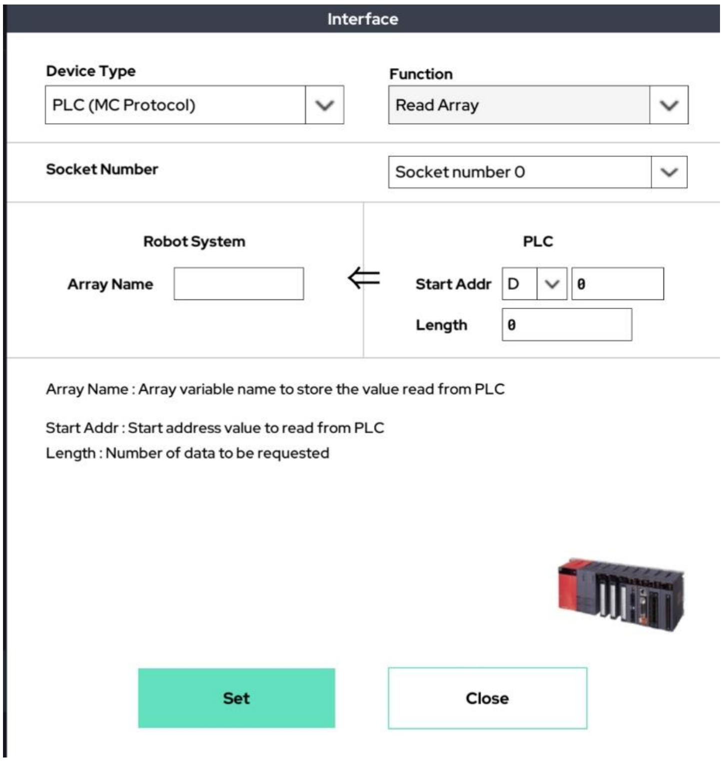

The ability to read values from one address in PLC. The read values are stored in the variable you specify (Variable).

PLC(MC Protocol)- Write Array



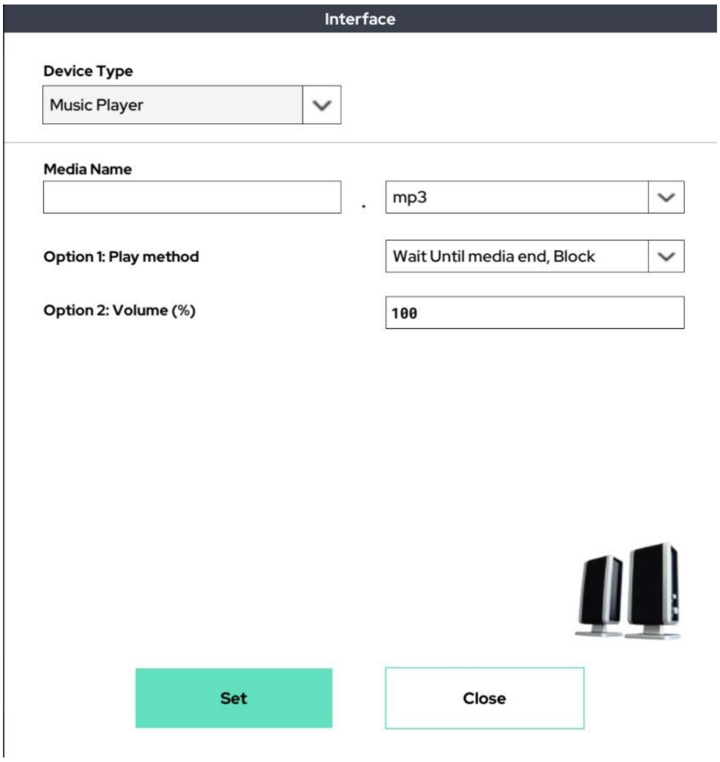

The ability to enter numbers from the starting address of the PLC to the specified number of addresses. The pre-declared array must be written to Array Name and should not exceed the maximum length of the array, 20

PLC(MC Protocol)- Read Array



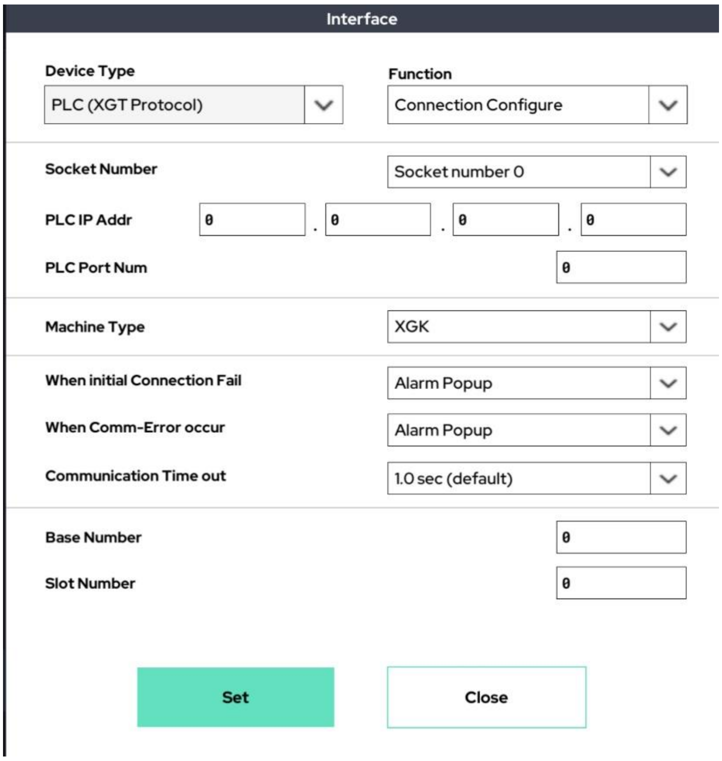

The ability to read data from the starting address of PLC to the specified number of addresses. The pre-declared array must be written to Array Name and should not exceed the maximum length of the array, 20.

Music Player



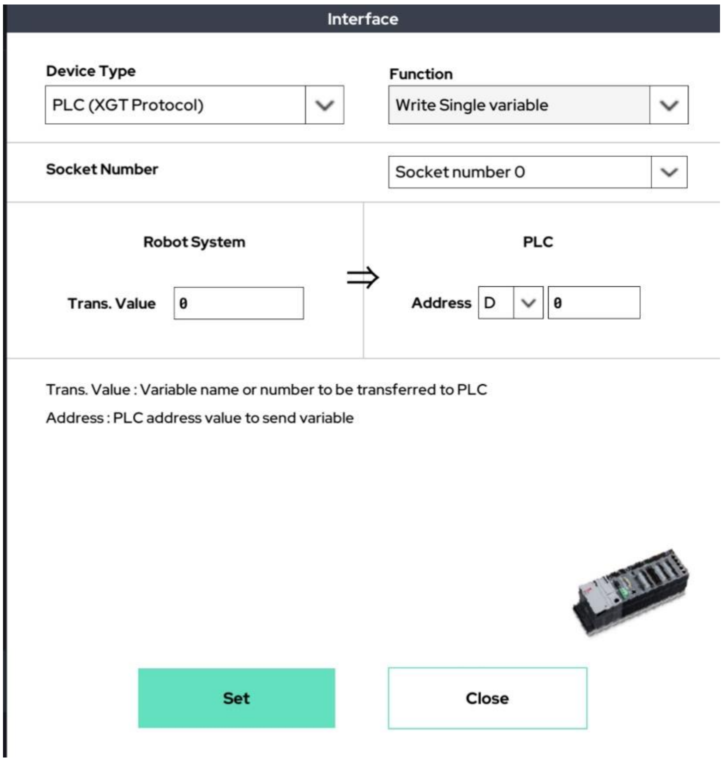

This function plays an mp3 file while the program is running. The Music driver must be installed through the RB Driver, and the mp3 file you want to play must exist in the specified path.

PLC(XGT Protocol)-Connection Configure



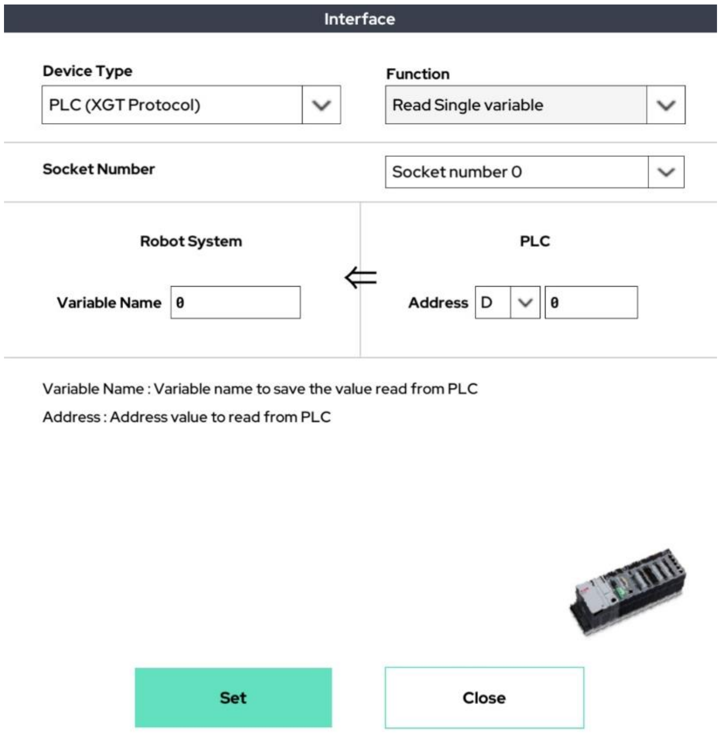

A function that connects communications between the LS Electric PLC and the RB system. User will enter the socket number, IP address, and port. User can also decide whether to turn on or ignore alarm pop-up in the event of a connection failure or communication error and set a communication timeout time.

PLC(XGT Protocol)-Write Single variable



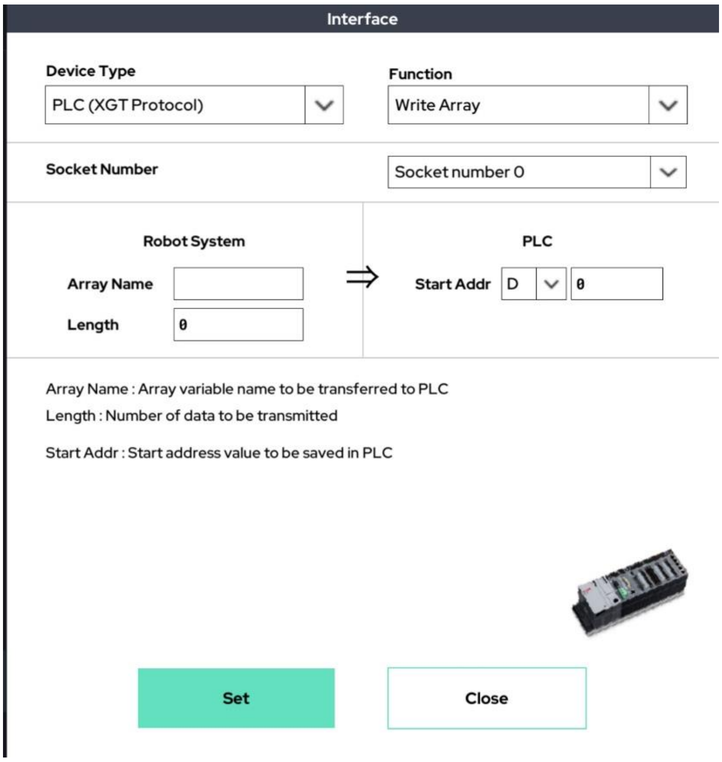

The ability to enter values for one address of PLC. Enter a number or variable name for the transfer value.

PLC(XGT Protocol)-Read Single variable



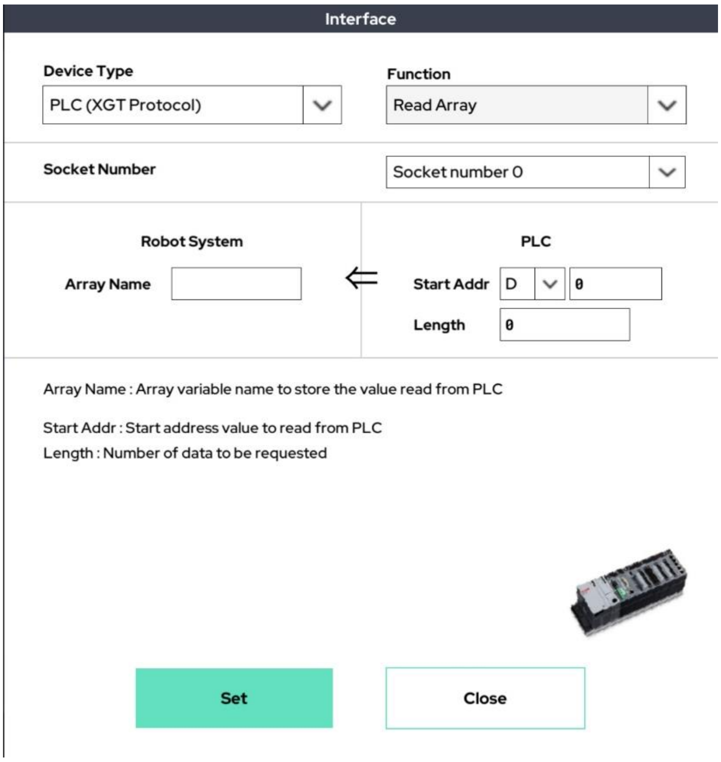

The ability to read values from one address in PLC. The read values are stored in the variable you specify (Variable).

PLC(XGT Protocol)-Write Array



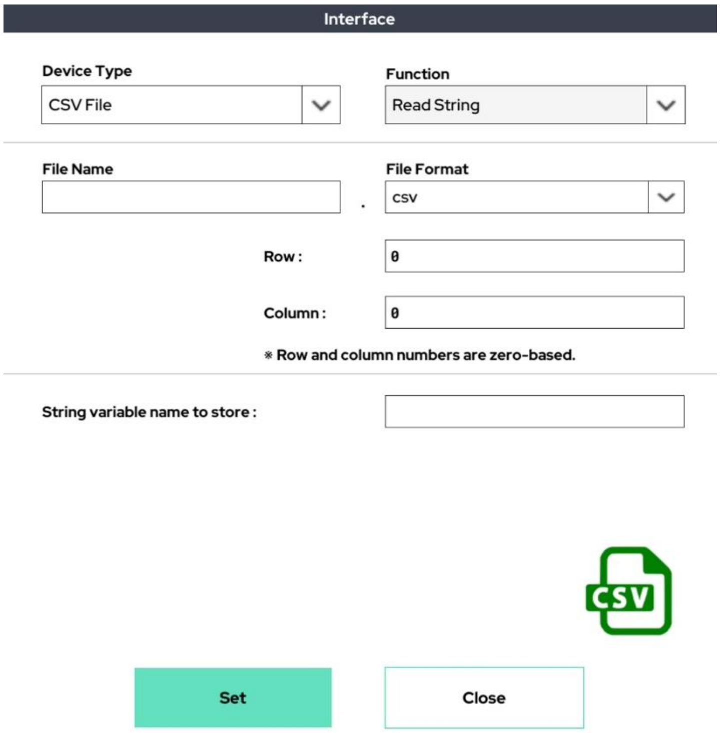

The ability to enter numbers from the starting address of the PLC to the specified number of addresses. The pre-declared array must be written to Array Name and should not exceed the maximum length of the array, 20

PLC(XGT Protocol)-Read Array



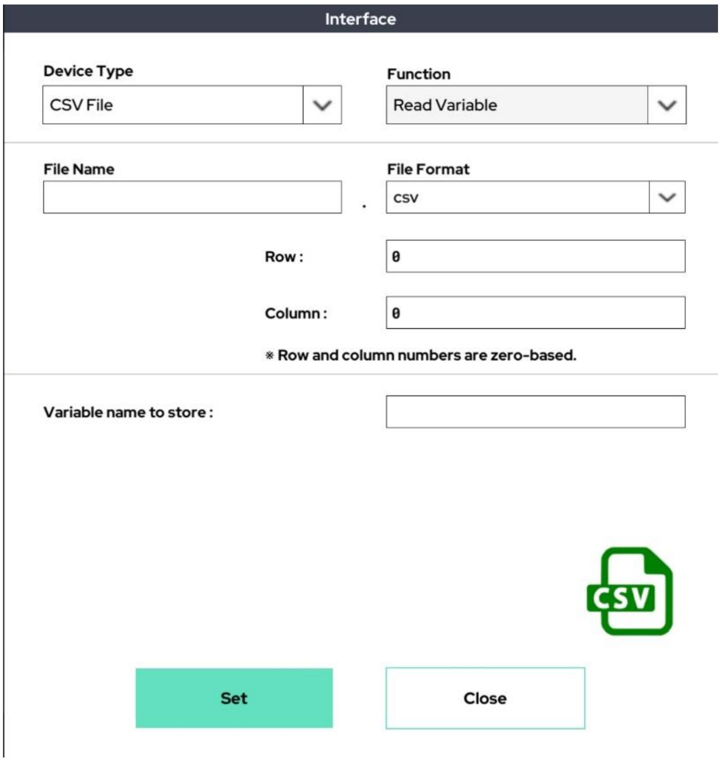

The ability to read data from the starting address of PLC to the specified number of addresses. The pre-declared array must be written to Array Name and should not exceed the maximum length of the array, 20.

CSV File-Read String



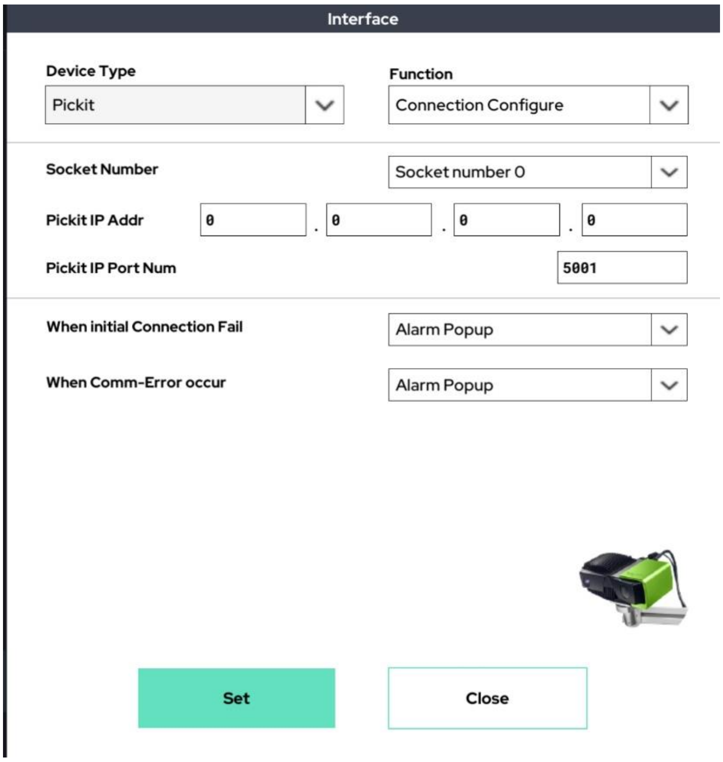

This function reads a string from a CSV file. The CSV file must be saved within the specified path.

CSV File-단일 숫자 읽기



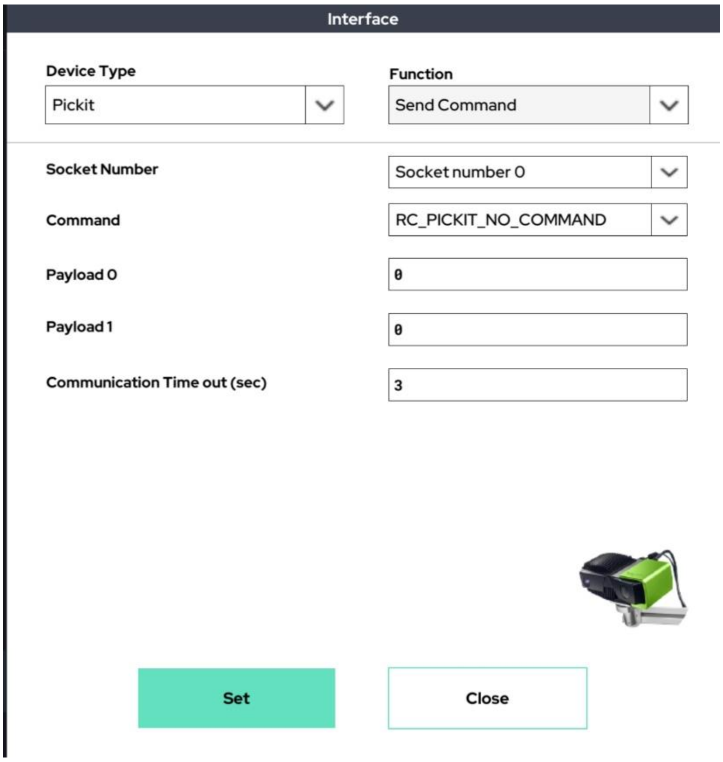

This function reads a single number from a CSV file. The CSV file must be saved within the specified path.

Pickit-Connection Configure



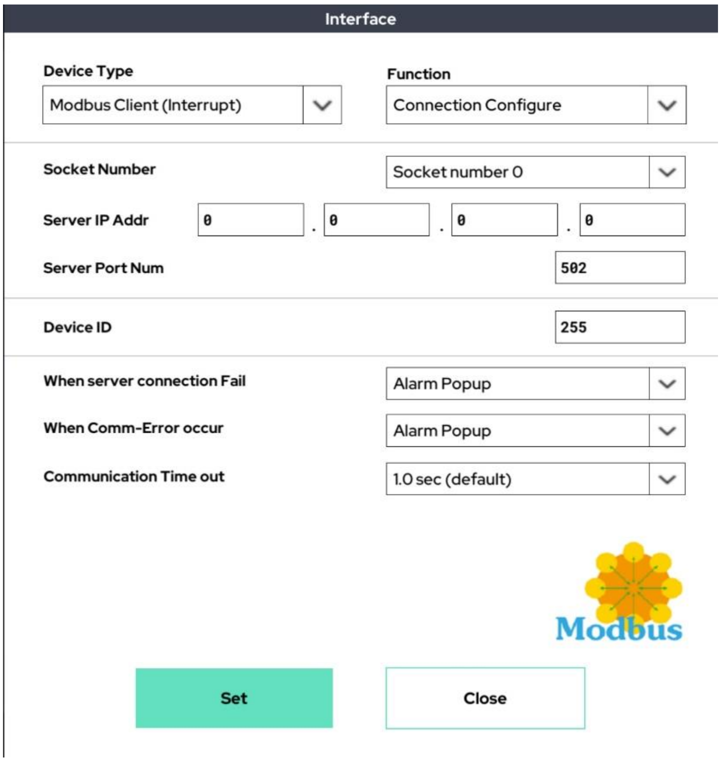

A function that connects communications between the LS Electric PLC and the RB system. User will enter the socket number, IP address, and port. User can also decide whether to turn on or ignore alarm pop-up in the event of a connection failure or communication error.

Pickit -Send Command



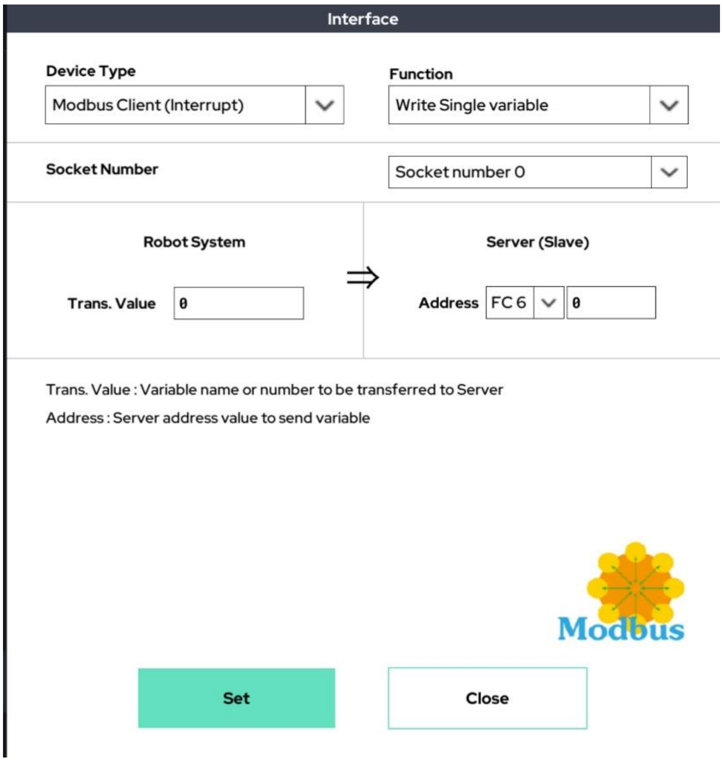

Set the command to be sent to Pickit and the data according to the command.

Modbus Client(Interrupt) -Connection Configure



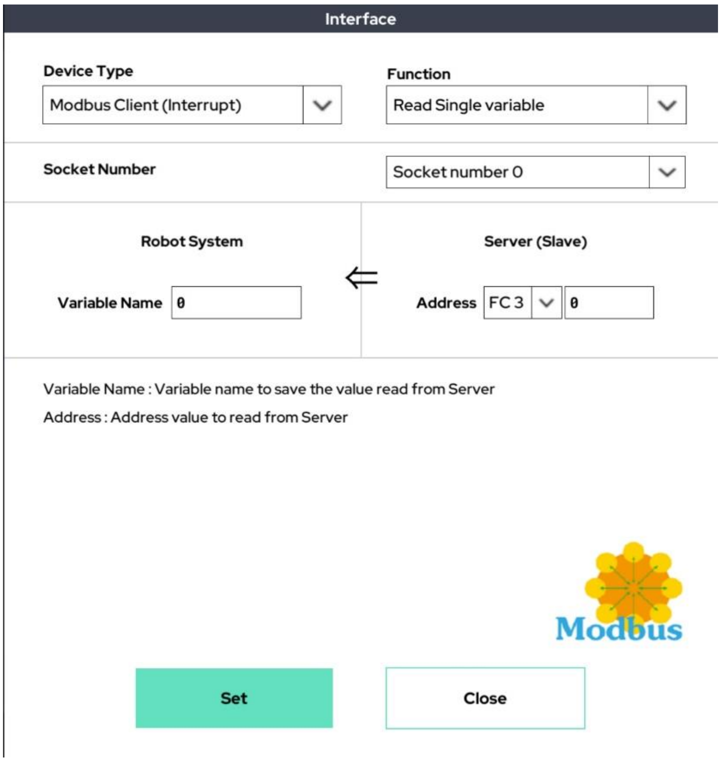

This is a function that connects the RB system as a client in Modbus communication. User will enter the socket number, IP address, and port. User can also decide whether to turn on or ignore alarm pop-up in the event of a connection failure or communication error and set a communication timeout time.

Modbus Client(Interrupt) -Write Single variable



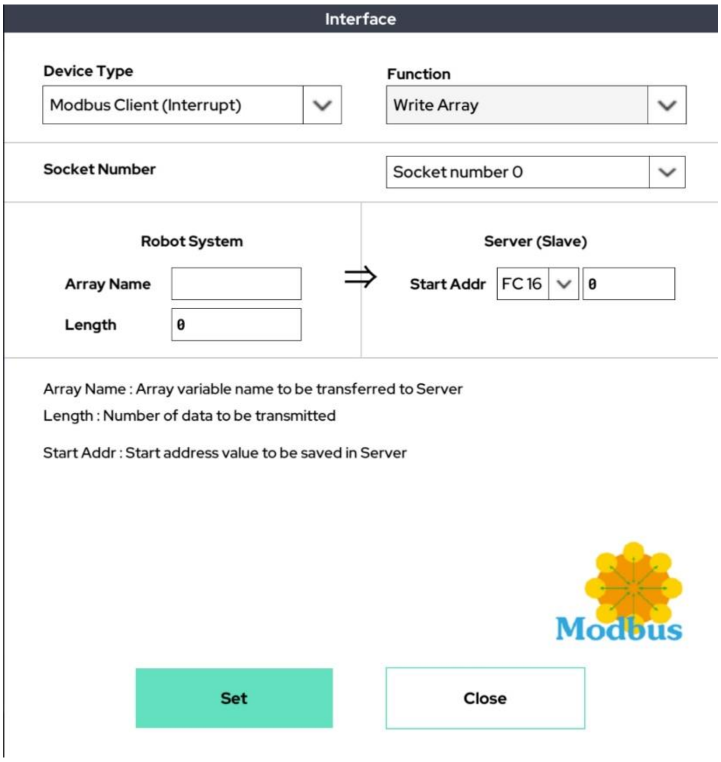

This is a function to input word type data to one address through Modbus communication. At this time, enter the name of a number or variable for the transfer value.

Modbus Client(Interrupt) -단일 숫자 읽기



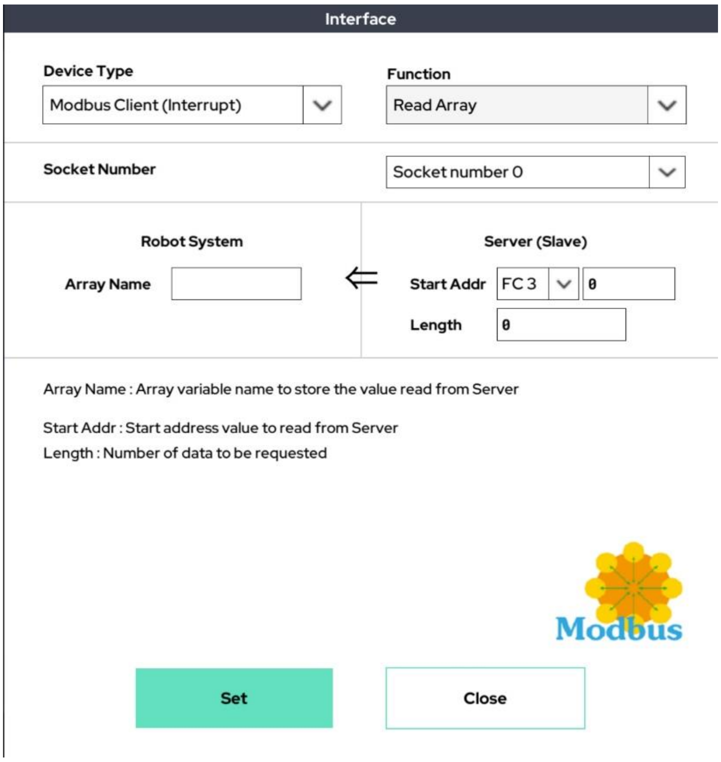

This is a function to read the value of one word type data from the address through Modbus communication. At this time, the read value is saved in the assigned variable.

Modbus Client(Interrupt) -다중 숫자(배열) 쓰기



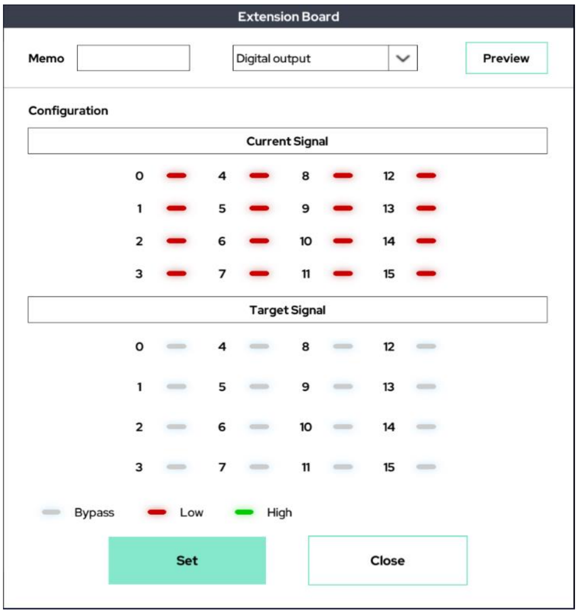

This is a function to input word data from the start address to the specified number of addresses through Modbus communication. At this time, the previously declared array should be written in ‘Array Name’ and the length should not exceed 20, the maximum length of the array.

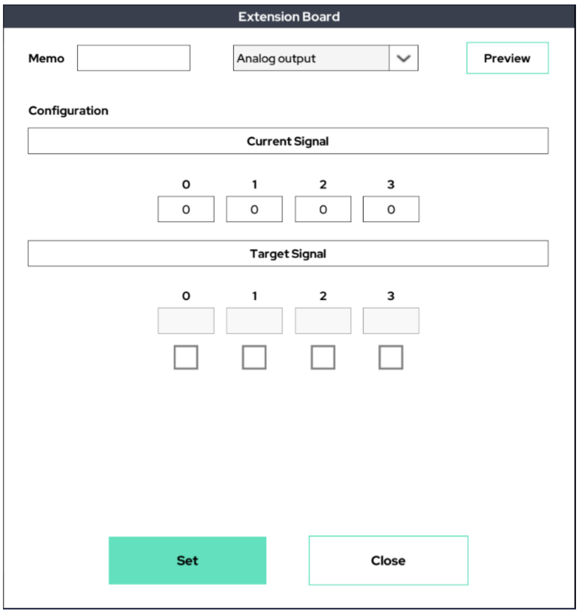
Modbus Client(Interrupt) -다중 숫자(배열) 읽기




This function reads data from the start address to the specified number of addresses through Modbus communication. At this time, the previously declared array should be written in ‘Array Name’ and the length should not exceed 20, the maximum length of the array.

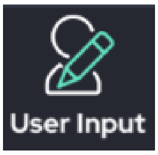
■ Extension Board Function :

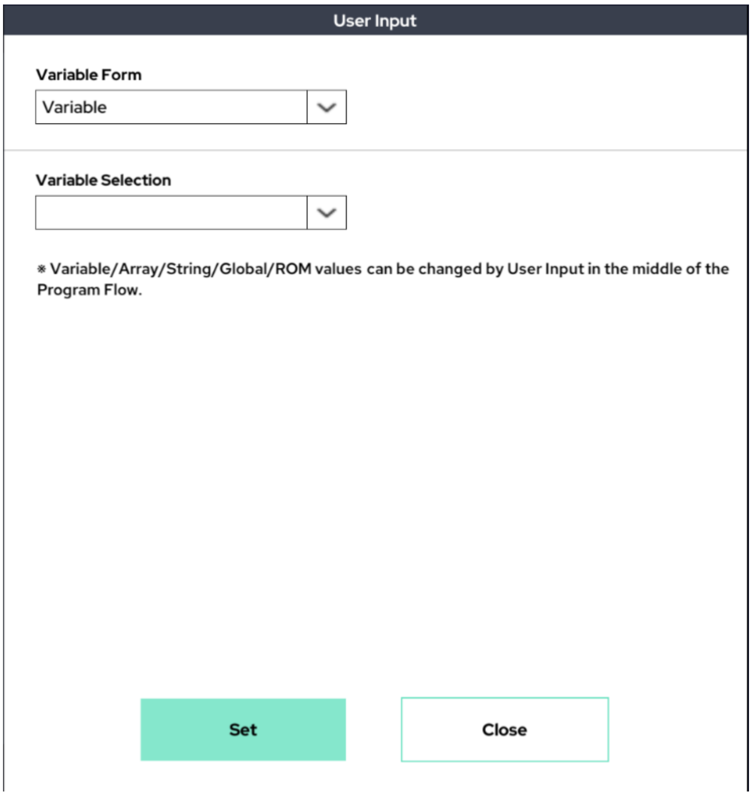




A feature that controls digital/analog output when purchasing and using an extended I/O module. The method of use is the same as the existing D.output and An.output.

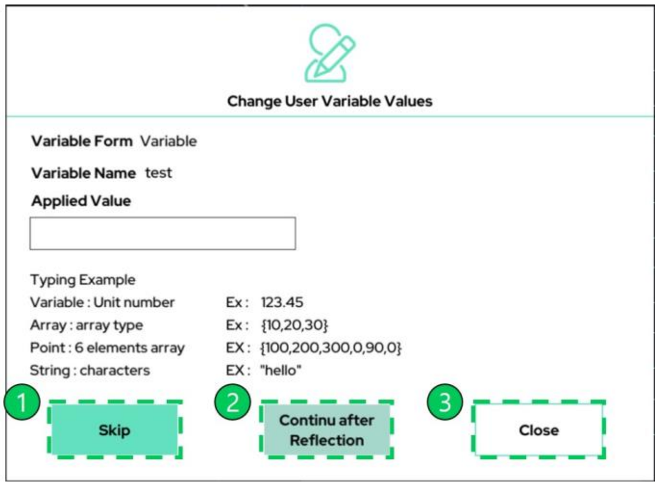
■ User Input Function :




This feature is used when a user wants to randomly change the value of a specific variable while the program is running. Available for Variable/Array/Point/String/Global/ROM variables.

When launched, a pop-up window will appear as follows:



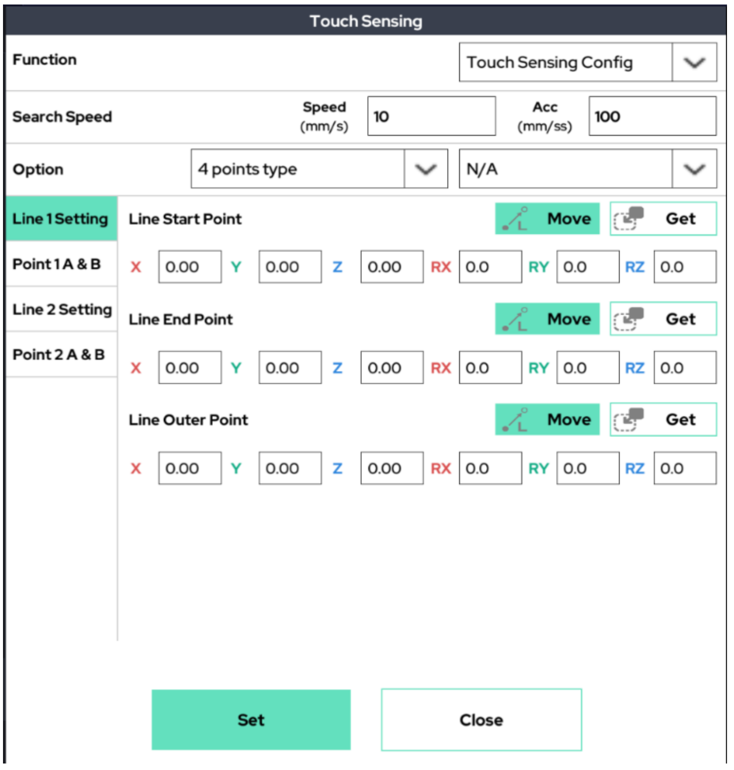

You can select three features in the pop-up window.

① Resume the program without replacing the corresponding variable.

② Enter the data you want to change in 'Applied Values', then press to reflect the data you entered in 'Applied Values', and then resume the program.

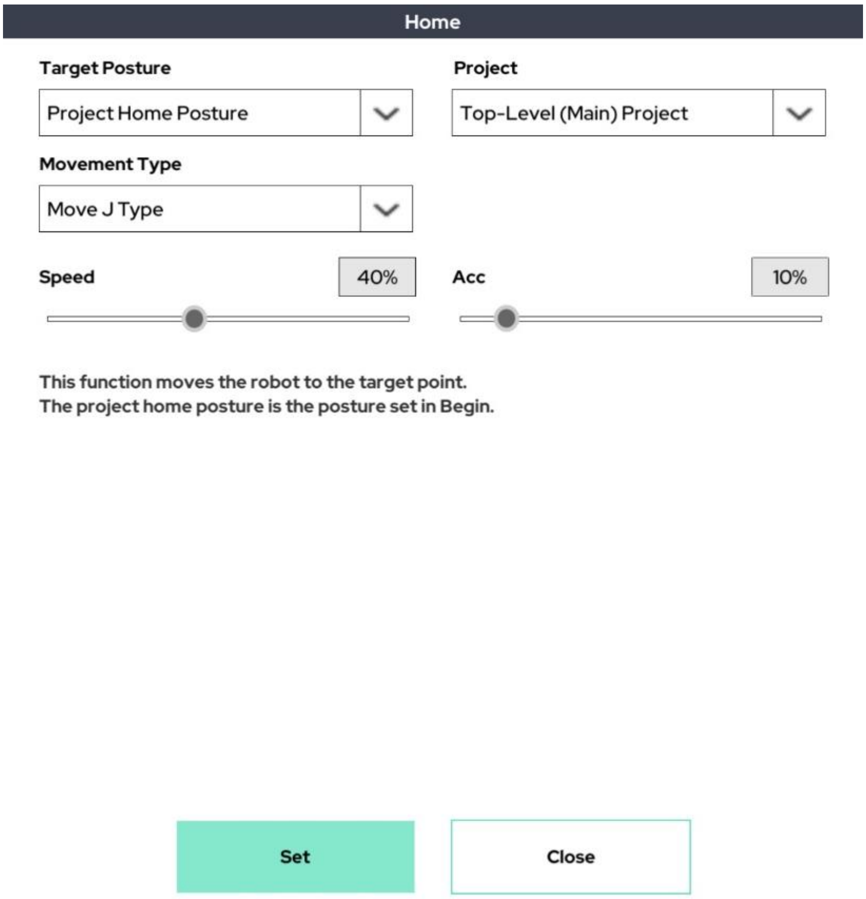
③ Exit the program.

■ Touch Sensing Function:

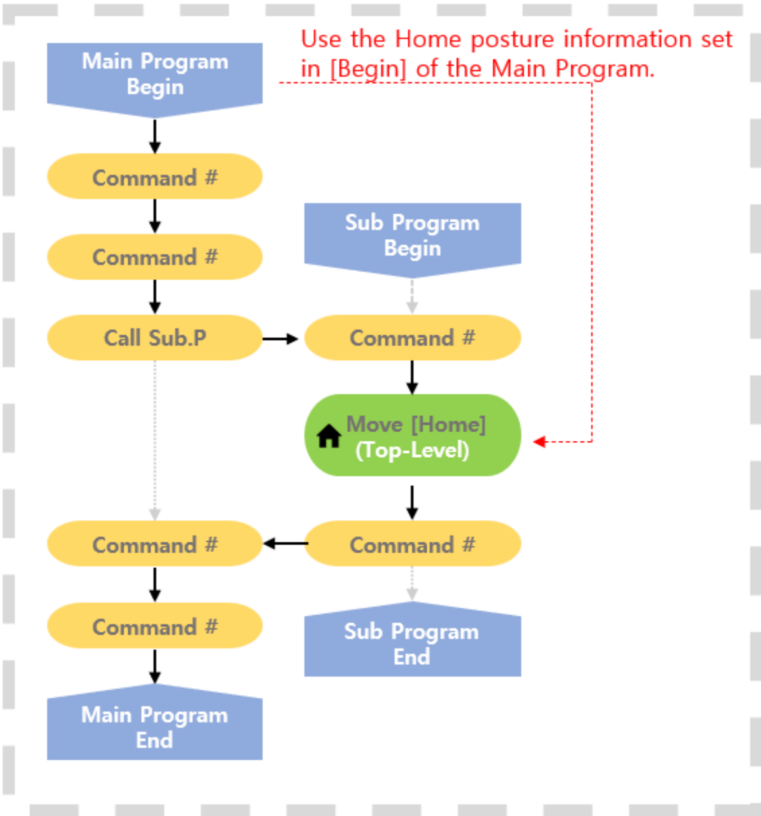

Touch sensing is intended to utilize welding applications. Detects the movement of the base material and reflects the direction of movement of the base material and is used for welding.
A detailed description of this feature is provided in a separate manual.

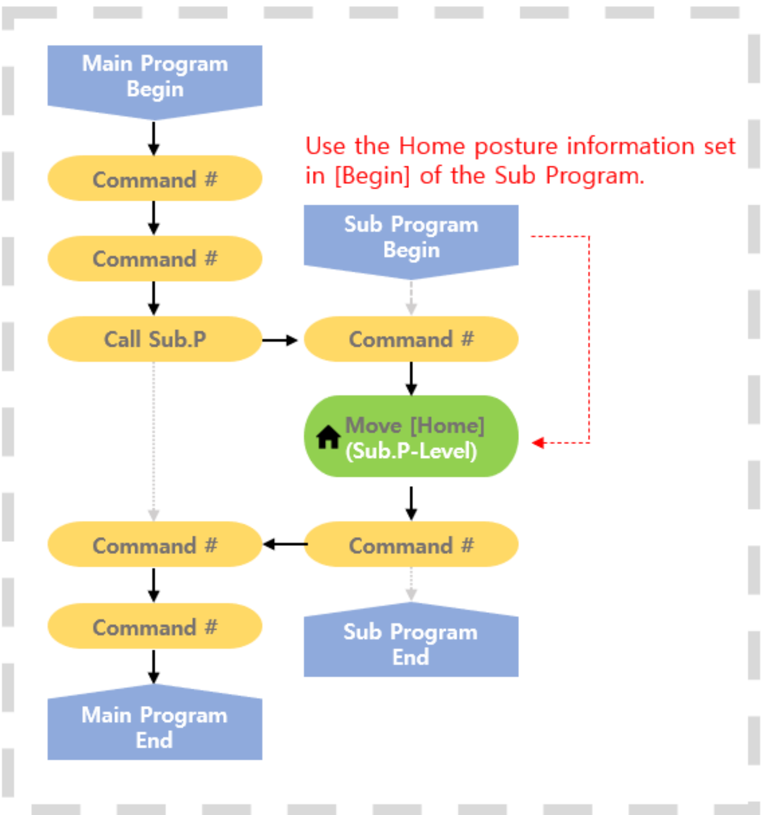
■ Home：

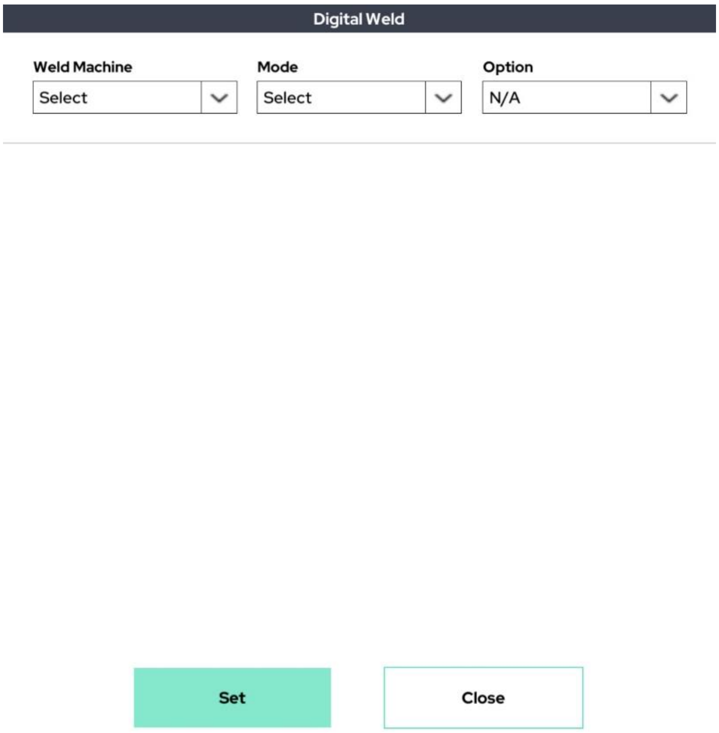

Home function is a function to move the robot with Project Home Posture or Joint Zero Posture. At this time, user can select the movement type. When going to the Project Home Posture, user can select Project Home Posture of the main program and Project Home Posture of the subprogram.

The diagram below shows the difference between the case of going to the Project Home Posture of the main project and the case of going to the Project Home Posture of the subproject when using the home function within the subprogram.



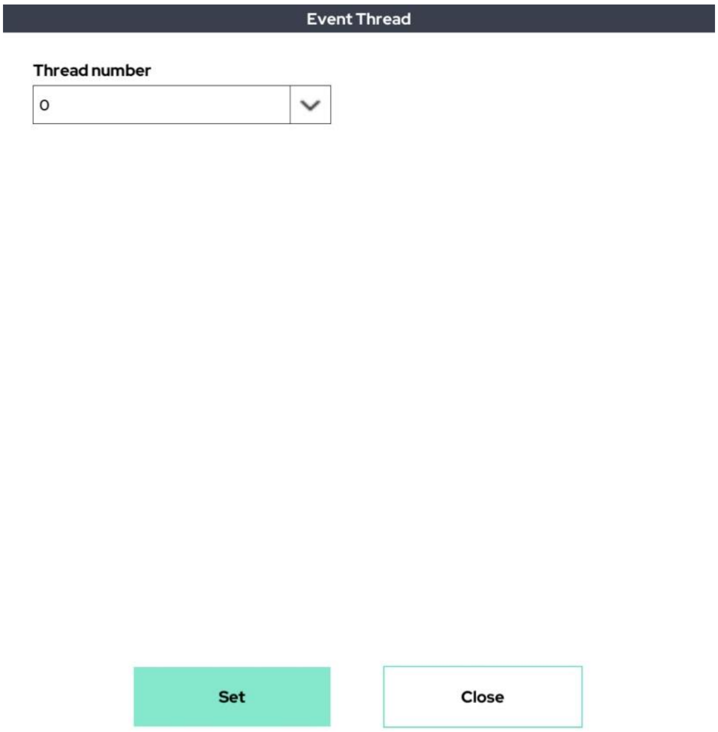



■ D.Weld :

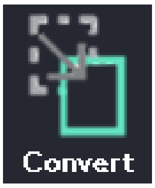

It is a function that can use the digital weld machine. After selecting the weld machine to be used, user can proceed with 'Weld Start', 'Weld Off', and 'Weld Setting'.

■ Event Thread Call :

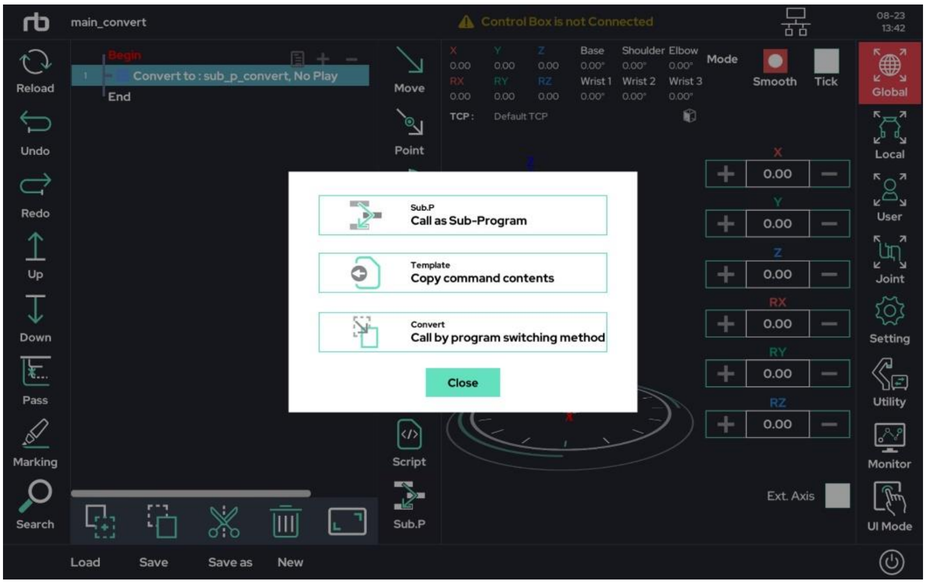

This is a function to run the event thread in the main program. The event thread is executed only when the event thread call is executed in the main program. In this case, the number of the event thread to be executed can be selected.

■ Convert :

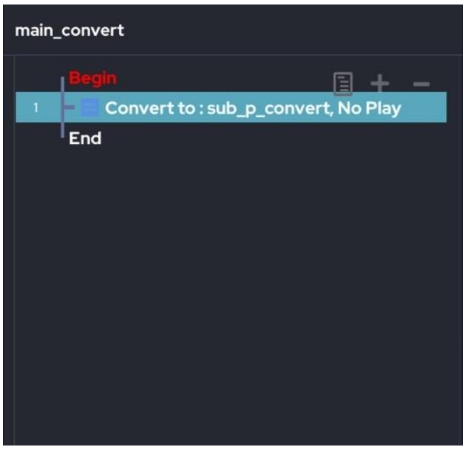



This function is to convert the main program. Unlike the existing Sub.P and Tmplate, this is a function that changes the main program itself, so the program displayed on the UI will also change.

If you click the Sub.P icon in the program, the following pop-up window appears, and at this time, press Call program conversion method.




If you select the program you want to switch to in the file explorer popup that appears after clicking, the command is created as shown below.



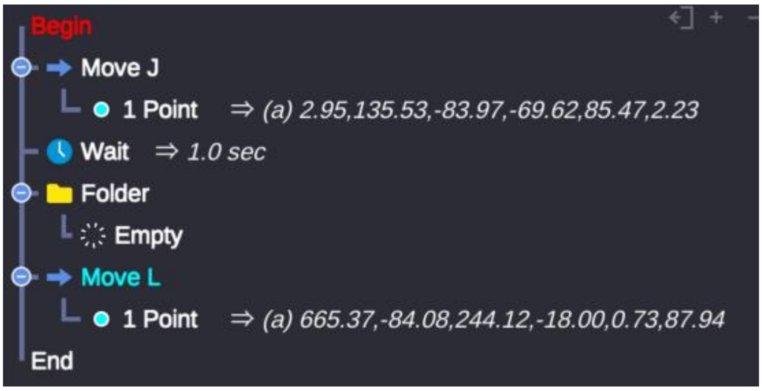
## 7.5 Editing The Program

The bar on the left of the screen contains icons that allow a user to change the order or structure of the instructions entered in the program tree.

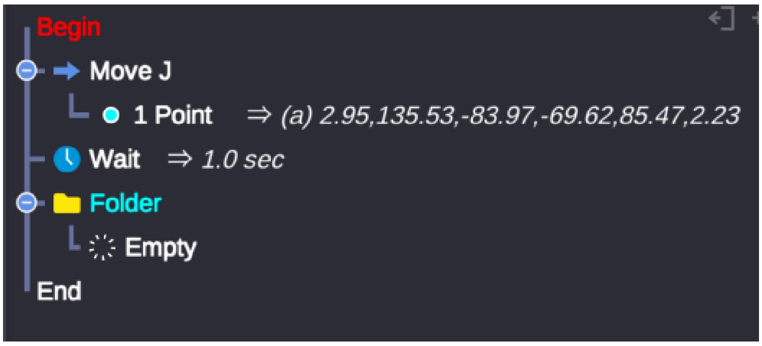
Please refer to section 6.1 for the description of the edit icon. The example explains how to edit the program.

■ Cut/ Paste  

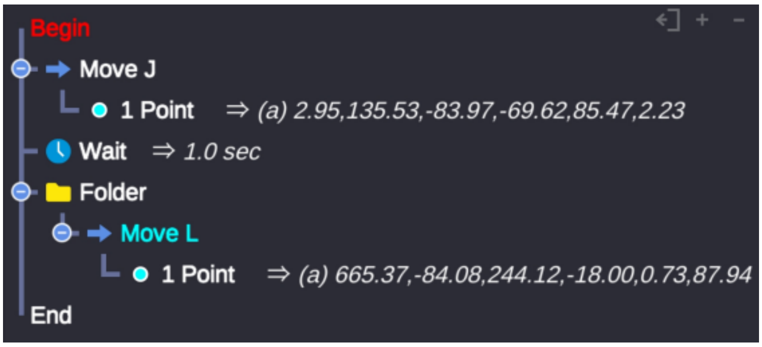
Step1) Select the command to cut. The selected command will be shown in blue. In the example below, the MoveL line is selected.

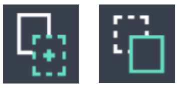



Step2) Press the Cut button. Once Cut is clicked, the line disappears from the program tree.

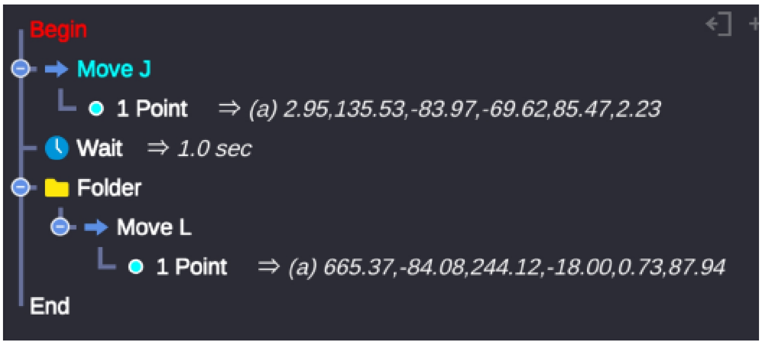


Step3) Click the location to paste and click the Paste button. In the example, the MoveL command is pasted inside the Folder.



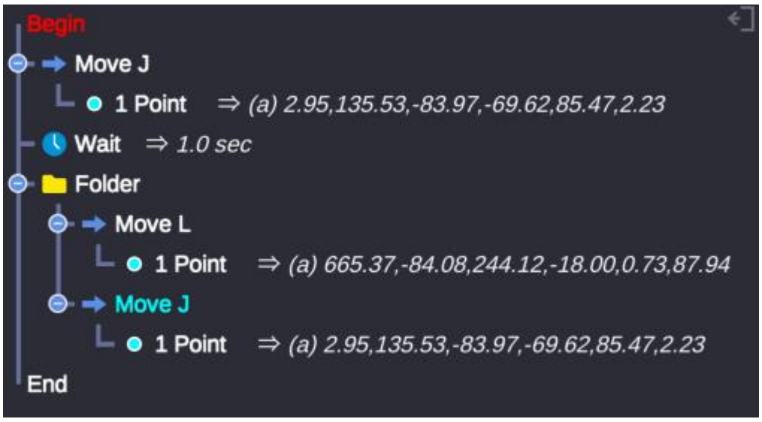

■ Copy/ Paste 

Step1) Select the item to copy. The selected command will be shown in blue. In the below example, the MoveJ line is selected.



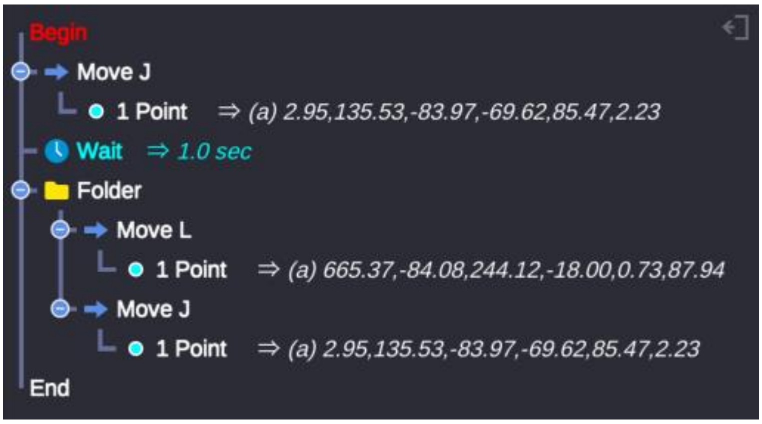

Step2) Press the Copy button.

Step3) Click desired location and click the Paste button. In the example, the MoveJ command is pasted under the Folder.

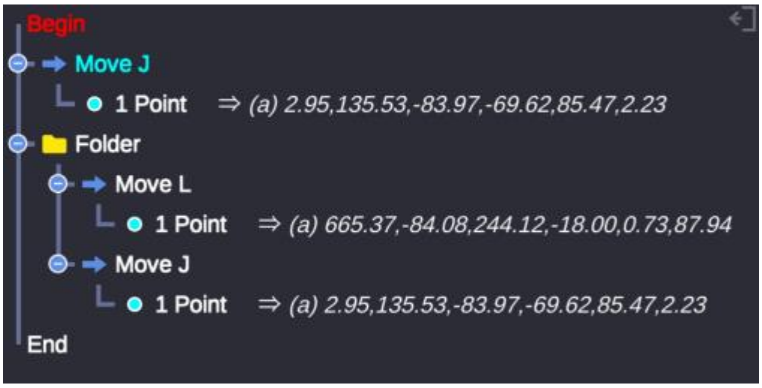


■ Delete 

Step1) Select the command to delete. The selected command will be shown in blue. In this example, the Wait command is selected.

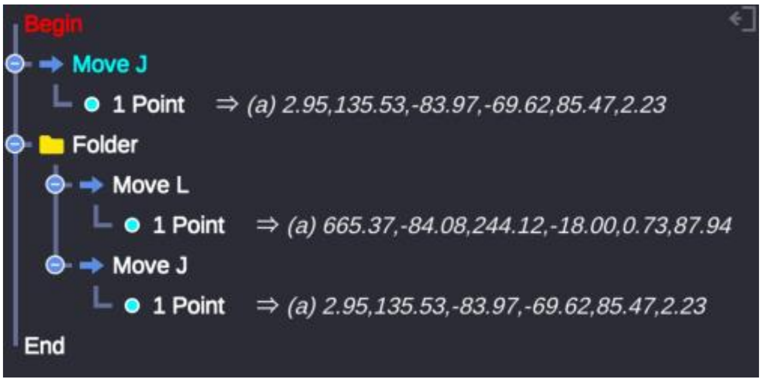



Step2) Click the Del button. The command has been removed as shown below.

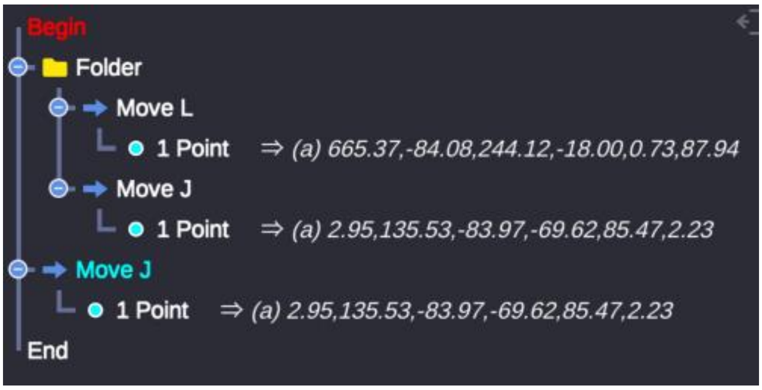



■ Move 


Step1) Select the command to move. The selected command is shown in blue. In this example, MoveJ at the top is selected.



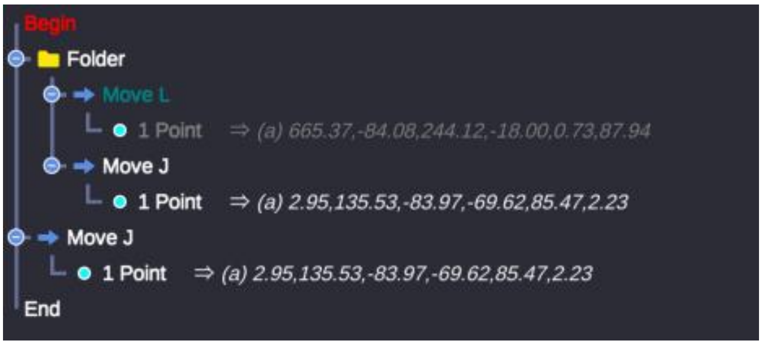
Step2) Click the Down button to move MoveJ down as shown below.




■ Pass 


Step1) Select a function to temporarily hold / block its execution. The selected command is shown in blue. In this example, the MoveL command is selected.

Step2) Click the Pass icon. The command turns dark as shown below and will not execute. To undo, simply select the command again and press the Pass button again.

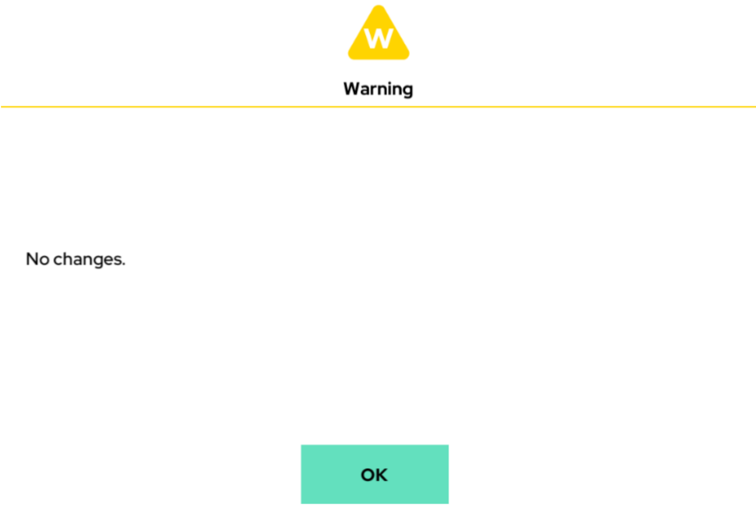


## 7.6 Program Management

Allows the user to save, load, or create a project

- Save Project

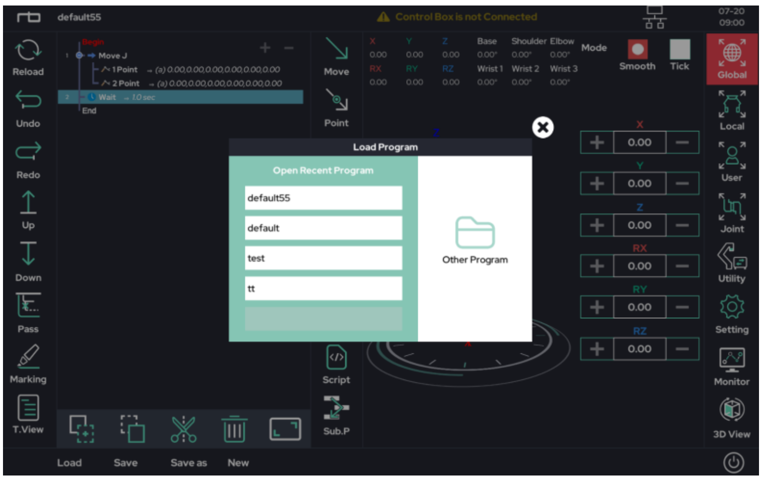
  To save the current project, click the save Save icon on the bottom left side of the UI work screen. If there is no change from the existing saved contents, it is shown as below.



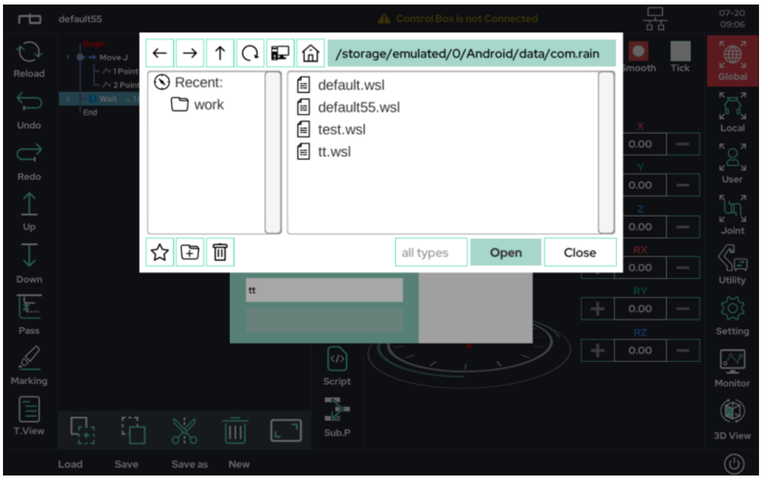

- Load File

  To load a saved project, click the FILE button at the bottom of the UI to display a list (shown below). If a user selects a file from the list, it will be loaded in as the current program. If there are unsaved changes to the current project, a prompt will request the user to save.

  Note: Only recently used files will appear in the list.

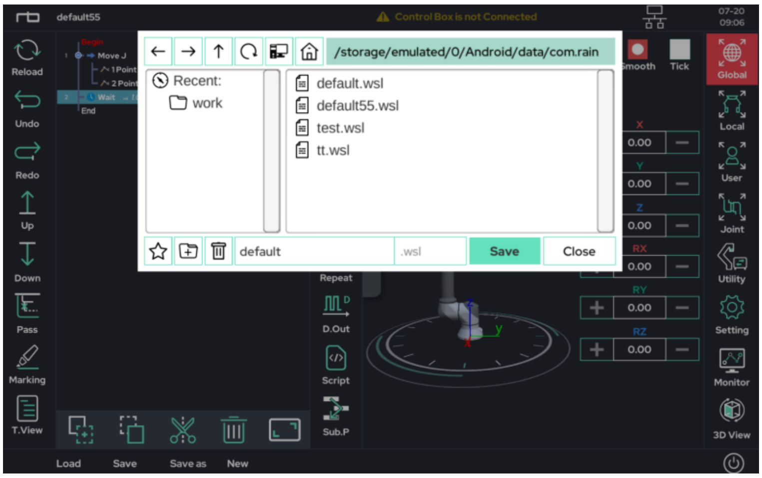


To open a file, users can click the Load option. Clicking Other Program button will open the File Explorer, which allows the user to look through saved files.




- Save As

To save a program with a different name, click the Save As option in the FILE list. The following popup window will appear. Using this window, users can save their current file with the desired program name. The program name cannot bet set to "default," as it is already in use by the system.



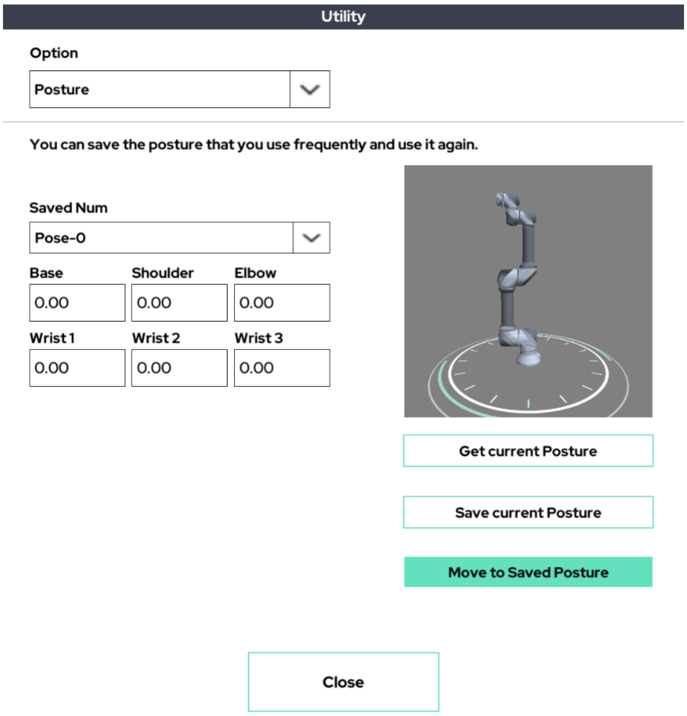
## 7.7 Operation Utilities

On the right side of the Make screen, there are other utility functions to help a user operate the system.


- Utility: A collection of additional functions, such as the posture saving function, the system input / output information view function, and the system output test function. These functions are also frequently used.
- Setting: Allows the user to use the jog function, as well as other utility functions to help the user's experience.
- Monitor: Provides a window that allows the user to monitor both system and user variables in real time.
- UI Mode: UI mode can be selected according to the user's level and the user's purpose.

■ Utility sub-functions

[Utility-Posture]

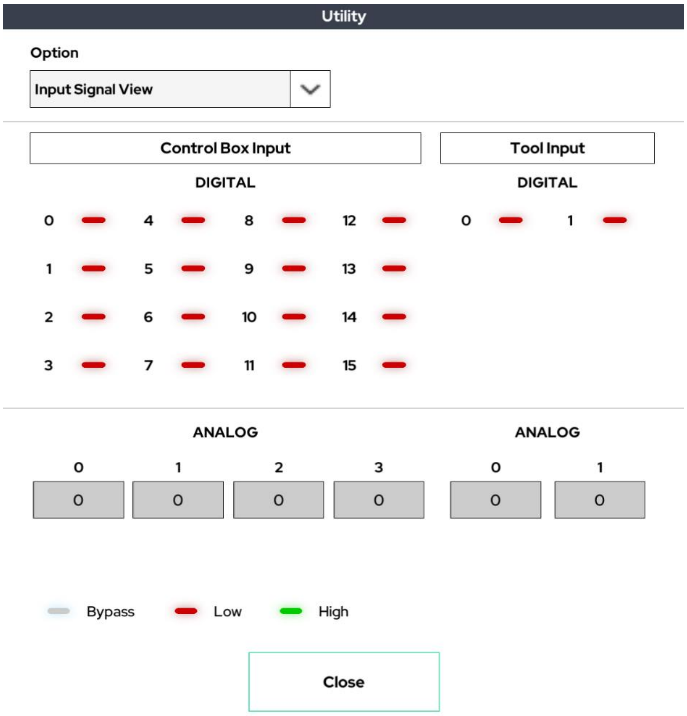



Up to 20 frequently used postures can be saved and used on the UI tablet.

Press the Get button to get the current position information and press the Set button to save it.

Hold down the Move button to move to the saved position.

[Utility-Input Signal View]

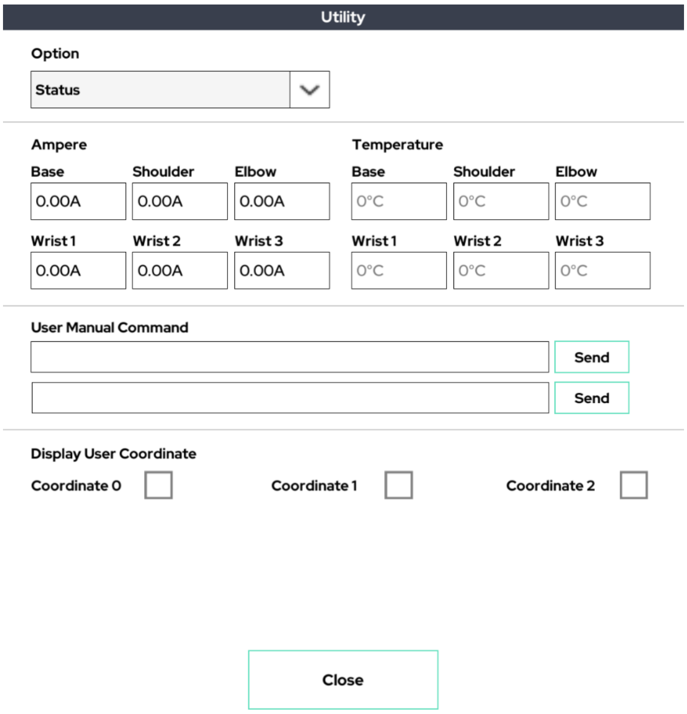



Input signal monitoring window for control box and tool flange.

[Utility-Output Signal View]

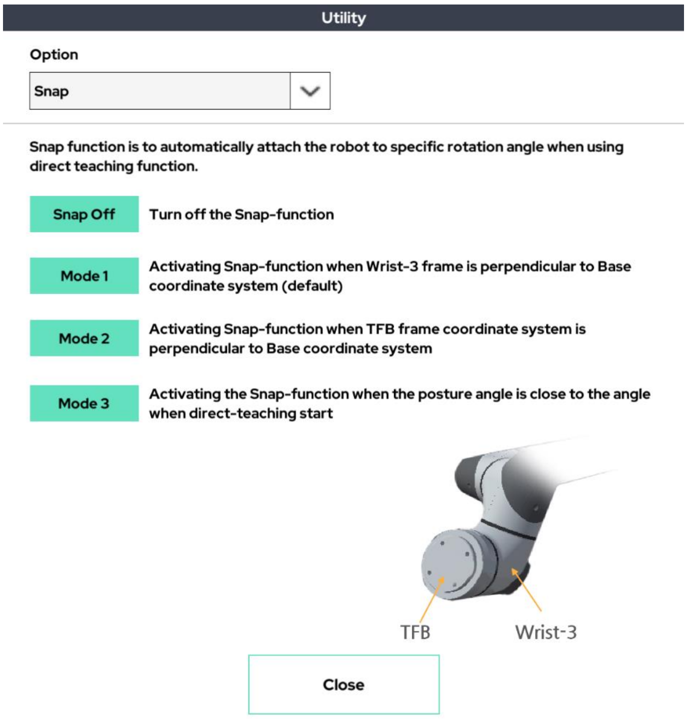

Output signal monitoring window for control box and tool flange.

[Utility-Status]



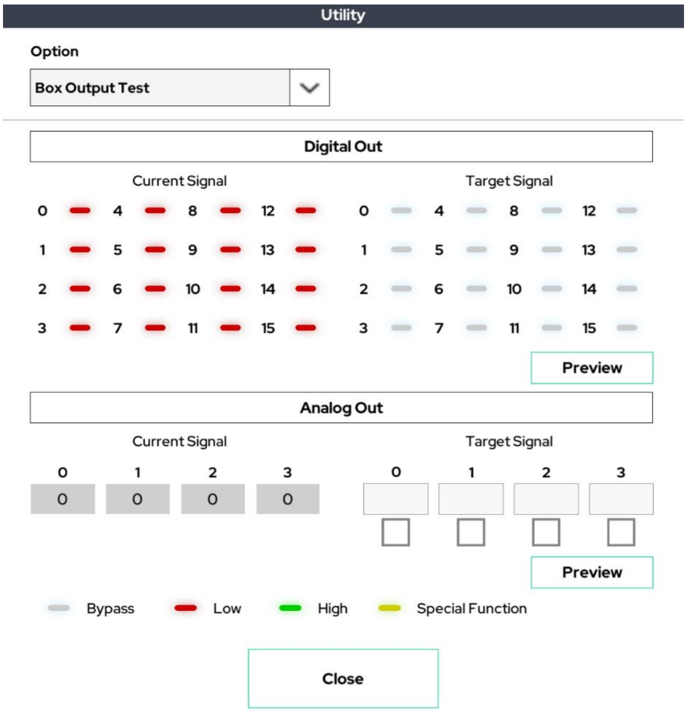

This window allows the user to see the robot arm's current and temperature. It also shows the user coordinate system settings.

[Utility-Snap]



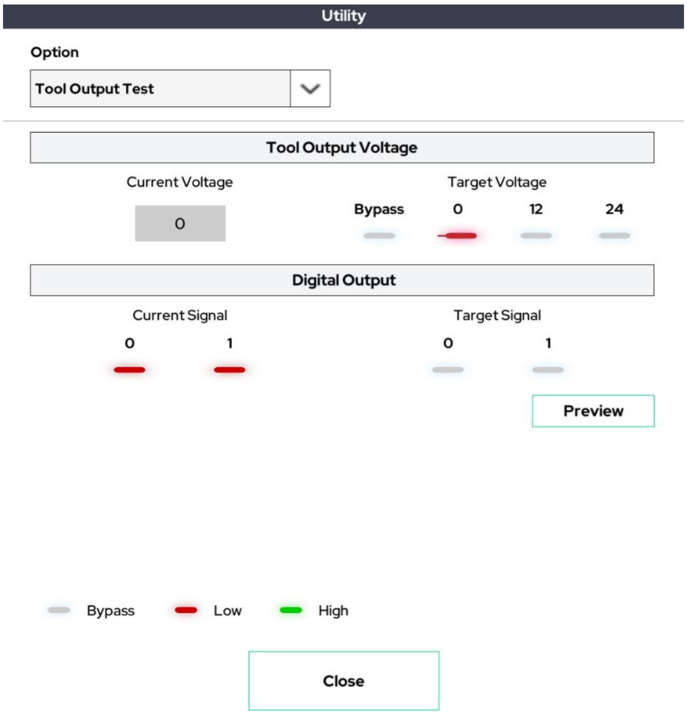

Snap mode selection window to be applied when using direct teaching mode.

[Utility-Box Output Test]



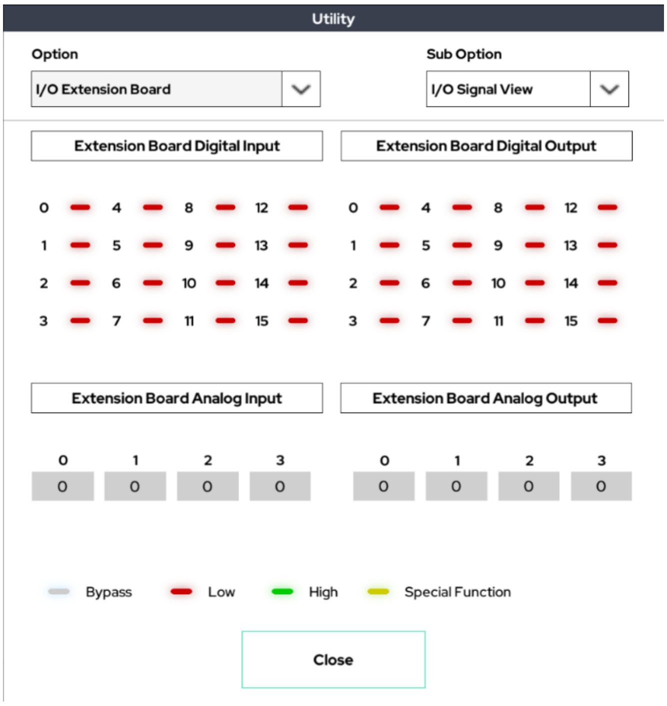

This window allows you to test the output of the control box.

[Utility-Tool Output Test]

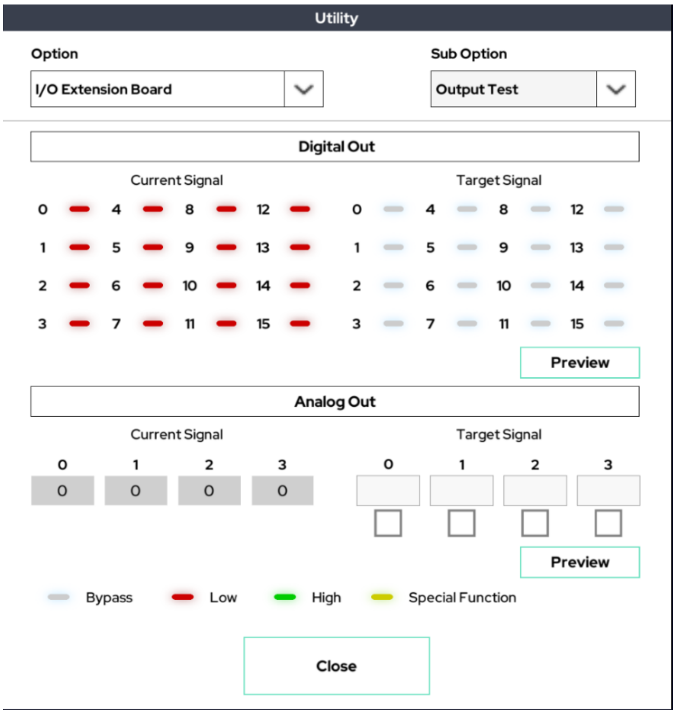



This window allows you to test the output of the tool flange.

[Utility-I/O Extension Board]



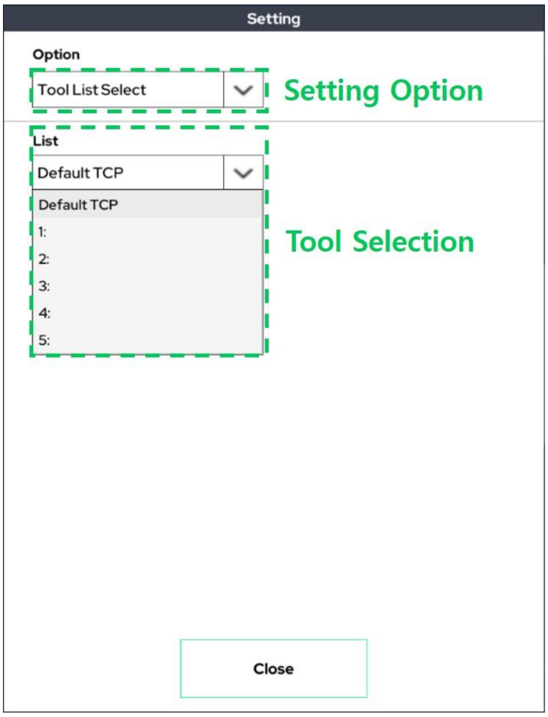

I/O expansion module's I/O signal monitoring window.




Window for testing the output of the I/O expansion module.

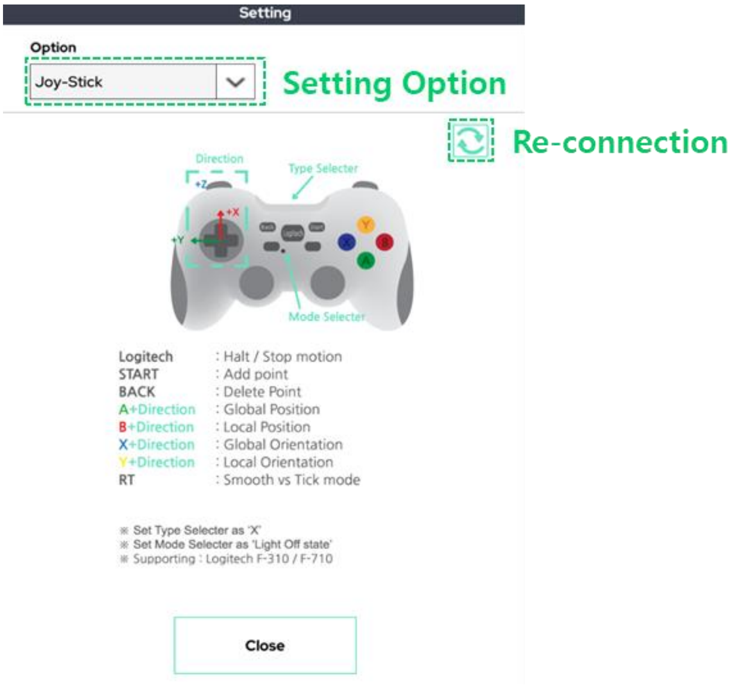
■ Setting sub functions

[Setting-Tool List Select]



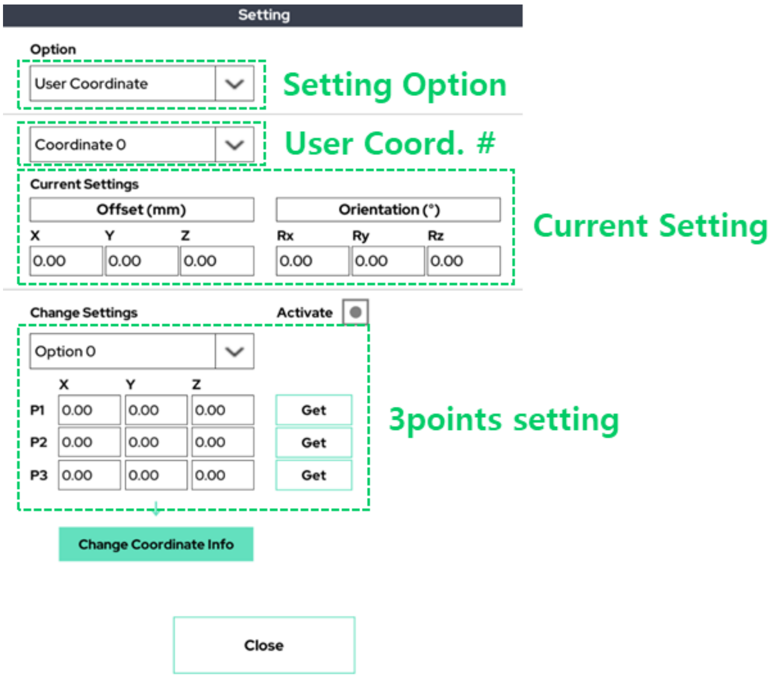

There is a Tool List Select setup feature that sets up TCP to use in a pre-saved TCP list.

[Setting-Joystick]



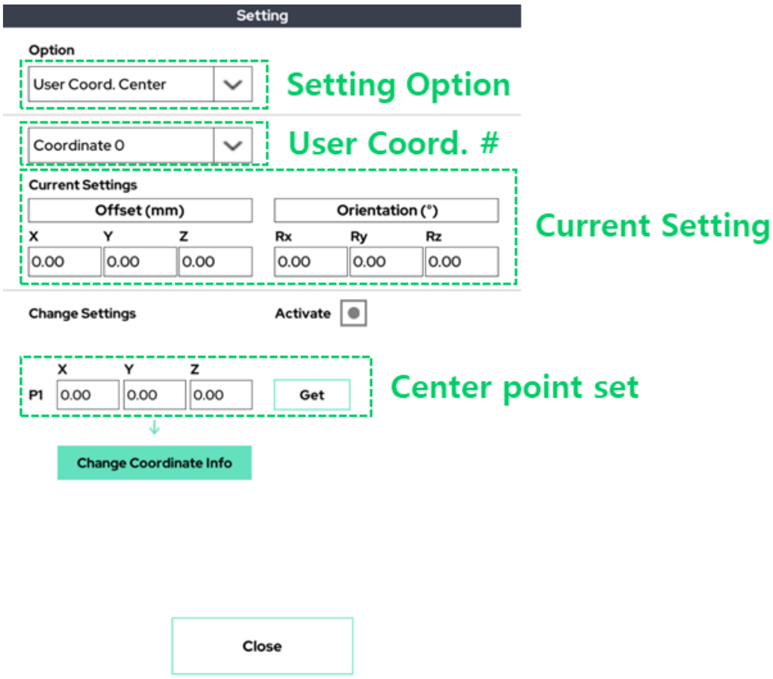

The Joystick setting allows the user to control the robot using a joystick connection.

[Setting-User Coordinate]




Allows the user to set their own coordinate system using the 3-point setting mode. For more explanation, see the Coordinate page on the Setup Screen.

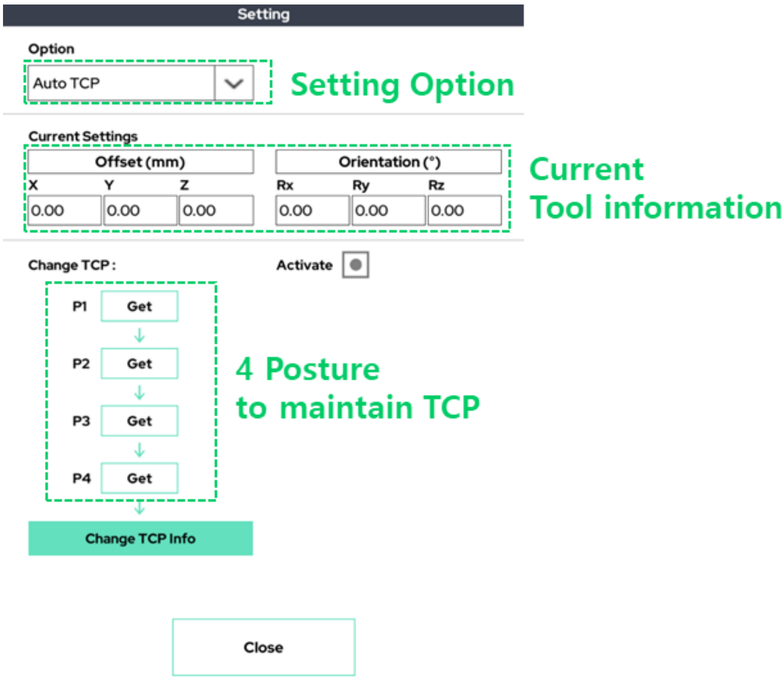
[Setting-User Coordinate Center]




- Offset (mm) - Change the X, Y, and Z of the user-defined coordinate system by providing an offset for the robot. Maintains the rotation of the coordinate system.

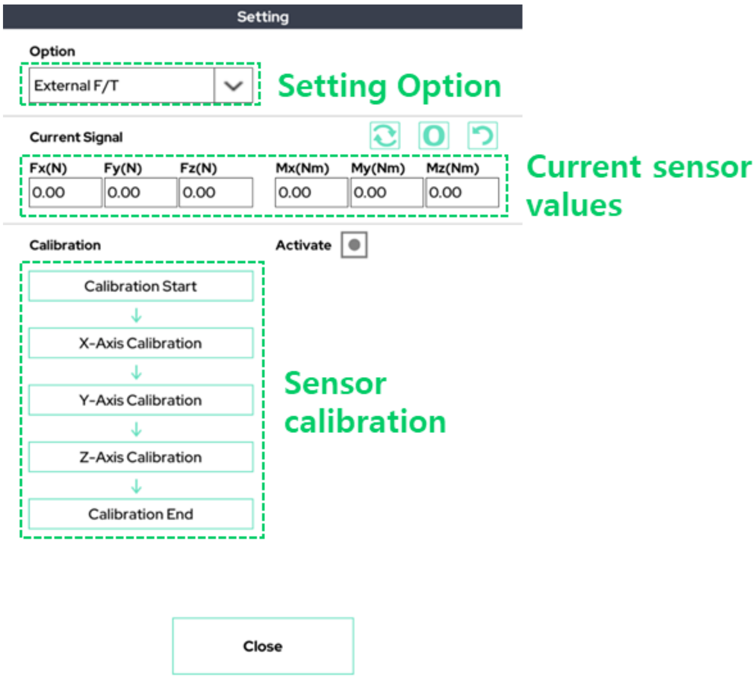
- Orientation (°) – Change the orientation of the user-defined coordinate system. Maintains the X, Y, and Z of the coordinate system.

[Setting-Auto TCP]



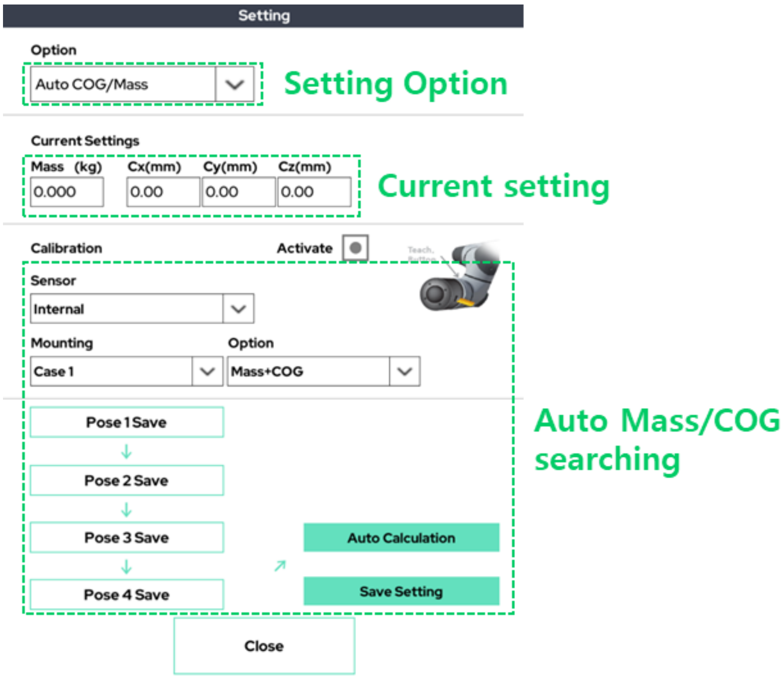

This function allows the user to find the position of the TCP automatically.

[Setting-External F/T]



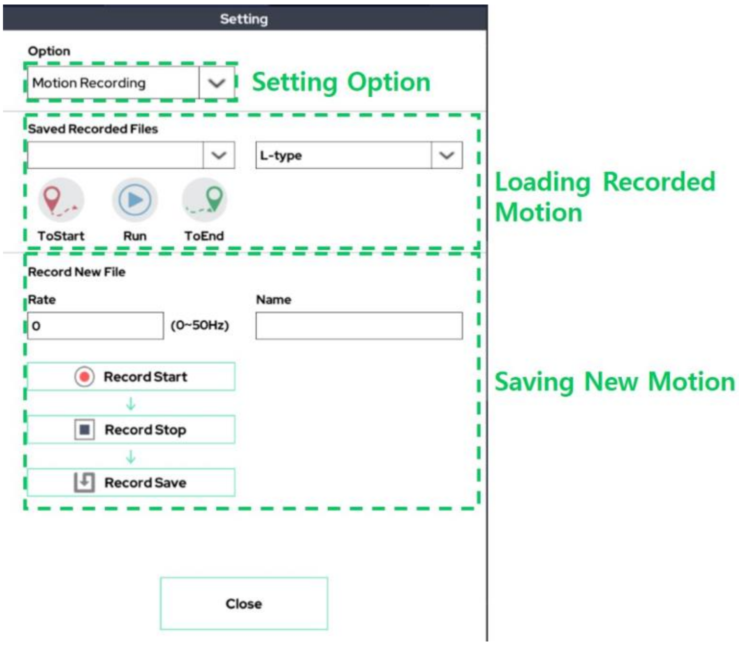

This window allows you to check and calibrate the external F/T sensor (e.g. Robotiq F/T sensor).

[Setting-Auto COG / Mass]



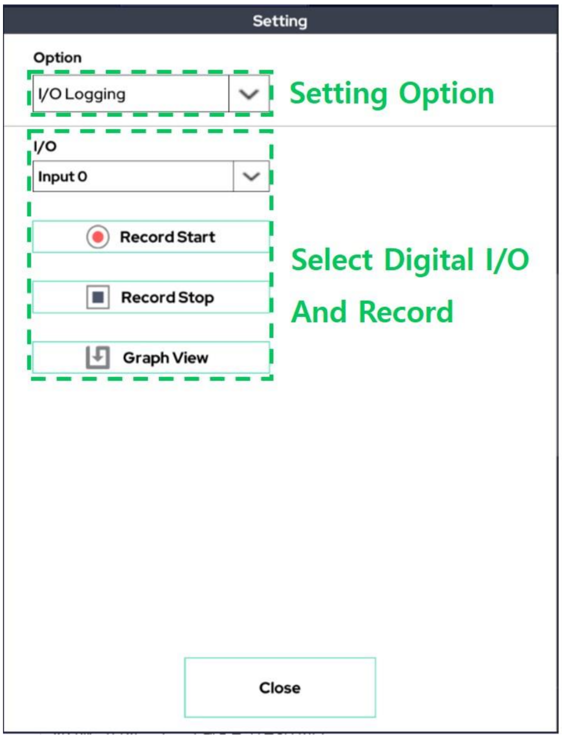

This function finds the weight and center of gravity attached to the tool using the internal / external F/T sensor.

[Setting-Motion Recording]



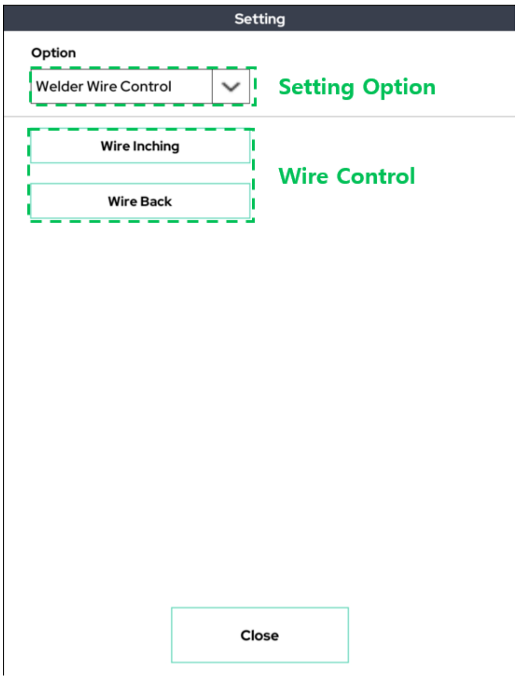

It is a function that records the motion through the direct-teaching(gravity compensation) function. The recorded action is available in the program via the Replay function.

[Setting-I/O Logging]



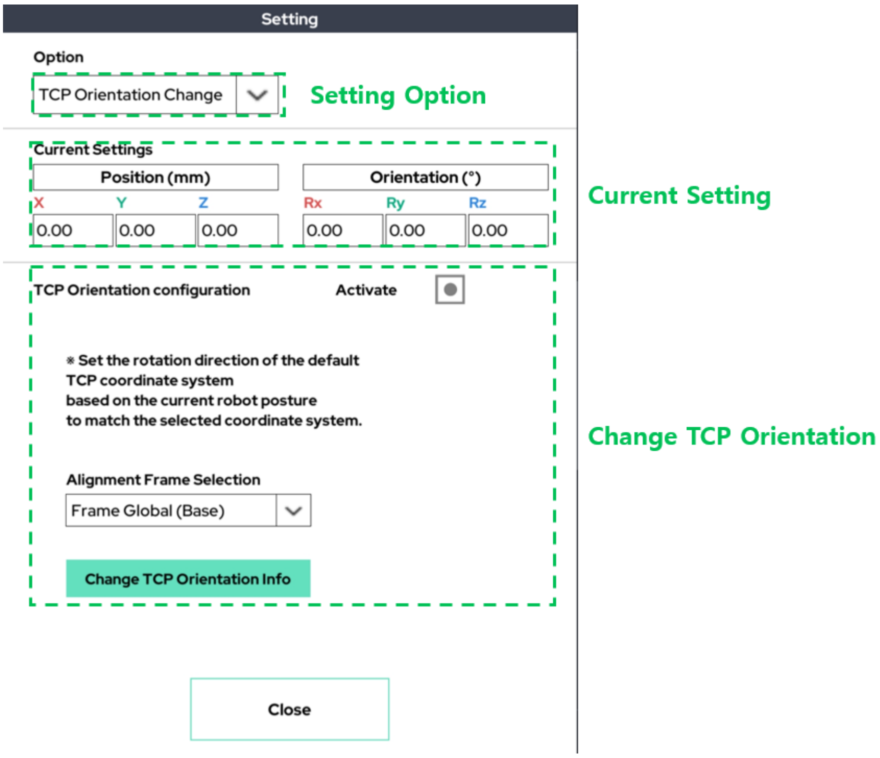

This function sets one digital input/output, records the change in the value of that input/output, and shows it graphically.

[Setting-Welder Wire Control]



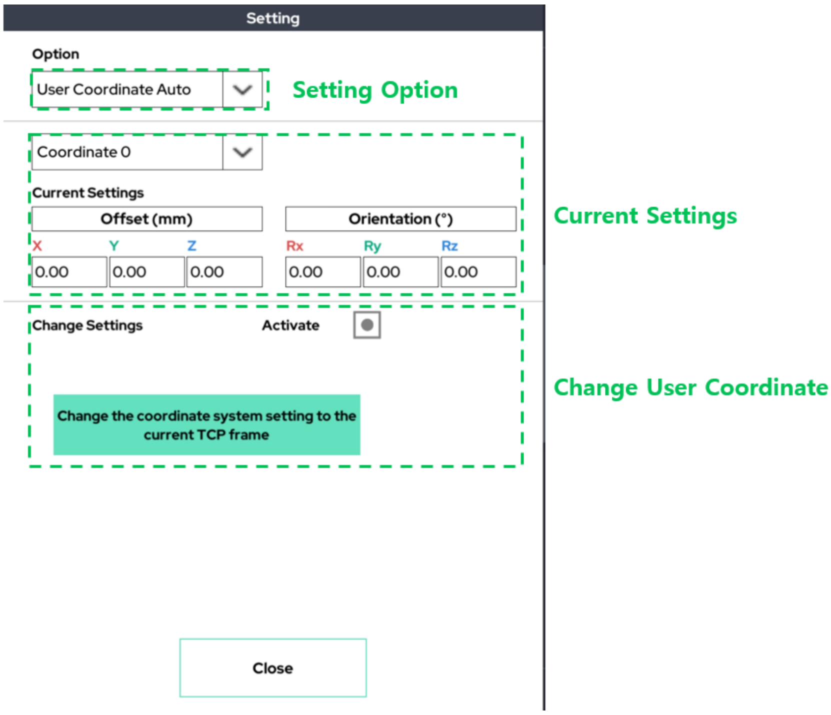

This function can control the welding machine's wire.

[Setting-TCP Orientation Change]




Sets the rotation direction of the default TCP coordinate system based on the current robot pose to match the selected coordinate system.

[Setting-User Coordinate Auto]

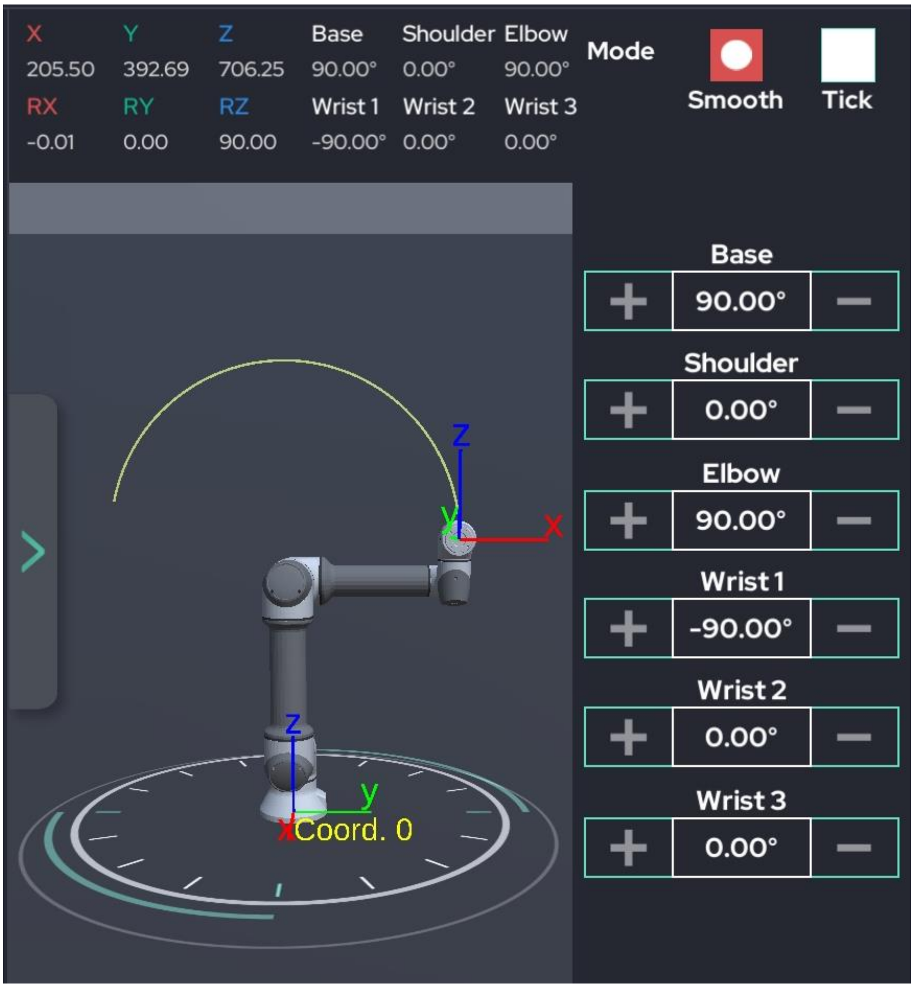



Change the coordinate system setting to the current TCP frame.

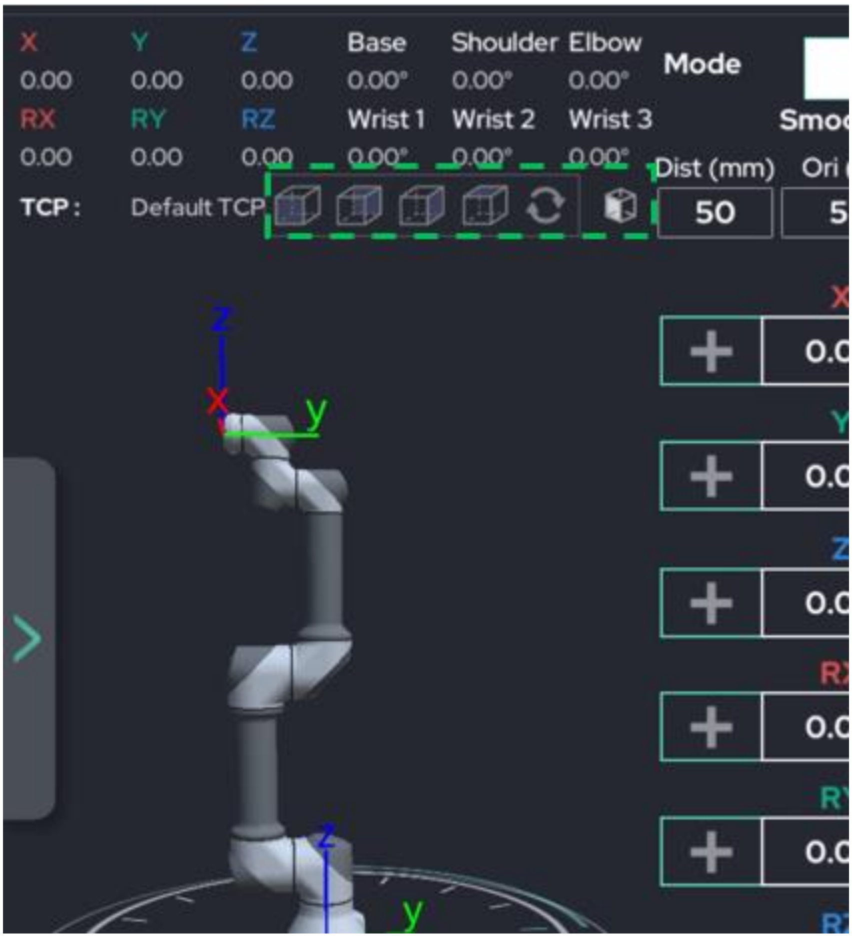
■ Monitor Function

This function is used in conjunction with the Monitor command in Section 6. This window allows the user to observe the system and user variables in real time.

By pressing the recording function on the upper right, the TCP trace of the robot tool is recorded in the 3D viewer in the 3D viewer. (Yellow solid line)



■ 3D View Function



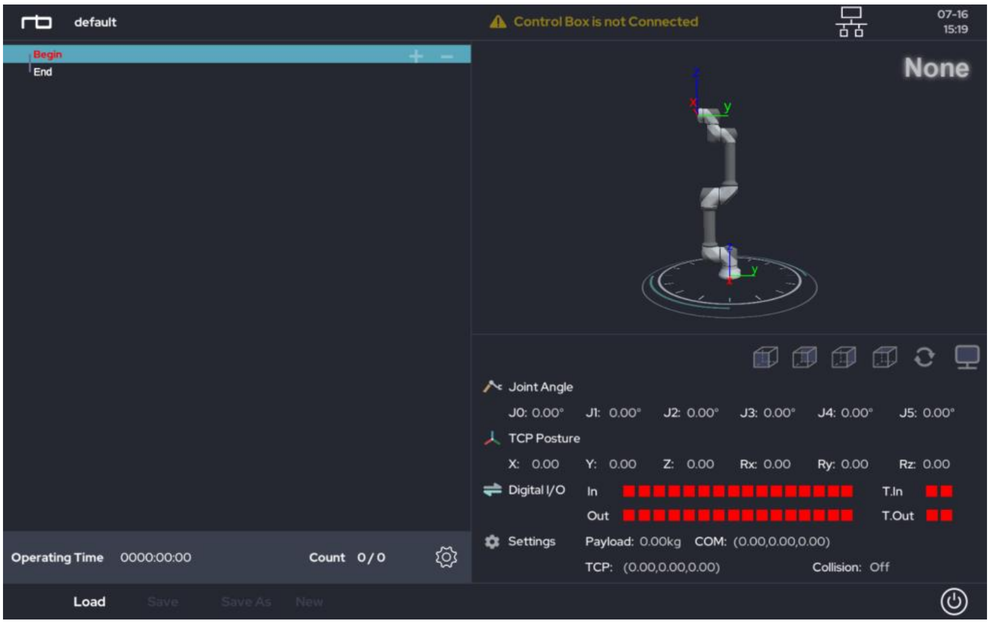

Allows users to change the perspective of the 3D viewer.

# Chapter 8. Robot Operation

## 8.1 Robot Operation

The ‘Play’ screen allows the user to use program files to move the robot in a continuous loop.

- ‘Play’ screen is shown below.




- Before using, please check the connection between tablet PC and control box. Check the  Icon to view the connection with the robot. Please refer to Chapter 6.2 for connection.
- Open the desired project. Please refer to Chapter 6.6 for more detail about how to open a project.
- Press the play (▷) button located at the bottom of the screen to run the robot.
- A dialog is pops up when the current robot position is different from the initial position specified. Press and hold the ‘approach’ button to move the robot to the initial position.
- In ‘Play’, the program loaded will repeat indefinitely if the ‘number of repeat’ is not specified. Press ‘Count’ at the top of the screen to set the ‘number of repeat’.
- The motion speed of the robot can be adjusted while the robot is in operation.

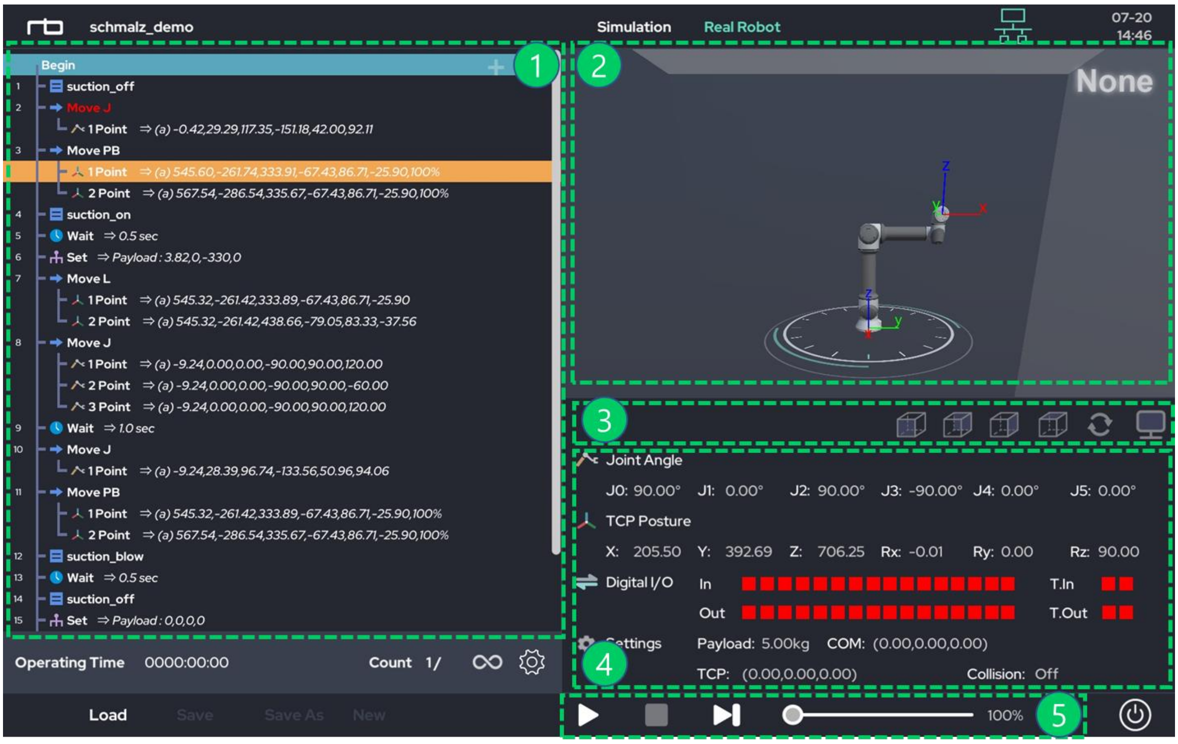
## Warning

Warning:

1) The risk assessment of the robot must be done, and all safety requirements must be satisfied before the robot operation.
2) The initialization of the robot may fail when the robot is not properly installed, the payload is not set accurately, or an issue occurs in the initialization process.
3) In ‘Play’, the robot physically moves immediately when the ‘Play’ button is clicked. Please read carefully all sections related to the robot operation.
4) To move to the ‘Make’ or ‘Setup’ screen, the program running must be terminated.
5) The USB cable between Tablet PC and control box can be unplugged during the robot operation.

## 8.2 Robot Status Check

The robot's current status is shown in the 'Play' screen during operation.

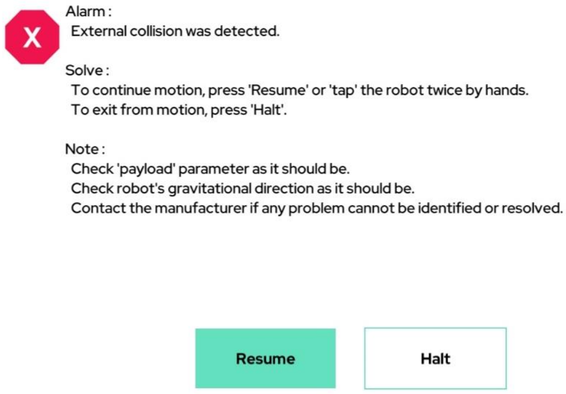



| Description | |
|---|---|
| ① | Program flow tree |
| ② | 3D viewer |
| ③ | 3D view angle changer |
| ④ | System information, system variable monitor |
| ⑤ | Play / Pause / Stop / Velocity slide bar |

## 8.3 Troubleshooting while Operating

Various problems can occur while the robot is in operation. Below are some of those problems and ways to troubleshoot.

1. External Collision



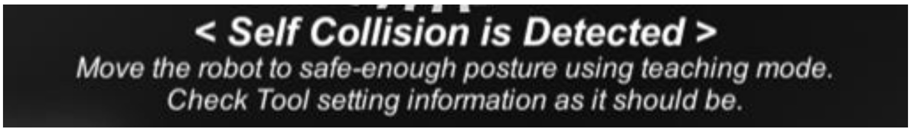

The robot will stop immediately when a collision is detected.

In order to resolve, please remove the object that collided with the robot. Press 'Resume' to resume the current program or 'Halt' to terminate.

TOK TOK (Tap to Resume)

Tap the robot twice to resume the previous task.

2. Self Collision



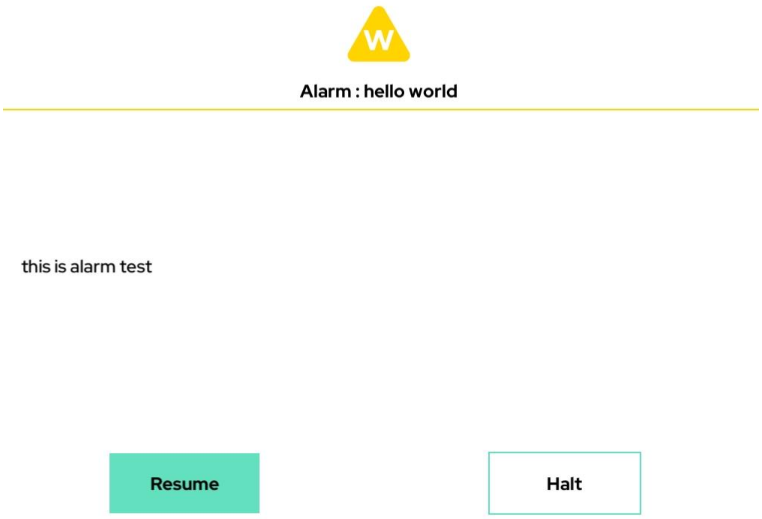

The robot automatically stops when it approaches a configuration where it will run into itself.

To recover from this situation, press the 'Teaching Button' at the tool flange and manually change the current pose of the robot. Then, please edit the command in the current program that caused the self-collision.

When the situation occurs in 'Simulation' mode on the 'Make' screen, any of the following instructions will recover the robot.

- Use any button on the 'Make' screen related to robot motion.
- Change 'Simulation' mode to 'Real' mode to get the current joint data of the robot.
- Use the 'Teaching Button' to get the current joint data of the robot.

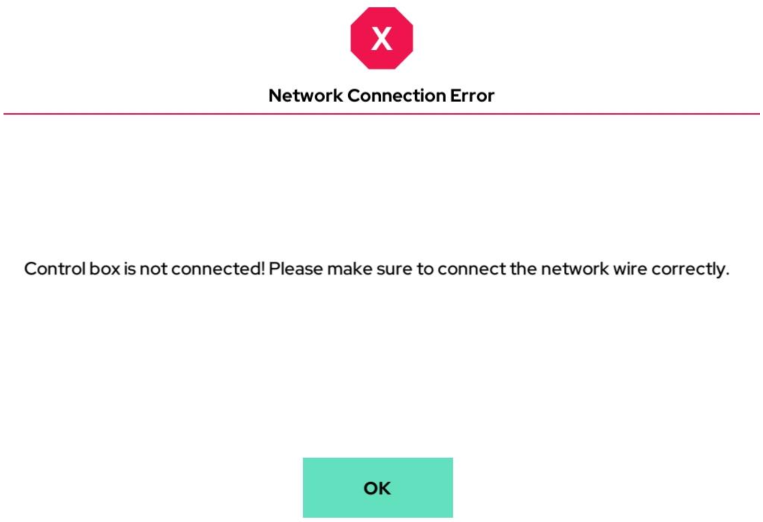
3. Alarm Message




When an 'Alarm' is set in the current program, the robot will pause once the 'Alarm' command is reached. A dialog will then pop up.

Press 'Resume' to continue the task or 'Halt' to stop.

## 4. Teaching Pendant (Tablet) Disconnection

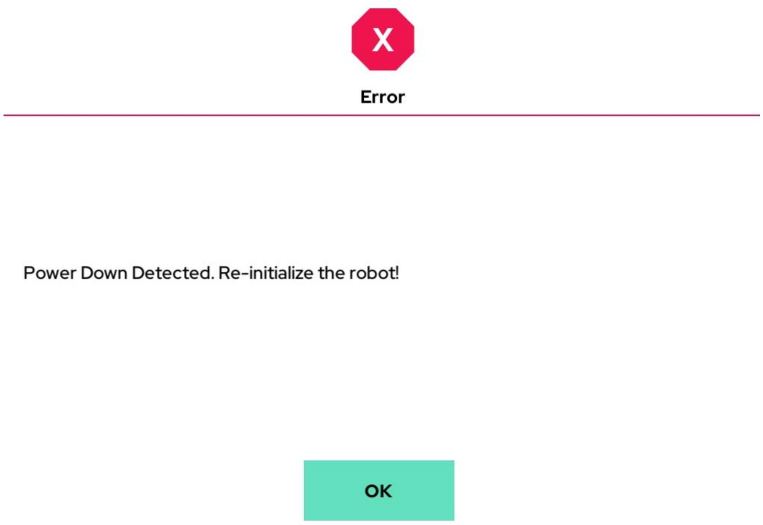



This message will occur when the Tablet PC is physically disconnected from the control box.

To recover, please plug the Tablet PC's USB cable to the control box.

If the Tablet PC's USB cable is plugged into both the Tablet PC and the control box, it may be damaged. Please replace the cable with new one.

## 5. Power Down of Robot ARM




This error will appear when the robot is not receiving enough power. It may appear when the Emergency Stop button is pressed. If the button is not pressed, however, the AC or DC power line may be damaged.

The robot should be rebooted and re-initialized to resolve this issue.

## 6. Joint Controller Errors

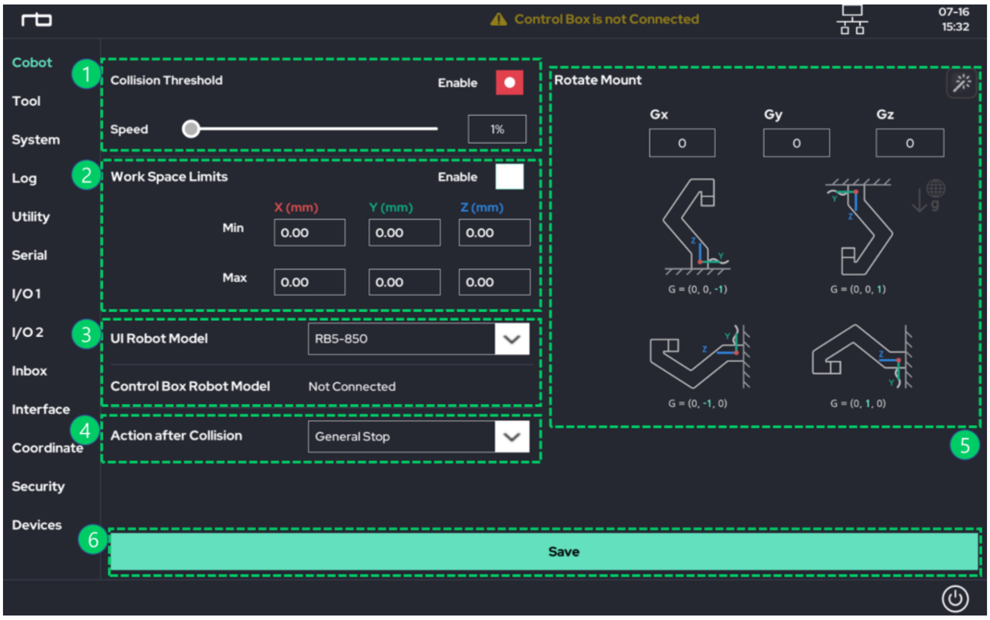
The robot will stop automatically when one of its joint controllers experiences one of the following errors:

| | |
|---|---|
| Big Error | The difference between the reference input and encoder value exceeds the factory-specified threshold. |
| Jam Error | The encoder value does not change, but a current is supplied that is over the factory-specified threshold. |
| Overcurrent Error | The current exceeds the maximum current threshold. |
| Temperature Error | The temperature exceeds the maximum temperature threshold. |
| Mode Error | The version of software in the main controller is different from the version in the joint controller. |

# CHAPTER 9. SETUP

## 9.1 SET-UP(COBOT)

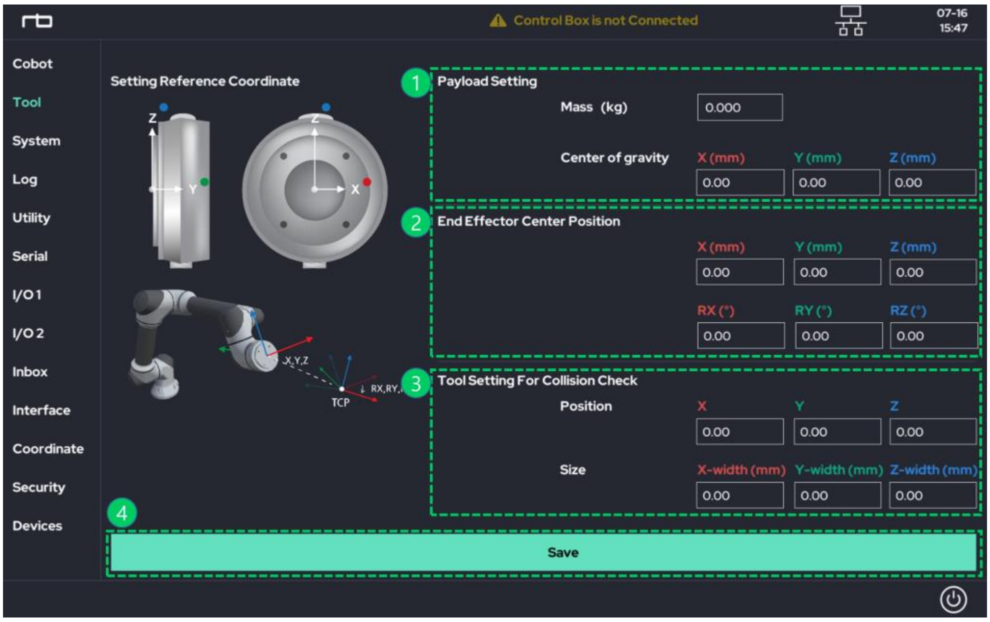
Sets the default settings for the robot arm.




| | |
|---|---|
| ① | Collision sensitivity. When it is enabled, the collision sensitivity can be adjusted. The robot stops with a smaller impact if the collision sensitivity is lower. |
| ② | Workspace boundary. The robot will stop when it crosses the boundary of workspace. |
| ③ | Displays the UI robot model currently in operation and the robot model of the connected control box. |
| ④ | Select the stop mode after the collision detection. |
| ⑤ | Robot’s installation angle. If there is a corresponding installation angle among the examples shown, click on the picture. To input directly, enter the direction vector of gravity based on the global coordinate system |
| ⑥ | Save current settings. |

# 9.2 SET-UP(TOOL)

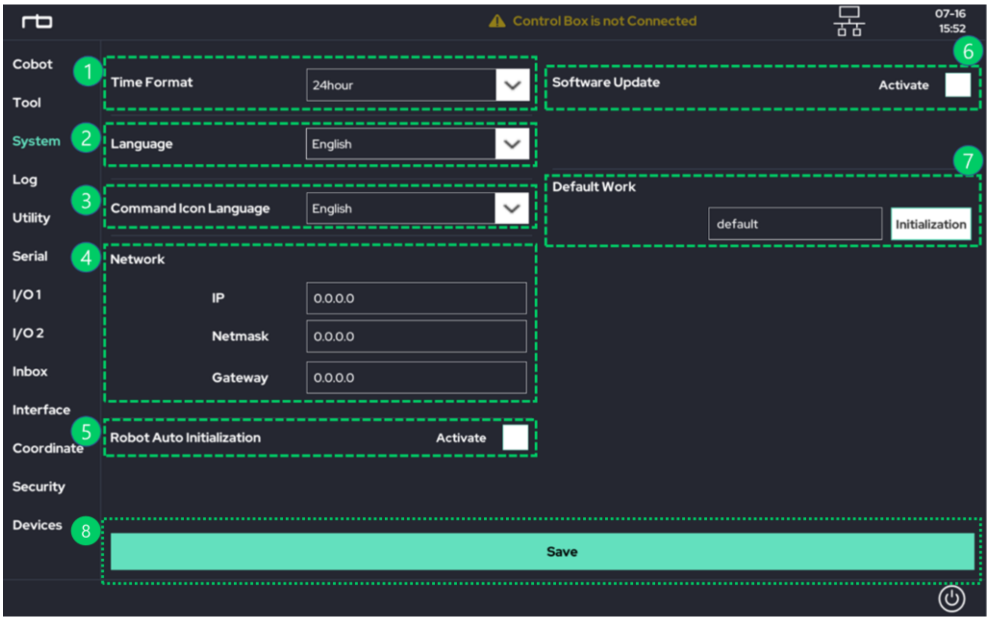
Sets the installed tools.




| | |
|---|---|
| ① | To prevent self-collision while moving, a virtual box-type boundary at the tool flange can be created. Define a size of the box and location of the center in respect to the tool's coordinate system. |
| ② | Payload setup. Mass in kg up to * kg (depending on robot model) and center of gravity in mm should be defined. |
| ③ | Define the relative TCP position and orientation in respect to the coordinate system of the tool flange. |
| ④ | Save current settings |

## 9.3 SET-UP(SYSTEM)

Set the display unit, date and time, UI password, system update, and more.




| | |
|---|---|
| ① | Date and time format |
| ② | Log-in password setup. |
| ③ | Network address setup. This address is used to communicate to other devices. |
| ④ | Language setup |
| ⑤ | Auto initialization setup. When this function is enabled, the robot initializes itself to be ready to move in 'Real-Mode'. To use this feature, the 'Auto-Initialization Key' in the digital I/O needs to be defined. When 'Auto-Initialization Key' is activated, the robot will initialize itself. |
| ⑥ | When enabled, will update software. Please refer to appendix (software update) for more details. |
| ⑦ | This Function can be used with 'Auto-Initialization'. After the robot initializes itself, the program specified in this Function will run automatically. |
| ⑧ | Save current settings |

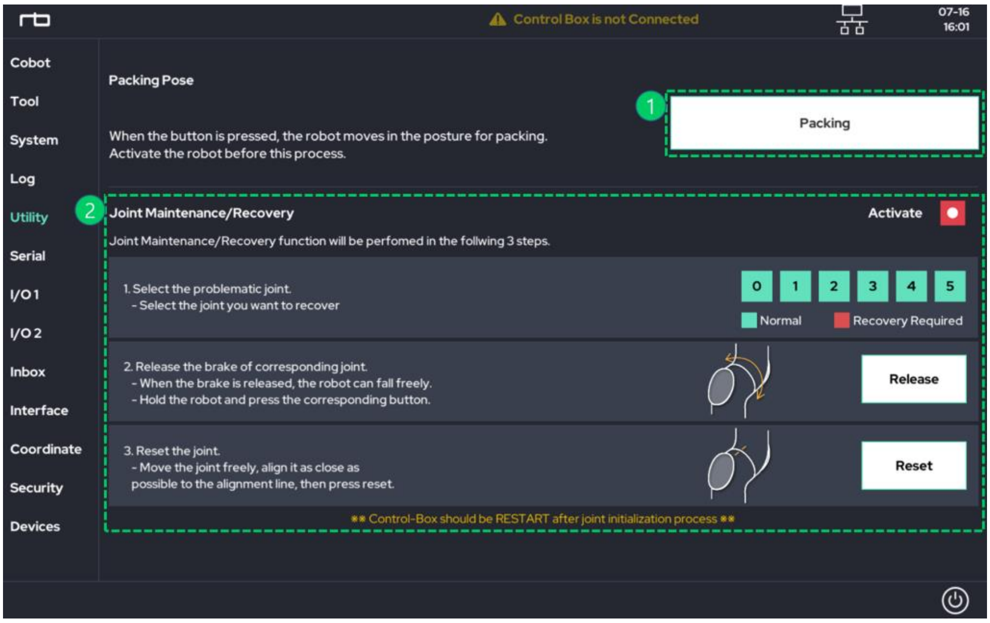
# 9.4 SET-UP(LOG)

Check the system log status of the robot arm.


| | |
|---|---|
| ① | Copy a log file from the control box to the Tablet PC. Depending on the size of file, it may take few seconds. |
| ② | Opens the log file copied. |
| ③ | Displays an internal log based on the following characters:<br>I : Information<br>W : Warning<br>E : Error<br>S : System<br>F : Fatal |
| ④ | This function checks the status of the robot arm. |
| ⑤ | Back up program files / log files / setup files stored in the Control Box to your tablet PC.<br>The copied (back-up) files are stored in a specific path on your tablet PC. |

# 9.5 Set-up(Utility)

Provides functionality for packaging and emergency recovery of robots.




| | |
|---|---|
| ① | A button pre-defined for packing pose. When a user presses and holds this button, the robot moves to the packing pose. This is the pose that the robot is originally shipped in. |
| ② | This function is used to reset a joint encoder back to its initial value.<br>This function is intended to recover the robot from abnormal operation and should be used with caution.<br>Step1. Select a joint to be reset.<br>Step2. Press the 'Release' button to have the joint move freely.<br>Step3. Align the marks at the joint. Press 'Reset' to re-initialize the joint. |

**Warning**

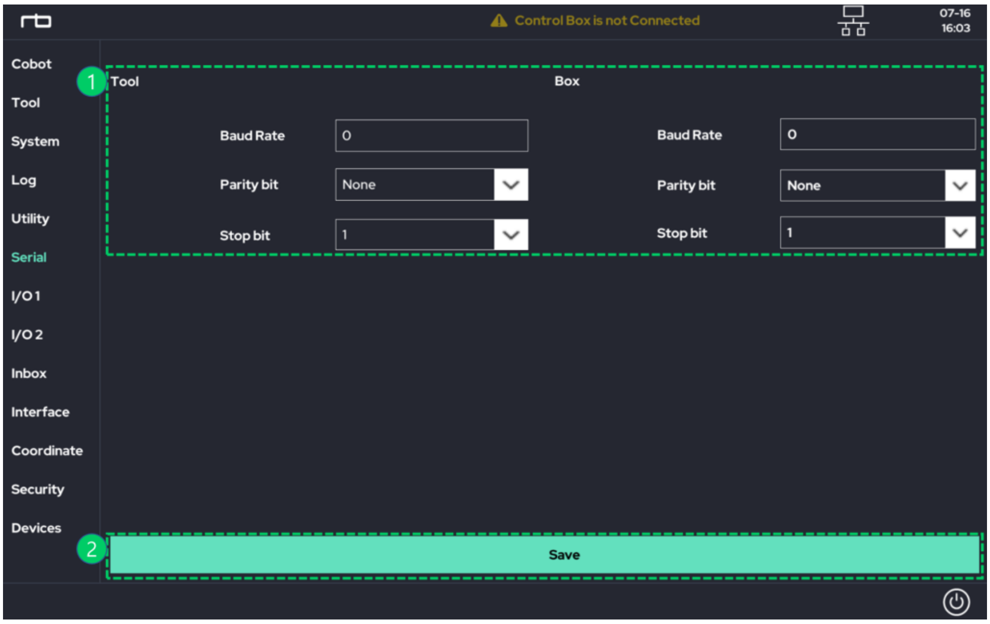
Warning:

1) Before using the 'Emergency Joint Recovery', please fully understand all related usages of the robot.

2) If shipping the robot, it should be packed within its original box.

# 9.6 SET-UP(SERIAL)

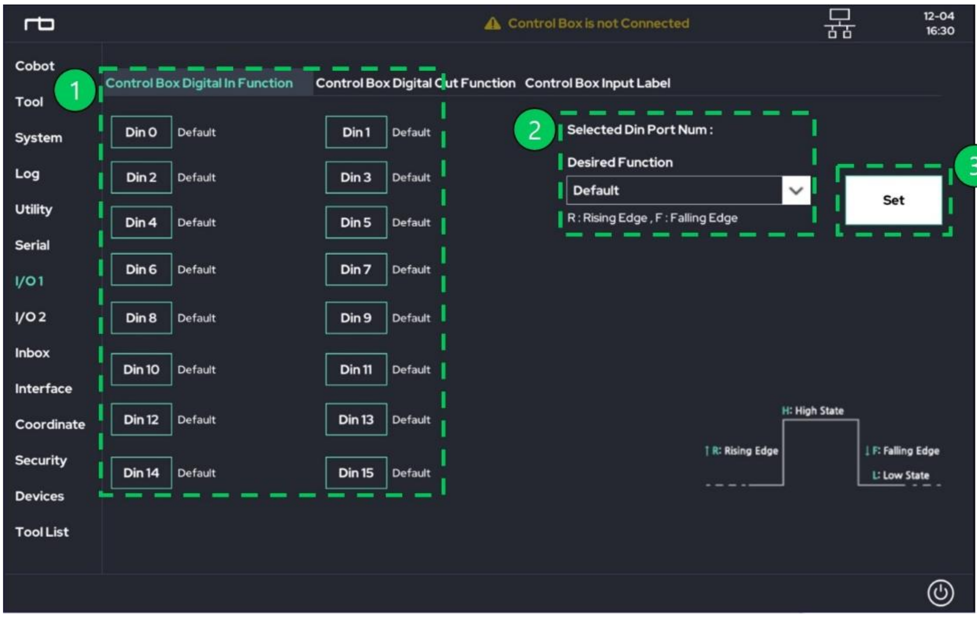
Sets serial communication between the robot tool and the control box.



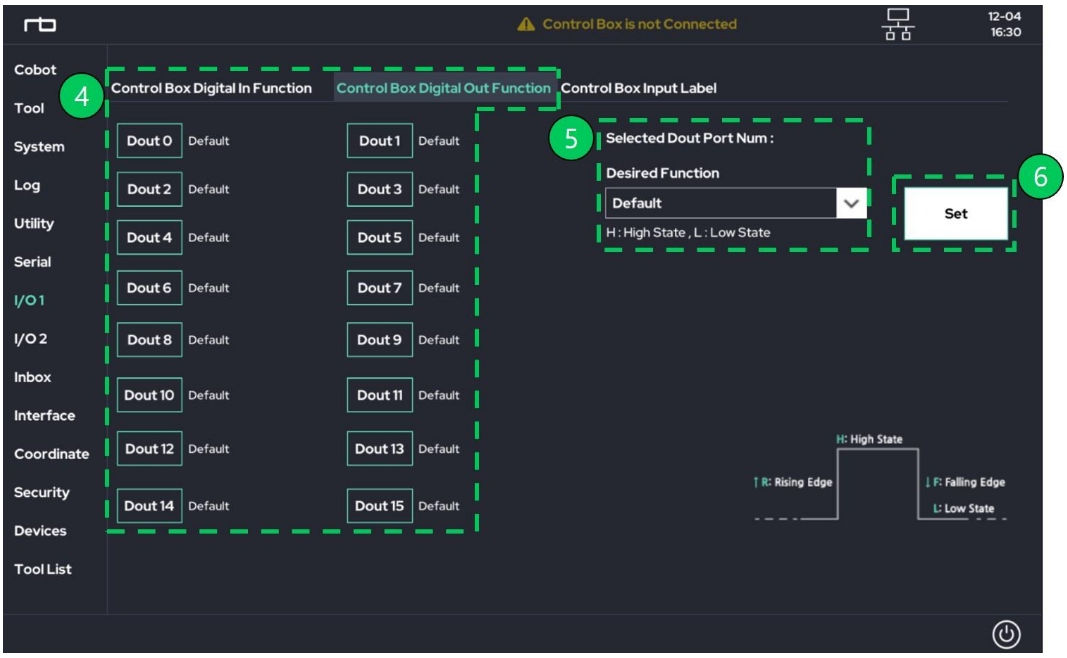

| | |
|---|---|
| ① | Settings for Serial (RS485) communication(Baud Rate, Stop bit, Parity bit). |
| ② | Save current settings |

# 9.7 SET-UP(I/O 1)

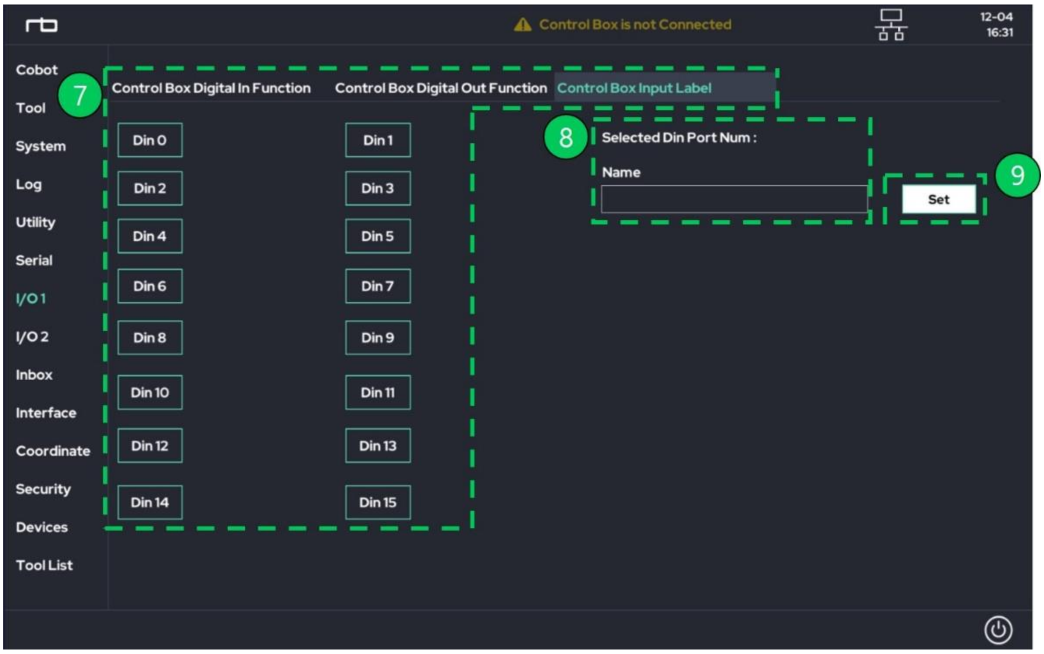
Sets the function of the GPIO port on the control box.



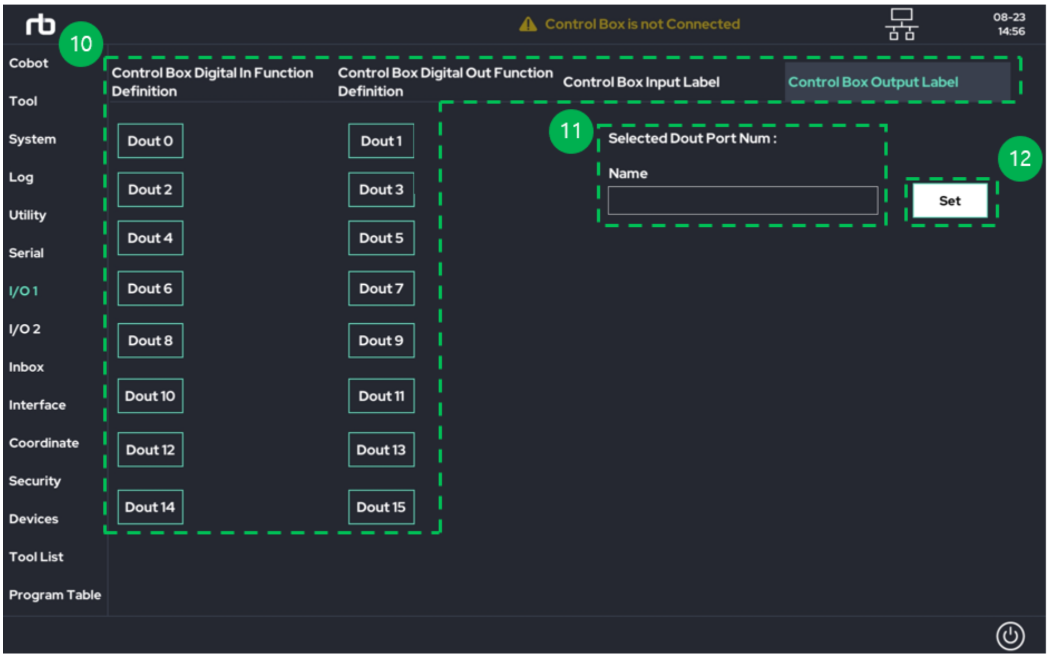

| | |
|---|---|
| ① | Select an input port to be changed. |
| ② | Specify the type of the input port. |
| ③ | Save changes. |




| | |
|---|---|
| ④ | Select an output port to be changed. |
| ⑤ | Specify the type of the output port. |
| ⑥ | Save changes. |




| | |
|---|---|
| ⑦ | Select the input port to be named. |
| ⑧ | Type the name you want to use. |
| ⑨ | Save as selected name. |




| | |
|---|---|
| ⑩ | Select the output port to be named. |
| ⑪ | Type the name you want to use. |
| ⑫ | Save as selected name. |

■ Description of all types available for the input port.

The input ports from Din0 to Din 15 in the control box can be set up as one of the following types. (R = Rising Edge, F = Falling Edge, H = High State).

0. Default (GPIO)
1. Run Program Once (Rising Edge)
2. Stop/Halt Program (Rising Edge)
3. Pause Program (Rising Edge)
4. R = On direct-teaching / F = Off direct-teaching
5. R = speed 100% / F = speed 0%
6. R = Convert to REAL mode / F = Convert to SIMULATION mode
7. R = Robot arm initialization (activate servo drive)
8. H = Collision detection off
9. H = Auto-Initialization Key
10. R = Resume the operation from pause state caused by external collision
11. Add Point in UI (Rising Edge)
12. Run Program Repeat (Rising Edge)
13. R=goto Begin posture / F=stop moving
14. R = Program Resume
15. H = Quick Freedrive Change
16. R = Pause / F = Resume Program
17. F = Pause / R = Resume Program
18. H = Speed 100% / L = 0%
19. R = Load Default Program
20. F = Robot Arm PowerDown
21. R = Touch Sensing
22. F = Touch Sensing
23. H = No Arc
24. H = Program Start Block
25. R = Ext.Joint0 Plus/F=stop

26. R = Ext.Joint0 Minus/F=stop
27. R = Ext.Joint1 Plus/F=stop
28. R = Ext.Joint1 Minus/F=stop
29. R = Ext.Joint2 Plus/F=stop
30. R = Ext.Joint2 Minus/F=stop
31. H = Safety Speed
32. F = UserCoord0 ←TCP frame
33. F = UserCoord1 ←TCP frame
34. F = UserCoord2 ←TCP frame
35. F = Load & Run Program Table

**Warning**

Warning:

1) Before using digital input, please fully understand electrical characteristics and all related manuals about digital input port.

■ Description of all types available for the output port.

The output ports from Dout 0 to Dout 15 in the control box can be set up as one of the following types. (R = Rising Edge, F = Falling Edge, H = High State).

All ports specified as one of types except for 'Default (0)' mode cannot be used in 'Teaching'

0. Default (GPIO)
1. H = Program/Robot is running / L = Idle
2. L = Program/Robot is running / H = Idle
3. H = External collision is detected
4. H = Direct teaching is running
5. Bypass the Digital input signal (same number Din port)
6. Bypass Tool Flange input 0
7. Bypass Tool Flange input 1
8. H = Robot's arm is in active status (servo on) / L = non-active
9. H = Real mode status / L = Simulation mode status
10. H = Robot is moving / L = Idle
11. L = Robot is moving / H = Idle
12. H = Robot activation (Servo-on) fail
13. H = Arm electric power is On / L = Power is Off
14. H = Collision detection is running / L = not-running
15. H = Pause state
16. H = Trap status in Inbox 0
17. H = Trap status in Inbox 1
18. PWM module
19. H = Teaching Pendant is connected
20. H = Program is running by MAKE page
21. H = Program is running by PLAY page
22. H = Is Conveyor mode
23. H = Control Box Boot

24. H = Force Control mode

25. PC Alive Pulse

26. H = Speed Bar 100%

27. H = Last Program Load Success

28. H = TCP is in InBox 0

29. H = TCP is in InBox 1

30. H = Is Alarm

31. H = Robot posture is Begin posture

32. H = Emergency Teaching Enable

33. H = Prog. Run in Sub.P area

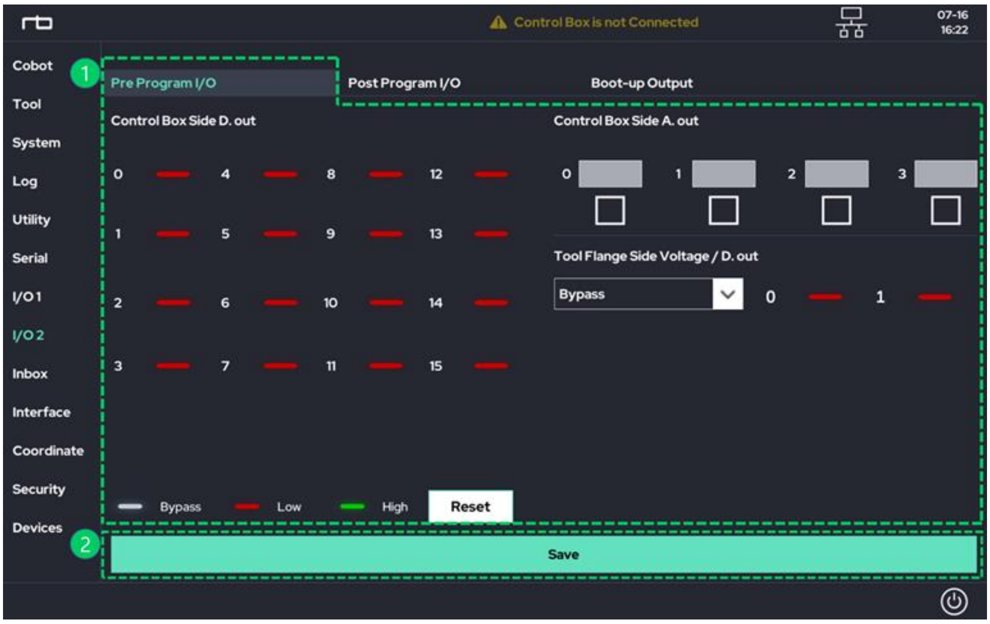
**Warning**

Warning:

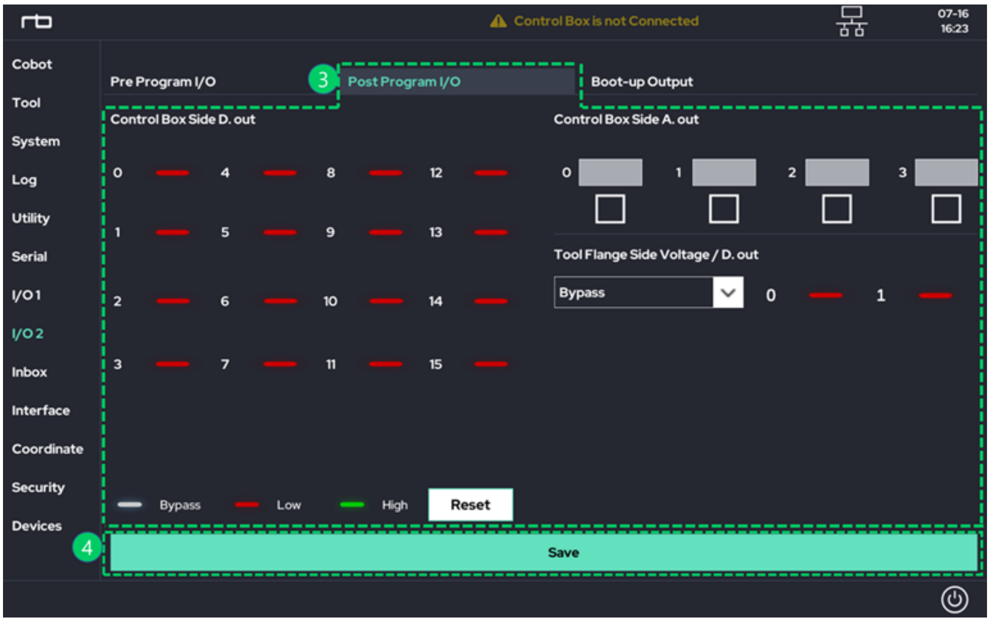
1) Before using digital output, please fully understand electrical characteristics and all related manuals about digital output port.

# 9.8 SET-UP(I/O 2)

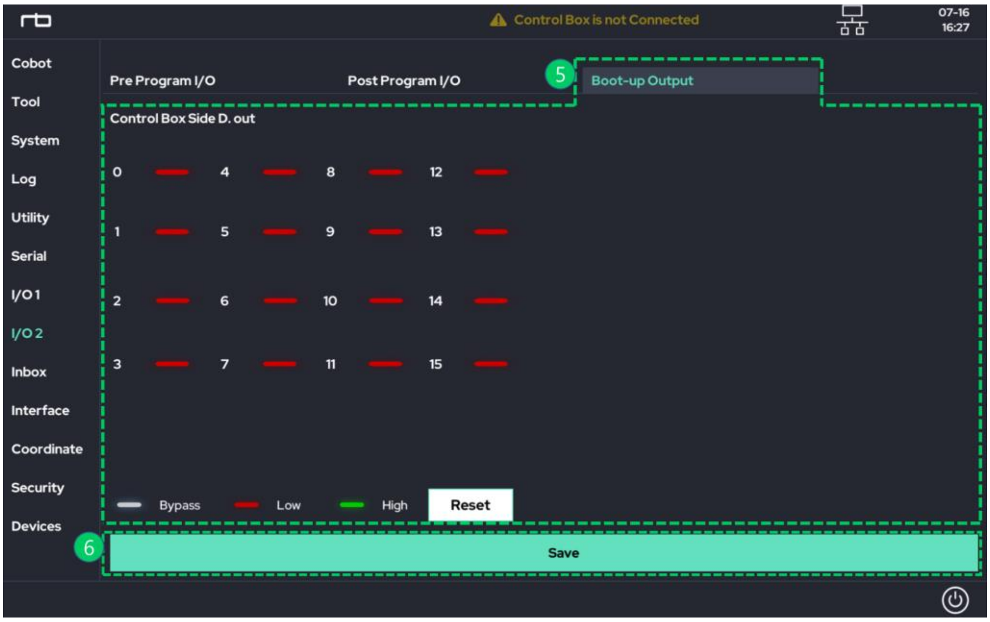
Set I / O value to always perform before / after program operation.




| | |
|---|---|
| ① | Set I / O transmission to be executed before program start.<br>Ports set here will send output to the corresponding settings as soon as the program starts. |
| ② | Save current settings |



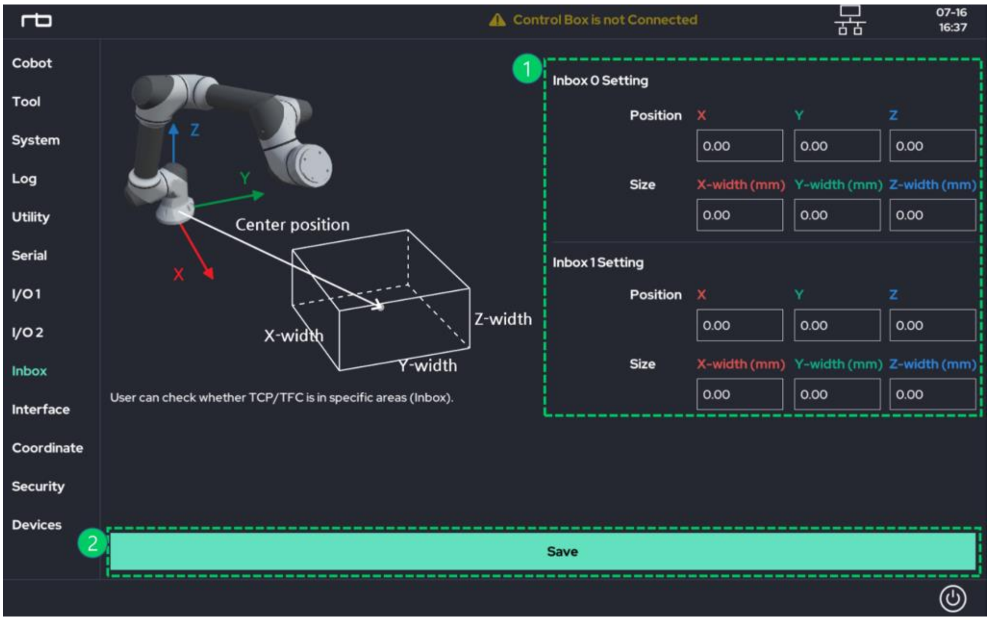

| | |
|---|---|
| ③ | Set I / O sending to be executed after program end.<br>Ports set here send output to the set value at the end of the program. |
| ④ | Save current settings |




| | |
|---|---|
| ⑤ | After the control box boots for the first time, select the digital output option for the control box. |
| ⑥ | Save current settings |

# 9.9 SET-UP(INBOX)

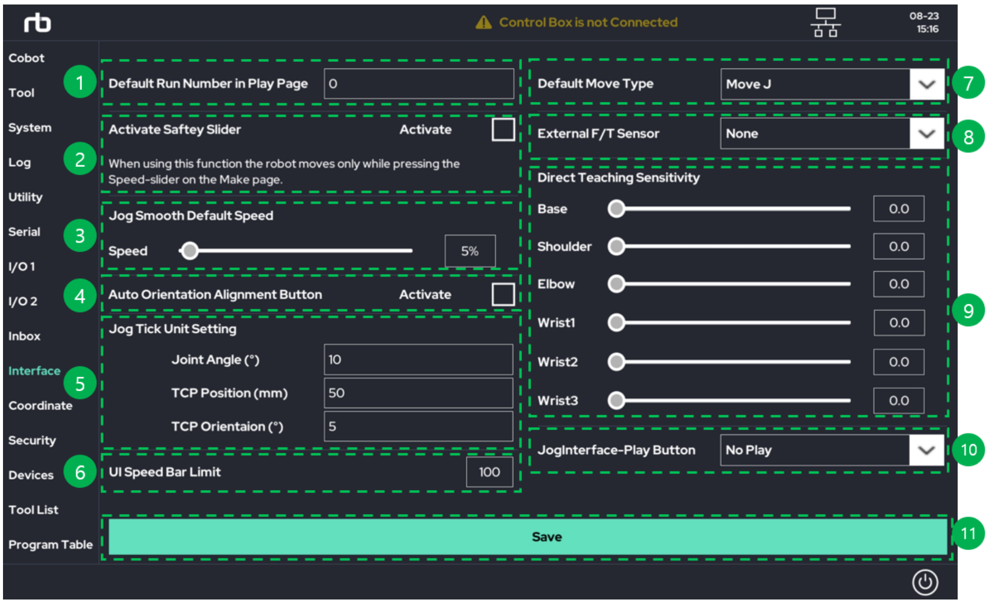
Sets the Inbox size and location information for using Inbox features




| | |
|---|---|
| ① | Input panel to specify center of mass and size for Inboxes 0 and 1. The coordinate system matches the manufacture's (robot base) coordinate system. |
| ② | Save changes |

# 9.10 SET-UP(INTERFACE )

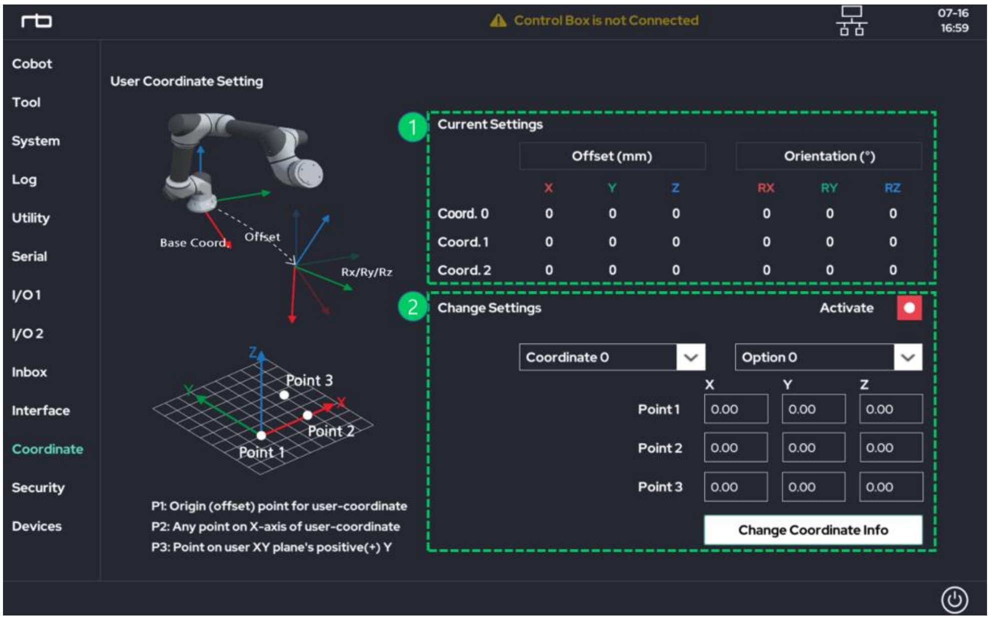
Perform the settings required for the user to operate the robot and UI.




| | |
|---|---|
| ① | Number of times to repeat loop in 'Play' screen. |
| ② | Enable/Disable Safety slider in 'Make' screen.<br><br>When enabled, a user must hold the slider to keep the speed. Otherwise, the speed is back to zero.<br>When disabled, the speed is maintained at the level that a user specifies. |
| ③ | Speed for the 'Smooth' option while using 'Jog'. |
| ④ | At the bottom of the jog on the Make screen, select whether or not to visualize the Auto Rotate Alignment function button. |
| ⑤ | Amount of movement per 'Tick' while using 'Jog'. |
| ⑥ | Limit the upper limit value of the speed bar on the UI |
| ⑦ | In the Make screen, select the action property to be created by default when creating Move. |
| ⑧ | Select the external-F/T sensor usage. |
| ⑨ | Joint sensitivity for direct-teaching. |
| ⑩ | When using a dedicated jog/emergency stop interface device provided by Rainbow Robotics, define the role of the play button on the device. |
| ⑪ | Save Changes |

# 9.11 Set-up(Coordinate)

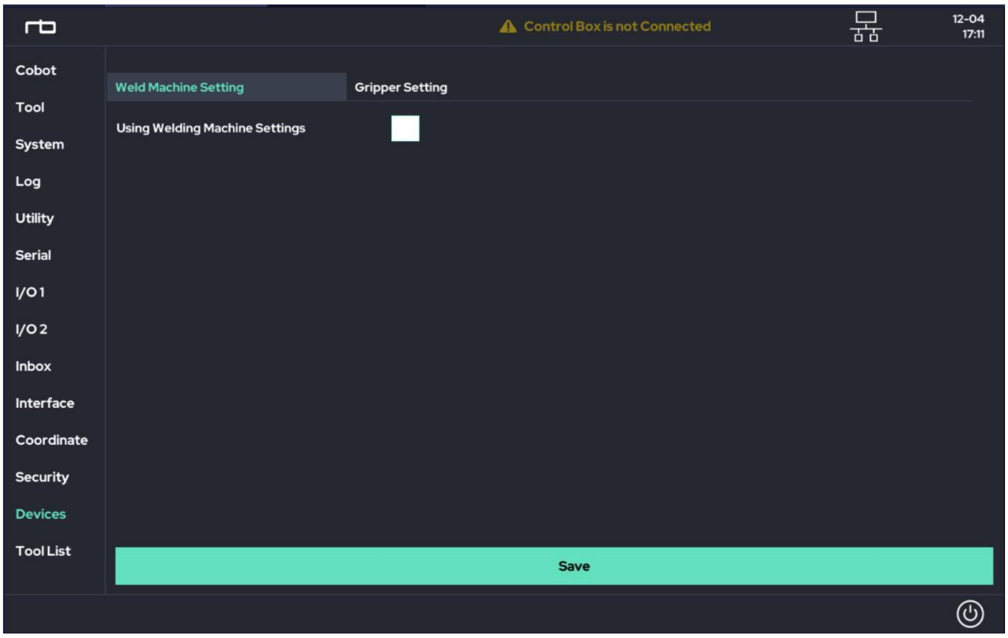
Contains information regarding user coordinate settings.




| | |
|---|---|
| ① | The current user-coordinate information manually set.<br>This coordinate is in respect to the manufacturer's base coordinate system. |
| ② | Add/Edit user-coordinate system.<br>The user-coordinate system is defined using the 3-point method. |

## 9.12 Set-up(Devices)

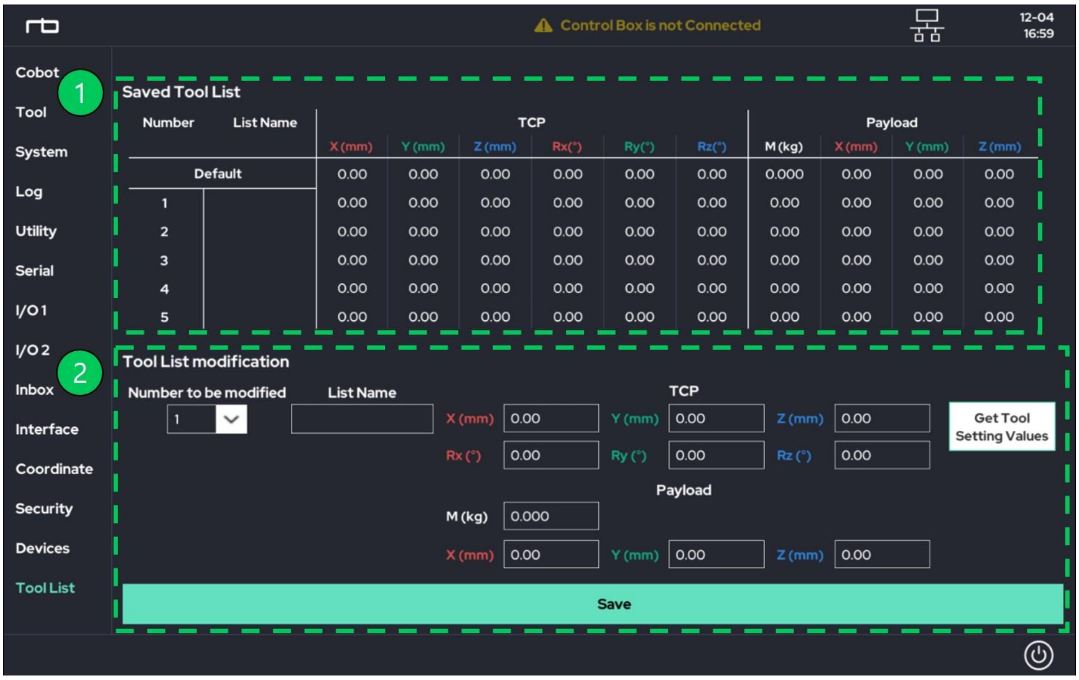
Set up additional equipment associated with the robot.




Proceed with the setup of the equipment connected to the robot.

# 9.13 Set-up(Tool List)

Set the Tool List.



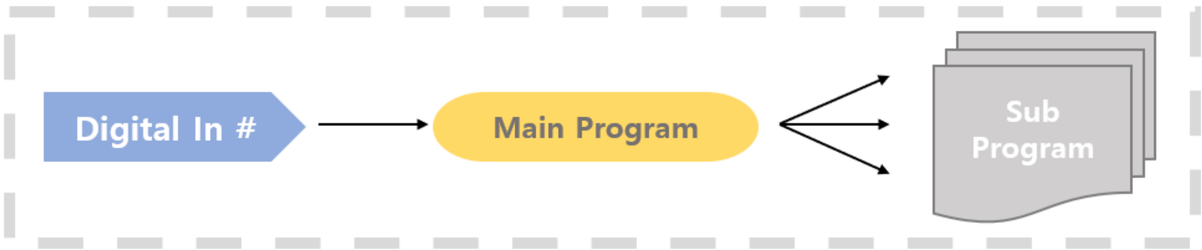

| | |
|---|---|
| ① | Indicates the TCP value currently set. |
| ② | Select the TCP number you want to modify. Set the name, TCP location, and center of gravity, and then save. |

# 9.14 Set-up(Program Table)

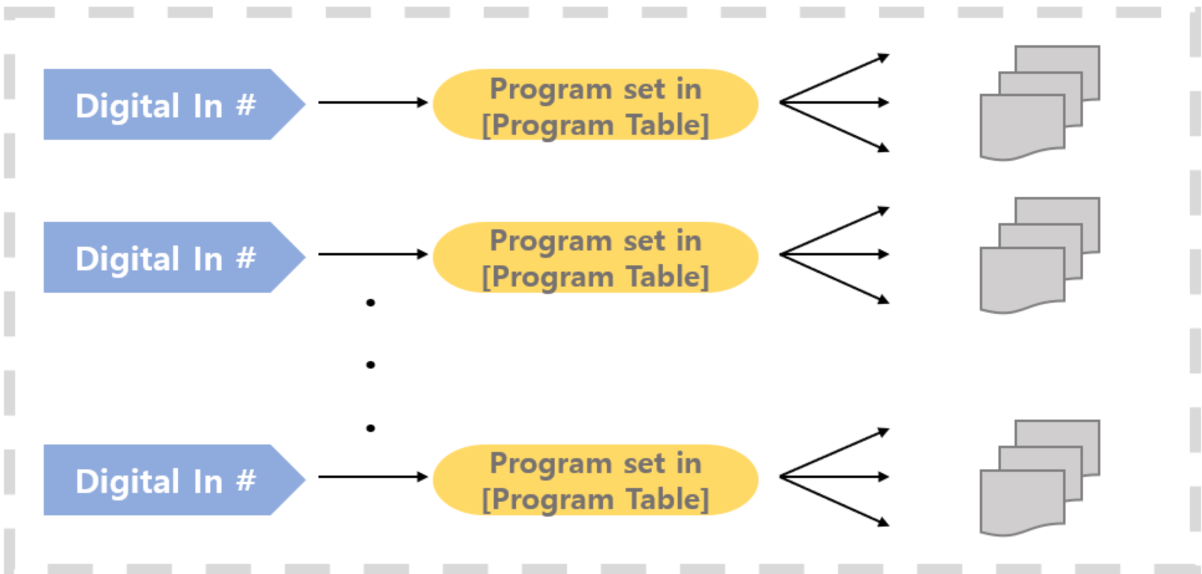

| | |
|---|---|
| ① | Import previously created settings.. |
| ② | Indicates which digital input port to use. |
| ③ | Select the function user want to use. The function is divided into Load, Load + Play(Once), and Load + Play(Repeat).. |
| ④ | Select the project to use via digital input. |
| ⑤ | Save the settings. |

The Start Program function used as the 'Control Box Digital In Function Definition' of the existing 'Setup > I/O1' can use only one digital input, and the program can run only main program currently uploaded to the control box. ‘Setup > Program Table’ can load different programs to different digital inputs. You can run additionally loaded programs.

The picture below shows the difference between the two features.




SetUp > I/O1




SetUp > Program Table

# CHAPTER 10. MAINTENANCE

## 10.1 CHECK LIST AND PERIOD

The robot requires regular maintenance to perform in the best condition. As such, a regular maintenance schedule is highly recommended. During maintenance, the following check list has to be done.

<table>
<tr><th>Check Item</th><th colspan="2">Check Point</th><th>Period</th></tr>
<tr><td rowspan="5">Robot Arm</td><td rowspan="3">Robot</td><td>Check that the robot moves to the desired location properly.</td><td rowspan="2">Daily</td></tr>
<tr><td>Check that the robot keeps its pose between being turned on and off.</td></tr>
<tr><td>Remove stains, dust, and any contamination.</td><td>Every<br>3 months</td></tr>
<tr><td>Motor</td><td>Check if a joint becomes irregularly hot or noisy</td><td>Daily</td></tr>
<tr><td>Screws</td><td>Check that all screws on the body are tightened</td><td>Every<br>3 months</td></tr>
<tr><td rowspan="2">Control Box</td><td>Cable</td><td>Check the connection of cables</td><td rowspan="2">Every<br>6 months</td></tr>
<tr><td>In-Box</td><td>Remove dust in the control box.</td></tr>
</table>

**Warning**

Warning:

1) During maintenance, cut off the power to the system (Control Box and Robot Arm) and perform work.

2) For pneumatic/electric line passing models, remove the connected energy source (pneumatic/electric power) and perform the work.

## 10.2 Robot Arm Maintenance

- Maintenance Period

  The robot arm requires an inspection at least per 1 year. Depending on the wear and tear, the maintenance period may differ.

- Maintenance Instruction

  1. Move the robot to the 'Home' position.
  2. Turn off the control box.
  3. Check the following list.
     ① Robot-Control Box Cable: Is it cut or stabbed?
     ② Screws: Are any loose?
     ③ Mechanical Parts (Motor, Brake, Reduction Gear): Are any louder than normal?
  4. Remove stains, dust, and any other contamination.

## 10.3 Control Box Maintenance

Dust in the control box may cause it to over-heat or generate electrostatic. These can potentially damage the control box. It is required to regularly clean up dust in the control box.

- Maintenance Period

  The control box requires an inspection and clean-up at least once per 6 months. Depending on the environmental condition around the robot, the period may differ.

- Maintenance Instructions

  1. Turn off the control box.
  2. Remove the cover of the control box.
  3. Remove dust in the control box with a vacuum cleaner.
  4. Check that all wires are connected properly.

# Robot Arm

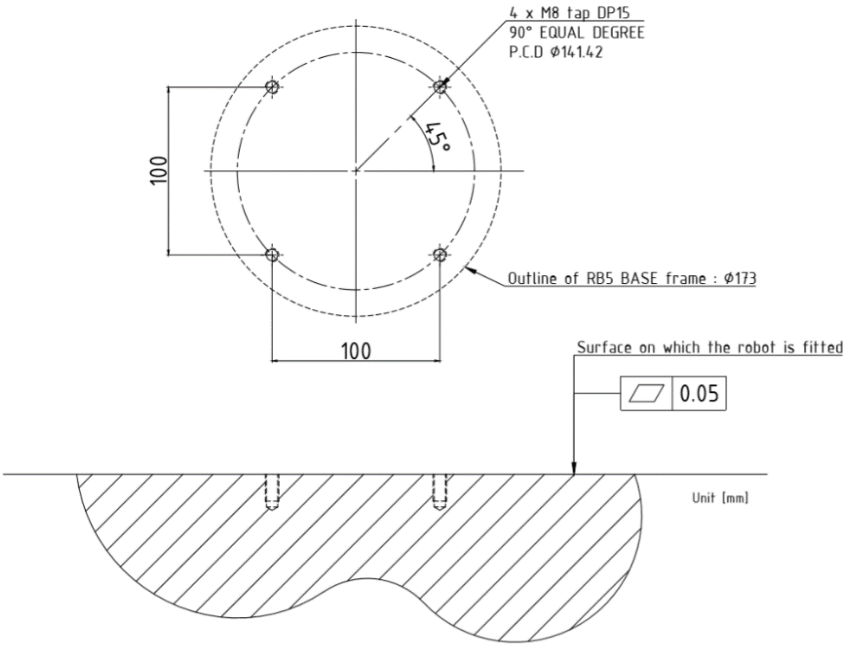
| | Specification |
|---|---|
| **Payload** | RB5-850E Series: 5 kg / 11 lbs<br>RB3-1200E Series: 3kg / 6.6 lbs<br>RB10-1300E Series: 10 kg / 22 lbs |
| **Weight** | RB5-850E Series, RB3-1200E Series: 22 kg / 48.5 lbs<br>RB10-1300E Series: 33 kg / 72.8 lbs |
| **Arm Reach** | RB5-850E Series: 850 mm / 33.5 in<br>RB3-1200E Series: 1200 mm / 47.2 in<br>RB10-1300E Series: 1300 mm / 51.1 in |
| **Degree of freedom** | 6 axis |
| **Joint Range** | ± 360°(Elbow: ± 165°) |
| **Joint Velocity** | Joint: 180°/s, TCP: 1m/s |
| **Repeatability** | ±0.1 mm |
| **Foot print** | RB5-850E/RB3-1200E Series: Φ173 mm<br>RB10-1300E Series: Φ196 mm |
| **Tool Flange Connector** | M10 12-pin |
| **Tool Flange I/O** | Non-E Version : Digital In 2, Digital Out 2, Analog In 2<br>E Version : Digital In 6, Digital Out 2 |
| **Tool Flange Comm.** | RS485 |
| **Tool Flange Output Vol.** | 12V/24V, 2A |
| **IP Rate** | IP66 |
| **Temperature / Noise** | 0 ~ 50 ˚C / <65dB |
| **Material** | Aluminum, Steel. |
| **Cable Length** | Power cable, RobotArm-ControlBox connection cable, Estop/Jog Interface cable : 5m |

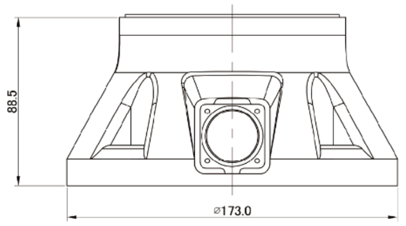
# Stand-type Control Box

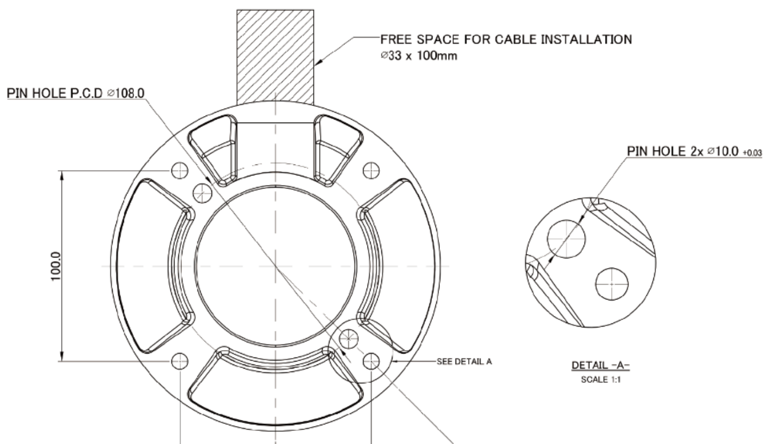
| | Specification |
|---|---|
| **Weight** | CB04: 17.3 kg / 38.14 lbs |
| **Size (W x H x D)** | 454 x 240 x 416.2 mm |
| **I/O Ports** | Digital Input 16 / Digital Output 16<br>Analog Input 4 / Analog Output 4 |
| **Communication** | Ethernet, TCP/IP |
| **Power** | 100 ~ 240 VAC, 50 ~ 60 Hz |
| **Material** | EGI |

# Appendix B. Foot Print Schematic

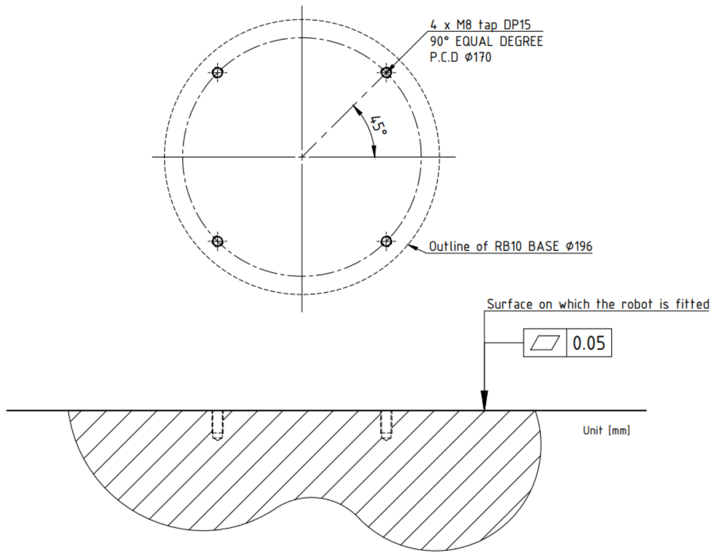
- RB5-850E / RB3-1200E Series Foot Print Schematic
  - ▶ P.C.D: Pitch Circle Diameter
  - ▶ DP: Depth

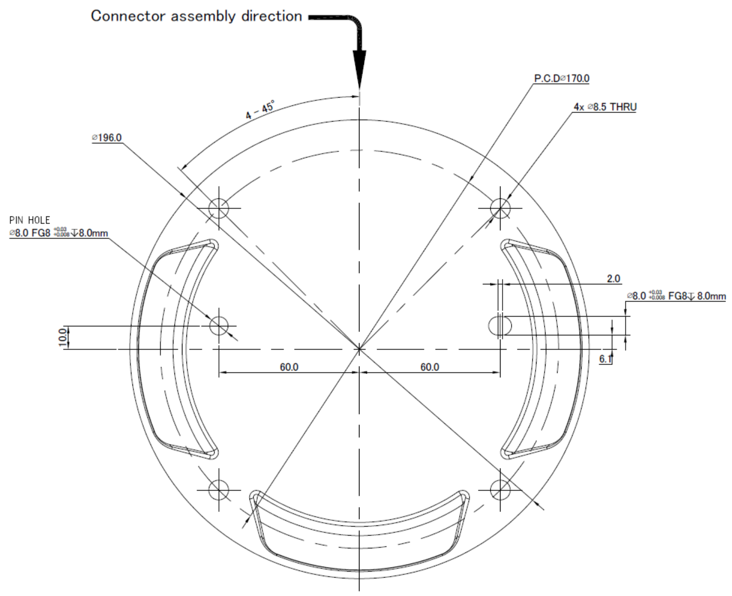


- RB10-1300E Series Foot Print Schematic
  - ▶ P.C.D: Pitch Circle Diameter
  - ▶ DP: Depth



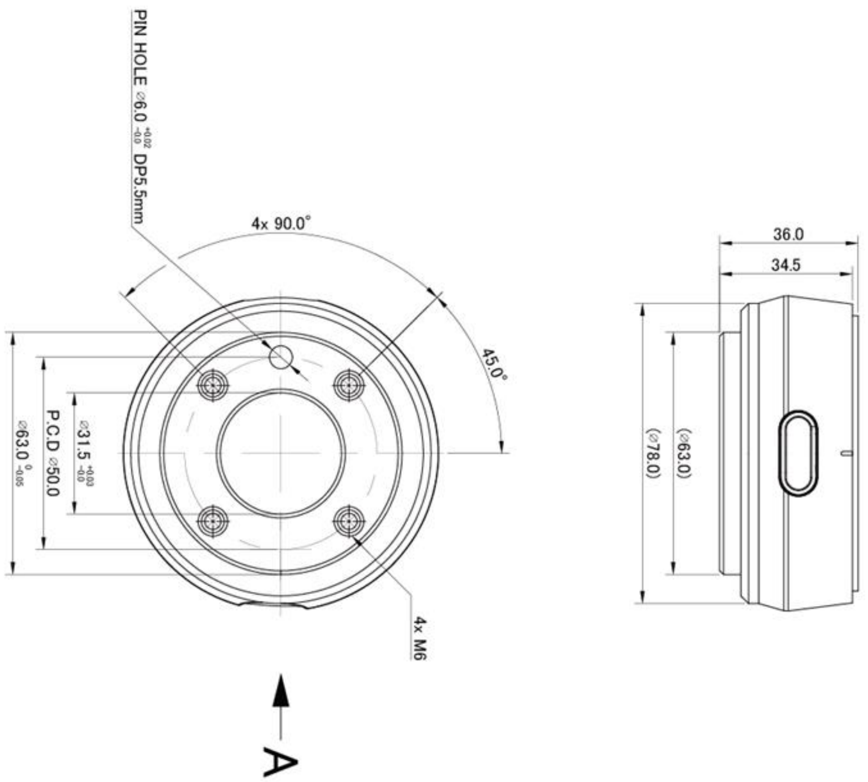



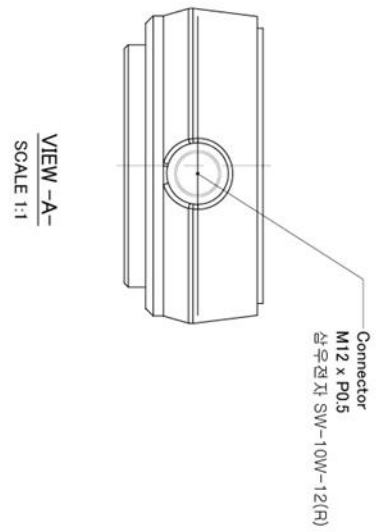
# APPENDIX C. TOOL FLANGE SCHEMATIC

■ RB5-850E / RB3-1200E Series Tool Flange Schematic

▶ P.C.D: Pitch Circle Diameter

▶ DP: Depth

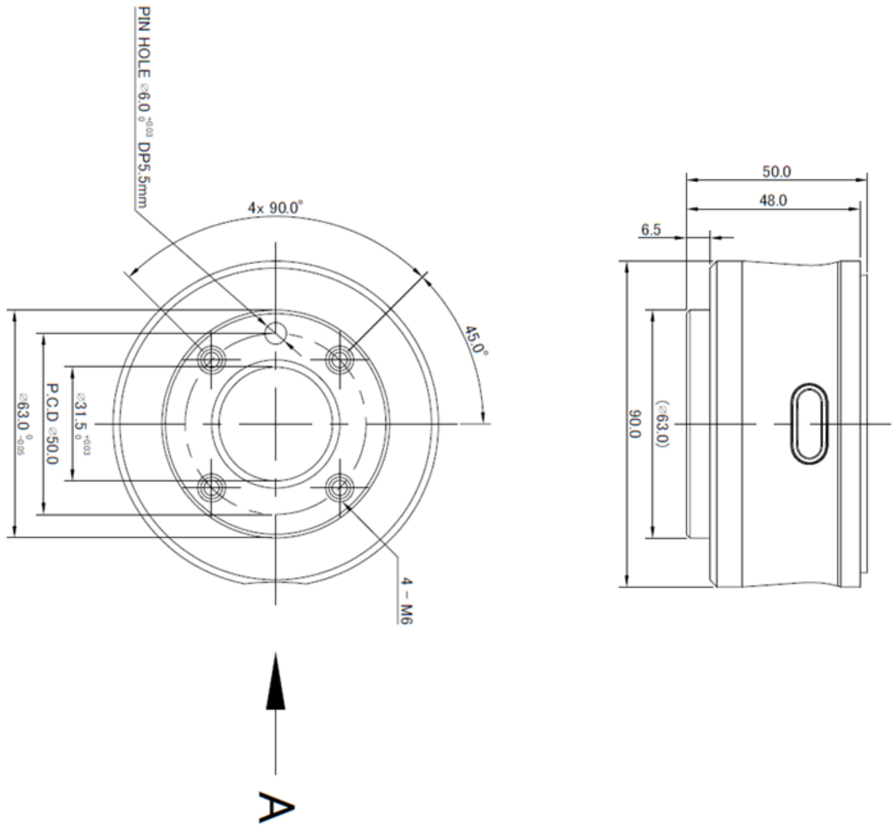


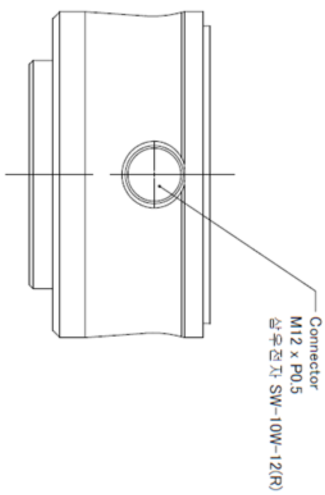



■ RB10-1300E Series Tool Flange Schematic

▶ P.C.D: Pitch Circle Diameter

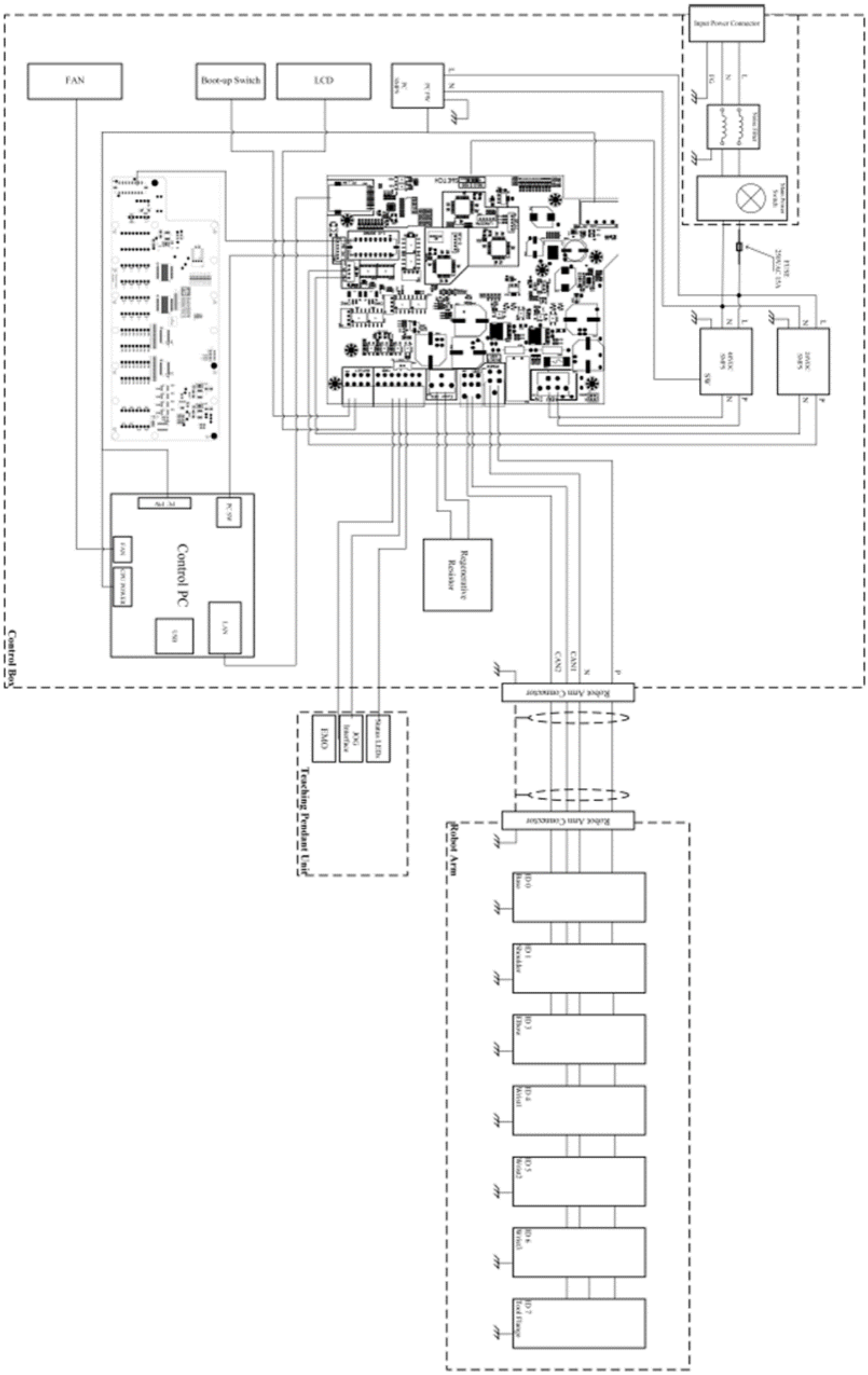
▶ DP: Depth.



# APPENDIX D. CONTROL BOX ELECTRICAL SCHEMATIC

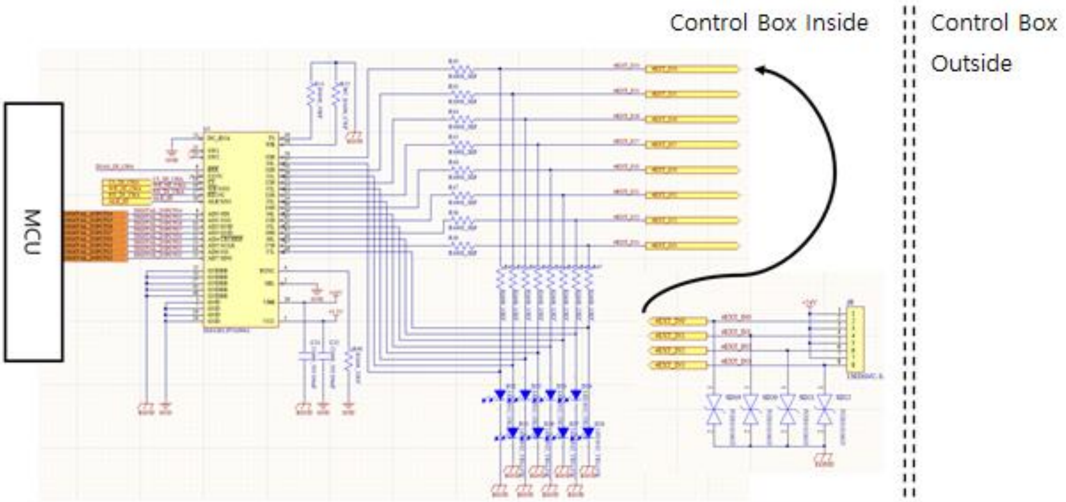
- Stand-type Control Box(CB04) Electrical Schematic



# Appendix D-1. Control Box Digital Input

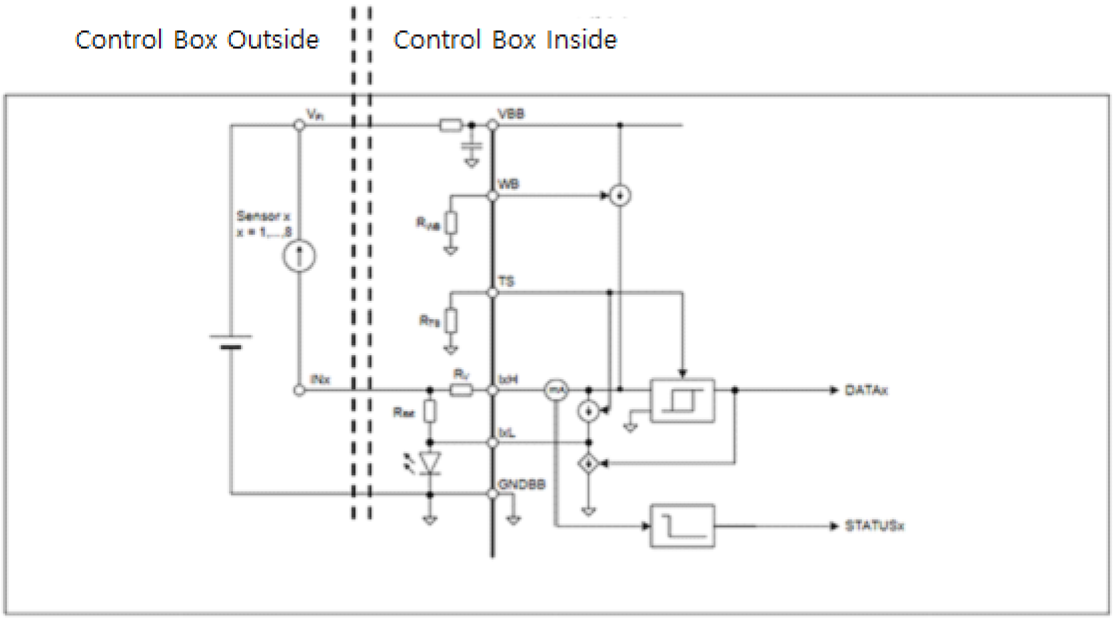
- **Warning**
  Before connecting Control Box Digital input port, the power should be cut off.

## 1. Internal Circuit Diagram of Digital Input [DI00 ~ DI15]




Device configuration that receives Control Box Digital input [DI00-DI15].
There is an internal 24V supply terminal. A malfunction will occur if an external 24V is supplied.

## 2. How to use digital input elements [DI00 ~ DI15]




How to use RB Control Box Digital input device [DI00-DI15].

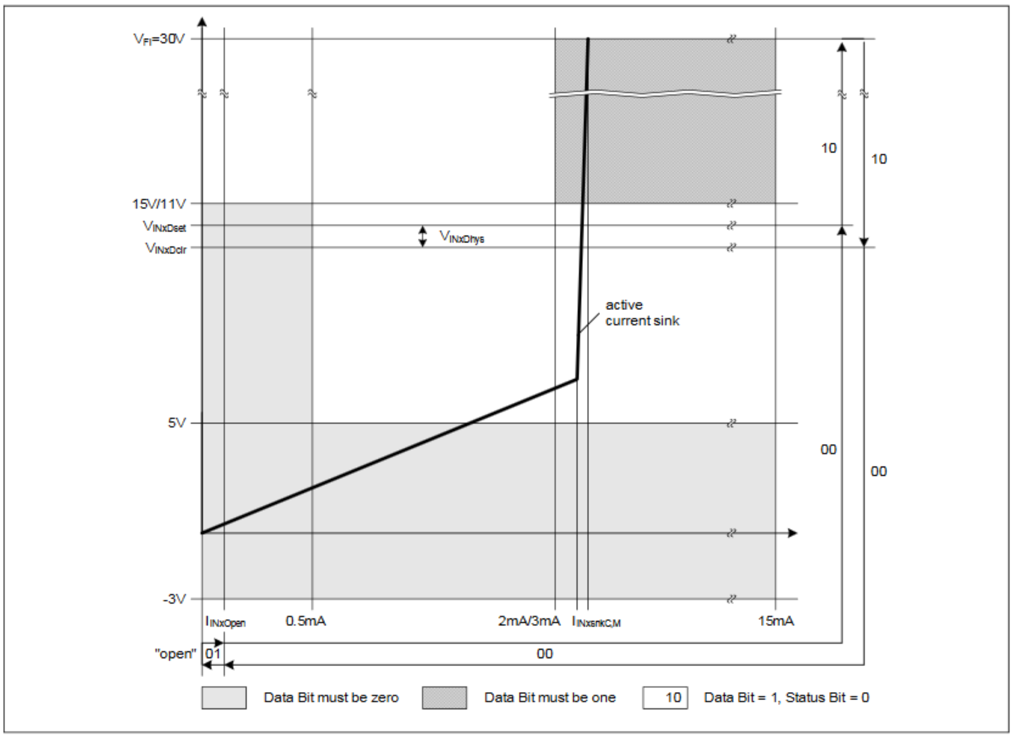



**Figure 11 Sensor Input Characteristics**

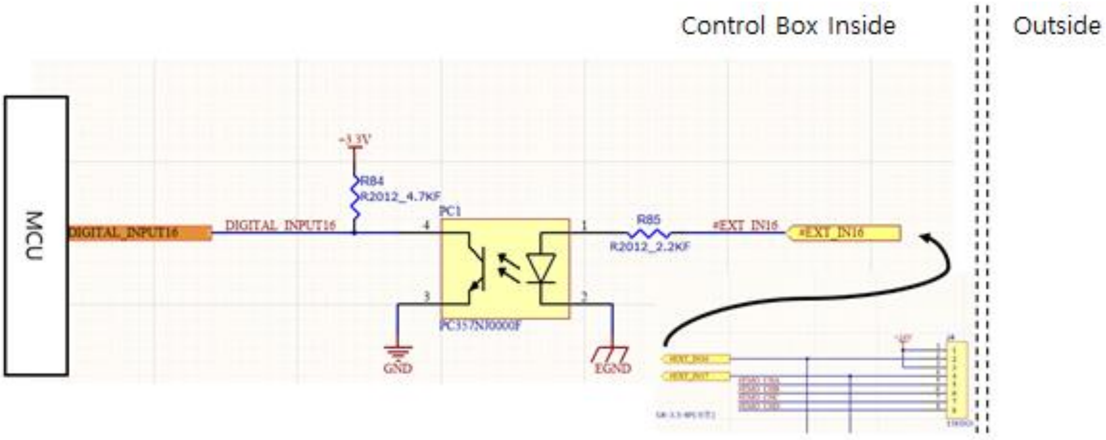
Voltage / current characteristic curve of digital input signal.

## 3. Digital input characteristics [DI00 ~ DI15]

| Terminals | Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| [DI00 – DI15] | Voltage | -3 | - | 30 | V |
| [DI00 – DI15] | OFF region | -3 | - | 5 | V |
| [DI00 – DI15] | ON region | 11 | - | 30 | V |
| [DI00 – DI15] | Current(11-30V) | 2 | - | 15 | mA |
| [DI00 – DI15] | Function | - | PNP+ | - | Type |
| [DI00 – DI15] | IEC 61131-2 | - | 1 | - | Type |

<u>This specification applies only to digital input 0 to digital input 15.</u>

4. Internal Circuit Diagram of Digital Input [DI16-DI17]




Device configuration that receives Control Box Digital input [DI16-DI17]. There is an internal 24V supply terminal. A malfunction occurs when an external 24V is supplied.

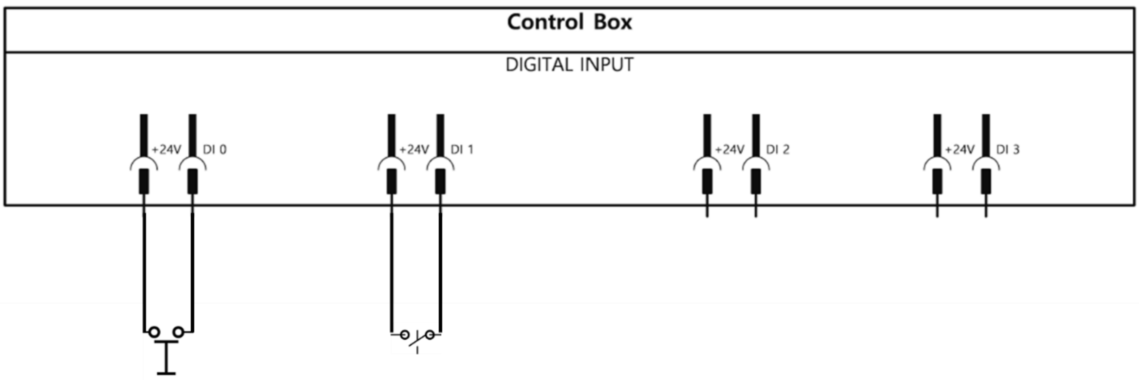
5. Digital input characteristics [DI16-DI17]

| Terminals | Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| [DI16 – DI17] | Voltage | 0 | - | 25 | V |
| [DI16 – DI17] | OFF region | 0 | - | 7 | V |
| [DI16 – DI17] | ON region | 7 | - | 25 | V |
| [DI16 – DI17] | Function | - | PNP+ | - | Type |

This applies only to digital inputs 16 and 17.

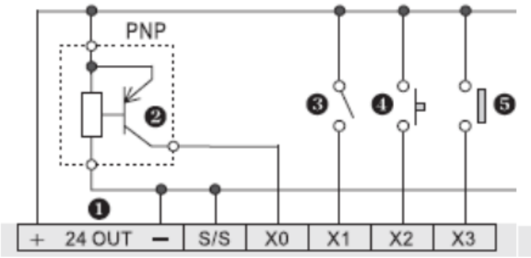
6. Testing environment

Digital input device test was conducted using Toggle switch, and the following configuration was tested.



## 7. How to use PNP sensor

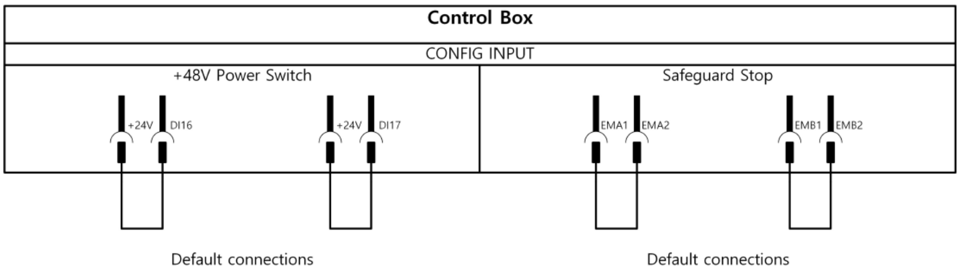
Ex source : https://blog.naver.com/mjg5080/97380010




PNP sensor can be used in the same way as above.

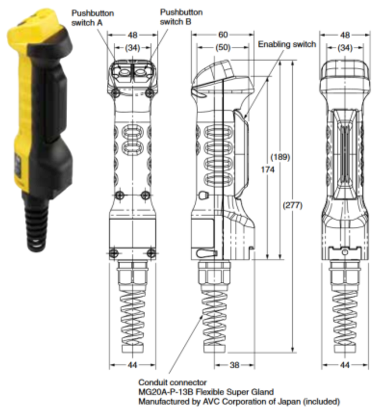
<u>This is a specification that applies to all digital inputs.</u>

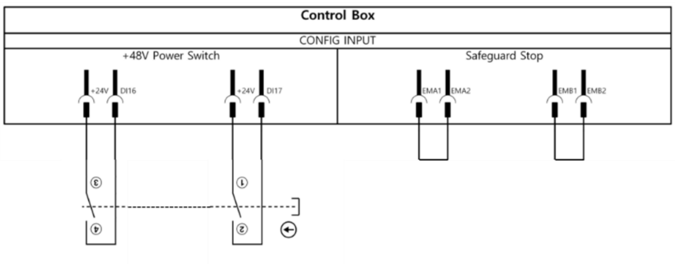
## 8. How to connect 3-Position Enabling Device

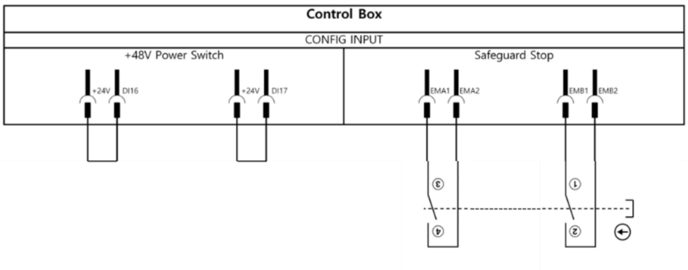



The initial factory condition is as above, and it is possible to install the operation.

Source : https://www.motionsolutions.com






<u>This applies to Enabling Device in accordance with ISO 10218, IEC 60204-1.</u>

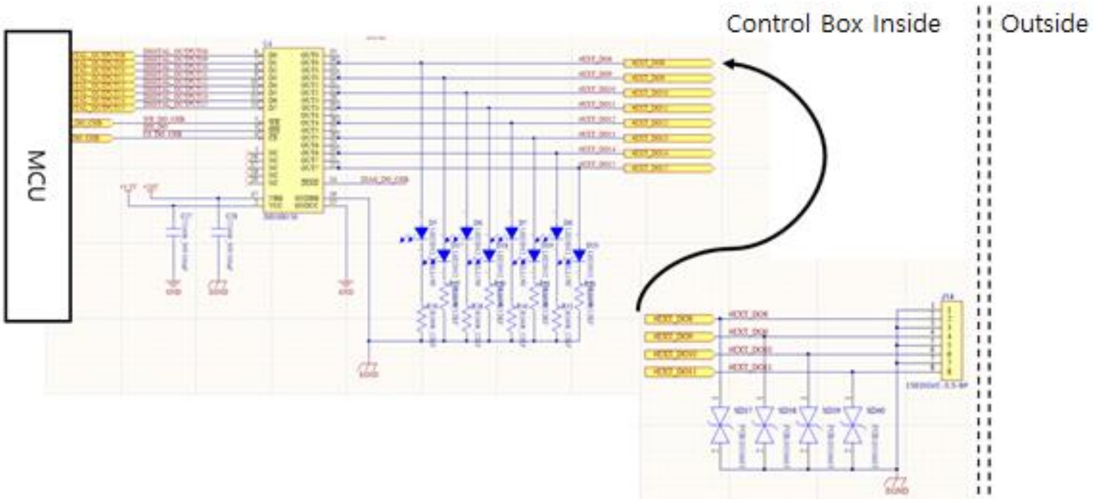
## 9. How to connect safety equipment

Safety device wiring using PNP type sensor and Enabling Device such as light curtain and safety door sensor is same as above.

# APPENDIX D-2. CONTROL BOX DIGITAL OUTPUT

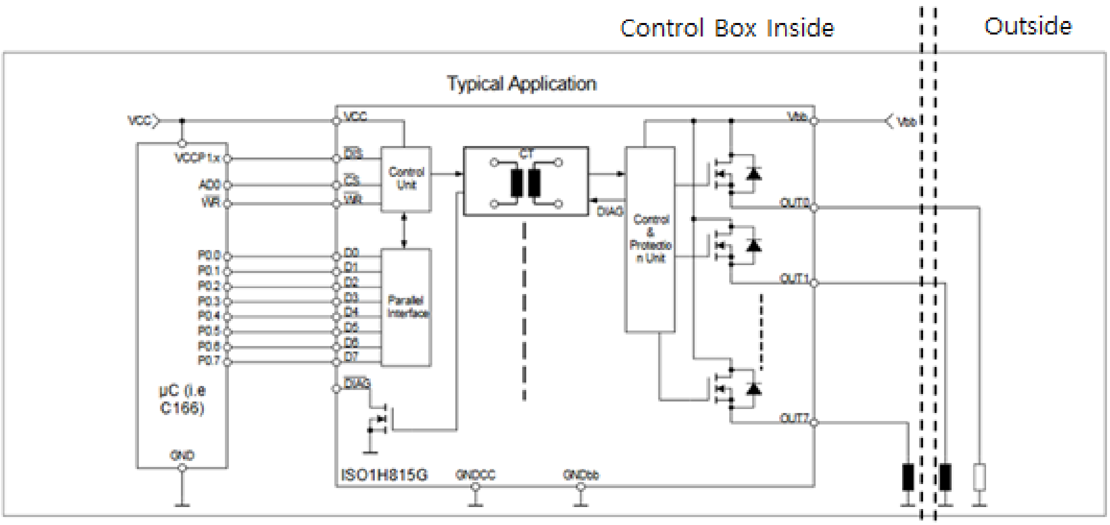
- **Warning**
  Before connecting the Control Box Digital output port, the power should be turned off.

### 1. Digital output internal circuit diagram [DO00-DO15]

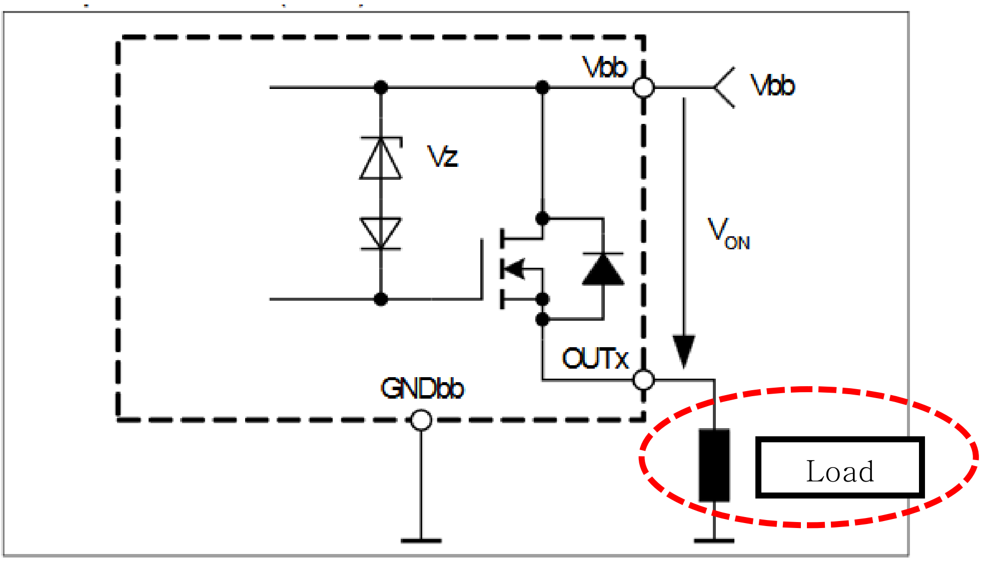



Device configuration that performs Control Box Digital output [DO00-DO15]. There is internal GND terminal, and it should be connected to GND of external sensor and equipment to be connected.

### 2. Digital output device usage [DO00-DO15]




How to use RB Control Box Digital Output Device [DO00-DO15].




How to use a single digital output.
Vbb power is supplied inside of the control box and its output is the source.

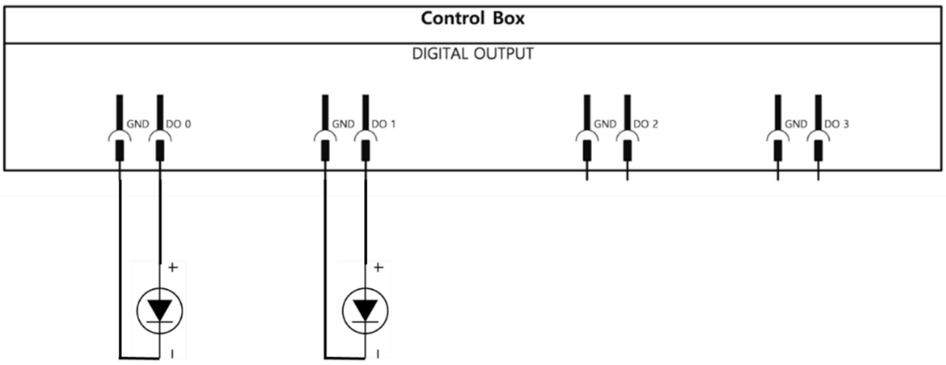
## 3. Digital output characteristics [DO00-DO15]

| Terminals | Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| [DO00 – DO15] | Voltage | - | 24 | - | V |
| [DO00 – DO15] | CURRENT | 0 | - | 1 | A |
| [DO00 – DO15] | Function | - | PNP | - | Type |

<u>Single channel 1A is possible, but the total current of all channels must be less than 2A</u>

## 4. Test environment

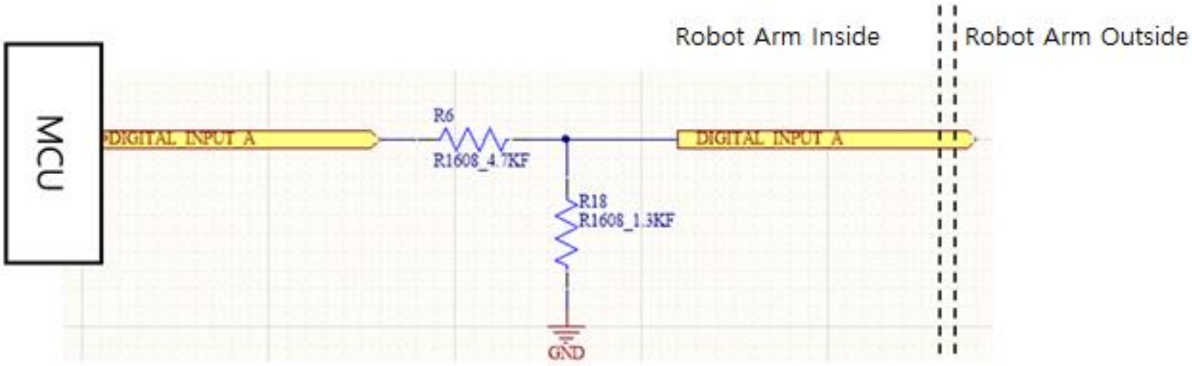
Digital output device test was conducted using 24Vdc LED and the following configuration was tested.



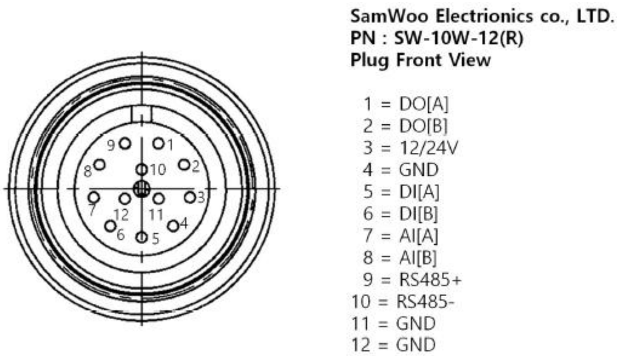
# APPENDIX D-3. TOOL FLANGE DIGITAL INPUT

- Warning
  Before connecting RB Tool Flange I / O input port, the power should be cut off.
- The electrical drawing below is for Non-E type only.

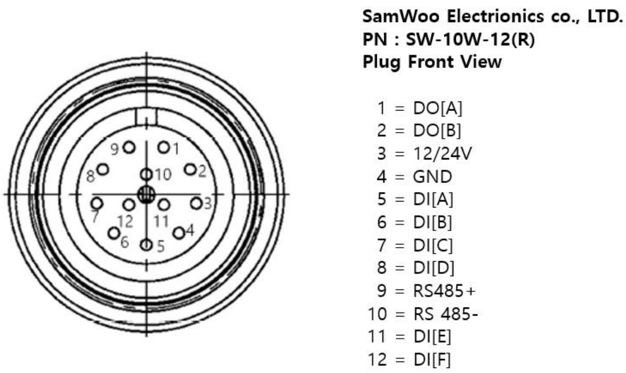
1. Digital input internal circuit diagram [DIA, DIB]




Device configuration for Tool Flange Digital input.




(1) Non-E Version Robot




(2) E Version Robot

Exposed connector wiring diagram.

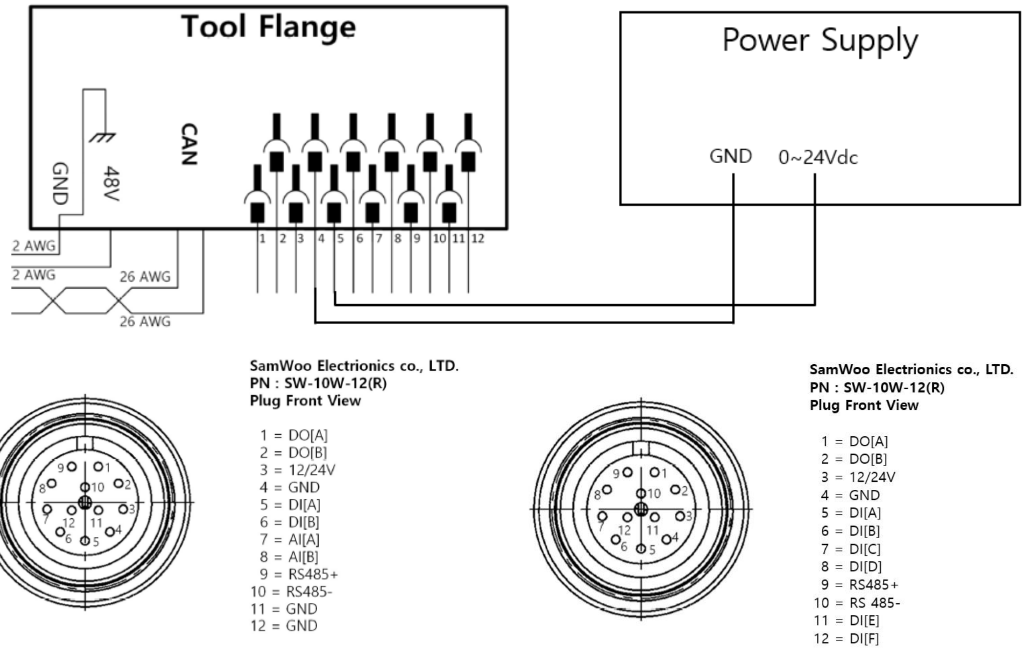
## 2. Digital input characteristics [DIA, DIB]

| Terminals | Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| [DIA, ... , DIF] | Voltage | 0 | - | 24 | V |
| [DIA, ... , DIF] | OFF region | 0 | - | 9 | V |
| [DIA, ... , DIF] | ON region | 10 | - | 24 | V |

<u>This is a specification that applies only to Tool Flange Digital input (At this time, only DIA and DIB for Non-E version Robot are applied.)</u>

## 3. Test environment

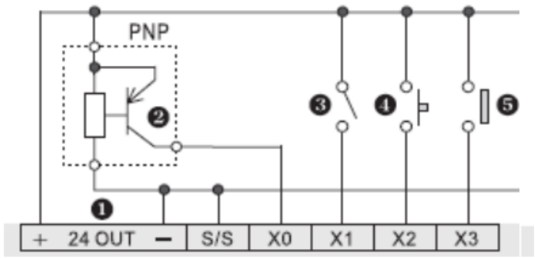
Digital input device test was conducted using power supply, and the following configuration was tested.




Non-E Version                                        E Version

## 4. How to use PNP sensor

Ex source : https://blog.naver.com/mjg5080/97380010



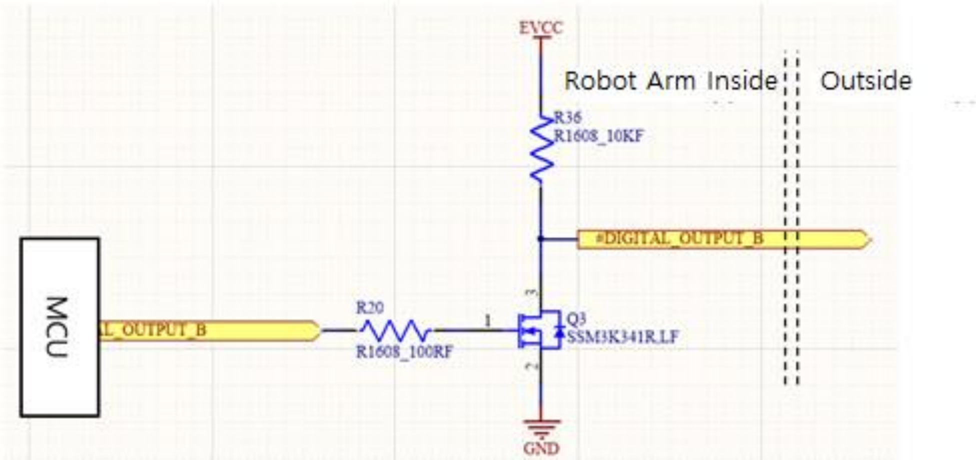

PNP sensor can be used in the same way as the above connection.

<u>This applies equally to the Control Box Digital input.</u>

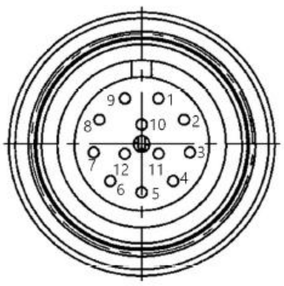
# Appendix D-4. Tool Flange Digital Output

- **Warning**
  Before connecting the Tool Flange I / O output port, the power should be cut off.
- The electrical drawing below is for Non-E type only.

1. Digital output internal circuit diagram



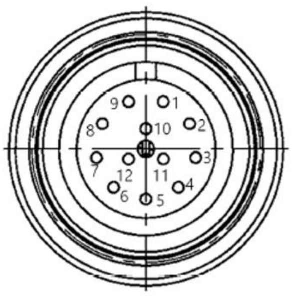

Device composition for Tool Flange Digital output [DOA, DOB].



**SamWoo Electrionics co., LTD.**
**PN : SW-10W-12(R)**
**Plug Front View**

1 = DO[A]
2 = DO[B]
3 = 12/24V
4 = GND
5 = DI[A]
6 = DI[B]
7 = AI[A]
8 = AI[B]
9 = RS485+
10 = RS485-
11 = GND
12 = GND

(1) Non-E Version Robot



**SamWoo Electrionics co., LTD.**
**PN : SW-10W-12(R)**
**Plug Front View**

1 = DO[A]
2 = DO[B]
3 = 12/24V
4 = GND
5 = DI[A]
6 = DI[B]
7 = DI[C]
8 = DI[D]
9 = RS485+
10 = RS 485-
11 = DI[E]
12 = DI[F]

(2) E Version Robot

External connector wiring diagram.

## 2. Digital output characteristics

| Terminals | Parameter | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| [DOA, DOB] | Voltage | 0 | 12/24 | 24 | V |
| [DOA, DOB] | CURRENT Ver 1. | 0 | 150 | 700* | mA |
| [DOA, DOB] | CURRENT Ver 2. | 0 | 2000 | 2000 | mA |

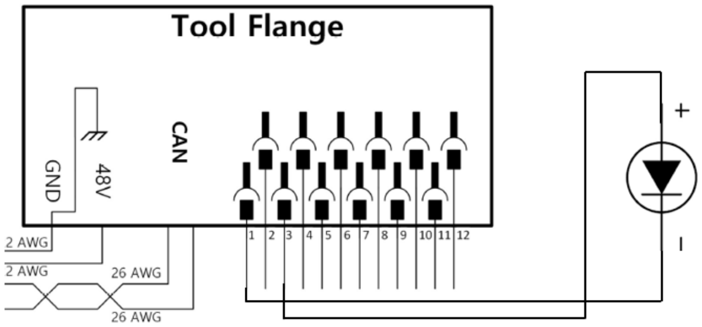
*Tsp=25°C; pulsed; tp≤10µs

<u>This specification applies only to Tool Flange Digital outputs A and B.</u>

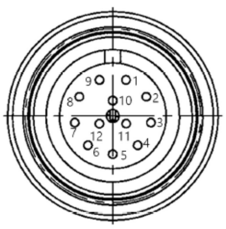
<u>As of July 24, 2019, version of RB5 shipped out is Ver 1.</u>

## 3. Test Environment

Digital output device test was conducted using 24V dc LED and the following configuration was tested.



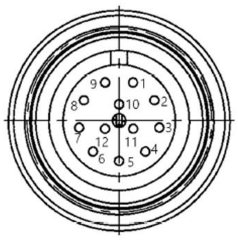

SamWoo Electrionics co., LTD.
PN : SW-10W-12(R)
Plug Front View




1 = DO[A]
2 = DO[B]
3 = 12/24V
4 = GND
5 = DI[A]
6 = DI[B]
7 = AI[A]
8 = AI[B]
9 = RS485+
10 = RS485-
11 = GND
12 = GND

Non-E Version

SamWoo Electrionics co., LTD.
PN : SW-10W-12(R)
Plug Front View

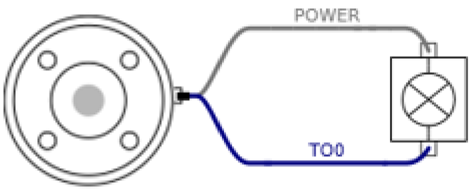



1 = DO[A]
2 = DO[B]
3 = 12/24V
4 = GND
5 = DI[A]
6 = DI[B]
7 = DI[C]
8 = DI[D]
9 = RS485+
10 = RS 485-
11 = DI[E]
12 = DI[F]

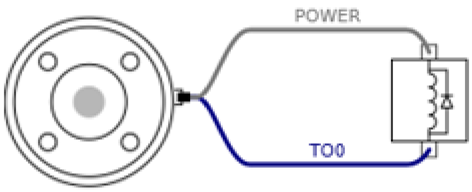
E Version

The following example is shown in this manual.

- The image shown below illustrates how to turn on/off a load with 12V or 24V. The voltage level can be specified in the Tool Out (TO0) block

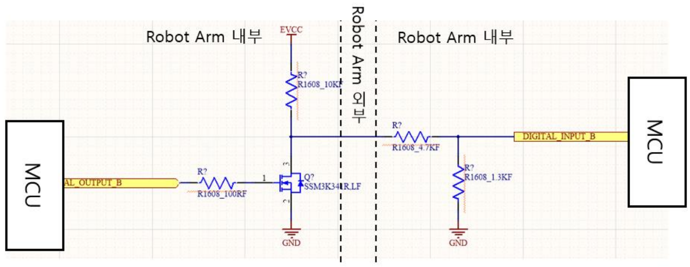



※ It is strongly recommended to use a diode to protect the tool using an inductive load

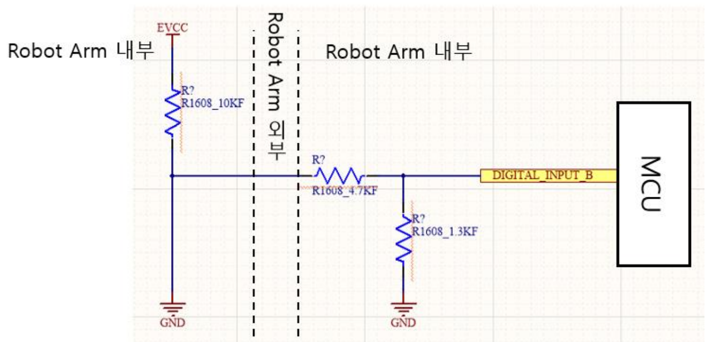



## 4. Precautions when using

Digital output device is **NPN** type but has internal 10K pullup resistor.
Most devices (LEDs, solenoid valves, relays) can be used in the test environment No. 3 or with the digital signal application function on commercial grippers. However, they may not work in the environment using the same voltage distribution as Rainbow Robotics' tool flange digital input devices.

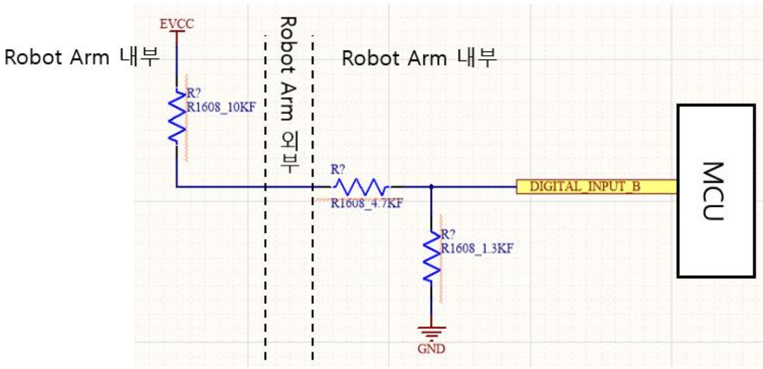



When Rainbow Robotics' digital output is connected to the digital input




Low Digital output

Digital input is output low with 0V input.




Digital output high

Digital input may not be recognized depending on the resistance value.

For the diagram above, if the voltage applied to MCU is EVCC 24Vdc, about 2V is applied to it and is detected as Low.

If users MUST operate as above diagram, digital input stage resistance ratio adjustment is necessary.

# Appendix E. External Script Control API

## E.0 Concept

The cooperative robot RB series can be operated for various environments and purposes. It can be used in conjunction with multiple RB series or other systems. In conjunction with the vision system, movement coordinates can be changed in real time, or used as part of a user's existing system.

Users can control the robot with teaching pendant (tablet UI), but it provides a way to control the robot from any external controller for user's convenience or operation.

The RB series receives script commands by default and executes those commands. The task of writing a motion using the teaching pendant (tablet UI) and executing the script of the file in order is a general operation method. The following method described in this document is an alternative method of receiving a command script from another external device to control a robot of the RB series.

The control syntax provided in the teaching pendant / tablet UI can be implemented by the user directly from the external control device, and the robot operation commands / IO control commands are sent according to the user's use case.

The following document describes an example of driving a robot with the above concepts.

## E.1 External Control Script API

The description of the scripts provided in this document looks similar to the scripts in the ".wsl" work document, which is written using a tablet as a dedicated script for external control. Work documents contain statements that control flows such as "repeat", "if-else", and "break", so that the completion of a statement is not directly related to the action, and the parent sentence of that statement must be completed.

For example, suppose there are Point Functions in the Move command Function.

| 1) |
|---|
| move joint {<br>  point ( ) absolute 0.4, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0<br>    point ( ) absolute 0.4, 0.1, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0<br>} |

| 2) |
|---|
| move joint {<br>  point ( ) absolute 0.4, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0<br>    point ( ) absolute 0.4, 0.1, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0 |

The difference between 1) and 2) is the presence or absence of "}" at the end. In both cases, the point statement is complete. However, unlike 1), 2) is a syntax that cannot operate because the move statement, which is the parent of point, is not completed, and the parser will wait for the statement to complete.

| 3) |
|---|
| folder( ) {<br>    move joint {<br>    point ( ) absolute 0.4, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0<br>        point ( ) absolute 0.4, 0.1, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0<br>    } |

In the same logic as above, the parser does not run because it waits for the folder statement to complete.

However, the above method is not suitable for external control method. The user expects the robot to operate by parsing the command the moment it sends it through external control. It does not send multiple commands and complete those lines of text like example 3).

So external control must be organized so that each command is sent separately as a string. External control does cannot access any features that control the flow. Commands such as "repeat", "if-else", "break" or "wait" in the work document will not be available externally and must be replaced by the same structure and logic within the external control.

The following commands are actual motion commands to move the robot. Each one contains an example string that matches how a user would control the robot from an external application.

There are five operation commands.

1) jointall
2) movetcp
3) movecircle
4) blend_jnt
5) blend_tcp

**jointall**

| Command | Jointall |
|---|---|
| Script | jointall spd, acc, joint1, joint2, joint3, joint4, joint5, joint6 |
| Descript. | This command moves joints in Joint Space.<br><br>The input values for joint1 to joint6 in the command denotes base, shoulder, elbow, wrist1, wrist2 and wrist3 accordingly. Each joint value represents the desired angle to go. The desired angle should be an absolute angle in degree.<br><br>The input values for spd and acc are used to define velocity and acceleration accordingly. The spd and acc should be a number between 0 and 1. Smaller number represents slower. When the input value is -1, the joint moves with the default value.<br><br>This command will be ignored if the previous command is not finished yet. |
| Example | “jointall 0.4, 0.1, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0” |

**movetcp**

| Command | Movetcp |
|---|---|
| Script | movetcp spd, acc, x, y, z, rx, ry, rz |
| Descript. | This command moves TCP in Cartesian Space.<br><br>The input values for x, y, z are used to define the desired position to go. The values should be a number in mm.<br><br>The input values for rx, ry, rz are used to define the desired orientation to go. It is represented as roll, pitch and yaw in Euler angle, accordingly. The values should be a number in degree. |

| | |
|---|---|
| | The input values for spd and acc are used to define velocity and acceleration accordingly. The spd and acc should be a number between 0 and 1. Smaller number represents slower. When the input value is -1, the joint moves with the default value.<br><br>This command will be ignored if the previous command is not finished yet. |
| Example | "movetcp 0.4, 0.1, 100.0, 100.0, 300.0, 0.0, 90.0, 0.0" |

### movecircle

| Command | movecircle(three points mode) |
|---|---|
| Script | movecircle threepoints orientation_option spd, acc, x1, y1, z1, rx1, ry1, rz1, x2, y2, z2, rx2, ry2, rz2 |
| Descript. | This command generates the circular motion of TCP using three points.<br><br>Three options determining the orientation of TCP in drawing a circle are available in orientation_option.<br><br>With 'intended', TCP follows the input orientation for mid-point (rx1, ry1, rz1) and end-point (rx2, ry2, rz2).<br>With 'constant', TCP keeps the current orientation during the circular motion.<br>With 'radial', the orientation of TCP changes in a way of the tangent direction to the center of the circle.<br><br>The input values for x1, y1, z1 are used to define the relative position of TCP at mid-point from the center of the circle. It is a number in mm.<br><br>The input values for rx1, ry1, rz1 are used to define the relative orientation of TCP at mid-point in Euler angle in respect to the center of the circle. It is a number in degree.<br><br>The input values for x2, y2, z2 are used to define the relative position of TCP at end-point from the center of the circle. It is a number in mm.<br><br>The input values for rx1, ry1, rz1 are used to define the relative orientation of TCP at end-point in Euler angle in respect to the center of the circle. It is a number in degree.<br><br>The input values for spd and acc are used to define velocity and acceleration accordingly. The spd and acc should be a number between 0 and 1. Smaller number represents slower. When the input value is -1, the joint moves with the default value.<br><br>This command will be ignored if the previous command is not finished yet. |
| Example | "movecircle threepoints intended 0.4, 0.1, 100.0, 100.0, 300.0, 0.0, 90.0, 0.0, 200.0, 200.0, 200.0, 0.0, 90.0, 45.0"<br>"movecircle threepoints constant 0.4, 0.1, 100.0, 100.0, 300.0, 0.0, 90.0, 0.0, 200.0, 200.0, 200.0, 0.0, 90.0, 45.0"<br>"movecircle threepoints radial 0.4, 0.1, 100.0, 100.0, 300.0, 0.0, 90.0, 0.0, 200.0, 200.0, 200.0, 0.0, 90.0, 45.0" |

| Command | movecircle(axis mode) |
|---|---|
| Script | movecircle axis orientation_option spd, acc, rot_angle, cx, cy, cz, ax, ay, az |
| Descript. | This command generates the circular motion of TCP using axes of rotation defined.<br><br>Three options determining the orientation of TCP in drawing a circle are available in orientation_option.<br>With 'intended' or 'constant', TCP keeps the current orientation during the circular motion.<br>With 'radial', the orientation of TCP changes in a way of the tangent direction to the center of the circle.<br><br>The input values for cx, cy, cz are used to define the position of axes of rotation (the center position of the circle). It is a number in mm.<br><br>The values for ax, ay, az are used to define the orientation of axes of rotation. It represents an unit vector.<br><br>The input value for rot_angle is used to define the amount of angle to rotate. It is a number in degree.<br><br>The input values for spd and acc are used to define velocity and acceleration accordingly. The spd and acc should be a number between 0 and 1. Smaller number represents slower. When the input value is -1, the joint moves with the default value.<br><br>This command will be ignored if the previous command is not finished yet. |
| Example | "movecircle axis constant 0.4, 0.1, 180.0, 200.0, 200.0, 200.0, 1.0, 0.0, 0.0"<br>"movecircle axis radial 0.4, 0.1, 180.0, 200.0, 200.0, 200.0, 1.0, 0.0, 0.0" |

## blend_jnt

| Command | blend_jnt |
|---|---|
| Script | blend_jnt clear_pt |
| Descript. | This command delete all desired joint values previously defined in the joint blending sequence.<br><br>This command should be used at the beginning of blend_jnt programming. |
| Example | "blend_jnt clear_pt" |

| Command | blend_jnt |
|---|---|
| Script | blend_jnt add_pt spd, acc, joint1, joint2, joint3, joint4, joint5, joint6 |
| Descript. | This command adds a desired joint value to the joint blending sequence.<br><br>The input values for joint1 to joint6 in the command denotes base, shoulder, elbow, wrist1, wrist2 and wrist3 accordingly. Each joint value represents the desired angle to go. The desired angle should be an absolute angle in degree.<br><br>The input values for spd and acc are used to define velocity and acceleration accordingly. The spd and acc should be a number between 0 and 1. Smaller number represents slower. When |

| | |
|---|---|
| | the input value is -1, the joint moves with the default value.<br><br>The speed and acceleration of the motion are defined by spd and acc in the last command. |
| Example | "blend_jnt add_pt 0.4, 0.1, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0" |

| | |
|---|---|
| Command | blend_jnt |
| Script | blend_jnt move_pt |
| Descript. | This command runs the joint blending motion.<br><br>Each joint follows the angles defined in the joint blending sequence. |
| Example | "blend_jnt move_pt" |

### blend_tcp

| | |
|---|---|
| Command | blend_tcp |
| Script | blend_tcp clear_pt |
| Descript. | This command delete all desired TCP values previously defined in the TCP blending sequence.<br><br>This command should be used at the beginning of blend_tcp programming. |
| Example | "blend_tcp clear_pt" |

| | |
|---|---|
| Command | blend_tcp |
| Script | blend_tcp add_pt spd, acc, radius, x, y, z, rx, ry, rz |
| Descript. | This command adds a desired TCP value to the TCP blending sequence.<br><br>The input value for radius determines the smoothness of blending. The value is in mm. Arithmetically it is the distance from the straight line between the first and third points to the second point. Thus, when it is set to 0, the blending becomes maximized and the robot skips the second point.<br><br>The input values for x, y, z are used to define the desired position to go. The values should be a number in mm.<br><br>The input values for rx, ry, rz are used to define the desired orientation to go. It is represented as roll, pitch and yaw in Euler angle, accordingly. The values should be a number in degree.<br><br>The input values for spd and acc are used to define velocity and acceleration accordingly. The spd and acc should be a number between 0 and 1. Smaller number represents slower. When the input value is -1, the joint moves with the default value.<br><br>The speed and acceleration of the motion are defined by spd and acc in the last command. |
| Example | "blend_tcp add_pt 0.4, 0.1, 30.0, 100.0, 100.0, 300.0, 0.0, 90.0, 0.0" |

| | |
|---|---|
| Command | blend_tcp |
| Script | blend_tcp move_pt |

| | |
|---|---|
| Descript. | This command runs the TCP blending motion.<br><br>TCP follows the positions and orientations of TCP defined in the TCP blending sequence. |
| Example | "blend_tcp move_pt" |

The following commands are commands to control the output values of the digital and analog ports of switchboards and tool flanges.

There are three commands.

1) digital_out
2) analog_out
3) tool_out

**digital_out**

| | |
|---|---|
| Script | digital_out d0, d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15 |
| Descript. | This command generates a signal through the digital output port.<br><br>The input values for d0 to d15 are used to activate the port. The number should be 0 or 1. 0 and 1 mean off and on, accordingly.<br><br>-1 can be used other than 0 or 1. In this case, the port with -1 keeps the previous status. |
| Example | "digital_out 1, 1, 1, 1, 0, 0, 0, 0, -1, -1, -1, -1, -1, -1, -1, -1" |

**analog_out**

| | |
|---|---|
| Script | analog_out a0, a1, a2, a3 |
| Descript. | This command generates a signal through the analog output port.<br><br>The input values for a0 to a3 are the output voltage of the port. The voltage should be a number between 0 and 10.<br><br>-1 can be used other than a number between 0 and 10. In this case, the port with -1 keeps the previous voltage. |
| Example | "analog_out 5.0, 5.0, -1, -1" |

**tool_out**

| | |
|---|---|
| Script | tool_out volt, d0, d1 |
| Descript. | This command sets the voltage and corresponding digital output ports at the tool flange.<br><br>The input value for volt is used to set the voltage to generate. The value should be 0, 12 or 24. Any number other than that will be ignored.<br><br>-1 can be used to keep the voltage previously defined.<br><br>The input values for d0 to d1 are used to activate the port. The number should be 0 or 1. 0 and 1 mean off and on, accordingly.<br><br>-1 can be used other than 0 or 1. In this case, the port with -1 keeps the previous status. |
| Example | "tool_out 12, 1, 0" |

The following commands are for initialization, termination, operation mode change, and speed change.

1) mc
2) shutdown
3) pgmode
4) sdw

**mc**

| | |
|---|---|
| Script | mc jall init |
| Descript. | This command starts initialization process. |
| Example | "mc jall init" |

**shutdown**

| | |
|---|---|
| Script | shutdown |
| Descript. | This command terminates the robot operation and turns off the power. |
| Example | "shutdown" |

**pgmode**

| | |
|---|---|
| Script | pgmode mode_type |
| Descript. | This command changes the mode between real and simulation modes.<br><br>The input values for mode_type should be "real" or "simulation".<br>In "real", the robot moves when commanded.<br>In "simulation", the robot does not moves but the internal reference values changes.<br><br>The default is "simulation". |
| Example | "pgmode real"<br>"pgmode simulation" |

**sdw (shared data write)**

| | |
|---|---|
| Script | sdw default_speed spd |
| Descript. | This command set the speed of the motion for overall program.<br><br>The input value for spd is a number between 0 and 1. Smaller value means slower. When the value is 0, the robot does not move even if a command is executed. In this case, the reference value does not change either.<br><br>When the pendent is connected to the robot while script programming is running, the speed can be adjusted via the pendent. Robot always follows the speed at the last command. |
| Example | "sdw default_speed 0.5" |

The last command explained is the task script.

**task**

| Script | task load work_file_name |
|---|---|
| Descript. | This command loads a work file previously programmed.<br><br>The format of the work file is ".wsl". The input value for work_file_name is the path and file name without ".wsl"<br><br>If the file is saved via the pendent, the file can be loaded without connecting to the pendent. |
| Example | "task load test_file" |

| Script | task play option |
|---|---|
| Descript. | This command runs the work file loaded.<br><br>The input value for option is blank or "once".<br><br>When option leaves empty, it runs the work file repeatably until the number of repeatation is met.<br>When "once" is set, it runs the work file once. |
| Example | "task play"<br>"task play once" |

| Script | task repeat num |
|---|---|
| Descript. | This command sets the number of repeatation for the work file.<br><br>The input value for num is the number of repeatation. The number should be an integer.<br>-1 can be used to run the work file unlimitedly.<br><br>The number of repeatation set by this command is maintained until power off. After rebooting the robot, this value is set by a number in the pendent. |
| Example | "task repeat 5"<br>"task repeat -1" |

| Script | task pause |
|---|---|
| Descript. | This command pauses the motion.<br><br>To resume the motion, use "task resume_a" command.<br><br>During pausing, the robot ignores all other commands. |
| Example | "task pause" |

| Script | task stop |
|---|---|
| Descript. | This command terminates the motion completely.<br><br>This command results in immediate stop of the motion. It is recommended using “task pause before this command to smoothly stop the motion. |
| Example | “task stop” |

| Script | task resume_a |
|---|---|
| Descript. | This command resumes the moiton paused by “task pause”, “alarm” or “debug”. |
| Example | “task resume_a” |

| Script | task resume_b |
|---|---|
| Descript. | This command resumes the motion paused by the collision. |
| Example | “task resume_b” |

In order to use external control, the external computer must be connected to the control box. The connection uses TCP / IP communication and the corresponding IP address can be set in the pendant. The result is displayed on the screen on control panel. Ports 5000 and 5001 open for external control. Port 5000 is a port for receiving commands, and port 5001 is a port for requesting and sending data indicating robot status. For convenience, port 5000 is called the command port and port 5001 is called the data port.

Users can send the script command described above to the command port. The command port has a filter for the first command, so if the start is not a script command as described above, such as "jointall", "movetcp", "mc", "pgmode", etc., the response is "The command is not allowed". If the command starts with a normal command and passes the input statement to the parser, the response is “The command was executed”.

When the command “reqdata” is sent to the data port, robot status information is sent to the data port in response. The format of the data is shown below.

| Header (4 Byte) | | | | Data (n Byte) |
|---|---|---|---|---|
| 0x24 | Size&0xFF | (Size>>8)&0xFF | 0x03 | Data |

The format of the data is shown below. Depending on the system version, the size of the data may be different. However, the order is consistent, please refer to the table below.

| Offset | Type | Description |
|---|---|---|
| 0 | Float | Task Time elapsed in second (reset at the beginning of the task) |
| 1 | Float | Reference angle of base joint in degree. |
| 2 | Float | Reference angle of shoulder joint in degree. |
| 3 | Float | Reference angle of elbow joint in degree. |
| 4 | Float | Reference angle of wrist1 joint in degree. |
| 5 | Float | Reference angle of wrist2 joint in degree. |
| 6 | Float | Reference angle of wrist3 joint in degree. |
| 7 | Float | Encoder angle of base joint in degree. |
| 8 | Float | Encoder angle of shoulder joint in degree. |
| 9 | Float | Encoder angle of elbow joint in degree. |
| 10 | Float | Encoder angle of wrist1 joint in degree. |
| 11 | Float | Encoder angle of wrist2 joint in degree. |
| 12 | Float | Encoder angle of wrist3 joint in degree. |
| 13 | Float | Current value of base joint in ampere |
| 14 | Float | Current value of shoulder joint in ampere. |
| 15 | Float | Current value of elbow joint in ampere. |
| 16 | Float | Current value of wrist1 joint in ampere. |
| 17 | Float | Current value of wrist2 joint in ampere. |
| 18 | Float | Current value of wrist3 joint in ampere. |
| 19 | Float | Reference position of TCP in X direction in mm. |
| 20 | Float | Reference position of TCP in Y direction in mm. |
| 21 | Float | Reference position of TCP in Z direction in mm. |
| 22 | Float | Reference orientation of TCP in Roll (RX) in degree. |
| 23 | Float | Reference orientation of TCP in Pitch (RY) in degree. |
| 24 | Float | Reference orientation of TCP in Yaw (RZ) in degree. |
| 25 | Float | Same as 19. |
| 26 | Float | Same as 20. |
| 27 | Float | Same as 21. |
| 28 | Float | Same as 22. |
| 29 | Float | Same as 23. |
| 30 | Float | Same as 24. |
| 31 | Float | Voltage at analog input port #0. |
| 32 | Float | Voltage at analog input port #1. |
| 33 | Float | Voltage at analog input port #2. |
| 34 | Float | Voltage at analog input port #3. |
| 35 | Float | Voltage at analog output port #0. |
| 36 | Float | Voltage at analog output port #1. |
| 37 | Float | Voltage at analog output port #2. |
| 38 | Float | Voltage at analog output port #3. |
| 39 | Int | On/Off status at digital input port #0 (on:1 / off:0). |
| 40 | Int | On/Off status at digital input port #1 (on:1 / off:0). |
| 41 | Int | On/Off status at digital input port #2 (on:1 / off:0). |
| 42 | Int | On/Off status at digital input port #3 (on:1 / off:0). |
| 43 | Int | On/Off status at digital input port #4 (on:1 / off:0). |
| 44 | Int | On/Off status at digital input port #5 (on:1 / off:0). |
| 45 | Int | On/Off status at digital input port #6 (on:1 / off:0). |
| 46 | Int | On/Off status at digital input port #7 (on:1 / off:0). |
| 47 | Int | On/Off status at digital input port #8 (on:1 / off:0). |
| 48 | Int | On/Off status at digital input port #9 (on:1 / off:0). |

| | | |
|---|---|---|
| 49 | Int | On/Off status at digital input port #10 (on:1 / off:0). |
| 50 | Int | On/Off status at digital input port #11 (on:1 / off:0). |
| 51 | Int | On/Off status at digital input port #12 (on:1 / off:0). |
| 52 | Int | On/Off status at digital input port #3 (on:1 / off:0). |
| 53 | Int | On/Off status at digital input port #14 (on:1 / off:0). |
| 54 | Int | On/Off status at digital input port #15 (on:1 / off:0). |
| 55 | Int | On/Off status at digital output port #0 (on:1 / off:0). |
| 56 | Int | On/Off status at digital output port #1 (on:1 / off:0). |
| 57 | Int | On/Off status at digital output port #2 (on:1 / off:0). |
| 58 | Int | On/Off status at digital output port #3 (on:1 / off:0). |
| 59 | Int | On/Off status at digital output port #4 (on:1 / off:0). |
| 60 | Int | On/Off status at digital output port #5 (on:1 / off:0). |
| 61 | Int | On/Off status at digital output port #6 (on:1 / off:0). |
| 62 | Int | On/Off status at digital output port #7 (on:1 / off:0). |
| 63 | Int | On/Off status at digital output port #8 (on:1 / off:0). |
| 64 | Int | On/Off status at digital output port #9 (on:1 / off:0). |
| 65 | Int | On/Off status at digital output port #10 (on:1 / off:0). |
| 66 | Int | On/Off status at digital output port #11 (on:1 / off:0). |
| 67 | Int | On/Off status at digital output port #12 (on:1 / off:0). |
| 68 | Int | On/Off status at digital output port #13 (on:1 / off:0). |
| 69 | Int | On/Off status at digital output port #14 (on:1 / off:0). |
| 70 | Int | On/Off status at digital output port #15 (on:1 / off:0). |
| 71 | Float | Temperature of motor drive at base joint in Celsius. |
| 72 | Float | Temperature of motor drive at shoulder joint in Celsius. |
| 73 | Float | Temperature of motor drive at elbow joint in Celsius. |
| 74 | Float | Temperature of motor drive at wrist1 joint in Celsius. |
| 75 | Float | Temperature of motor drive at wrist2 joint in Celsius. |
| 76 | Float | Temperature of motor drive at wrist3 joint in Celsius. |
| 77 | Int | Location of program counter in the task (The location where Step command executes). |
| 78 | Int | Desired number of repetitions. |
| 79 | Int | Current action number of the task. |
| 80 | Int | Current number of repetitions. |
| 81 | Float | Task time elapsed in seconds (not reset at the beginning of the task) |
| 82 | Int | Task status (1: Idle, 2: Paused, 3: Run) |
| 83 | Float | Motion speed (0~1). |
| 84 | Float | Robot status (1: stopped, 3: in operation) |
| 85 | Float | Status of power in terms of LSB offset<br>0: 48V input<br>1: 48V output<br>2: 24V status<br>3: E-stop status<br>4: PC switch status<br>5: Motion controller status |
| 86 | Float | Not used |
| 87 | Float | Not used |
| 88 | Float | Not used |
| 89 | Float | Not used |
| 90 | Float | Not used |
| 91 | Float | Not used |
| 92 | Int | Status of motor controller at base joint. |
| 93 | Int | Status of motor controller at shoulder joint. |
| 94 | Int | Status of motor controller at elbow joint. |
| 95 | Int | Status of motor controller at wrist1 joint. |
| 96 | Int | Status of motor controller at wrist2 joint. |

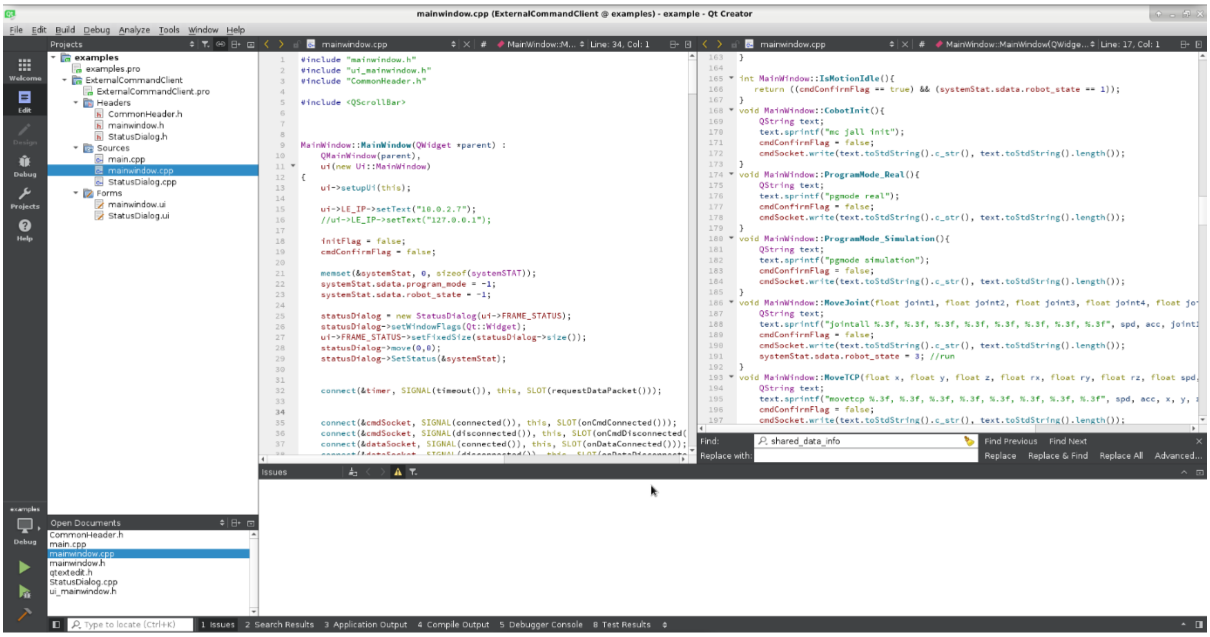
| | | |
|---|---|---|
| 97 | Int | Status of motor controller at wrist3 joint. |
| | | Status of motor controller in terms of LSB offset.<br>0: FET<br>1: Position control<br>2: Status of initialization<br>3: In control mode<br>4: Nonius error<br>5: Low battery<br>6: Calibration mode<br>7: Multiturn error<br>8: JAM error<br>9: Over Current error<br>10: Big error<br>11: Input error<br>12: FET drive error<br>13: Temperature error<br>14: Position error (Low)<br>15: Position error (High) |
| 98 | Int | On/Off collision detection (1:on, 0:off) |
| 99 | Int | On/Off teaching mode (1:on, 0:off) |
| 100 | Int | Operation mode (1:simulation mode, 0:real mode) |
| 101 | Int | Information of initialization process<br>0: Default<br>1: Voltage check<br>2: Device check<br>3: Position control start<br>4: Parameter check<br>5: Collision check<br>6: Initialization done |
| 102 | Int | Error codes in initialization<br>0: Initialization completed without error<br>1: SMPS error<br>2: E-Stop switch error<br>3: Power conversion error 1 (in control box)<br>4: Power conversion error 2 (in control box)<br>5: Connection error<br>6: Initialization error<br>7: Payload error<br>8: Tool flange connection error<br>9: Tool flange orientation error<br>10: Motor controller encoder error 1<br>11: Motor controller encoder error 2<br>12: Digital input 16/17 short error<br>13: 48V switch error<br>14: Teaching button error |
| 103 | Float | Voltage of analog signal port #0 at tool flange. |
| 104 | Float | Voltage of analog signal port #1 at tool flange. |
| 105 | Int | On/Off status of digital signal port #0 at tool flange (on:1 / off:0). |
| 106 | Int | On/Off status of digital signal port #1 at tool flange (on:1 / off:0). |
| 107 | Int | On/Off output status of digital signal port #0 at tool flange (on:1 / off:0). |
| 108 | Int | On/Off output status of digital signal port #1 at tool flange (on:1 / off:0) |
| 109 | Float | Voltage output at tool flange. |
| 110 | Int | Status of collision (detected:1) |
| 111 | Int | Status of device errors<br>0: No error<br>1: PVL error |

| | | |
|---|---|---|
| | | 2: CPU error<br>3: Big error<br>4: Input error<br>5: JAM error<br>6: Over current error<br>7: Joint angle error<br>8: Control mode error<br>9: Offset error between reference and encoder<br>10: Current error at upper level controller<br>11: Temperature error<br>12: Speed error in teaching |
| 112 | Int | Self collision (on:1 / off:0) |
| 113 | Int | Robot paused (paused:1) |
| 114 | Int | Status of motion errors<br>0: No error<br>1: TCP motion commanded when the robot is fully stretched out.<br>2: TCP command unreachable<br>3: Joint command crossed mechanical limit<br>4: TCP command singularity |
| 115 | Int | On/Off status of digital input port #16 (on:1 / off:0). |
| 116 | Int | On/Off status of digital input port #17 (on:1 / off:0). |
| 117 | Int | Inbox 0 Trap occurred |
| 118 | Int | Inbox 1 Trap occurred |
| 119 | Int | Inbox 0 check mode |
| 120 | Int | Inbox 1 check mode |

### E.2 Example Program Development Environment

This example has been tested on Debian 9.8 and Ubuntu 18.04. It may work on similar Linux systems. No separate kernel patch is required.

As an integrated development environment (IDE) for programming, use Qt version 5.8 (https://www.qt.io).





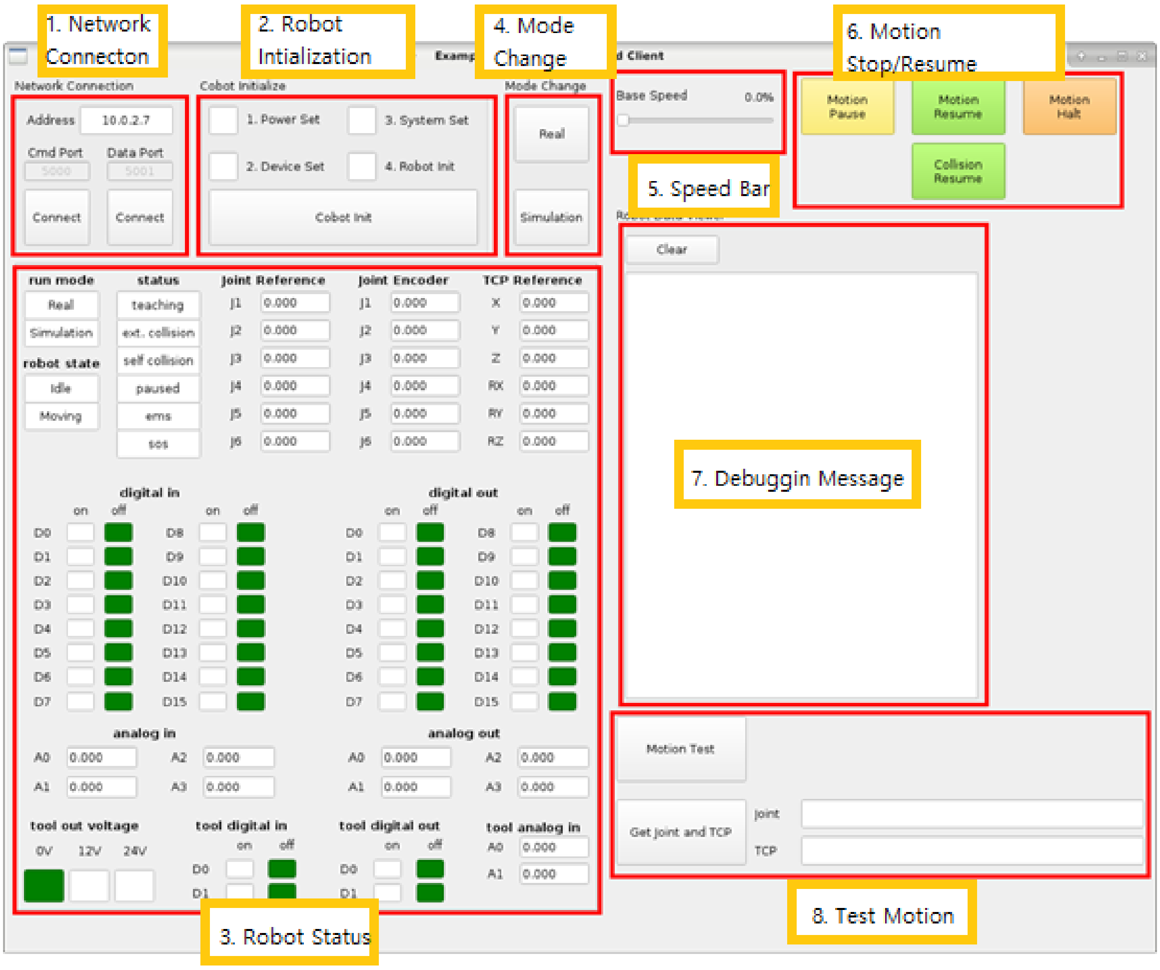
Warning:

Qt-based C++ examples, Visual Studio-based C# examples, and more. Sample programs can be obtained from the manufacturer or distributor.

## E.3 Programming Method

This example does not include all the functionality provided by the tablet user interface (UI). Only the information that is useful for monitoring by the user while moving the robot through external control is implemented.

The following image is the programming UI when the example program is executed




The function of each item is as follows.

1) Network Connection

Connect to the robot's main controller through the LAN port on the control box of the RB5 robot. The default IP address for external control is fixed at '10 .0.2.7 '. The server for receiving external control commands connects to port 5000, and the server for requesting and receiving robot status information connects to port 5001. There is a separate button for connecting each one. If the connection is successful, the word 'Connect' on the button is changed to 'Disconnect'. The reverse happens when the connection is lost.

2) Initializing the Robot

After connecting to the robot's main controller press the button marked ‘Cobot Init’ to start the initialization process. Robots go through a series of processes called ‘Power Set’, ‘Device Set’, ‘System Set’ and ‘Robot Init’. As robot’s initialization process continues, the white edit box in front of each course turns yellow. Processes that have been completed turn green and processes that have not been performed remain red. When all four boxes turn green, the robot's initialization process is complete and ready for use.

3) Robot Status

The status of the robot can be known from the data received from the main program in the control box. This data is sent to the main program in response to a request for "reqdata" on port 5001. The format of the data is passed in the form of the 'systemSTAT' structure in 'CommonHeader.h'.

run mode: Displays the operation mode of the robot. There are real mode and simulation mode. In real mode, motion commands are actually applied to the robot and the robot moves. In simulation mode, the motion is performed but the command is not sent to the robot. The teaching pendant will show the translucent robot moving. The robot operation mode is represented by the value of the 'program_mode' variable in 'systemSTAT'. A value of 0 for this variable is real mode, and a value of 1 for simulation mode.

robot state: Indicates whether the robot is currently moving or in a state capable of receiving motion commands. The robot state can be known from the value of the 'robot_state' variable of 'systemSTAT'. If this value is 1, the idle state can receive motion commands. If the value is 3, the robot is moving. Motion commands are ignored while the robot is in motion. If the value is 2, the robot is stopped due to unspecified reasons or stopped by the pause command. In this case, it is displayed as paused in the ‘status’ column.

status: Displays current robot special operation status or abnormal status. ‘Teaching’ if teaching directly, ‘ext. ' collision ',' self-collision 'if it is just before self-collision during operation,' paused 'if stopped by pause command,' ems' if input without solution in robot control algorithm comes in, power problem or robot control problem Will change the color of the 'sos' edit window. This is displayed by referring to the values of 'op_stat_collision_occur', 'op_stat_sos_flag', 'op_stat_self_collision', 'op_stat_soft_estop_occur', 'op_stat_ems_flag' and 'robot_state' in 'systemSTAT'.

joint reference: Displays the reference input value for each joint (in degrees).
joint encoder: Displays the current encoder value of each joint (in degrees).
TCP reference: Displays the reference position value of TCP (in mm and degree).
digital in: Displays the digital input value of the control box.
digital out: Displays the digital output value of the control box.
analog in: Displays the analog input value of the control box (in voltage).
analog out: Displays the analog output value of the control box (in voltage).
tool out voltage: Displays the output voltage of the currently set tool flange board (0V, 12V or 24V).
tool digital in: Displays the digital input value of the tool flange board.
tool digital out: Displays the digital output value of the tool flange board.
tool analog in: Displays the analog input value of the tool flange board

4) Mode Change

The robot can have two modes of operation: simulation mode and real mode. In simulation mode, the robot does not move but the value of the input reference can be changed. In real mode, the robot actually moves in response to user input. To change the robot's operation mode by pressing the button marked 'Real' and 'Simulation'. Immediately after the initialization process, the robot is in simulation mode.

5) Speed Change

Adjust the overall speed of robot motion. Users can move the slider bar between 0% and 100%. This speed is multiplied by the speed given to the robot's motion command.

6) Stop and Resume Motion

Press the 'Motion Pause' button to pause and press 'Motion Halt' to stop the motion completely. In the case of 'Motion Halt', the robot stops abruptly, so it is recommended to use pause first in order to use the robot stably. If the robot is in the paused state, it will not be executed even if another robot is given a motion command. In order to stop the current operation and perform another operation, must finish the current operation completely through the 'Motion Halt' button after the 'Motion Pause' button.

Conversely, users can resume motion paused or stopped by external collision detection. Press the 'Motion Resume' button to resume a paused motion or press the 'Collision Resume' button to resume a motion stopped due to external collision detection.

7) Debugging Message Screen

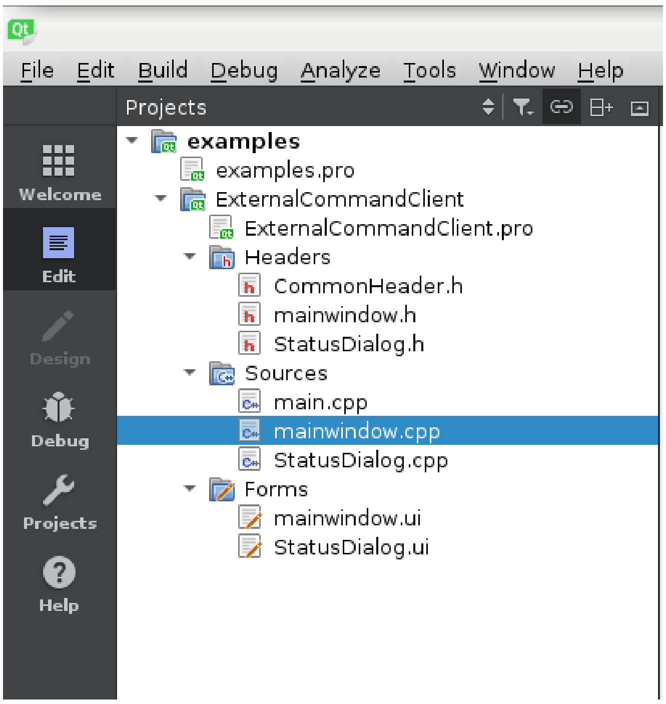
This is the window where users can view messages for debugging.

8) Test Motion

Press the ‘Motion Test’ button to perform three basic motions in sequence. Please consider the environment around the robot.

If the user presses the ‘Get Joint and TCP’ button, the reference angle and TCP value of the current robot's joint will be expressed using the ',' separator in the edit window next to it. It is helpful to copy this value when coding the robot motion sequence into the program.

This example is a single process example with a GUI. In Qt, users can easily place the GUI, generate events like button clicks, and associate them with user’s program. See ‘mainwindow.ui’.




The core contents of the example are included in ‘mainwindow.cpp’ and ‘mainwindow.h’. In ‘CommonHeader.h’, users can check the shape of the robot status data.

```
// Cobot Control API -------------------
// <CobotInit>
// : initialize Cobot
void CobotInit();

// <MoveJoint>
// : move to target posture in joint coordinate
// joint1~joint6 : target joint angle in deg unit
// spd : speed parameter (0~1: user define   or   -1: default setting)
// acc : acceleration parameter (0~1: user define   or   -1: default setting)
void MoveJoint(float joint1, float joint2, float joint3, float joint4, float joint5, flo

// <MoveTCP>
// : move to target posture in cartesian coordinate
// x, y, z : target TCP(tool center point) position in mm unit
// rx, ry, rz : target TCP orientation (Yaw-Pitch-Roll Euler angle) in degree unit
// spd : speed parameter (0~1: user define   or   -1: default setting)
// acc : acceleration parameter (0~1: user define   or   -1: default setting)
void MoveTCP(float x, float y, float z, float rx, float ry, float rz, float spd = -1, fl

// <ControlBoxDigitalOut>
// control digital out ports in control box
// d0~d15 : digital out value (0 or 1)
void ControlBoxDigitalOut(int d0, int d1, int d2, int d3, int d4, int d5, int d6, int d7

// <ControlBoxAnalogOut>
// control analog out ports in control box
// a0~a3 : analog out value in voltage unit (0~10)
void ControlBoxAnalogOut(float a0, float a1, float a2, float a3);

// <ToolOut>
// control digital out ports and voltage level in tool flange board
// volt : reference voltage of tool flange board in voltage unit(0, 12, 24)
// d0, d1 : digital out value (0 or 1)
void ToolOut(int volt, int d0, int d1);

// <ProgramMode_Real>
// change to 'real robot' mode -- robot will move
void ProgramMode_Real();

// <ProgramMode_Simulation>
// change to 'simulation' mode -- robot will not move except teaching
void ProgramMode_Simulation();

// <BaseSpeedChange>
// change base speed -- base speed will be multiplied to motion velocity
// spd : normalized base speed (0~1)
void BaseSpeedChange(float spd);

// <MotionPause>
// pause the current motion
void MotionPause();

// <MotionHalt>
// halt the current motion
// !! CAUTION : user would better escape the motion sequence
//            : if not, the next motion will be activated immediately
void MotionHalt();

// <MotionResume>
// resume the paused motion
void MotionResume();

// <CollisionResume>
// resume the motion which is paused due to external collision
void CollisionResume();
// ------------------------------------
```

The robot control commands that can be used by the user are specified in 'mainwindow.h' as above. The detailed description is as follows.

| | |
|---|---|
| Function | CobotInit(void) |
| Script | “mc jall init” |
| Descript. | This commande starts initialization process.<br><br>Progress in initialization is shown in ‘init_stat_info’ and ‘init_error’ inside ‘systemSTAT’. |

| | |
|---|---|
| Function | MoveJoint(float joint1, float joint2, float joint3, float joint4, float joint5, float joint6, float spd = -1, float acc = -1); |
| Script | “jointall spd, acc, joint1, joint2, joint3, joint4, joint5, joint6” |
| Descript. | This command moves joints to the desired angles in Joint Space<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | MoveTCP(float x, float y, float z, float rx, float ry, float rz, float spd = -1, float acc = -1); |
| Script | “movetcp spd, acc, x, y, z, rx, ry, rz” |
| Descript. | This command moves TCP to the given position and orientation in Cartesian Space.<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | MoveCircle_ThreePoint(int type, float x1, float y1, float z1, float rx1, float ry1, float rz1, float x2, float y2, float z2, float rx2, float ry2, float rz2, float spd = -1, float acc = -1); |
| Script | "movecircle threepoints intended spd, acc, x1, y1, z1, rx1, ry1, rz1, x2, y2, z2, rx2, ry2, rz2"<br><br>"movecircle threepoints constant spd, acc, x1, y1, z1, rx1, ry1, rz1, x2, y2, z2, rx2, ry2, rz2"<br><br>"movecircle threepoints radial spd, acc, x1, y1, z1, rx1, ry1, rz1, x2, y2, z2, rx2, ry2, rz2" |
| Descript. | This command generates the circular motion of TCP using three points.<br><br>type=0 : 'intended' in script programming<br><br>type=1 : 'constant' in script programming<br><br>type=2 : 'radial' in script programming<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | MoveCircle_Axis(int type, float cx, float cy, float cz, float ax, float ay, float az, float rot_angle, float spd = -1, float acc = -1); |
| Script | "movecircle axis intended spd, acc, rot_angle, cx, cy, cz, ax, ay, az"<br><br>"movecircle axis constant spd, acc, rot_angle, cx, cy, cz, ax, ay, az"<br><br>"movecircle axis radial spd, acc, rot_angle, cx, cy, cz, ax, ay, az" |
| Descript. | This command generates the circular motion of TCP using axes of rotation defined.<br><br>type=0 : 'intended' in script programming<br><br>type=1 : 'constant' in script programming<br><br>type=2 : 'radial' in script programming<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | MoveJointBlend_Clear(void); |
| Script | “blend_jnt clear_pt” |
| Descript. | This command delete all desired joint values previously defined in the joint blending sequence.<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | MoveJointBlend_AddPoint(float joint1, float joint2, float joint3, float joint4, float joint5, float joint6, float spd = -1, float acc = -1); |
| Script | “blend_jnt add_pt spd, acc, joint1, joint2, joint3, joint4, joint5, joint6” |
| Descript. | This command adds a desired joint value to the joint blending sequence.<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | MoveJointBlend_MovePoint(void); |
| Script | “blend_jnt move_pt” |
| Descript. | This command runs the joint blending motion.<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | MoveTCPBlend_Clear(void); |
| Script | “blend_tcp clear_pt” |
| Descript. | This command delete all desired TCP values previously defined in the TCP blending sequence.<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | MoveTCPBlend_AddPoint(float radius, float x, float y, float z, float rx, float ry, float rz, float spd = -1, float acc = -1); |
| Script | "blend_tcp add_pt spd, acc, radius, x, y, z, rx, ry, rz" |
| Descript. | This command adds a desired TCP value to the TCP blending sequence.<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | MoveTCPBlend_MovePoint(void); |
| Script | "blend_tcp move_pt" |
| Descript. | This command runs the TCP blending motion.<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | ControlBoxDigitalOut(int d0, int d1, int d2, int d3, int d4, int d5, int d6, int d7, int d8, int d9, int d10, int d11, int d12, int d13, int d14, int d15) |
| Script | "digital_out d0, d1, d2, d3, d4,d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15" |
| Descript. | This command generates a signal through the digital output port.<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | ControlBoxAnalogOut(float a0, float a1, float a2, float a3) |
| Script | "analog_out a0, a1, a2, a3" |
| Descript. | This command generates a signal through the analog output port.<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | ToolOut(int volt, int d0, int d1) |
| Script | "tool_out volt, d0, d1" |
| Descript. | This command sets the voltage and corresponding digital output ports at the tool flange.<br><br>Please refer to script programming. |

| | |
|---|---|
| Function | ProgramMode_Real(void) |
| Script | "pgmode real" |
| Descript. | This command changes operation mode to Real Mode. |

| | |
|---|---|
| Function | ProgramMode_Simulation(void) |
| Script | "pgmode simulation" |
| Descript. | This command changes operation mode to Simulation Mode |

| | |
|---|---|
| Function | BaseSpeedChange(float spd) |
| Script | “sdw default_speed spd” |
| Descript. | This command set the speed of the motion for overall program.<br><br>Please refert to script programming. |

| | | |
|---|---|---|
| Function | 함수 | MotionPause(void) |
| Script | 스크립트 | “task pause” |
| Descript. | 설명 | This command pauses the motion.<br><br>Please refer to script programming.<br><br>To execute the other commands, the robot should be resumed by MotionResume or terminated by MotionHalt. |

| | |
|---|---|
| Function | MotionResume(void) |
| Script | “task resume_a” |
| Descript. | This command resumes the motion paused by MotionPause.<br><br>The command does not resume the motion paused by a collision. |

| | |
|---|---|
| Function | CollisionResume(void) |
| Script | “task resume_b” |
| Descript. | This command resumes the motion paused by a collision.<br><br>This command does not resume the motion paused by MotionPause. |

| | |
|---|---|
| Function | MotionHalt(void) |
| Script | “task stop” |
| Descript. | This command terminates the motion completely. |

```
int test_flag = false;
int test_state = 0;
void MainWindow::onLogic(){

    // Motion Sequence ==============================================
    if(test_flag == true){
        switch(test_state){
        case 0:
            if(IsMotionIdle()){
                print("test_state 0\n");
                MoveJoint(0,0,0,0,0,0);
                test_state = 1;
            }
            break;

        case 1:
            if(IsMotionIdle()){
                print("test_state 1\n");
                MoveJoint(50,50,50,50,50,50);
                test_state = 2;
            }
            break;
        case 2:
            if(IsMotionIdle()){
                print("test_state 2\n");
                MoveTCP(400,400,400,150,30,-100);
                test_state = 3;
            }
            break;
        case 3:
            if(IsMotionIdle()){
                print("test_state 3\n");
                test_flag = false;
                test_state = 0;
            }
            break;
        }
    }
    // ==============================================================

}
```

The code above is an action code that performs two joint control motions and one TCP control motion sequentially. There is a 'test_flag' which decides whether or not to execute the motion sequence, and if this value is true, it moves sequentially from the previous motion to the next motion according to the 'test_state' value indicating the progress of the sequence.

At this point, check whether the previous motion is over or not, and there is an 'IsMotionIdle' function to make it easier. This function sends instructions to the robot's main controller.

```
int MainWindow::IsMotionIdle(){
    return ((cmdConfirmFlag == true) && (systemStat.sdata.robot_state == 1));
}
```

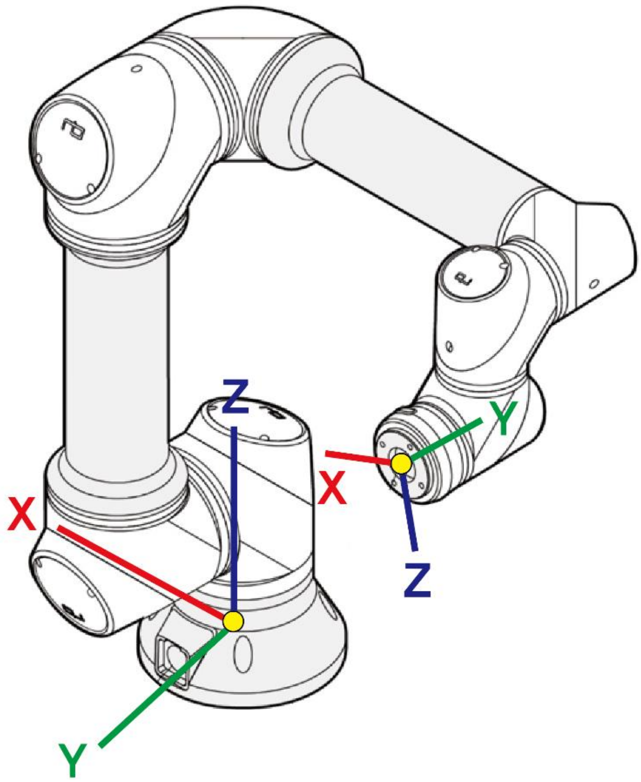
The ‘onLogic’ function, which contains an action sequence, is linked to a timer provided by Qt. In this example, it is set at 10ms intervals, and this function is executed every 10ms.

```

void MainWindow::on_BTN_TEST_clicked()
{
    test_state = 0;
    test_flag = true;
}

```

Executing robot motion is simple. Set the 'test_state' value representing the motion sequence state to 0, the starting point of the motion, and set the 'test_flag' value to perform the motion to move the robot.

The behavior shown in the example code provided is very simple, but free from structure constraints. Users can build own application based on this example code, or build a separate application by understanding only the script.

# Appendix F. Coordinate System




- Global Coordinate (Base coordinate)

  Once the robot is fixed with the coordinate system fixed to the base of the robot, the global coordinate system is also fixed.
  The center of the base surface is the origin. Set the robot direction to the + Z direction from the origin and the connector direction to the + Y direction from the origin.

- Local Coordinate (Tool coordinate)

  Coordinate system fixed to TCP (Tool Center Point) of the robot, the direction of the axis changes in real time by setting or moving the TCP offset. Set TCP as the origin and set the robot direction from the origin to the + Y direction and the teach button direction from the origin to the + Z direction.

# APPENDIX G. STOPPING TIME/DISTANCE

In the RB Series of collaborative robots, the time and distance between the robots stop and the distance are generated by the safety monitoring function.

The graph below shows the stop time and stop distance for stop category 1 for Joint 0 (Base axis), Joint 1 (Shoulder axis), and Joint 2 (Elbow axis).

## Warning

Depending on the situation, the actual stop motion may differ from the results below. Joint 0 is the result of horizontal movement, and Joint 1 and 2 are the result of vertical downward movement. For the length of the arm, the maximum length is applied.

| RB5-850E Series Base (Joint 0) | | |
|---|---|---|
| | Stop Distance (mm) | Stop Time (sec) |
| Test 1 | 183.38 | 0.19 |
| Test 2 | 160.1 | 0.24 |
| Test 3 | 191.03 | 0.24 |
| Maximum | 191.03 | 0.24 |
| Average | 178.17 | 0.22 |
| Condition | Max. Reach / Max. Velocity / Horizontal Motion | |

| RB5-850E Series Shoulder (Joint 1) | | |
|---|---|---|
| | Stop Distance (mm) | Stop Time (sec) |
| Test 1 | 183.71 | 0.13 |
| Test 2 | 177.53 | 0.13 |
| Test 3 | 183.39 | 0.21 |
| Maximum | 183.71 | 0.21 |
| Average | 181.54 | 0.16 |
| Condition | Max. Reach / Max. Velocity / Vertical Downward Motion | |

| RB5-850E Series Elbow (Joint 2) | | |
|---|---|---|
| | Stop Distance (mm) | Stop Time (sec) |
| Test 1 | 90.935 | 0.14 |
| Test 2 | 94.809 | 0.14 |
| Test 3 | 81.987 | 0.14 |
| Maximum | 94.809 | 0.14 |
| Average | 89.24 | 0.14 |
| Condition | Max. Reach / Max. Velocity / Vertical Downward Motion | |

# APPENDIX H. NAMEPLATE

The nameplate of the robot is divided into the robot arm and the control box as shown below.

[Robot Arm]

RB5-850E Series

RAINBOW ROBOTICS
34122, 10-19, Expo-ro 339beon-gil, Yuseong-gu, Daejeon, Korea
TEL : +82.42.719.8070
FAX : +82.42.719.8071

| | | | |
|---|---|---|---|
| **Designation** | Robot Arm | **Max. Payload** | 5 kg |
| **Model No.** | RB5-850E | **Mfg. Date** | 2020-01 |
| **Serial No.** | R585E-2001001 | | |
| **Reach** | 850 mm | | |
| **Weight** | 21.5 kg | | |
| **Supply Power** | 48 VDC | | |

RAINBOW ROBOTICS
34122, 10-19, Expo-ro 339beon-gil, Yuseong-gu, Daejeon, Korea
TEL : +82.42.719.8070
FAX : +82.42.719.8071

| | | | |
|---|---|---|---|
| **Designation** | Robot Arm | **Max. Payload** | 5 kg |
| **Model No.** | RB5-850EA1 | **Max. Pressure** | 10 bar |
| **Serial No.** | R585E-2001001 | **Mfg. Date** | 2020-01 |
| **Reach** | 850 mm | | |
| **Weight** | 21.5 kg | | |
| **Supply Power** | 48 VDC | | |

RAINBOW ROBOTICS
34122, 10-19, Expo-ro 339beon-gil, Yuseong-gu, Daejeon, Korea
TEL : +82.42.719.8070
FAX : +82.42.719.8071

| | | | |
|---|---|---|---|
| **Designation** | Robot Arm | **Max. Payload** | 5 kg |
| **Model No.** | RB5-850EA2 | **Max. Pressure** | 10 bar |
| **Serial No.** | R585E-2001001 | **Mfg. Date** | 2020-01 |
| **Reach** | 850 mm | | |
| **Weight** | 21.5 kg | | |
| **Supply Power** | 48 VDC | | |

RB3-1200E Series

RAINBOW ROBOTICS
34122, 10-19, Expo-ro 339beon-gil, Yuseong-gu, Daejeon, Korea
TEL : +82.42.719.8070
FAX : +82.42.719.8071

| | | | |
|---|---|---|---|
| **Designation** | Robot Arm | **Max. Payload** | 3 kg |
| **Model No.** | RB3-1200E | **Mfg. Date** | 2020-01 |
| **Serial No.** | R312E-2001001 | | |
| **Reach** | 1200 mm | | |
| **Weight** | 22 kg | | |
| **Supply Power** | 48 VDC | | |

RAINBOW ROBOTICS
34122, 10-19, Expo-ro 339beon-gil, Yuseong-gu, Daejeon, Korea
TEL : +82.42.719.8070
FAX : +82.42.719.8071

| | | | |
|---|---|---|---|
| **Designation** | Robot Arm | **Max. Payload** | 3 kg |
| **Model No.** | RB3-1200EA1 | **Max. Pressure** | 10 bar |
| **Serial No.** | R312E-2001001 | **Mfg. Date** | 2020-01 |
| **Reach** | 1200 mm | | |
| **Weight** | 22 kg | | |
| **Supply Power** | 48 VDC | | |

RAINBOW ROBOTICS
34122, 10-19, Expo-ro 339beon-gil, Yuseong-gu, Daejeon, Korea
TEL : +82.42.719.8070
FAX : +82.42.719.8071

| | | | |
|---|---|---|---|
| **Designation** | Robot Arm | **Max. Payload** | 3 kg |
| **Model No.** | RB3-1200EA2 | **Max. Pressure** | 10 bar |
| **Serial No.** | R312E-2001001 | **Mfg. Date** | 2020-01 |
| **Reach** | 1200 mm | | |
| **Weight** | 22 kg | | |
| **Supply Power** | 48 VDC | | |

RB10-1300E Series

| | | | |
|---|---|---|---|
| **RAINBOW ROBOTICS** | | | |
| 34122, 10-19, Expo-ro 339beon-gil, Yuseong-gu, Daejeon, Korea | | | |
| TEL : +82.42.719.8070 | | | |
| FAX : +82.42.719.8071 | | | |
| **Designation** | Robot Arm | **Max. Payload** | 10 kg |
| **Model No.** | RB10-1300E | **Mfg. Date** | 2020-01 |
| **Serial No.** | R1013E-2001001 | | |
| **Reach** | 1300 mm | | |
| **Weight** | 33 kg | | |
| **Supply Power** | 48 VDC | | |

| | | | |
|---|---|---|---|
| **RAINBOW ROBOTICS** | | | |
| 34122, 10-19, Expo-ro 339beon-gil, Yuseong-gu, Daejeon, Korea | | | |
| TEL : +82.42.719.8070 | | | |
| FAX : +82.42.719.8071 | | | |
| **Designation** | Robot Arm | **Max. Payload** | 10 kg |
| **Model No.** | RB10-1300EA1 | **Max. Pressure** | 10 bar |
| **Serial No.** | R1013E-2001001 | **Mfg. Date** | 2020-01 |
| **Reach** | 1300 mm | | |
| **Weight** | 33 kg | | |
| **Supply Power** | 48 VDC | | |

| | | | |
|---|---|---|---|
| **RAINBOW ROBOTICS** | | | |
| 34122, 10-19, Expo-ro 339beon-gil, Yuseong-gu, Daejeon, Korea | | | |
| TEL : +82.42.719.8070 | | | |
| FAX : +82.42.719.8071 | | | |
| **Designation** | Robot Arm | **Max. Payload** | 10 kg |
| **Model No.** | RB10-1300EA2 | **Max. Pressure** | 10 bar |
| **Serial No.** | R1013E-2001001 | **Mfg. Date** | 2020-01 |
| **Reach** | 1300 mm | | |
| **Weight** | 33 kg | | |
| **Supply Power** | 48 VDC | | |

## [Control Box]

RB5-850E Series, RB3-1200E Series: Stand type(CB04)

| | | | |
|---|---|---|---|
| **RAINBOW ROBOTICS** | | | |
| 34122, 10-19, Expo-ro 339beon-gil, Yuseong-gu, Daejeon, Korea | | | |
| TEL : +82.42.719.8070 | | | |
| FAX : +82.42.719.8071 | | | |
| **Designation** | Control Box | **Rated Current** | 15 A |
| **Model No.** | CB04 | **Rated Frequency** | 50~60 Hz |
| **Serial No.** | C04-5-2001001 | **S.C.C.R** | 2.5 kA |
| **Drawing No.** | RB-ES-020 | **Weight** | 17.3kg |
| **Mfg. Year & Month** | 2020-01 | | |
| **Rated Power Supply** | Single phase 100-240 VAC | | |

# APPENDIX I. MODBUS TCP SERVER

- **Warning**
  This manual describes the Modbus server (slave controller). See Section 6 for a description of the Modbus client features.

## 1. Overview

RB's Modbus TCP server (slave controller) is fixed at port number 502. The IP address changes depending on the network settings through the UI. (The initial IP address is 10.0.2.7.)
RB's Modbus server allows the connection of multiple clients and executes the following operation commands.

| | Function Code | Function Name |
|---|---|---|
| Bit Address | 2 | Read Discrete Inputs |
| | 1 | Read Coils |
| | 5 | Write Single Coil |
| | 15 | Write Multiple Coils |
| 16-bit (Word) Address | 4 | Read Input Registers |
| | 3 | Read Multiple Holding Registers |
| | 6 | Write Single Holding Register |
| | 16 | Write Multiple Holding Registers |

## 2. Exception Code

The following error message is returned when accessing the wrong address, incorrect range of values, or invalid command sent.

| Exception Code | Exception Name |
|---|---|
| 1 | Illegal Function |
| 2 | Illegal Data Address |
| 3 | Illegal Value |

3. Bit Address Map

| Bit Address | | | |
|---|---|---|---|
| Address | Function | Read | Write |
| 0 | Box digital input 0 | o | x |
| 1 | Box digital input 1 | o | x |
| 2 | Box digital input 2 | o | x |
| 3 | Box digital input 3 | o | x |
| 4 | Box digital input 4 | o | x |
| 5 | Box digital input 5 | o | x |
| 6 | Box digital input 6 | o | x |
| 7 | Box digital input 7 | o | x |
| 8 | Box digital input 8 | o | x |
| 9 | Box digital input 9 | o | x |
| 10 | Box digital input 10 | o | x |
| 11 | Box digital input 11 | o | x |
| 12 | Box digital input 12 | o | x |
| 13 | Box digital input 13 | o | x |
| 14 | Box digital input 14 | o | x |
| 15 | Box digital input 15 | o | x |
| 16 | Box digital output 0 | o | o |
| 17 | Box digital output 1 | o | o |
| 18 | Box digital output 2 | o | o |
| 19 | Box digital output 3 | o | o |
| 20 | Box digital output 4 | o | o |
| 21 | Box digital output 5 | o | o |
| 22 | Box digital output 6 | o | o |
| 23 | Box digital output 7 | o | o |
| 24 | Box digital output 8 | o | o |
| 25 | Box digital output 9 | o | o |
| 26 | Box digital output 10 | o | o |
| 27 | Box digital output 11 | o | o |
| 28 | Box digital output 12 | o | o |
| 29 | Box digital output 13 | o | o |
| 30 | Box digital output 14 | o | o |
| 31 | Box digital output 15 | o | o |
| 32 | Tool digital input 0 | o | x |
| 33 | Tool digital input 1 | o | x |
| 34 | Tool digital output 0 | o | o |
| 35 | Tool digital output 1 | o | o |

## 4. Word(16 bit) Address Map

| Word Address | | | | |
|---|---|---|---|---|
| Address | Function | Read | Write | Comments |
| 0 | Box digital input 0~15 | o | x | [BBBB BBBB BBBB BBBB] |
| 1 | Box digital output 0~15 | o | o | [BBBB BBBB BBBB BBBB] |
| 2 | Box analog input 0 | o | x | 1mV unit |
| 3 | Box analog input 1 | o | x | 1mV unit |
| 4 | Box analog input 2 | o | x | 1mV unit |
| 5 | Box analog input 3 | o | x | 1mV unit |
| 6 | Box analog output 0 | o | o | 1mV unit |
| 7 | Box analog output 1 | o | o | 1mV unit |
| 8 | Box analog output 2 | o | o | 1mV unit |
| 9 | Box analog output 3 | o | o | 1mV unit |
| 10 | Extend digital input 0~15 | o | x | [BBBB BBBB BBBB BBBB] |
| 11 | Extend digital output 0~15 | o | x | [BBBB BBBB BBBB BBBB] |
| 12 | Extend analog input 0 | o | x | 1mV unit |
| 13 | Extend analog input 1 | o | x | 1mV unit |
| 14 | Extend analog input 2 | o | x | 1mV unit |
| 15 | Extend analog input 3 | o | x | 1mV unit |
| 16 | Extend analog output 0 | o | o | 1mV unit |
| 17 | Extend analog output 1 | o | o | 1mV unit |
| 18 | Extend analog output 2 | o | o | 1mV unit |
| 19 | Extend analog output 3 | o | o | 1mV unit |
| 20~29 | Reserved (Box IO) | | | |
| 30 | Tool output voltage | o | o | 0, 12, 24 |
| 31 | Tool digital input 0~1 | o | x | [TTxx xxxx xxxx xxxx] |
| 32 | Tool digital output 0~1 | o | o | [TTxx xxxx xxxx xxxx] |
| 33 | Tool analog input 0 | o | x | 1mV unit |
| 34 | Tool analog input 1 | o | x | 1mV unit |
| 35~49 | Reserved (Tool IO) | | | |
| 50 | Is Robot Activated | o | x | 0 or 1 |
| 51 | Is Real-mode | o | x | 0 or 1 |
| 52 | Is Collision Detected | o | x | 0 or 1 |
| 53 | Is Robot arm power engaged | o | x | 0 or 1 |
| 54 | Is Direct Teaching mode | o | x | 0 or 1 |
| 55 | Is Robot moving | o | x | 0 or 1 |
| 56 | Is Pause state | o | x | 0 or 1 |
| 57 | Is Teaching pendant is connected | o | x | 0 or 1 |
| 58 | Is Program Run | o | x | 0 or 1 |
| 59 | Is No-Arc mode is on | o | x | 0 or 1 |
| 60 | Is EMG button released | o | x | 0 or 1 |
| 61 | Is First Program Run | o | x | 0 or 1 |
| 62~99 | Reserved (Future System) | | | |
| 100 | Command: Start Program Once | o | o | Rising Edge is command |
| 101 | Command: Start Program Repeat | o | o | Rising Edge is command |
| 102 | Command: Pause Program | o | o | Rising Edge is command |
| 103 | Command: Stop Program | o | o | Rising Edge is command |
| 104 | Command: Resume from pause | o | o | Rising Edge is command |
| 105 | Command: Resume from collision | o | o | Rising Edge is command |
| 106 | Command: Load default Program | o | o | Rising Edge is command |

| | | | | |
|---|---|---|---|---|
| 107 | Command: Robot Arm activation | o | o | Rising Edge is command |
| 108 | Command: Change to Real-mode | o | o | Rising Edge is command |
| 109 | Command: Power off the robot arm | o | o | Rising Edge is command |
| 110~127 | Reserved (Future System) | | | |
| 128~255 | User General Purpose Register | o | o | User Define Area |
| 256 | Joint reference 0 | o | x | 0.02deg unit / Signed |
| 257 | Joint reference 1 | o | x | 0.02deg unit / Signed |
| 258 | Joint reference 2 | o | x | 0.02deg unit / Signed |
| 259 | Joint reference 3 | o | x | 0.02deg unit / Signed |
| 260 | Joint reference 4 | o | x | 0.02deg unit / Signed |
| 261 | Joint reference 5 | o | x | 0.02deg unit / Signed |
| 262 | Joint angle 0 | o | x | 0.02deg unit / Signed |
| 263 | Joint angle 1 | o | x | 0.02deg unit / Signed |
| 264 | Joint angle 2 | o | x | 0.02deg unit / Signed |
| 265 | Joint angle 3 | o | x | 0.02deg unit / Signed |
| 266 | Joint angle 4 | o | x | 0.02deg unit / Signed |
| 267 | Joint angle 5 | o | x | 0.02deg unit / Signed |
| 268 | Joint current 0 | o | x | 10mA unit / Signed |
| 269 | Joint current 1 | o | x | 10mA unit / Signed |
| 270 | Joint current 2 | o | x | 10mA unit / Signed |
| 271 | Joint current 3 | o | x | 10mA unit / Signed |
| 272 | Joint current 4 | o | x | 10mA unit / Signed |
| 273 | Joint current 5 | o | x | 10mA unit / Signed |
| 274 | Joint information 0 | o | x | |
| 275 | Joint information 1 | o | x | |
| 276 | Joint information 2 | o | x | |
| 277 | Joint information 3 | o | x | |
| 278 | Joint information 4 | o | x | |
| 279 | Joint information 5 | o | x | |
| 280 | Joint temperature 0 | o | x | celcius unit |
| 281 | Joint temperature 1 | o | x | celcius unit |
| 282 | Joint temperature 2 | o | x | celcius unit |
| 283 | Joint temperature 3 | o | x | celcius unit |
| 284 | Joint temperature 4 | o | x | celcius unit |
| 285 | Joint temperature 5 | o | x | celcius unit |
| 286 | Joint 0 Estimated Current | o | x | 10mA unit / Signed |
| 287 | Joint 1 Estimated Current | o | x | 10mA unit / Signed |
| 288 | Joint 2 Estimated Current | o | x | 10mA unit / Signed |
| 289 | Joint 3 Estimated Current | o | x | 10mA unit / Signed |
| 290 | Joint 4 Estimated Current | o | x | 10mA unit / Signed |
| 291 | Joint 5 Estimated Current | o | x | 10mA unit / Signed |
| 292 | Joint 0 Gap(Esti.-Meas.) Current | o | x | 10mA unit / Signed |
| 293 | Joint 1 Gap(Esti.-Meas.) Current | o | x | 10mA unit / Signed |
| 294 | Joint 2 Gap(Esti.-Meas.) Current | o | x | 10mA unit / Signed |
| 295 | Joint 3 Gap(Esti.-Meas.) Current | o | x | 10mA unit / Signed |
| 296 | Joint 4 Gap(Esti.-Meas.) Current | o | x | 10mA unit / Signed |
| 297 | Joint 5 Gap(Esti.-Meas.) Current | o | x | 10mA unit / Signed |
| 298 | Joint 0 Gap(Esti.-Meas.) Curr+LPF | o | x | 10mA unit / Signed |
| 299 | Joint 1 Gap(Esti.-Meas.) Curr+LPF | o | x | 10mA unit / Signed |
| 300 | Joint 2 Gap(Esti.-Meas.) Curr+LPF | o | x | 10mA unit / Signed |
| 301 | Joint 3 Gap(Esti.-Meas.) Curr+LPF | o | x | 10mA unit / Signed |

| | | | | |
|---|---|---|---|---|
| 302 | Joint 4 Gap(Esti.-Meas.) Curr+LPF | o | x | 10mA unit / Signed |
| 303 | Joint 5 Gap(Esti.-Meas.) Curr+LPF | o | x | 10mA unit / Signed |
| 304~329 | Reserved (Joint Information) | | | |
| 330 | TCP reference X | o | x | 0.1mm unit / Signed |
| 331 | TCP reference Y | o | x | 0.1mm unit / Signed |
| 332 | TCP reference Z | o | x | 0.1mm unit / Signed |
| 333 | TCP reference RX | o | x | 0.02deg unit / Signed |
| 334 | TCP reference RY | o | x | 0.02deg unit / Signed |
| 335 | TCP reference RZ | o | x | 0.02deg unit / Signed |
| 336 | TCP position X | o | x | 0.1mm unit / Signed |
| 337 | TCP position Y | o | x | 0.1mm unit / Signed |
| 338 | TCP position Z | o | x | 0.1mm unit / Signed |
| 339 | TCP position RX | o | x | 0.02deg unit / Signed |
| 340 | TCP position RY | o | x | 0.02deg unit / Signed |
| 341 | TCP position RZ | o | x | 0.02deg unit / Signed |
| 342~389 | Reserved (TCP Information) | | | |

# APPENDIX J. SYSTEM UPDATE

- **Warning**
  It is recommended to back up the program files (.wsl) inside the tablet UI before the system update.

## 1. Overview

Rainbow Robotics' system update is a two-step process.

| |
|---|
| UI update through APK install → System software (control box) update |

## 2. Backup Program file

Connect the tablet and personal / business PC and obtain the program file (.wsl) from the path below and back it up.

| |
|---|
| Tablet → Android → data → com.rainbow.cobot → files → work→ GET .wsl files |

(※ It is recommended that you back up the acquired files before proceeding to the next step.)

## 3. UI Update

Rainbow Robotics' tablet UI program is distributed in the form of APK.

This is the same installation file as a regular Android application. Therefore, UI program is updated by moving the installation APK file to the tablet and installing it.

(※ Rainbow Robotics recommends installing after deleting an existing application.)

(※ When deleting an existing application, the program file (.wsl) is deleted together. Back up the program file in step 1 and proceed with this process.)

| |
|---|
| Copy the distributed APK file into Table → APK install |

## 4. Connection between Tablet PC and Control Box

Connect the tablet to the control box and access the UI program. After connecting, connect the control box communication with the tablet.

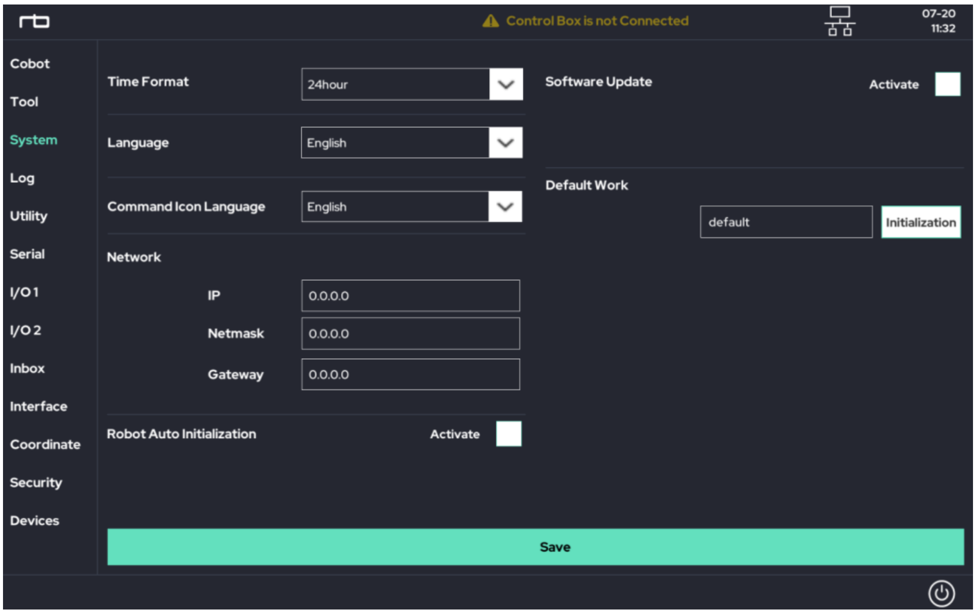
| |
|---|
| UI Home → Make → Click 'State' button → Connect |

(If the communication between the tablet and the control box is normal, the first box will be lit blue. For safety reasons, it is recommended not to initialize the robot.)

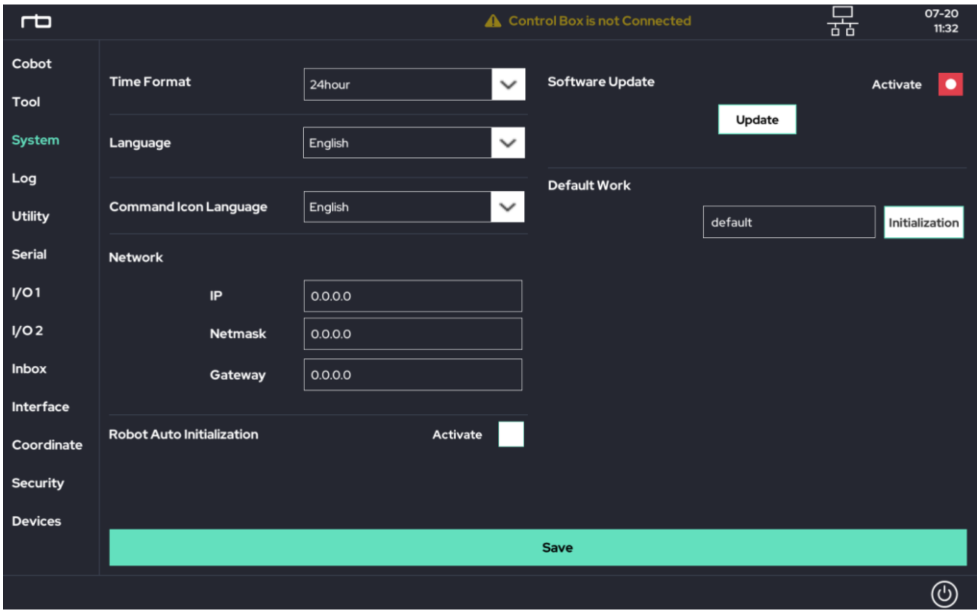
### 5. Go to and activate the system software update

Navigate to the system software update path as shown below.

| UI Home → Setup → System Tab |
| --- |

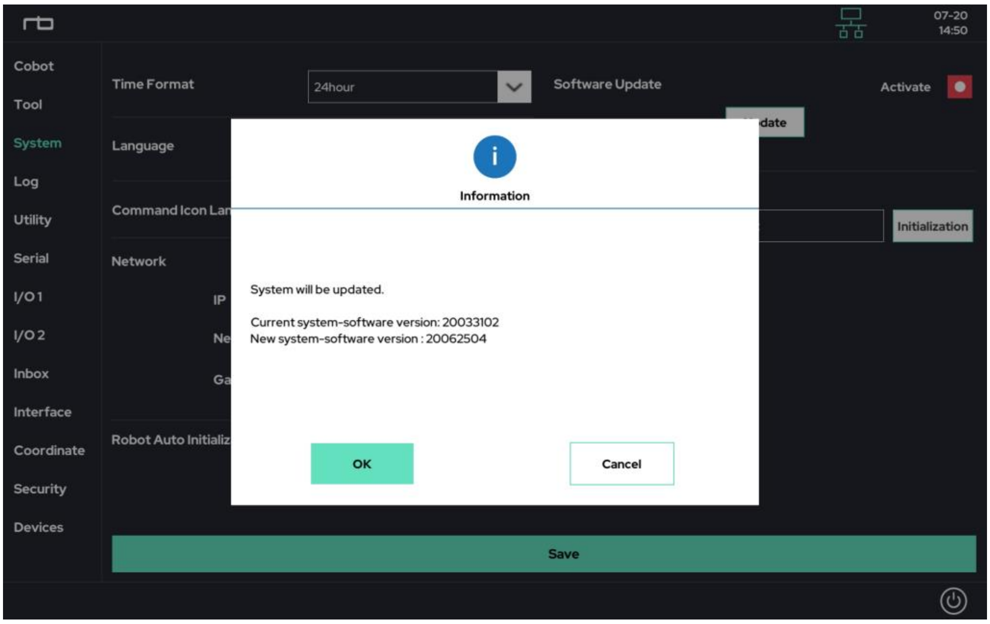



In the "Software Update" section on the right, click the Activate checkbox.



## 6. Progress System Software Update

The Update button will appear, and click this button to open a popup window.




Press "OK" button to update the software.

If the update is completed normally after clicking the OK button, the PC of the control box (controller) will automatically restart within 5 ~ 15 seconds.

During the restart process, "Please Wait…" is displayed on the LCD of the control box. Is displayed temporarily. This indicates that the control box is rebooting.

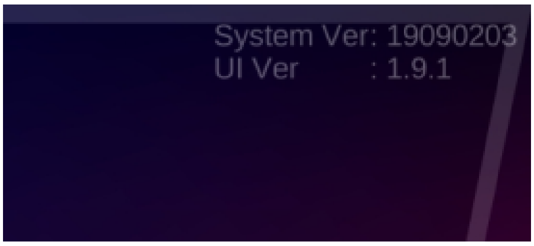
After the reboot is completed, "Normal Operation" is displayed on the LCD of the control box.

## 7. Check the Update

Reconnect the UI tablet and control box.

| |
|---|
| UI Home → Make → Click 'State' button → Connect |

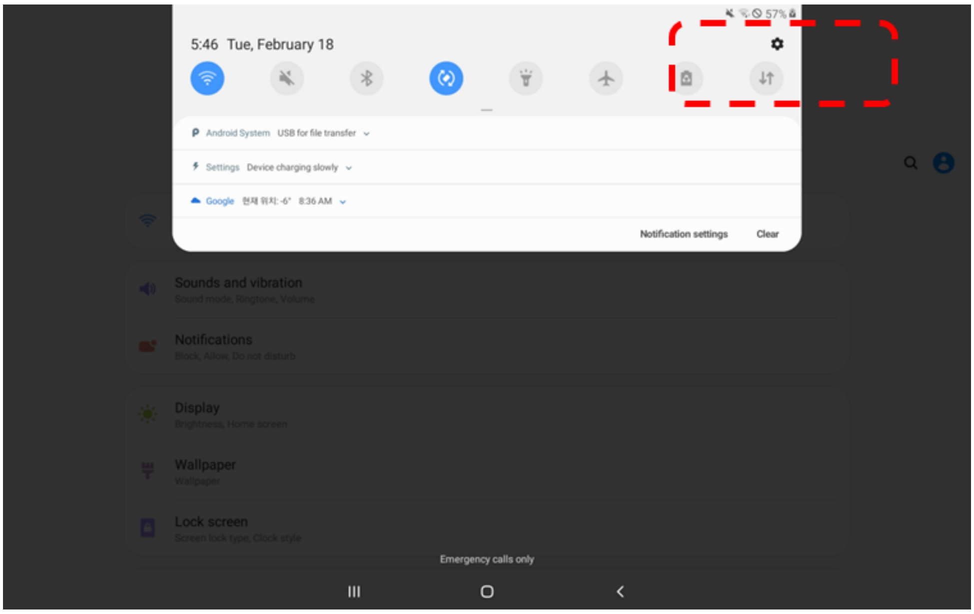
When you go back to the home screen of the UI, the software version is displayed on the upper right (or lower left). Check if it is updated to the correct version.



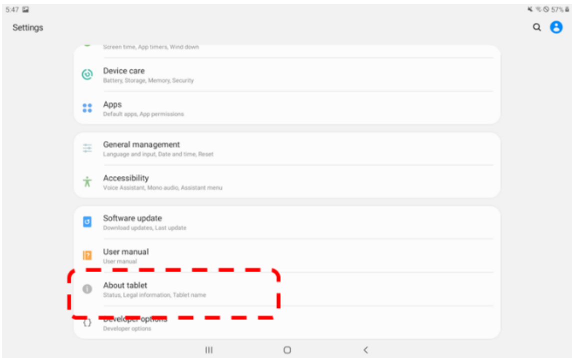
# APPENDIX K. ANDROID TABLET CONFIGURATION

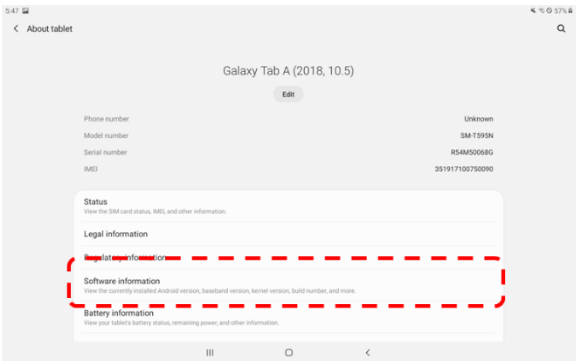
Before using the UI program, the following tablet settings are required.

1. Goto Setting section of the Android.

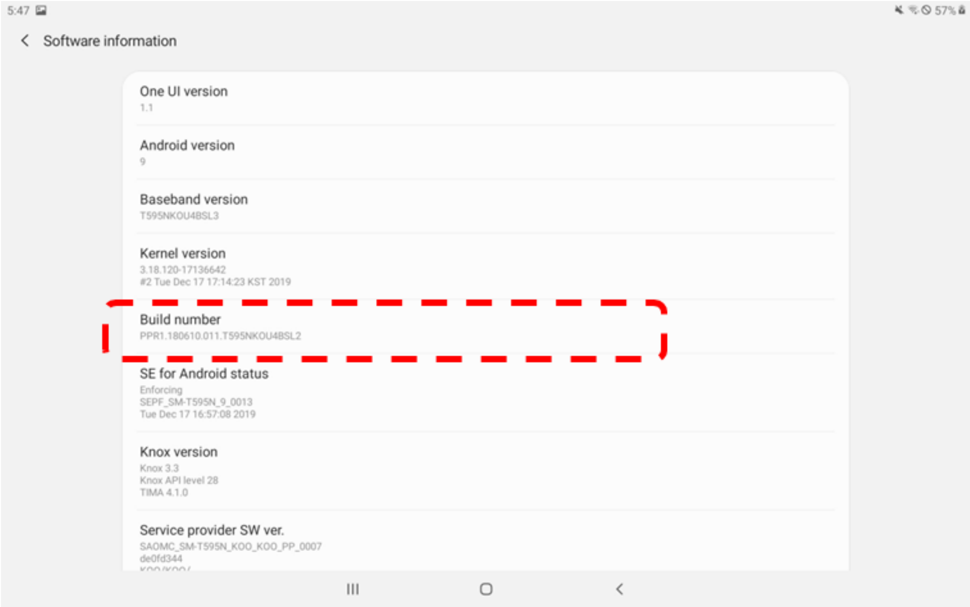



2. Goto “About Tablet” > “Software Information”.

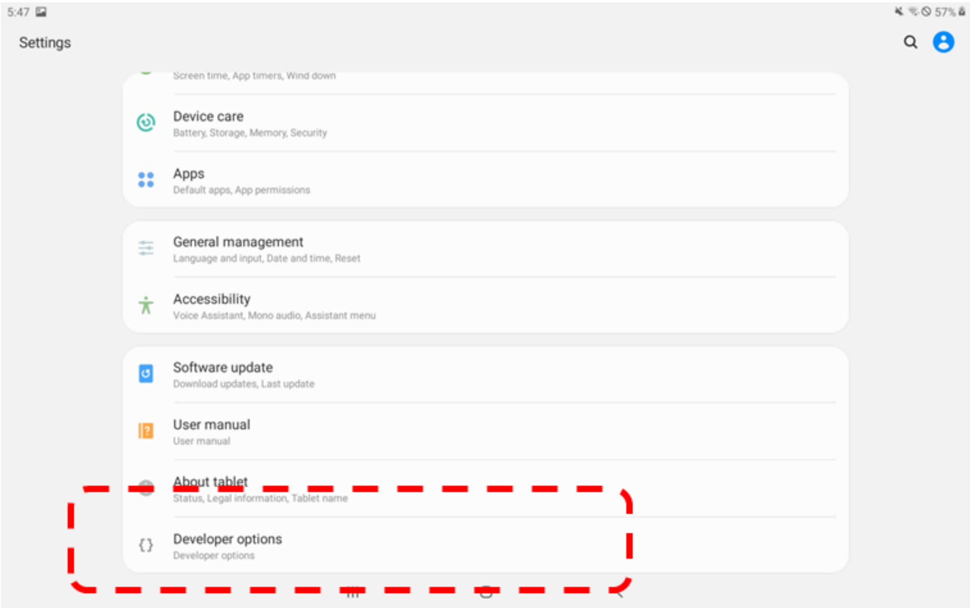


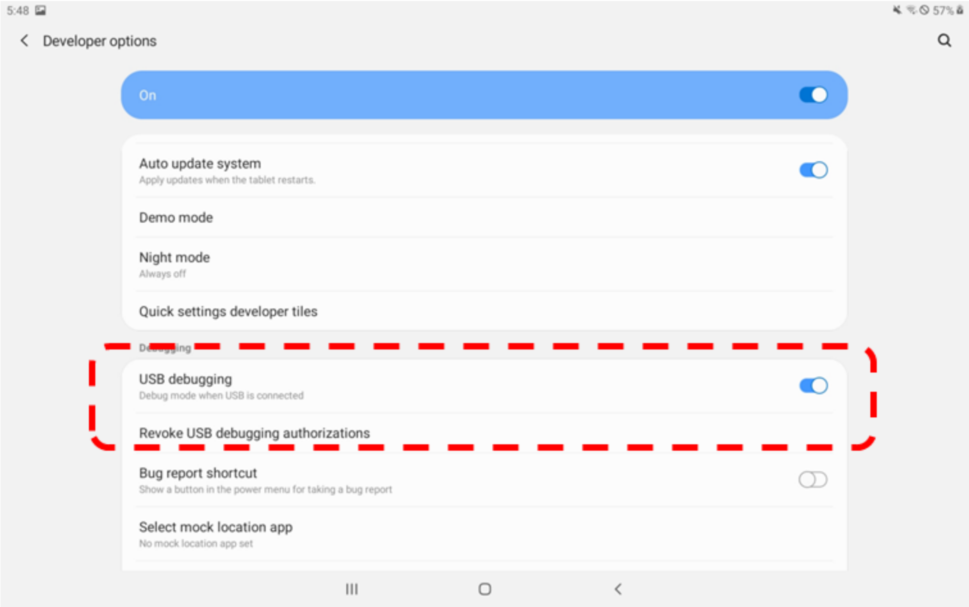
3. Multi-click (7 or more times) “Build Number” of tablet information.




4. A menu called "Developer Options" will appear under "About Tablet" as shown below.



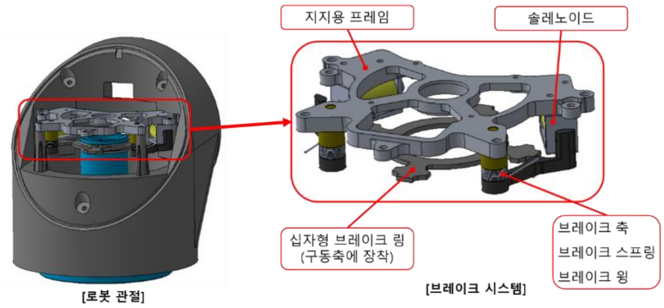
5. Activate "USB Debugging" in "Developer Options".




6. Run the APK distributed by Rainbow Robotics to install the UI program on your tablet.

# Appendix L. Brake System

The configuration of the Brake System on each axis of the robot arm consists of a support frame, solenoid, brake ring, brake shaft, brake spring and brake wing, which are installed on the robot joint as shown below.




If the solenoid is on, the physical interference between the turning radius of the brake ring and the brake wing is released, and if the solenoid is off, the physical interference between the end of the brake ring and the brake wing occurs, which stops the rotation of the driveshaft.

When the brake ring rotates and pushes through the brake wing, the wing returns to the spring force, and then a bi-directional brake occurs through physical interference, keeping both bi-directional rotations of the driveshaft stationary.

WE
TOUCH
THE CORE .

Rainbow Robotics