JANOME DESKTOP ROBOT

JR3000 Series

JANOME CARTESIAN ROBOT

JC-3 Series

JANOME SCARA ROBOT

JS3 Series

Operation Manual

Functions II

(Variables/Commands/Functions)

Thank you for purchasing this Janome Robot.

- **Before using your robot, read this manual thoroughly and always make sure you use the robot correctly. In particular, be sure to thoroughly read "For Your Safety" as it contains important safety information.**

- **After reading this manual, store in a safe place that can be easily accessed at any time by the operator.**

JANOME

PREFACE

This manual covers the JR3200, JR3300, JR3400, JR3500, JR3600, JC-3, and the JS3 Series.

There are several manuals pertaining to these robots.

Manual	Details	JR3000	JC-3	JS3
Read This First	• For Your Safety Be sure to thoroughly read “For Your Safety” as it contains important safety information. (Refer to the operation manual *Special Specifications* if you use the special specifications.) • Package Contents (JS3 Series only) Check the items included with your robot. • CD-ROM Contents Explains the CD-ROM contents.	✓	✓	✓
Setup (JR3000 / JC-3) Installation (JS3)	Explains how to set up the robot. ■ Make sure you read this manual when installing the robot ■ NOTE: This manual is designed for people who have received safety and installation training regarding the robot.	✓	✓	✓
Maintenance	Explains maintenance procedures for the robot. ■ Make sure you read this manual when performing maintenance ■ NOTE: This manual is designed for people who have received safety and maintenance training regarding the robot.	✓	✓	✓
Basic Instructions	Provides part names, data configurations, and the basic knowledge necessary to operate the robot.	✓ (Common)		✓
Quick Start	Explains the actual operation of the robot by creating and running simple programs.	✓ (Common)		✓
Teaching Pendant Operation	Explains how to operate the robot via the teaching pendant.	✓ (Common)		✓
Functions I	Explains point teaching.	✓ (Common)		
Functions II	Explains commands, variables, and functions.	✓ (Common)		
Functions III	Explains functions such as All Program Common Settings and PLC programs.	✓ (Common)		
Functions IV	Explains Customizing Functions.	✓ (Common)		
External Control	Explains I/O and Fieldbus. Refer to this manual if you are using Fieldbus.	✓	✓	✓
Communication Control	Explains COM 1 – 3 and LAN communication control.	✓ (Common)		

Manual	Details	JR3000	JC-3	JS3
Camera & Sensor Functions	Explains the functions of the attachable camera and Z position sensor.	✓ (Common)		
Specifications	Outlines general specifications such as the robot's operating range, mass, etc.	✓	✓	—
Auxiliary Axis Functions	Explains the auxiliary axis functions.	✓ (Common)		
Application Specifications	Explains the specialized functions of the various application specifications.	Standard model: - Application model: ✓		
Special Specifications	Explains installation and maintenance procedures for the robot. ■ Make sure you read this manual when performing installation and maintenance ■ NOTE: This manual is designed for people who have received safety, and installation and maintenance trainings regarding the robot.	—	✓	—

⚠ Warning

Do not handle or operate the robot in ways not covered in the manuals listed here.
Contact Janome (listed on the back of this manual) for repairs.
Failure to do so can cause electric shock or injury.

⚠ Caution

To make full use of the machine's functions and capabilities, make sure that you use the robot according to the correct handling/operation procedures that are written in the manuals pertaining to this robot.

If you turn OFF the power after making changes to robot's settings or data without saving, those changes are lost and the robot will revert to its original settings.
Make sure that you save any changes to data and/or settings.

Before using this robot for the first time, make sure you back up robot data and save the individual configuration information. Individual configuration information is needed when replacing internal circuit boards.
For details on how to back up robot data, refer to "3. BACKING UP AND RESTORING ROBOT DATA" in the operation manual *Setup* for the JR3000 Series, "6.1 Backing Up and Restoring Robot Data" in the operation manual *Setup* for the JC-3 Series, and "9.1 Backing Up and Restoring Robot Data" in the operation manual *Installation* for the JS3 Series.

- The descriptions within this manual are based on standard specifications. The menu item names etc. may vary depending on the model type.
- Menu items related to the Z axis may appear with 2 axis specifications; however settings made for these items are not applied.
- For information regarding optional additions for this robot, refer to “24. SPECIFICATIONS” in the operation manual *Specifications* for the JR3000 Series,“14. SPECIFICATIONS” in the operation manual *Specifications* for the JC-3 Series, and “15. SPECIFICATIONS” in the operation manual *Basic Intructions* for the JS3 Series. The notation “optional” is not used in the main text of this manual except for diagrams.
- Machine specifications may be modified without prior notice to improve quality.

Remarks:

- The operation methods described in this manual are indicated as follows:

  TP Operation via the teaching pendant

  PC Operation via PC (JR C-Points II)

- Click text that appears blue and is underlined to jump to that section.
  Example: Refer to “1. CREATING POINT JOB DATA.”

CONTENTS

FOR YOUR SAFETY

The safety notes outlined below are provided in order to ensure safe and correct usage of the product, and to prevent injury to the operator or other people, and damage to property.

・・・・・Be sure to follow the safety guidelines detailed here・・・・・

Symbols are also listed alongside the safety note explanations. Refer to the information below for understanding these terms and symbols.

■ Symbols that indicate the level of danger and/or damage.
The level of danger or damage that could occur as a result of ignoring these safety guidelines and misusing the robot are classified by the following symbols.

Danger	This symbol indicates an imminent risk of serious injury or death.
Warning	This symbol indicates a risk of serious injury or death.
Caution	This symbol indicates the possibility of serious injury or damage to property.

■ The following symbols indicate the nature of the danger and any necessary safety precautions to be taken.

	Indicates caution must be taken
	Take Caution (General Precaution)
	Indicates a forbidden action
	Never do this (General Prohibition)
	Do not disassemble, modify or repair.
	Do not touch (Contact Prohibition)
	Indicates a required action
	Be sure to follow instructions (General Requirement)
	Be sure to unplug the power cord
	Make sure the machine is grounded

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■■■ JR3000 Series ■■■■■■■■■■■■■■■■■■■■

If using auxiliary axis functions to operate a motor, such as a servo motor, that produces feedback and/or a motor with high output etc., or when using auxiliary axes in the robot setup etc., we ask that you perform a risk assessment on your side and take any necessary safety measures.

If Using Auxiliary Axis Functions in a Way that Require Safety Measures

⚠ Danger

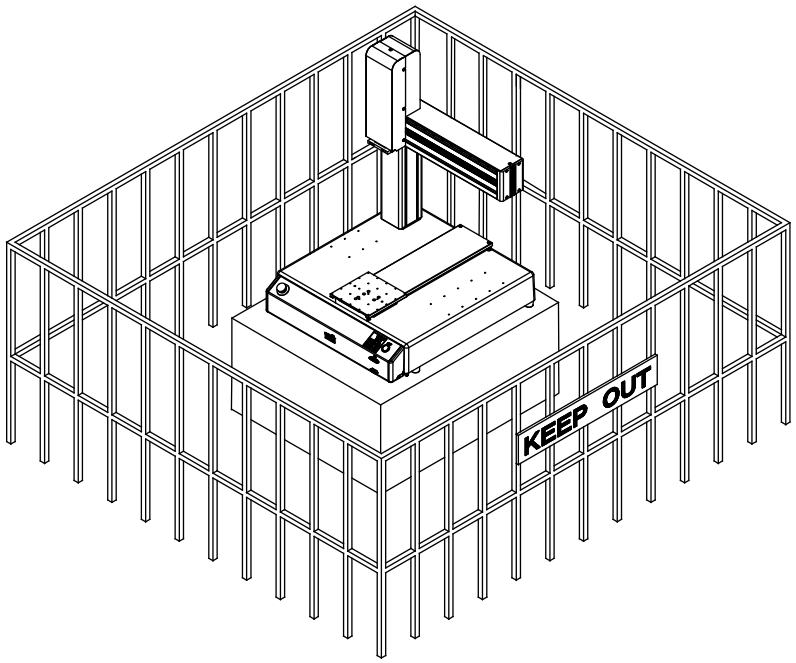
Always set up safety guards around the robot or the auxiliary axes so the moveable parts cannot be touched.

Anyone within the maximum reach of the robot and the auxiliary axes being controlled by the robot may be injured. Set up an **emergency stop interlock device that cuts OFF the motor power to the auxiliary axes when the entrance to the safety guard is opened** and make sure this entrance is the only way to access the machine.

NOTE: **A stop made via a device connected to the I/O-S connector is a category 2 stop. Make sure to perform a separate risk assessment of the interlock device**.

Furthermore, put up a **"Keep Out"** or **"Do Not Operate"** warning sign in a clearly visible place.

Example:



FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■■■ JR3000 Series ■■■■■■■■■■■■■■■■■■■■

If Using Auxiliary Axis Functions in a Way that Require Safety Measures

⚠ Danger

When power to the robot is ON, never enter the safety guard or put your head, hands, or any part of your body inside.
Entering the safety guard could result in injury.

When entering the safety guard due to something wrong with the robot or a peripheral device, or to inspect or lubricate the machine etc., **with both the power supply breaker and the robot switched OFF, make sure to lockout and tagout and confirm there is no electricity flowing to the robot.**
Failure to do so can cause electric shock or injury.

⚠ Warning

When creating a robot system using auxiliary axis functions, if the system can be categorized as an industrial robot, make sure to use the robot in accordance with the laws and guidelines of the country where it is used.

Before performing a run or operation, always check the following:

- **Obstacles** : Make sure there are no obstacles or people within the safety guard.
- **Installation** : Make sure the robot is installed properly, that there are no abnormalities with the robot and the surrounding devices, and that the teaching pendant and tools are in the appropriate places.
- **Emergency Stop Switch** : Make sure the I/O-S circuit (interlock) and emergency stop switch(es) are functioning properly.

It is potentially dangerous to operate the robot without making these checks first.

FOR YOUR SAFETY

If Using Auxiliary Axis Functions in a Way that Require Safety Measures

⚠ Warning	
	**Construct safety guards that are strong enough to protect the operator against such dangers as the tool or workpiece splintering, etc.** When working within the safety guard, use protective gear such as a helmet, protective gloves, protective goggles, and safety shoes. Failure to follow these safety measures can result in injury.
	If objects that the robot grasps have a risk of falling or being projected, **take into account the size, mass, and chemical composition of the objects for the required safety precautions.** Failure to do so can result in injury or unit breakdown.
	**When working within the safety guard, make sure not to come within the maximum range of the robot.** Failure to do so can cause injury.
	When starting a run, first confirm there are **no people inside of the safety guard and there are no obstacles that could interfere with the run.** Failure to do so can cause injury or unit breakdown.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JR3000 Series ■■■■■■■■■■■■■■■■■■

⚠ Danger

Do not use where flammable or corrosive gas is present.
Leaked gas accumulating around the unit causes explosions or fire.

⚠ Warning

Use protective gear such as a helmet, protective gloves, protective goggles, and safety shoes when installing the robot and performing maintenance.
Failure to do so can cause injury.

Make sure that you securely install the unit in a place that can fully withstand both the unit's weight and its usage. Install the robot on a workbench 600 mm or higher above floor level, in the center of the workbench.
In addition, **for units with a cooling fan on the back, allow for 300 mm or more clearance between the back of the unit and the wall.**
Install the switchbox 600 mm or more above floor level in an easily accessible place.
If installation is inadequate, the unit can drop or fall over causing injury and unit breakdown. Also, inadequate installation causes overheating or fire.

Make sure to power the unit within its rated current range.
Failure to do so causes electric shock, fire, or unit breakdown.

Plug the power cord into the power outlet firmly.
Failure to do so causes the plug to heat up resulting in fire.

Be sure to use the unit within its indicated voltage range.
Failure to do so causes unit breakdown, fire, or electric shock.

When replacing fuses, or inspecting or lubricating the unit, unplug the power cord from the power outlet, then remove the cord from the main unit and make sure there is no electrical current. Also, do not touch any of the power inlet pins within 5 seconds of removing the power cords. Failure to follow these steps causes electric shock or injury.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■■■ JR3000 Series ■■■■■■■■■■■■■■■■■■■■

⚠ Warning	
	Always **make sure the machine is grounded through the power cord.** **Do not use the machine when it is not grounded.** Improper grounding causes electric shock, fire, malfunction, or unit breakdown.
	**Wipe the power plug with a clean, dry cloth periodically to eliminate dust.** Dust accumulation deteriorates the electrical insulation or causes fire.
	**Be sure to unplug the power cord from the power outlet when the unit is not in use for long periods of time.** Dust accumulation causes fire.
	**Be sure to turn OFF the unit before inserting or removing cords and cables such as the teaching pendant cable or LAN cable.** Failure to do so causes electric shock, data loss, unit breakdown, or malfunction.
	**If disassembling this machine, follow the instructions in the operation manual *Maintenance*, and do not disassemble in any other way other than those specified.** Do not modify the machine in any way. Inappropriate disassembly or modification causes electric shock or unit breakdown.
	**Do not allow water or oil to come in contact with the unit, control box or the power cord.** Contact with water or oil causes electric shock, fire, or unit breakdown. IP Protection Rating: IP20.
	**If anything unusual occurs, such as a burning smell or unusual sound, stop operation and unplug the power cord immediately. Contact Janome (details on the back of this manual) or a Janome dealer.** Continuing to use the robot without addressing the problem causes electric shock, fire, or unit breakdown.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JR3000 Series ■■■■■■■■■■■■■■■■■■

⚠ Caution	
	**Do not drop or jar the unit during transport and/or installation.** This can cause injury or damage the unit.
	**Before performing any operation, ensure there is no imminent danger to any of the operators.** Failure to do so causes injury.
	**Use the unit in an environment between 0 and 40 °C, with a humidity level of 20 to 90 %, and without condensation.** Use outside of these conditions can cause unit breakdown or malfunction.
	**Use the unit in an environment where no electrical noise is present.** Failure to do so causes unit malfunction or breakdown.
	**For models with I/O-S circuits*, when installing the unit, take safety measures such as setting up area sensors and safety guards.** If there are no safety measures in place and someone enters the area of operation when the robot is running, they may be injured.
	**Keep the emergency stop switch within reach of the operator when running or operating the robot.** If the robot is operated when the emergency switch is not within reach, it may not be possible to stop the robot immediately and safely. This is potentially dangerous.
	**Make sure that you regularly perform a function check of the emergency stop switch(es). Also, for models with I/O-S circuits*, regularly perform an I/O-S circuit function check.** If the robot is operated without making these checks, it may not be possible to stop the robot immediately and safely in an emergency. This is potentially dangerous.

* A stop made via a device connected to the I/O-S connector is a category 2 stop. Make sure to perform a separate risk assessment of the interlock device.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■■■ JR3000 Series ■■■■■■■■■■■■■■■■■■■■

⚠ Caution	
	**When attaching tools, a USB camera, or any other device, make sure they are securely fitted before running the robot.** Failure to do so causes injury or breakdown.
	**When using the machine for extended periods of time, check and make sure none of the main unit's mounting screws are loose, and perform a routine inspection every 3 months or after every 750 hours of operation.** Failure to do so causes injury or breakdown.
	**Be sure to check the connections of the cords and cables to the main unit.** Improper connection causes unit malfunction or breakdown.
	**When lifting or transporting the robot, do so in accordance with the following:** • **Secure the movable parts of the unit (the Z/ZR mechanism) before transportation.** • **Lift and transport the JR3200 Series with 2 or more people.** • **Lift and transport the JR3300 – JR3600 Series with 2 or more people or use a lifter.** • **Hold the bottom of the base and keep the robot horizontal.** • **Do not hold the robot by the column or Y body.** Failure to adhere to this can cause injury or damage the robot.
	**Use the unit in an environment that is not exposed to direct sunlight.** Direct sunlight causes unit malfunction or breakdown.
	Individual Configuration Information varies for each individual unit even if they are the same model. **Do not use backup data with a different robot. The robot cannot function normally with backup data from a different robot.**

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JC-3 Series ■■■■■■■■■■■■■■■■■■

Industrial Robot Safety Standards

Make sure to use the robot in accordance with the laws and guidelines of the country where it is used.

FOR YOUR SAFETY

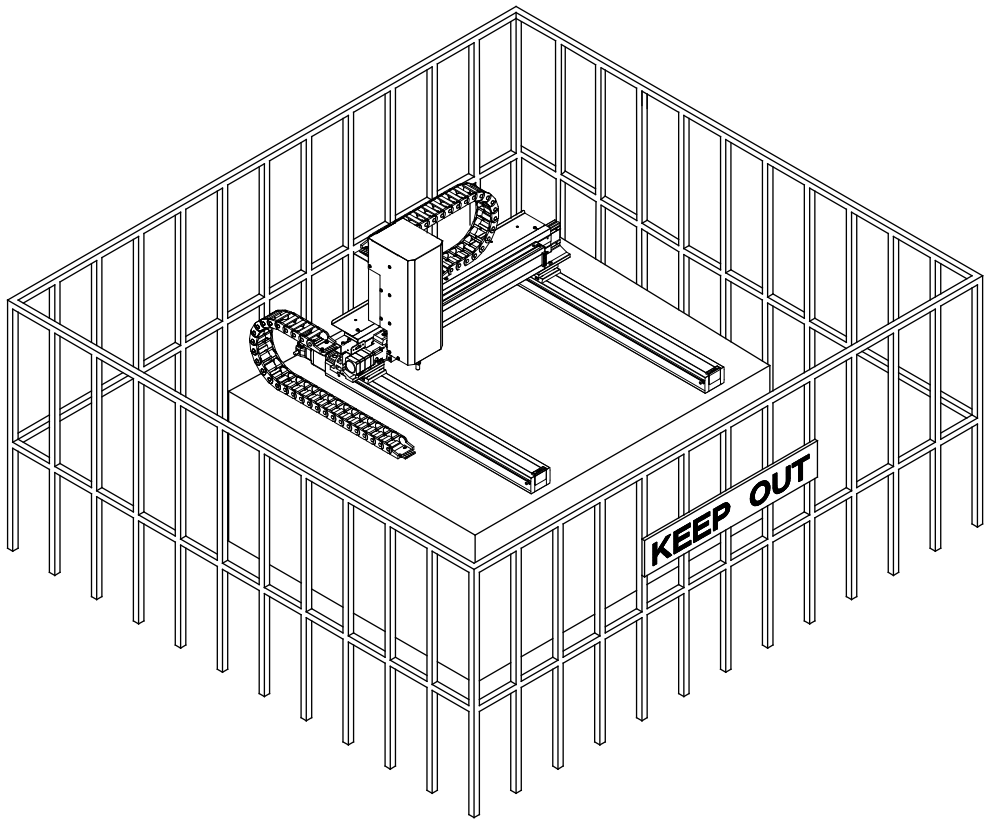
■■■■■■■■■■■■■■■■■■■■ JC-3 Series ■■■■■■■■■■■■■■■■■■■■

⚠ Danger

Do not use where flammable or corrosive gas is present.
Leaked gas accumulating around the unit causes explosions or fire.

Always set up safety guards.
Anyone within the maximum reach of the robot may be injured.
Using the included EMG-OUT connector (unit), set up an emergency stop interlock system that is triggered when the entrance to the safety guard is opened and make sure this entrance is the only way to access the machine.
Furthermore, put up a **"Keep Out"** or **"Do Not Operate"** warning sign in a clearly visible place.

Example:



FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JC-3 Series ■■■■■■■■■■■■■■■■■■

⚠ Danger

Construct a safety circuit before operating the robot.
Use the EMG-OUT connector (unit) to maintain safety by installing a relay such as an external stop device on the power supply line which cuts the DC 48 V power input. Construct the safety circuit so it operates as a category 0 stop when the emergency stop switch is pressed. Without the construction of a safety circuit, motor power 48 V will not be cut OFF even if the emergency stop switch is pressed, leading to unexpected accidents and injury. An error will occur if you construct the safety circuit without it operating as a category 0 stop when the emergency stop switch is pressed.

When power to the robot is ON, never enter the safety guard or put your head, hands, or any part of your body inside.
Entering the safety guard could result in injury.

Before entering the safety guard or inspecting or performing maintenance on the robot, make sure there is no electrical current.
・ Unplug the controller power cord from the power outlet. Do not touch any of the power inlet pins within 5 seconds of removing the power cords.
・ If using a controller with a terminal block (DC 48 V), turn the power OFF and remove the wiring from the terminal block.
Failure to follow these steps can cause unexpected operation, electric shock, or injury.

Before performing a run or operation, always check the following:

- **Obstacles** : Make sure there are no obstacles or people within the safety guard.
- **Installation** : Make sure the robot is installed properly, that there are no abnormalities with the robot and surrounding devices, and that the teaching pendant and tools are in the appropriate places.
- **Emergency Stop Switch** : Make sure the EMG OUT circuit (interlock) and emergency stop switch(s) are functioning properly.

It is potentially dangerous to operate the robot without making these checks first.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JC-3 Series ■■■■■■■■■■■■■■■■■■

⚠ Danger	
	**Keep the emergency stop switch within reach of the operator when running or operating the robot.** If the robot is operated when the emergency switch is not within reach, it may not be possible to stop the robot immediately and safely. This is potentially dangerous.
	**Make sure that you regularly perform a function check of the emergency stop switch(s). Also regularly perform an EMG OUT circuit function check.** If the robot is operated without making these checks, it may not be possible to stop the robot immediately and safely in an emergency. This is potentially dangerous.

⚠ Warning	
	**Use protective gear such as a helmet, protective gloves, protective goggles, and safety shoes when installing the robot and performing maintenance.** Failure to do so can cause injury.
	**Make sure to power the unit within its rated current range.** Failure to do so causes electric shock, fire, or unit breakdown.
	**Plug the power cord into the power outlet firmly.** Failure to do so causes the plug to heat up resulting in fire.
	**Make sure to connect and use crimp terminals with the power cord connecting to the terminal block (DC 48 V input) and to securely tighten the terminal block screws.** Failure to do so causes electric shock, fire, or unit breakdown.
	**Make sure to perform work from outside of the safety guards when the power is ON.** Failure to do so can cause injury.
	**Be sure to use the unit within its indicated voltage range.** Failure to do so causes unit breakdown, fire, or electric shock.
	**Install the controller within an industrial control panel, and make sure when the industrial control panel door is opened, the controller power is automatically cut OFF.** In addition, for controllers with a cooling fan, allow for a clearance of 300 mm or more from the top of the controller, as well as 100 mm or more from the air vent on the side. Inadequate installation can cause overheating, fire, electric shock, or injury.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JC-3 Series ■■■■■■■■■■■■■■■■■■

⚠ Warning

With the 3 axis specifications, if releasing the brake, do so after either removing the attached tool or taking measures to prevent the tool from dropping.
If you release the brake when turning the power ON in Teaching Mode, Switch Run Mode, or External Run Mode, the axis may drop down depending on the mass attached to the Z axis. If the Z axis drops down, it can cause injury or unit breakdown.

When inspecting or performing maintenance on the controller, **make sure there is no electrical current and perform the following**:

- **Unplug the controller power cord from the power outlet. Do not touch any of the power inlet pins within 5 seconds of removing the power cords.**
- **If using a controller with a terminal block (DC 48 V), turn the power OFF and remove the wiring from the terminal block.**

Failure to follow these steps can cause electric shock, injury, data loss, unit breakdown, or malfunction.

Always **make sure the machine is grounded through the power cord.**
Do not use the machine when it is not grounded.
Improper grounding causes electric shock, fire, malfunction, or unit breakdown.

Wipe the power plug with a clean, dry cloth periodically to eliminate dust.
Dust accumulation deteriorates the electrical insulation or causes fire.

Be sure to unplug the power cord from the power outlet when the unit is not in use for long periods of time. Dust accumulation causes fire.

If disassembling this machine, follow the instructions in the operation manual *Maintenance*, and do not disassemble in any other way other than those specified. Do not modify the machine in any way.
Inappropriate disassembly or modification causes electric shock or unit breakdown.

Do not allow water or oil to come in contact with the unit, controller or the power cord. Contact with water or oil causes electric shock, fire, or unit breakdown.
IP Protection Rating: IP20.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JC-3 Series ■■■■■■■■■■■■■■■■■■

⚠ Warning

If anything unusual occurs, such as a burning smell or unusual sound, stop the run, unplug the controller power cord from the power outlet, and make sure there is no electrical current. Contact Janome (details on the back of this manual) or a Janome dealer.

- **Unplug the controller power cord from the power outlet. Do not touch any of the power inlet pins within 5 seconds of removing the power cords.**
- **If using a controller with a terminal block (DC 48 V), turn the power OFF and remove the wiring from the terminal block.**

Continuing to use the robot without addressing the problem causes electric shock, fire, or unit breakdown.

⚠ Caution

Do not drop or jar the unit during transport and/or installation.
This can cause injury or damage the unit.

Before performing any operation, ensure there is no imminent danger to any of the operators.
Failure to do so causes injury.

Use the unit in an environment between 0 and 40°C, with a humidity level of 20 to 85 %, and without condensation.
Use outside of these conditions can cause unit breakdown or malfunction.

Use the unit in an environment where no electrical noise is present.
Failure to do so causes unit malfunction or breakdown.

When attaching tools, a USB camera, or any other device, make sure they are securely fitted before running the robot.
Failure to do so causes injury or breakdown.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JC-3 Series ■■■■■■■■■■■■■■■■■■

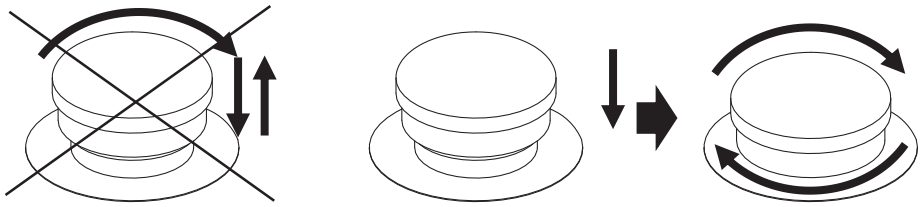
⚠ Caution

When using the machine for extended periods of time, check and make sure none of the main unit's mounting screws are loose, and perform a routine inspection every 3 months or after every 750 hours of operation. Failure to do so causes injury or breakdown.

Be sure to check the connections of the cords and cables to the main unit and controller.
Improper connection causes unit malfunction or breakdown.

Make sure to wait for at least 1 second before power cycling the robot after the external power supply to the controller (DC 48 V) is cut OFF. Failure to do so can cause unit breakdown.
Make sure to wait for at least 1 second between the following operations:

- When power cycling after the external power supply DC 48 V is cut OFF
- When releasing the emergency stop after the emergency stop switch is pressed down
- When releasing the emergency stop after an external emergency stop device is activated
- Operations to reset the emergency stop switch on the switchbox and teaching pendant when it is released



Do not push down emergency stop switch again when it is already rotated for release

Press straight down

Turn and release (Do not pull)

Make sure to construct a power circuit allowing for the rated current so that the external power supply DC 48 V interruption does not occur. Failure to do so can cause breakdown.
External Power Supply Output :DC 48 V, 6.25 A or more
DC 48 V, 8.15 A or more (Absolute Encoder Model)

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JC-3 Series ■■■■■■■■■■■■■■■■■■

⚠ Caution	
	**When using the external safety circuit using the EMG-OUT connector (unit) and the emergency stop device is activated, make sure to release the emergency stop after resolving the cause of the emergency stop.** Failure to do so can cause injury or unit breakdown. Adhere to the followings: • Do not use the external safety circuit or external emergency stop device to stop a run. • When not in emergency, use the start / stop switch on the switchbox or use the I/O-SYS signal to stop a run. • Using the external power supply DC 48 V to stop a run by cutting OFF / re-supply the power can cause breakdown.
	**Secure the movable parts of the unit before transportation.** Failure to do so causes injury or breakdown.
	**When lifting and transporting the robot, do so with 2 or more people.** Failure to do so causes injury or breakdown.
	**Use the unit in an environment that is not exposed to direct sunlight.** Direct sunlight causes unit malfunction or breakdown.
	Individual Configuration Information varies for each individual unit even if they are the same model. **Do not use backup data with a different robot. The robot cannot function normally with backup data from a different robot.**

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

Industrial Robot Safety Standards

Make sure to use the robot in accordance with the laws and guidelines of the country where it is used.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

Safety Precautions Regarding Installation

Robot Unit

⚠ Danger

Anyone within the maximum reach of the robot may be injured.
Install safety guards in adherence with the following:

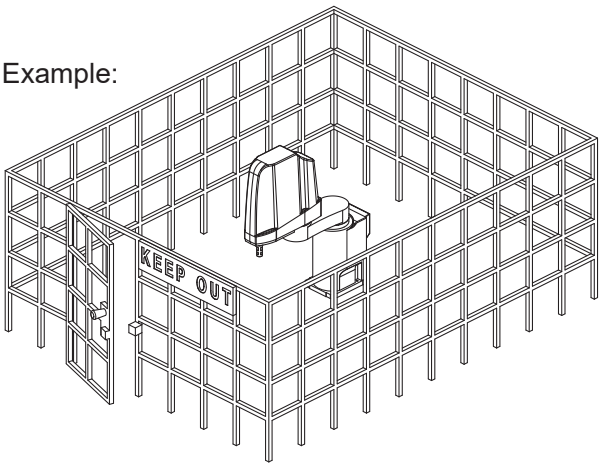
- The safety guards cannot easily be moved.
- The safety guards cannot easily fall over or be broken.
- Allow enough clearance between the robot and the safety guards so that even if the robot falls over, it does not hit the safety guards.
- No part of your body, such as your head or hands, can enter the safety guards.
- Install an interlock device on the entrance to the safety guards that activates an emergency stop when opened and make sure this entrance is the only way to access the machine. Connect the interlock device to the controller by using the included I/O-S connector.
- Place a warning sign such as “Keep Out” or “Do not Operate” on the safety guard entrance in a location that is easily visible.
- Affix the included danger sticker (shown below) in a location that is easily visible.

NOTE:

- A stop made via a device connected to the I/O-S connector is a category 1 stop. Make sure to perform a separate risk assessment for the interlock device.
- Refer to the operation manual *Installation* for details regarding I/O-S connections.
- After installing the unit, make sure to perform pre-operation checks from outside of the safety guards.

Warning Sticker

Example:



FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Danger

Do not remove the fixing plates from the robot arm until after transportation is complete.
Failure to do this can cause injury or breakdown.

If temporarily placing the robot on a mount, etc., make sure to secure the robot to the mount by tightening 2 or more bolts into the mounting holes on the mounting base to prevent the robot from tipping over.
Failure to do this may cause the robot to fall, resulting in injury or breakdown.

To set the home position for the J3 or J4-axis, release the brake and move the J3-axis (shaft) manually with two hands. When releasing the brake, take the following precautions:

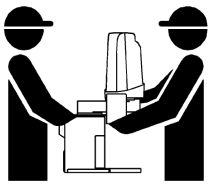
- Always release the brake with at least 2 people present.
- After the brake is released, the tool mass may cause the J3-axis to drop down. Make sure no people are within the operating range of the robot before releasing the brake.
- If there is risk that the robot may hit a peripheral device, make sure to position the robot arm appropriately before releasing the brake.

Failure to adhere to this can cause injury or breakdown.

⚠ Warning

Construct safety guards that are strong enough to protect the operator against such dangers as the tool or workpiece splintering, etc.
For the safety of the operator when working within the safety guard, use protective gear such as a helmet, protective gloves, protective goggles, and safety shoes.
Entering the safety guards could result in injury.

Always have 2 people carry the robot with the fixtures attached as shown in the illustration to the right.
Mass: JS3-3520: Approx. 39 kg, JS3-4520: Approx. 40 kg
JS3-5520: Approx. 41 kg
Refer to “2.3 Transporting the Robot Unit” in the operation manual *Installation* for further details.



FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Warning	
	Avoid using the robot in operating environments such as the ones below. If using the robot in environments such as these, take measures to protect the robot from the direct effects of the work environment. • An environment where the robot vibrates or is bumped at 3.5 G or more during transportation. • An environment where the robot vibrates or is bumped at 0.5 G or more during operation. Using the robot in these environments can cause malfunction or breakdown.
	Avoid using the robot in an environment that exceeds the radiation tolerance value for the average person. If using the robot in an environment such as this, take measures to protect the robot from the direct effects of the work environment. Using the robot in this environment can cause malfunction or breakdown.

⚠ Caution	
	**Do not hold the robot from the left and right sides or hold the robot by the covers.** Doing so may tip the robot over or the covers may break; causing the robot to fall resulting in injury.
	Do not hold the robot by the J2 arm cover during transportation. Doing so can damage the cover.
	**Do not apply force to the shaft (J3-axis).** Doing so can damage the shaft and cause an overload error to occur when the robot is run.
	**Do not tip or place the robot on its side.** Doing so can cause a grease leak or breakdown.
	Mount the robot on a steel base that is designed to withstand both the robot's mass and the force generated during operations. Inadequate installation may cause the robot to fall, resulting in injury or breakdown.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Caution	
	Firmly fix the robot to a specified horizontal mounting surface using the mounting bolts and make sure that positional displacements cannot occur.
	Install the robot in a location that provides enough clearance to replace the battery at the front of the robot and enough clearance to connect the motor power cable and the encoder cable to the back of the robot. Also, do not install the robot in a place where it is exposed to direct sunlight or heat from lighting. The robot surface may heat up and cause errors to occur.
	The robot requires periodic lubrication. Make sure you install the robot in a location where greasing is possible. (Refer to “4.8 Lubrication” in the operation manual *Maintenance*.
	Check the air tubing is connected correctly, has not come loose, or cannot easily come loose. If there is a problem, reconnect the tubing as required. Failure to do so can cause injury or breakdown.
	**After installing the robot, make sure to remove the fixing plates attached to the robot arm.** Failure to do so may cause breakdown.
	After installation, perform a periodic inspection (every 3 months or 750 hours of use, depending on how often the robot is in use) to check the tightening torque of the mounting screws with a torque wrench, etc.
	Adhere to the following when constructing or selecting a hand for the robot: • Make sure the total mass of the hand and the workpiece does not exceed the rated payload capacity of the robot. Especially if attaching an unconventional hand, check that the wrist does not exceed the allowable moment of inertia. • Use a hand or tool that has sufficient ability to grasp workpieces. • Secure the hand to the mechanical interface using bolts that are sized accordingly for the attachment dimensions. Make sure the tool has no protruding parts or sharp edges, other than those required for operation, and construct a cover for the tool if required. Failure to adhere to this can cause injury or breakdown.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Caution	
	If connecting cables or hoses to the hand, make sure they do not restrict the robot movements and make sure the robot operations do not cause the cables or hoses to get tangled and/or cause them to break. Improperly attached cables or hoses can cause breakdown.
	If using a pneumatic hand, make sure to provide clean air at the specified pressure. Also, make sure the air pressure does not exceed 0.7 MPa (7 kgf/cm^2). Air pressure higher than this may cause the robot's internal air hoses to burst.

Robot Unit and Controller

⚠ Danger	
	**Do not use the robot where flammable or corrosive gas is present.** Leaked gas accumulating around the unit causes explosions and fire.

⚠ Warning	
	**Use protective gear such as a helmet, protective gloves, protective goggles, and safety shoes when installing the robot.** Entering the safety guards could result in injury.
	**Before wiring the power cords, make sure there is no electrical current and perform the following:** • **Lockout/tagout with the power source circuit breaker in the OFF position, and remove the power cords from the terminal block.** • Do not touch the terminal block within 5 seconds of removing the power cords. Failure to adhere to this may cause electric shock, injury, data loss or breakdown.
	**Be sure to use the unit within its indicated voltage range.** **Failure to do so causes unit breakdown, fire, or electric shock.**

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Warning

Make sure to isolate the robot motor power cable, the encoder cable, and external I/O cables from the power cable or grounding wire of other devices. Also make sure the external I/O cables are shielded.
Do not apply voltages to terminals other than those specified in the operation manuals. Doing so can damage the robot or cause the terminal to explode.

⚠ Caution

When transporting the robot, **do not excessively shock or vibrate the robot.** Doing so can cause malfunction or breakdown.

Do not drop or jar the unit during transport and/or installation.
This can cause injury or damage the unit.

Use the robot in an indoor environment where it is not exposed to direct sunlight.
Direct sunlight can cause unit malfunction or breakdown.

Place the machine in a well-ventilated area for the health and safety of the operator.

Use the robot in an environment that meets the following conditions:

- **Ambient temperature of 0 to 40°C**
- **Relative humidity of 45 to 85 % (no condensation)**
- **Altitude not exceeding 1000 m**
- **The installation space is not in a vacuum.**

Use outside of these conditions may cause malfunction or unit breakdown. If the unit is used where these conditions are not met, we cannot guarantee its operation. IP protection rating for the both the robot and the controller is IP20.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Caution	
	**Use the unit in an environment where no electrical noise is present.** Noise may cause unit malfunction or breakdown.
	**Do not use the machine in an environment that is damp or dusty.** Dust and moisture can cause malfunction or breakdown. IP protection rating for the both the robot and the controller is IP20 (not water resistant).
	Install the cables and connectors so there is no stress on them and do not forcefully bend or pull, or stand on them after they are installed. Protect the cables and connectors by using piping and covers, etc., as necessary. Make sure the latches and screws securing the cables and connectors are firmly secured. If the cables and connectors are not secure, they may come loose during a run.
	If the hand input cable (optional) has free terminals, make sure to insulate them before supplying power to the hand input cable (optional). Uncovered terminals can cause a short circuit to occur.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

Controller

⚠ Danger	
	**Mount the controller outside of the safety guards in a location where the switches can easily be reached and the controller can always be monitored by the operator without turning their back on the robot unit itself.** Mount the controller so that the operation panel is 600 mm or more above floor level for maintenance work.
	Before connecting a Fieldbus, make sure safety can be maintained at all times when the robot is run. If signals such as a start signal, etc., are assigned to the Fieldbus, the Fieldbus may standby waiting to send signals and cause the robot to start running immediately after it is connected. Failure to do so can cause injury or breakdown.

⚠ Warning	
	Use a lifter, etc., to transport the controller. Failure to do so can cause injury or breakdown.
	If external vibrations can cause the controller to move, fix it in place using the base plate of the controller. (Do so without removing the rubber feet)
	**Make sure that you securely install the unit in a place that can fully withstand both the unit's mass and its usage.** If installation is inadequate, the unit can drop or fall over causing injury and unit breakdown. Inadequate installation can also cause overheating and fire.
	The controller has a fan and ventilation intakes for cooling. Do not block these openings and make sure there is a clearance of at least 20 mm on both the left and right sides and at least 40 mm on the back side. Blocking these openings can cause overheating or breakdown.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Warning	
	Leave approximately 200 mm or more clearance from the front of the controller so there is no stress on the connectors and enough room to work. Failure to do so can cause malfunction or breakdown.
	In addition to the clearance required for installation, leave sufficient space around the controller for removing covers (with a screwdriver) as a contingency for maintenance work. The controller may get hot during a malfunction. Do not touch the controller immediately after use.
	Make sure the power cord prepared on your side uses the **correct crimped terminals for connecting to the terminal block. Make sure they are within the sizes outlined below and never forcefully connect them.** Incorrect connections may cause fire or breakdown. • Conductor size: AWG10 (cross-sectional area: 6.0 mm^2) • Crimped terminal: M4, with a width of 9.5 mm or less
	After wiring the power supply terminal block, attach the cover included to prevent electric shock. An unprotected terminal block can cause electric shock, fire, or unit breakdown.
	**Always make sure to connect the protective ground through the power cord. Do not use the machine when the protective ground is not connected. Make sure the protective grounding resistance is 100 Ω or less.** Improper grounding causes electric shock, fire, malfunction, or unit breakdown, etc.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Caution	
	Do not apply pressure to any protruding parts, such as a switch, the terminal block, or a connector when transporting the controller. Doing so can cause breakdown.
	If you want to use the controller, operation box (optional), or teaching pendant as a monitor while in Run Mode, **mount the respective device 600 mm or more above floor level in an easily accessible place so that the emergency stop switch can be immediately reached in the event of an emergency.** It is dangerous to operate the machine without the emergency stop switch within reach; you will not be able to stop the robot immediately and safely in the event of an emergency. Installing one of these devices too low can also result in malfunction from people knocking the device with their feet, etc.
	Make sure installation allows for access to the memory port during maintenance.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

Safety Precautions Regarding Usage

Robot Unit

⚠ Warning	
	If objects that the robot grasps have a risk of falling or being projected, **take into account the size, mass, and chemical composition of the objects for the required safety precautions.** Failure to do so can result in injury or unit breakdown.
	**When manually moving the robot arm, do not insert your hands or fingers into any of the joints or openings.** Your hands or fingers may get caught in these openings depending on the robot pose.

⚠ Caution	
	Before performing any operation, make sure there is no imminent danger to any of the operators. Failure to do so causes injury.
	**When attaching tools, a USB camera, or any other device, make sure they are securely fitted before running the robot.** A loose tool can cause injury or breakdown.
	If weight is applied to the J3 (Z) / J4 (R) axis, the load may cause the J3/J4 axis to drop down when the power to the robot is turned OFF. To prevent this from happening, remove the load from the J3/J4 axis or install a safety block, etc.
	When performing work inside the safety guards, perform your own risk assessment and establish **"work regulations", as outlined below, with thorough planning for safety.** Entering the safety guards may result in injury. • Work regulations should be relevant and appropriate for the type of work, and consist of details such as robot operating procedures and signs to be used between operators. • When creating work regulations, incorporate the opinion of operators and work safety specialists. Make sure to review and update the contents of the work regulations regularly.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Caution	
	If manually moving the robot arm, do so slowly. Moving the arm at high speed can cause large amounts of backlash, reducing the robot's accuracy and damage the backup data.
	Depending on the arm pose, the shaft may come in contact with the robot base even when operating the robot within the work envelope. Make sure the shaft does not come in contact with the base while making JOG movements*. * A JOG movement refers to moving the robot arm via the teaching pendant.
	The J3 (Z) axis is equipped with a brake. Do not forcibly move the J3-axis using an external source when the brake is active. Forcibly moving the axis can reduce accuracy and/or wear the brake, damaging the reduction drive.
	If the robot picks up a workpiece that is charged with static electricity, the electricity is discharged through the hand and the robot arm. To prevent the robot from malfunctioning, make sure to insulate the hand and robot arm. Also, when the robot places a charged workpiece on a device, the electricity is discharged through the device and may cause it to malfunction. Configure the system so that it provides a route for appropriate electrical discharge from charged workpieces. Failure to do so can cause malfunction or unit breakdown.
	The lubricant is not completely effective at low temperatures. Make sure to use the warmup function to prevent loss of positioning accuracy and to prevent servo errors such as Excessive Margin of Error.
	Take care when affixing tape that has a strong adhesive, such as gummed tape, to the painted surfaces on the robot. The tape may damage the painted surfaces when it is peeled off.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

Robot Unit and Controller

⚠ Danger	
	**When power to the robot is ON, never enter the safety guard or put your head, hands, or any part of your body inside.** Entering the safety guards could result in injury.
	When changing modes or starting a run, first confirm there are **no people inside of the safety guard and there are no obstacles that could interfere with the run.** Entering the safety guards could result in injury.
	**Before performing a run or operation, always check the following:** • **Obstacles** Make sure there are no obstacles or people within the safety guard. • **Installation** Make sure the robot is installed properly, that there are no abnormalities with the robot and the surrounding devices, and that the teaching pendant and tools are in the appropriate places. • Emergency Stop Function Check Make sure the I/O-S circuit (interlock) and emergency stop switch(es) are functioning properly. It is potentially dangerous to operate the robot without making these checks first.
	If entering the safety guards without cutting OFF the power, **always make sure the select switch on the teaching pendant is set to TEACH (Teaching Mode).** If the select switch is set to AUTO (Run Mode), external commands can start the robot while you are inside the safety guards. Failure to adhere to this can cause injury or breakdown.
	If there are any safety devices that you disable while teaching, make sure to enable them after teaching to reestablish full functionality. Example: Enable the interlock device on the safety guard entrance, etc. Failure to do so can result in injury.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Warning	
	**Make sure to use a power source that is in accordance with the range indicated on the rating identification plate.** Failure to do so causes electric shock, fire, or unit breakdown.
	**Do not allow water or oil to come in contact with the unit or power cord.** Contact with water or oil causes electric shock, fire, or unit breakdown. IP Protection Rating: IP20.
	**Make sure that no foreign objects can enter the robot or controller.** In particular, a conductive or flammable foreign matter, such as a screw, metal shard or oil, can cause an explosion or damage the unit, etc.
	**Be sure to turn OFF the unit before inserting or removing cords and cables such as the teaching pendant cable or LAN cable.** Failure to do so can cause electric shock, data loss, unit breakdown, or malfunction.
	**Firmly connect and secure the power cord after checking that the connection area is not covered with dust, etc.** If the power cord is not firmly plugged in, the connectors may heat up and cause a fire.
	**Be sure to unplug the power cord from the power outlet when the unit is not in use for long periods of time.** Dust accumulation may cause fire.
	**If anything unusual occurs, such as a burning smell or unusual sound, stop operation and turn the power supply circuit breaker OFF. Unplug the power cord after confirming there is no power supplied to the robot and then contact Janome or a Janome dealer.** Continuing to use the robot without addressing the problem causes electric shock, fire, or unit breakdown.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Caution	
	**Be sure to check the connections of the cords and cables to the main unit.** Improper connection may result in unit malfunction or breakdown.
	Make sure to save if you modify any settings or data. If you turn OFF the power after making changes to robot settings or data without saving, those changes are lost and the robot will revert to its original settings.
	Diagnostic Mode and Mechanical Adjustment Mode are for maintenance personnel* use only.

* Maintenance personnel are individuals who have received maintenance training from Janome or from a Janome dealer.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

Controller

⚠ Warning

Do not touch the terminal block when there is an electrical current present.
Touching the terminal block can result in electric shock or injury.

⚠ Caution

Keep the emergency stop switch within reach of the operator when running or operating the robot.
If you operate the robot without the emergency switch in reach, you may not be able to stop the robot immediately and safely. This is potentially dangerous.

Individual Configuration Information varies for each individual unit even if they are the same model.
Do not use backup data with a different robot.
The robot cannot function normally with backup data from a different robot.

Do not turn OFF the power while the robot is running.
Turning OFF the power during a run can cause the arm to drop down or coast to a stop, which may result in the arm hitting a peripheral device.

When an error occurs, the error code that appears on the 7 segment LED and the teaching pendant display are useful indicators for finding the source of the error. Record the error number and then refer to the operation manual *Maintenance* for the countermeasure.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

Safety Precautions Regarding Maintenance

Robot Unit

⚠ Warning	
	Do not touch or come in contact with any potentially hot components on the robot. Doing so can result in burns and serious accidents. The servomotor may get hot. Do not touch or come in contact with the servomotor while the power is ON, only do so when the power is OFF and after it has cooled down.

⚠ Caution	
	**Check that the mounting screws are always firmly tightened with a periodic inspection (every 3 months or 750 hours of use, depending on how often the robot is in use).** A loose tool can cause injury or breakdown.
	**Periodically replace the robot battery.** Failure to do so can cause malfunction or breakdown. Replace the battery approximately every 1 year.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

Robot Unit and Controller

⚠ Danger	
	**If entering the safety guards, turn the power source circuit breaker OFF, lock and tag it, and then make sure there is no power supplied to the robot before continuing.** Failure to do so can cause electric shock, injury, or the robot may move unexpectedly.
	**If entering the safety guards, perform your own risk assessment and establish "work regulations", as outlined below, with thorough planning for safety.** • Work regulations should be relevant and appropriate for the type of work, and consist of details such as robot operating procedures and signs to be used between operators. • When creating work regulations, incorporate the opinion of operators and work safety specialists. Make sure to review and update the contents of the work regulations regularly.
	**When setting the home position or other such work that requires you to enter the safety guards with the power ON, make sure to activate the emergency stop switch before entering the safety guards and to perform the work with the robot in this state.** Failure to do so can result in injury.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Warning

Use protective gear such as a helmet, protective gloves, protective goggles, and safety shoes when performing maintenance.
Failure to do so can cause injury.

When inspecting or performing maintenance on the controller, **make sure there is no electrical current and perform the following:**

- **Lockout/tagout with the power source circuit breaker in the OFF position, and remove the power cords from the terminal block.**
- Do not touch the terminal block within 5 seconds of removing the power cords.

Failure to adhere to this may cause electric shock, injury, data loss or breakdown.

If disassembling this machine, follow the instructions in the operation manual *Maintenance*, and do not dissemble in any ways other than as specified. Do not modify the machine in any way.
Inappropriate disassembly or modification causes electric shock or unit breakdown.

For your safety, **do not modify the machine in any way.**
Inappropriate disassembly or modification causes electric shock or unit breakdown.
We cannot accept any responsibility for machines that malfunction due to modification.

After cutting OFF power to the system, energy stored by the devices is potentially dangerous, so make sure to establish means to reduce this risk.

- Controller
  Make sure the energy stored by the controller is discharged before opening the controller.
- Hand
  Use a hose connected to the solenoid valve to expel compressed air from a pneumatic chuck hand.

⚠ Caution

Perform daily and periodic inspections to check and make sure there are no abnormalities with the robot or peripheral devices. Additionally, keep records of the inspections and store them for 3 years or more so that the details can be referred to for future inspections.

Place a sign such as “Robot Inspection in Progress” in the necessary locations and establish means so as to prevent operation of the robot by those who may be unaware of the maintenance work. Whenever possible, also perform maintenance with at least one other person present so as to stop any unexpected robot movements.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

⚠ Caution	
	For a smooth and long operating life, **lubricate the shaft once for every 2,000 km the robot is run.** If the robot is run for 24 hour periods, lubricate the machine more frequently because the running time between lubrication periods is longer. The lubrication periods are calculated based on runs at maximum speed.
	Make sure to use the specified grease. Use of grease other than the specified grease can adversely affect the robot's performance or cause breakdown.
	If the arm joints or the Z-axis is subject to only minute operational angles or distances, fretting may occur in the internal robot bearings. Fretting refers to wear that occurs when minute operation angles do not allow the lubricant within the bearings to fully coat the working parts as required. This is also applies to axes that are inactive, as the counterforce from other axis operations or vibrations from the robot mounting surface cause minute movements in the inactive axes, which may result in fretting damage. To prevent fretting damage, we recommend running the arm joints more than 30 degrees and the Z-axis more than 20 mm per day.
	Do not charge, dispose of in a fire, or reuse the robot unit battery or controller battery (unit) in any way.
	The robot unit battery and controller battery (unit) are considered industrial waste. Make sure they are disposed of in accordance with the national and/or local authority laws and legislations.

FOR YOUR SAFETY

■■■■■■■■■■■■■■■■■■ JS3 Series ■■■■■■■■■■■■■■■■■■

Controller

⚠ Danger

Before entering the safety guard because of something wrong with the robot or a peripheral device, or to **inspect or lubricate the machine etc., always make sure to turn the controller and power source circuit breakers OFF, lock and tag them, and make sure there is no electrical current.**
Failure to follow these steps can cause electric shock or injury.

⚠ Warning

When replacing fuses, or inspecting or lubricating the unit, turn OFF the power supply, then remove the cord from the main unit and make sure there is no power supplied to the robot before continuing.
Also, **do not touch the terminal block within 5 seconds of removing the power cords.** Failure to follow these steps can cause electric shock or injury.

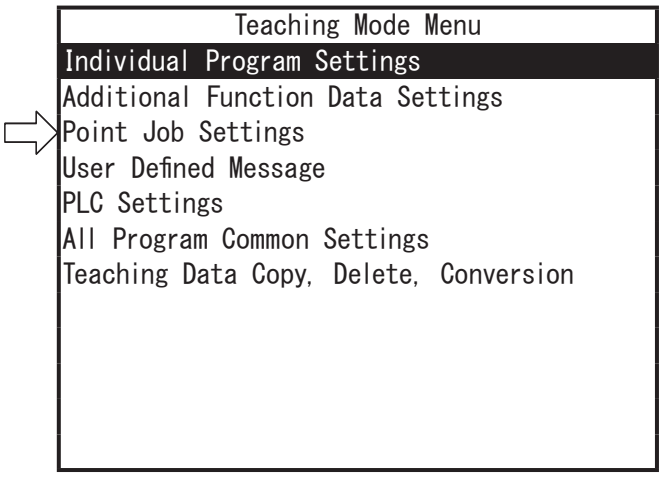
1. CREATING POINT JOB DATA

You can create point job data using the teaching pendant connected to the robot or using an external device connected to the robot via an Ethernet cable.

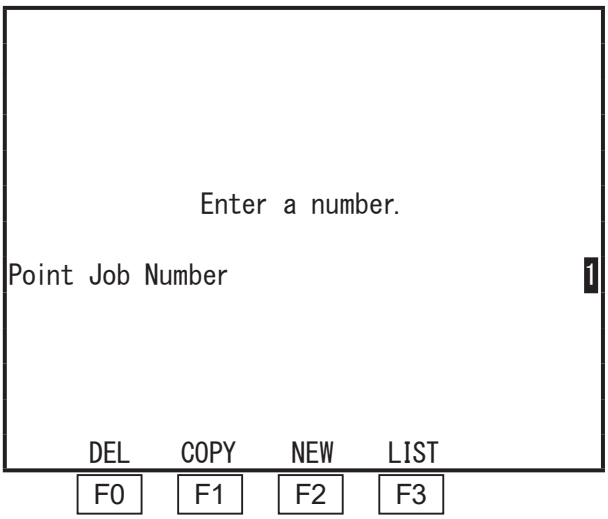
1.1 Via the Robot Teaching Pendant

To create point job data, you need to first display the input screen. You can use either of the two procedures below to display the input screen.

1. Display the point job data input screen
   Press the [MENU] key and then select [Point Job Settings] from the Teaching Mode Menu.



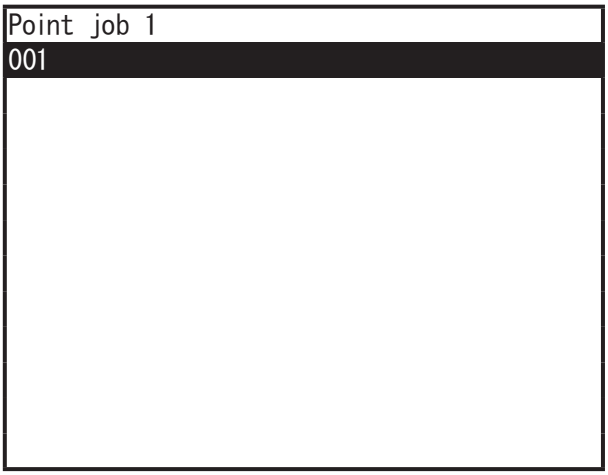
A point job entry screen is displayed. Enter the number for the point job data you want to create, and press the [ENTR] key. The point job entry screen appears.
On the number entry screen, press the [F2] (NEW) key to display a list of unused point job numbers. Press the [F3] (LIST) key to display point job numbers with teaching data already assigned. You can also select a number from these lists.



NOTE: From the Teaching Mode base screen, if you press the [CURSOR ▽] key and open the point job number settings screen for jobs such as [Job Before Moving], [Job While Moving], and [Job After Moving], the [F4] (VIEW) key is displayed. From the screen displaying the [F4] (VIEW) key, enter a number and press the [ENTR] key to set a point job data number to a point. Note this does not take you to the point job entry screen.

When starting from a point job number setting screen such as one for [Job Before Moving], [Job While Moving], and [Job After Moving], press the F4 (VIEW) key or press the F2 (NEW) key to select a point job number from a list of un-entered point job numbers to bring up the point job data entry screen.

Also, by registering a number with your created point job data to a point, the robot performs this job when it is run. Assign the numbers of your created point job data to points.

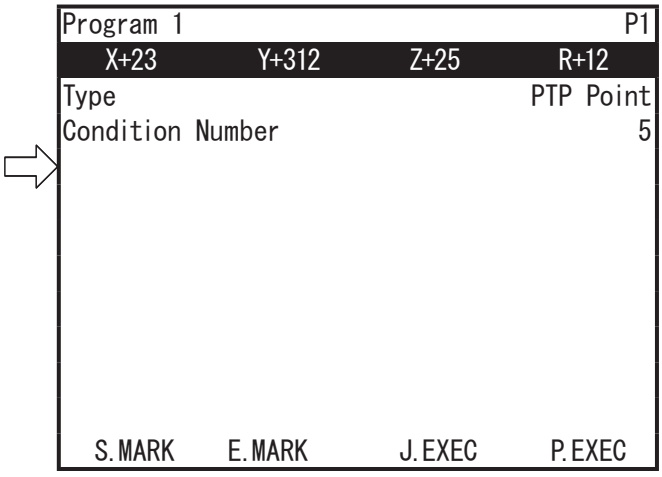


Point Job Data Entry Screen: New

2. Display the point job data input screen

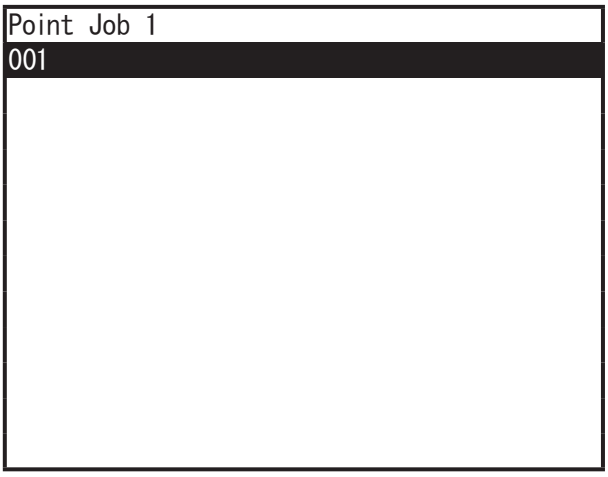
Move the cursor (highlight-bar) to an empty line on the point setting value display screen to show a list of the point jobs and additional functions you can assign to the current point.

Select any of the following: [Job Before Moving], [Job While Moving], [Job After Moving], [Job While CP Moving].

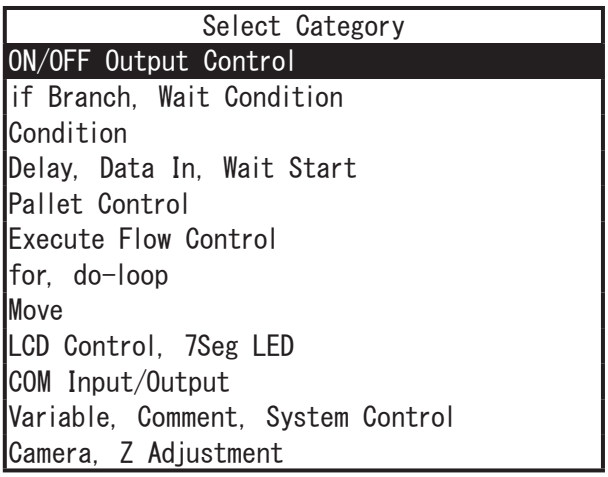


■ Inputting Point Job Data

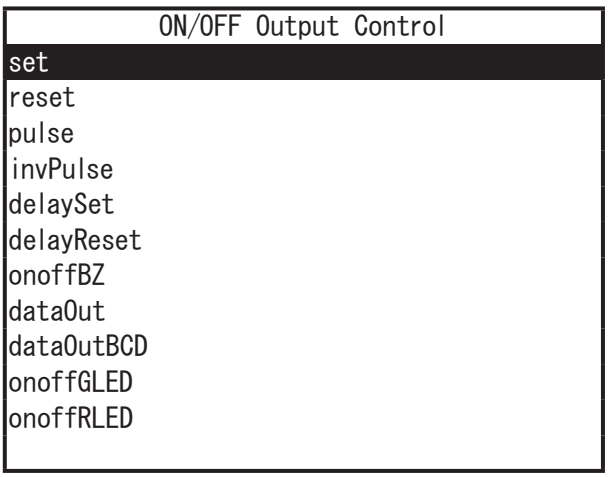
The point job data input screen appears as shown on the right when creating new point job data. [001] is a command number. Select command number [001]. To modify existing point job data, select the command line you want to modify.



Select a command number and the point job command category screen appears (right).



Once a command category is chosen, the command selection screen appears. The diagram on the right is an example of the [ON/OFF Output Control] selection. Select the command you want to enter. A selection screen for that command's necessary parameters appears. For example, if the [set] command is chosen, an output destination selection screen is displayed. Depending on the command, there are times when multiple parameters are necessary and times when there are none.



Enter/select the necessary parameters and the command input is completed in a single line. Repeat the same process to enter more commands.

2. EXPRESSION STRUCTURE

2.1 Expressions

Constant numbers, variables, and functions (string type and numeric type) combined with operators are called expressions.

2.2 Constant Numbers

There are two types of constant numbers, numeric types (e.g. 125, 2.0, 2e15) and string types (e.g. “ABC”). Numeric type constants are handled as 8 byte real type (double type), and string type constant numbers are handled as 255 byte.

For string type constant numbers, characters can be specified in hexadecimal code using the % symbol.

Example: eoutCOM port2, “%0D%0A” : CR ・ LF code output.

If there is any character other than 0 – 9, A – F, and a second % symbol after the first %, the second % is treated as a character.

Example: eoutCOM port2, “%G01” : Output as %G01.

If there is any character from 0 – 9 and A – F, enter %% as is to output %.

Example: eoutCOM port2, “%%300” : Output as %300.

2.3 Variable

A variable is a receptacle which numeric and string values are placed.
With this robot, you can use the built-in variables (which are built into the robot as a function) and the user-defined variables (which you can define).
Within user-defined variables there are local variables (variables effective only in defined point job data which are defined by the *declare* command), global variables, keeping variables, common setting variables, condition setting variables, and program setting variables

Boolean type (boo): 1 bit variable which only holds 1 (true) or 0 (false)
Numeric type (num): 8 byte real type (double type) variable
String type (str): 255 byte variable

2.4 Functions

A function is a setup which processes and returns numerical and string values sent to the robot. You can use the built-in functions (which are built into the robot as a function) and the user-defined functions (which you can define).
Depending on the format of the values returned, there are numerical type functions and character string type functions.

2.5 Operators

Operator	Description	Value*
+	Adds the left values to the right values.	num
-	Deducts the right values from the left values.	num
*	Multiplies the left values by the right values.	num
/	Divides the left values by the right values.	num
&	Combines the left values with the right values. e.g. “A” & “B” → “AB”	str
=	Assigns the right values to the left values.	num, str
>	Returns 1 if the left value is larger than the right value. Returns 0 if the left value is smaller than or the same as the right value.	num, str
<	Returns 1 if the left value is smaller than the right value. Returns 0 if the left value is larger than or the same as the right value.	num, str
>=, =>	Returns 1 if the left value is larger than or the same as the right value. Returns 0 if the left value is smaller than the right value.	num, str
<=, =<	Returns 1 if the left value is smaller than or the same as the right value. Returns 0 if the left value is larger than the right value.	num, str
<>, ><	Returns 1 if the left value is not equal to the right value. Returns 0 if they are equal.	num, str
==	Returns 1 if the left value is equal to the right value. Returns 0 if they are not equal.	num, str

* Numerical value (num), character string (str)

■ The priority of operators is as follows:

1. Expressions in brackets
2. Functions and variables
3. Independent “+” and “ – “
4. “*” and “/”
5. “+”, “ – “, and “&”
6. Relational operators (“>”, “>=”, “=>”, “=<”, “<=”, “<”, “<>”, “><”)
7. Assignment operators (“=”)

3. COMMAND LIST

If you assign point job data that includes a highlighted ( ) command to a CP passing point, the command is ignored.

3.1 Point Job Data

Category	Command	Necessary Parameter	Description
ON/OFF Output Control	set	Output Destination	Output ON.
	reset	Output Destination	Output OFF.
	pulse	Output Destination, Pulse Width	ON output pulses of predetermined length.
	invPulse	Output Destination, Pulse Width	OFF output pulses of predetermined length.
	delaySet	Output Destination, Delay Time	ON output after the predetermined delay time.
	delayReset	Output Destination, Delay Time	OFF output after the predetermined delay time.
	onoffBZ	ON Time, OFF Time	Sound the buzzer intermittently.
	onoffGLED	ON Time, OFF Time	The green LED in the following locations blink: • On the robot front (JR3000 Series only) • On the switchbox (JR3000/JC-3 Series only) • On the controller front or operation box (JS3 Series only)
	onoffRLED	ON Time, OFF Time	The red LED in the following locations blink: • On the robot front (JR3000 Series only) • On the switchbox (JR3000/JC-3 Series only) • On the controller front or operation box (JS3 Series only)
	dataOut	Output Value, Output, Output Bit Number	Outputs numeric data or a tag code assigned to a point to the I/O.
	dataOutBCD	Output Value, Output, Output Bit Number	Outputs numeric data or a tag code assigned to a point to the I/O in BCD.
	motorPowerON	-	Turns the motor power ON. (JS3 Series only)
	servoON	Specified axis	Turns the servomotor ON for the specified axis. (JS3 Series only)
	servoOFF	Specified axis	Turns the servomotor OFF for the specified axis. (JS3 Series only)

Category	Command	Necessary Parameter	Description
if Branch, Wait Condition	if	-	Conditional branching
	then	-	Perform if true.
	else	-	Perform if false.
	endIf	-	End of conditional branching
	waitCondTime	Wait Time	Wait for conditions for a designated period.
	timeUp	-	Perform when time is up.
	endWait	-	End of wait condition
	waitCond	-	Wait for conditions.
Condition	ld	Boolean variable or Expression	Input ON.
	ldi	Boolean variable or Expression	Input OFF.
	and	Boolean variable or Expression	Serial input ON.
	ani	Boolean variable or Expression	Serial input OFF.
	or	Boolean variable or Expression	Parallel input ON.
	ori	Boolean variable or Expression	Parallel input OFF.
	anb	–	Serial block connection
	orb	–	Parallel block connection
Delay	delay	Delay Time	Stop for the exact specified time.
	dataIn	Numeric Variable Name, Input Source, Input Bit Number	Read numeric data from the I/O.
	dataInBCD	Numeric Variable Name, Input Source, Input Bit Number	Read numeric data in BCD from the I/O.
	waitStart	–	Wait for a start instruction.
	waitStartBZ	–	Wait for a start instruction while sounding the buzzer intermittently.
Pallet	loopPallet	Pallet Routine Number, go Point Number	Pallet loop
	resPallet	Pallet Routine Number	Reset the pallet counter.
	incPallet	Pallet Routine Number	Increase the pallet counter number. (+1)

Category	Command	Necessary Parameter	Description
Execute Flow Control	callBase	–	At a user-defined point with a point job number set to it, call the point job defined by that point type.
	callJob	Point Job Number	Subroutine call point job data specified by number.
	callPoints	Point String Identifier	Perform a specified point string (defined in Customizing Mode).
	returnJob	–	End of point job
	returnFunc	Expression	Terminate the function by assigning the value of the specified expression as a return value. (This command is valid with functions only.)
	callProg	Program Number	Subroutine call a program specified by number.
	endProg	–	End of program
	goPoint	Movement Condition Number, Point Number	Jump to a specified point.
	goRPoint	Movement Condition Number, Relative go Point Number	Jump to a relatively-specified point.
	goCRPoint	PTP Movement Condition Number, Select Destination	Jump to a selected destination during a CP movement.
	jump	Jump destination, Label Number	Jump to a specified label.
	Label	Label Number	A label (a jump destination marker)
For, do-loop	for	Control Variable, Initial Value, End Value, Step Value	Repeat commands between *for* and *next* until the specified variable changes from the initial value to the end value.
	next	–	
	exitFor	–	Break from *for* loop.
	do	–	Repeat commands between *do* and *loop*.
	loop	–	
	exitDo	–	Break from *do* loop.

Category	Command	Necessary Parameter	Description
Move	upZ	Speed, Distance	Raise Z
	downZ	Speed, Distance	Descend Z
	movetoZ	Speed, Z movement pos.	Move Z
	lineMove	Speed, Movement/Rotation Distance of Each Axis	Make an axis move a specified distance (relative distance) at a specified speed with a CP line movement (relative move command). Entering this command displays the specified speeds, and distances of each direction as follows: Example: lineMoveSpeed 20 lineMoveX 10 lineMoveY 20 lineMoveZ 0 lineMoveR 0
	lineMoveStopIf	–	Terminate a *lineMove* command movement in the middle made if the conditions are met.
	endLineMove	–	End of *lineMoveStopIf* condition statements.
	initMec	Specified Axis	Initialize the specified axis.
	checkPos	–	Detect a position error (JR3000 only).
	monoMove	Specified Axis	Makes movement for 1 specified axis. You can specify the axis from among the X (J1), Y (J2), Z (J3), R (J4) and the auxiliary MT1 and MT2 axes.
	mMoveDistance	Distance	This specifies the distance for movement using the monoMove command. The unit parameter varies depending on the axis specified.
	mMoveSpeed	Speed	Specifies the speed for movement using the monoMove command. The unit parameter varies depending on the axis specified.
	mMoveAccel Rate	Acceleration rate (%)	These specify the acceleration for movement using the monoMove command. Specify either the mMoveAccelRate or mMoveAccelTime command. With the mMoveAccelRate command, acceleration is specified as a percentage (%) of the default acceleration. With the mMoveAccelTime command, acceleration is specified as the time (msec) it takes to reach the speed specified for the mMoveSpeed command.
	mMoveAccel Time	Acceleration time (msec)	

Category	Command	Necessary Parameter	Description
Move	monoMoveStopIf	-	This ends the movement made by the monoMove command when the conditions are met. You only need to input this command when using conditions to stop the movement.
	endMonoMove	-	This indicates the end of the movement for the monoMove command.
LCD/7SLED	clrLCD	-	Clear the LCD display.
	clrLineLCD	Clear Line (1 – 13)	Clear a specified line on the LCD display.
	outLCD	Display Line (1 – 13), Display Column (1 – 40), Display Data	Display strings on the LCD display.
	eoutLCD	Display Line (1 – 13), Display Column (1 – 40), Display Data	Display the expression result on the LCD display.
	sys7SLED	-	Return the 7 segment LED display changed by *out7SLED*.
	out7SLED	Type, Output Value	7 segment LED output.
COM/Ethernet I/O	outCOM	Port, Character String	Character string output from COM and Ethernet.
	eoutCOM	Port, Character String Expression	Equation result output from COM and Ethernet.
	setWTCOM	Port, Wait Time	Set a COM and Ethernet wait time (time-out period) for receiving data.
	inCOM	Variable Name, Port, Character Length	Assign the received data from COM and Ethernet to a specified variable.
	cmpCOM	Port, Character String	Compare the received data from COM and Ethernet and the string. The result is entered into the system flags (*sysFlag*(1) – *sysFlag*(15)).
	ecmpCOM	Port, Character String Expression	Compare the received data from COM and Ethernet and string expression. The result is entered into the system flags (*sysFlag*(1) – *sysFlag*(15)).
	clrCOM	Port	Clear the COM and Ethernet receive buffer.
	shiftCOM	Port, Shift Number	Shift data received from COM and Ethernet. Delete the amount of byte data shifted, from the top.

Category	Command	Necessary Parameter	Description
COM/Ethernet I/O	stopPC	-	Stop COM1 and Ethernet communication. Communication is not made until the power is cycled or the startPC command is received.
	startPC	-	Start COM1 and Ethernet communication. Communication is made possible when the power is turned ON so this command is not needed.
	connect	Port, numerical value	Connect to the communication destination (refer to “21.1.2 Connection Process.”)
	disconnect	Port, numerical value	Disconnect from the communication destination (refer to “21.1.2 Connection Process.”)
Variable, Comment, System Control	declare	Variable Type, Identifier	Local variable declaration.
	let	Assigned Character String Expression	Assign the right side sum results of the expression to the left side variables. The symbols +, − , *, /, =, (,), & can be used.
	rem	Character String	One line comment
	crem		End of line comment (only displayed when decompiled)
	setProgNum	Program Number	Change the program number. NOTE: Do not carry out this command while the robot is running. Use the command *callProg* if you want to change programs during a run.
	setSeqNum	PLC Program Number	Change the PLC program number in “system data”.
Camera, Z Adjustment	cameraWadj	Work Adjustment Number	Acquire an image with the camera and calculate the offset values according to the [Workpiece Adjustment] settings.
	wCameraWadj	Work Adjustment Number, Shot Number	Use this command when calculating [Workpiece Adjustment] offset values from two camera images.
	multiCamWadj	Work Adjustment Number	Execute camera with counter adjustment.
	incMultiCam Wadj	Work Adjustment Number	Increment the camera with counter adjustment sub-counter values.
	clrMultiCam Wadj	Work Adjustment Number	Clear the sub-values of the camera adjustment with counter.

Category	Command	Necessary Parameter	Description
Camera, Z Adjustment	cameraTool	Tool Data Number	Acquire an image with the camera and calculate [TCP-X] and [TCP-Y] from the data gained according to [Point Tool Data Settings].
	cameraPallet	Pallet Routine Number	Acquire an image with the camera and set the number of marks and coordinates acquired as the routine number and coordinates for the [Pallet Routine].
	takeZWadj	Work Adjustment Number	Calculate the Z offset from the data gained from the distance/touch sensor according to the [Workpiece Adjustment] settings.
	multiTakeZ Wadj	Work Adjustment Number	Execute the content set in the Z adjustment menu of the CCD camera adjustment with counter settings. The adjustment amount is recorded in the workpiece adjustment counter each time this is executed.

NOTE: When the start channel is set to anything other COM1, COM1 communication operations are handled differently depending on the system software version.

- System Software Versions 6 or Lower (JR3000/JC-3 only)
  All communication command functions for COM1 are disabled when the start channel is set to anything other [COM1].
- System Software Versions 6 or Higher
  Communication commands not related to movements are enabled even if the start channel is set to anything other [COM1]. If you want to process arbitrary communication with COM1 using commands such as *inCOM* and *outCOM*, use *stopPC* in advance to stop other communication functions. If you do not use *stopPC* to stop other communication functions, you cannot communicate properly with the robot and you will receive an error response from the input of character strings arbitrarily defined by you.

NOTE: For information regarding the Camera, Z Adjustment commands, refer to the operation manual *Camera & Sensor Functions* (JC-3 Series only).
For information regarding the Mono Movement commands, refer to the operation manual *Auxiliary Axis Functions* (JR3000/JC-3 Series only). The Mono Movement commands are only valid when using a robot equipped with the I/O-MT connector. (The I/O-MT connector is optional for all models. Select between the I/O-1 connector or I/O-MT connector for JR3200 Series models).

3.2 Execute Condition

Category	Command	Necessary Parameter	Description
Condition	ld	Boolean variable or expression	Input ON.
	ldi	Boolean variable or expression	Input OFF.
	and	Boolean variable or expression	Serial input ON.
	ani	Boolean variable or expression	Serial input OFF.
	or	Boolean variable or expression	Parallel input ON.
	ori	Boolean variable or expression	Parallel input OFF.
	anb	–	Serial block connection
	orb	–	Parallel block connection

3.3 PLC Programs

Category	Command	Necessary Parameter	Description
Calculate	ld	Boolean variable	Input ON.
	ldi	Boolean variable	Input OFF.
	and	Boolean variable	Serial input ON.
	ani	Boolean variable	Serial input OFF.
	or	Boolean variable	Parallel input ON.
	ori	Boolean variable	Parallel input OFF.
Coil	out	Output Destination	Coil movement
	set	Output Destination	Hold operation output.
	reset	Output Destination	Hold operation release
	pls	Output Destination	Rising pulse output.
	plf	Output Destination	Falling pulse output.
Connection	anb	–	Parallel circuit serial block connection
	orb	–	Serial circuit parallel block connection
	mps	–	Mid-calculation result storage
	mrd	–	Mid-calculation result readout
	mpp	–	Mid-calculation result readout and reset
Others	nop	–	No operation

4. VARIABLE LIST

You can use built-in variables (which are built into the robot as a function), and user-defined variables (which can be freely defined by the user).

■ User-defined Variables

Within local variables (variables valid only in defined point job data which are defined by the *declare* command) there are global variables, keeping variables, common setting variables, condition setting variables, and program setting variables.

■ Built-in Variables

In the character and expression entry screen, when [BVar] is displayed at the very bottom of the LCD (above the F3 key), press the F3 key to view a list of the built-in variables.

The built-in variables are listed in the following tables.

"Type" refers to the type of variable.

- boo Boolean type, 1 bit variable that holds only 1 (true) and 0 (false).
- num Numeric type, 8 byte real type variable.
- str Character string type. Maximum of 255 bytes.

"Access" refers to the read/write access.

- R Read only type variable.
- W Write only type variable.
- R/W Read and write type variable.

• Free Variable

Type	Identifier	Description	Access	Remarks
boo	#mv (1 to 99)	Boolean variable	R/W	
boo	#mkv (1 to 99)	Boolean variable (keeping variable)	R/W	*
num	#nv (1 to 99)	Numerical variable	R/W	
num	#nkv (1 to 99)	Numerical variable (keeping variable)	R/W	*
str	#sv (1 to 99)	Character string variable	R/W	
str	#skv (1 to 99)	Character string variable (keeping variable)	R/W	*

* Variables which hold their values even if the robot is turned OFF are keeping variables.

• Input Variable

Type	Identifier	Description	Access	Remarks
boo	#sysIn1 to #sysIn16	I/O-SYS input (JR3000/JC-3 Series)	R	
boo	#sysIn1 to #sysIn15	I/O-SYS input (JS3 Series)	R	
boo	#genIn1 to #genIn8	I/O-1 input (JR3000/JC-3 Series)	R	
boo	#genIn1 to #genIn18	I/O-1 input (JS3 Series)	R	
boo	#handIn1 to #handIn8	I/O-H input (JS3 Series only)	R	

num	#fbIn (a, b)
	a=I/O address (0x1000 to 0x17FF)
	b=bit width (1 to 32)
		Fieldbus I/O input	R	*

* Fieldbus I/O variables:

- Variables accessed with a bit width of 1 are Boolean variables. Variables accessed with a bit width of 2 or more are Numerical variables.
- Specify the I/O address as 4 hexadecimal digits continuing on from 0x.
- You can specify a bit width of 1 to 32 and up to 2 words (4 bytes) maximum. However, if the address specified exceeds the output area due to the Fieldbus settings, the exceeded bits are not included.

• Output Variable

Type	Identifier	Description	Access	Remarks
boo	#sysOut1 to #sysOut16	I/O-SYS output (JR3000/JC-3 Series)	W	
boo	#sysOut1 to #sysOut14	I/O-SYS output (JS3 Series)	W	
boo	#genOut1 to #genOut8	I/O-1 output (JR3000/JC-3 Series)	W	
boo	#genOut1 to #genOut22	I/O-1 output (JS3 Series)	W	
boo	#handOut1 to #handOut8	I/O-H output (JS3 Series only)	W	

num	#fbOut (a, b)
	a=I/O address (0x1800 to 0x1FFF)
	b=bit width (1 to 32)
		Fieldbus I/O output	W	*

* Fieldbus I/O variables:

- Variables accessed with a bit width of 1 are Boolean variables. Variables accessed with a bit width of 2 or more are Numerical variables.
- Specify the I/O address as 4 hexadecimal digits continuing on from 0x.
- You can specify a bit width of 1 to 32 and up to 2 words (4 bytes) maximum. However, if the address specified exceeds the output area due to the Fieldbus settings, the exceeded bits are not included.

• System Flag

Type	Identifier	Description	Access	Remarks
boo	#sysFlag(1 to 999)	System flag. Refer to "6. SYSTEM FLAG LIST."	R	

• Hardware

Type	Identifier	Description	Access	Remarks
boo	#optionLED (1 to 3)	Option LEDs	R/W	

• Specialized Variable

Type	Identifier	Description	Access	Remarks
num	#downTimer1 to #downTimer10	A countdown timer (msec unit).	R/W	
num	#jobStartHight	The job start height after moving (mm unit)	R/W	
num	#jobStartX	The job start X (J1) axis position after moving (mm unit)	R/W	
num	#jobStartY	The job start Y (J2) axis position after moving (mm unit)	R/W	
num	#jobStartR	The job R (J4) axis position after moving (deg unit)	R/W	
num	#priorityPTPCondNum	Prioritized PTP condition number	R/W	

• Pallet Routine

Type	Identifier	Description	Access	Remarks
boo	#palletFlag (1 to 100)	Pallet flag	R	*
num	#palletCount (1 to 100)	Pallet counter	R/W	*

* If using a JS3 Series pick and place application model, pallet flags and pallet counters 41 to 100 are reserved by the internal system software. Do not use these flags or counters.

• PLC Program

Type	Identifier	Description	Access	Remarks
boo	#seqT (1 to 99)	PLC timer. Returns 1 (true) when the timer reaches a value greater than the count specified in #seqTCount.	R/W	
num	#seqTCount (1 to 99) 1 to 50: incremental timer 51 to 99: decremental timer	Pallet counter value	R/W	
boo	#seqC (1 to 99)	PLC counter. Returns 1 (true) when the timer reaches a value greater than the count specified in #seqCCount.	R/W	
num	#seqCCount (1 to 99)	PLC counter value	R/W	

• Workpiece Adjustment (additional function data)

Type	Identifier	Description	Access	Remarks
num	#workAdj_X (1 to 3000)	X adjustment amount (mm unit)	R/W	
num	#workAdj_Y (1 to 3000)	Y adjustment amount (mm unit)	R/W	
num	#workAdj_Z (1 to 3000)	Z adjustment amount (mm unit)	R/W	
num	#workAdj_R (1 to 3000)	R adjustment amount (deg unit)	R/W	
num	#workAdj_Rotation (1 to 3000)	Rotate adjustment amount (deg unit)	R/W	

• CCD Camera Workpiece Adjustment with Counter

Type	Identifier	Description	Access	Remarks
num	#mulWorkAdj_Wrt_Cam	Write counter for camera adjustment amount.	R	
num	#mulWorkAdj_Wrt_Zadj	Write counter for Z adjustment amount.	R	
num	#mulWorkAdj_Read	Read counter for adjustment value.	R/W	
num	#mulWorkAdj_Num	Counter for workpiece adjustment number.	R/W	
num	#mulWorkAdj_Result	Counter for data result 0=fail, 1=success	R/W	

• Current Point in Current Program

Type	Identifier	Description	Access	Remarks
num	#point_X	X (J1) axis coordinates (mm unit)	R/W	
num	#point_Y	Y (J2) axis coordinates (mm unit)	R/W	
num	#point_Z	Z (J3) axis coordinates (mm unit)	R/W	
num	#point_R	R (J4) axis coordinates (deg unit)	R/W	
num	#point_MT1	MT1 axis coordinates (arbitrary unit)	R/W	*
num	#point_MT2	MT2 axis coordinates (arbitrary unit)	R/W	*
num	#point_LineSpeed	Line speed (mm/s unit)	R/W	
num	#point_CondNum	Condition number	R/W	
num	#point_MoveBeforeNum	Job Before Moving number	R/W	
num	#point_MovingNum	Job While Moving number	R/W	
num	#point_MoveAfterNum	Job After Moving number	R/W	
num	#point_CPWorkNum	Job while CP Moving number	R/W	
num	#point_PTPCondNum	PTP condition number	R/W	
num	#point_CPCondNum	CP condition number	R/W	
num	#point_ToolNum	Tool data number	R/W	
num	#point_PalletNum	Pallet routine number	R/W	
num	#point_WorkAdjNum	Workpiece adjustment number	R/W	
num	#point_RunCondNum	Execution condition number	R/W	
num	#point_TagCode	Tag code	R/W	

* The unit type used for MT1 and MT2 position coordinates is the unit type set separately in auxiliary axis configuration settings (JR3000/JC-3 Series only).

- Arbitrary Point in Current Program

  a = point number (0 to last point no)

Type	Identifier	Description	Access	Remarks
num	#P_X (a)	X (J1) axis coordinates (mm unit)	R/W	*2
num	#P_Y (a)	Y (J2) axis coordinates (mm unit)	R/W	*2
num	#P_Z (a)	Z (J3) axis coordinates (mm unit)	R/W	*2
num	#P_R (a)	R (J4) axis coordinates (deg unit)	R/W	*2
num	#P_MT1 (a)	MT1 axis coordinates (arbitrary unit)	R/W	*1 *2
num	#P_MT2 (a)	MT2 axis coordinates (arbitrary unit)	R/W	*1 *2
num	#P_LineSpeed (a)	Line speed (mm/s unit)	R/W	*2
num	#P_CondNum (a)	Condition number	R/W	*2
num	#P_MoveBeforeNum (a)	Job Before Moving number	R/W	*2
num	#P_MovingNum (a)	Job While Moving number	R/W	*2
num	#P_MoveAfterNum (a)	Job After Moving number	R/W	*2
num	#P_CPWorkNum (a)	Job while CP Moving number	R/W	*2
num	#P_PTPCondNum (a)	PTP condition number	R/W	*2
num	#P_CPCondNum(a)	CP condition number	R/W	*2
num	#P_ToolNum (a)	Tool data number	R/W	*2
num	#P_PalletNum (a)	Pallet routine number	R/W	*2
num	#P_WorkAdjNum (a)	Workpiece adjustment number	R/W	*2
num	#P_RunCondNum (a)	Execution condition number	R/W	*2
num	#P_TagCode (a)	Tag code	R/W	*2

*1: The unit type used for MT1 and MT2 position coordinates is the unit type set separately in auxiliary axis configuration settings (JR3000/JC-3 Series only).

*2: If you specify the point number 0, you can access to the Work home point data. The access destination depends on the Individual/Common selections. Read and Write are available for [Individual Work Home]. However, [Common Work Home] is Read only; Write is unavailable.

- Arbitrary Point in Arbitrary Program Number

  a = program number (1 to 999), b = point number (0 to last point no)

Type	Identifier	Description	Access	Remarks
num	#prog_P_X (a, b)	X (J1) axis coordinates (mm unit)	R/W	*2
num	#prog_P_Y (a, b)	Y (J2) axis coordinates (mm unit)	R/W	*2
num	#prog_P_Z (a, b)	Z (J3) axis coordinates (mm unit)	R/W	*2
num	#prog_P_R (a, b)	R (J4) axis coordinates (deg unit)	R/W	*2
num	#prog_P_MT1 (a, b)	MT1 axis coordinates (arbitrary unit)	R/W	*1 *2
num	#prog_P_MT2 (a, b)	MT2 axis coordinates (arbitrary unit)	R/W	*1 *2
num	#prog_P_LineSpeed (a, b)	Line speed (mm/s unit)	R/W	*2
num	#prog_P_CondNum (a, b)	Condition number	R/W	*2
num	#prog_P_MoveBeforeNum (a, b)	Job Before Moving number	R/W	*2
num	#prog_P_MovingNum (a, b)	Job While Moving number	R/W	*2
num	#prog_P_MoveAfterNum (a, b)	Job After Moving number	R/W	*2
num	#prog_P_CPWorkNum (a, b)	Job while CP Moving number	R/W	*2
num	#prog_P_PTPCondNum (a, b)	PTP condition number	R/W	*2
num	#prog_P_CPCondNum (a, b)	CP condition number	R/W	*2
num	#prog_P_ToolNum (a, b)	Tool data number	R/W	*2
num	#prog_P_PalletNum (a, b)	Pallet routine number	R/W	*2
num	#prog_P_WorkAdjNum (a, b)	Workpiece adjustment number	R/W	*2
num	#prog_P_RunCondNum (a, b)	Execution condition number	R/W	*2
num	#prog_P_TagCode (a, b)	Tag code	R/W	*2

*1: The unit type used for MT1 and MT2 position coordinates is the unit type set separately in auxiliary axis configuration settings (JR3000/JC-3 Series only).

*2: If you specify the point number 0, you can access to the Work home point data. The access destination depends on the Individual/Common selections. Read and Write are available for [Individual Work Home]. However, [Common Work Home] is Read only; Write is unavailable.

• Tool Data for All Program Common Settings

Type	Identifier	Description	Access	Remarks
num	#comm_ToolData_Mass	Tool mass (select tool mass no.)	R/W	*
num	#comm_ToolData_X	TCP-X (mm unit)	R/W	
num	#comm_ToolData_Y	TCP-Y (mm unit)	R/W	
num	#comm_ToolData_DeltaZ	TCP-deltaZ (mm unit)	R/W	

* The tool mass numbers and kg mass unit varies depending on the model you are using, as shown below:

	JR3200	JR3300 - JR3600	JR3303F JR3403F	JC-3 3 axis single sided	JC-3 3 axis double sided	JC-3 4 axis double sided	JS3
0	1 kg	1 kg	1 kg	4 kg	8 kg	3 kg	1 kg
1	3.5 kg	4 kg	5 kg	-	-	-	3 kg
2	-	7 kg	10 kg	-	-	-	6 kg
3	-	-	15 kg	-	-	-	-

• PTP Condition for All Program Common Settings

Type	Identifier	Description	Access	Remarks
num	#comm_PTPData_Speed	PTP speed (% unit)	R/W	
num	#comm_PTPData_R_Speed	R axis rotate speed (% unit)	R/W	
num	#comm_PTPData_R_Acc	R axis rotate acceleration (% unit)	R/W	
num	#comm_PTPData_Archmotion	Arch motion 0 = Z movement relative distance specification. 1 = Z movement absolute position specification.	R/W	
num	#comm_PTPData_Z_Height	Z movement height (mm unit)	R/W	
num	#comm_PTPData_Z_Up_Dis	Z axis up distance (mm unit)	R/W	
num	#comm_PTPData_Z_Down_Dis	Z axis down distance (mm unit)	R/W	
num	#comm_PTPData_Move_Dis_Pos	Horizontal movement pos. (mm unit)	R/W	
num	#comm_PTPData_Move_Start_Pos	Horizontal movement starting pos. (mm unit)	R/W	
num	#comm_PTPData_Down_Start_Pos	Down starting pos. (mm unit)	R/W	

• CP Conditions for All Program Common Settings

Type	Identifier	Description	Access	Remarks
num	#comm_CPData_Acc	CP acceleration (% unit)	R/W	
num	#comm_CPData_R_Speed	R axis rotate speed (% unit)	R/W	
num	#comm_CPData_R_Acc	R axis rotate acceleration (% unit)	R/W	

• Tool Data for Individual Program Settings

Type	Identifier	Description	Access	Remarks
num	#prog_ToolData_EachCommon (1 to 999)	Common/individual selection 0 = common, 1 = individual	R/W	
num	#prog_ToolData_Mass (1 to 999)	Tool mass (select tool mass no.)	R/W	*
num	#prog_ToolData_X (1 to 999)	TCP-X (mm unit)	R/W	
num	#prog_ToolData_Y (1 to 999)	TCP-Y (mm unit)	R/W	
num	#prog_ToolData_DeltaZ (1 to 999)	TCP-deltaZ (mm unit)	R/W	

* The unit type used for MT1 and MT2 position coordinates is the unit type set separately in auxiliary axis configuration settings (JR3000/JC-3 Series only).

• PTP Condition for Individual Program Settings

Type	Identifier	Description	Access	Remarks
num	#prog_PTPData_EachCommon(1 to 999)	Common/individual selection 0 = common, 1 = individual	R/W	
num	#prog_PTPData_Speed (1 to 999)	PTP speed (% unit)	R/W	
num	#prog_PTPData_R_Speed (1 to 999)	R axis rotate speed (% unit)	R/W	
num	#prog_PTPData_R_Acc (1 to 999)	R axis rotate acceleration (% unit)	R/W	
num	#prog_PTPData_Archmotion (1 to 999)	Arch motion 0 = Z movement relative distance specification. 1 = Z movement absolute position specification.	R/W	
num	#prog_PTPData_Z_Height (1 to 999)	Z movement height (mm unit)	R/W	
num	#prog_PTPData_Z_Up_Dis (1 to 999)	Z axis up distance (mm unit)	R/W	
num	#prog_PTPData_Z_Down_Dis (1 to 999)	Z axis down distance (mm unit)	R/W	
num	#prog_PTPData_Move_Dis_Pos (1 to 999)	Horizontal movement pos. (mm unit)	R/W	
num	#prog_PTPData_Move_Start_Pos (1 to 999)	Horizontal movement starting pos. (mm unit)	R/W	
num	#prog_PTPData_Down_Start_Pos (1 to 999)	Down starting pos. (mm unit)	R/W	

• CP Condition for Individual Program Settings

Type	Identifier	Description	Access	Remarks
num	#prog_CPData_EachCommon (1 to 999)	Common/individual selection 0 = common, 1 = individual	R/W	
num	#prog_CPData_Acc (1 to 999)	CP acceleration (% unit)	R/W	
num	#prog_CPData_R_Speed (1 to 999)	R axis rotate speed (% unit)	R/W	
num	#prog_CPData_R_Acc (1 to 999)	R axis rotate acceleration (% unit)	R/W	

• Tool Data for Additional Point Function

Type	Identifier	Description	Access	Remarks
num	#tool_Mass (1 to 100)	Tool mass (select tool mass no.)	R/W	*
num	#tool_X (1 to 100)	TCP-X (mm unit)	R/W	
num	#tool_Y (1 to 100)	TCP-Y (mm unit)	R/W	
num	#tool_Z (1 to 100)	TCP-deltaZ (mm unit)	R/W	
num	#tool_R(1 to 100)	R axis rotate amount (deg unit) (when using [Set TCP by Camera])	R/W	

* The tool mass numbers and kg mass unit varies depending on the model you are using, as shown below:

	JR3200	JR3300 - JR3600	JR3303F JR3403F	JC-3 3 axis single sided	JC-3 3 axis double sided	JC-3 4 axis double sided	JS3
0	1 kg	1 kg	1 kg	4 kg	8 kg	3 kg	1 kg
1	3.5 kg	4 kg	5 kg	-	-	-	3 kg
2	-	7 kg	10 kg	-	-	-	6 kg
3	-	-	15 kg	-	-	-	-

• PTP Condition for Additional Point Function

Type	Identifier	Description	Access	Remarks
num	#ptp_Speed (1 to 100)	PTP speed (% unit)	R/W	
num	#ptp_R_Speed (1 to 100)	R axis rotate speed (% unit)	R/W	
num	#ptp_R_Acc (1 to 100)	R axis rotate acceleration (% unit)	R/W	
num	#ptp_Archmotion (1 to 100)	Arch motion 0 = Z movement relative distance specification. 1 = Z movement absolute position specification.	R/W	
num	#ptp_Z_Height (1 to 100)	Z movement height (mm unit)	R/W	
num	#ptp_Z_Up_Dis (1 to 100)	Z axis up distance (mm unit)	R/W	
num	#ptp_Z_Down_Dis (1 to 100)	Z axis down distance (mm unit)	R/W	
num	#ptp_Move_Dis_Pos (1 to 100)	Horizontal movement pos. (mm unit)	R/W	
num	#ptp_Move_Start_Pos (1 to 100)	Horizontal movement starting pos. (mm unit)	R/W	
num	#ptp_Down_Start_Pos (1 to 100)	Down starting pos. (mm unit)	R/W	

• CP Condition for Additional Point Function

Type	Identifier	Description	Access	Remarks
num	#cp_Acc (1 to 100)	CP acceleration (% unit)	R/W	
num	#cp_R_Speed (1 to 100)	R axis rotate speed (% unit)	R/W	
num	#cp_R_Acc (1 to 100)	R axis rotate acceleration (% unit)	R/W	

• Model Information, etc.

Type	Identifier	Description	Access	Remarks
num	#Model_Series	Series information. “3” is always returned.	R	
num	#Model_AxisNum	Number of mechanical axes: 2 = 2 axis model (X, Y) 3 = 3 axis model (X, Y, Z) 4 = 4 axis model (X, Y, Z, R) (J1, J2, J3, J4)	R	
num	#Model_Info	Model information: 0 = Desktop Robot JR3000 1 = SCARA Robot JS3 2 = Cartesian Robot JC-3	R	
num	#Model_AuxAxis	Auxiliary axis (I/O-MT) existence: 0 = no 1 = yes	R	
num	#Model_Language	Current language setting: 0 = English, 1 = Japanese, 2 = German, 3 = Italian, 4 = Spanish, 5 = French, 6 = Korean, 7 = Simplified Chinese, 8 = Czech, 9 = Vietnamese 10 = Traditional Chinese	R	

5. FUNCTION LIST

You can use built-in functions (which are built into the robot as a function) and user-defined functions.

User-defined Functions: These are defined in Teaching Mode or Customizing Mode. (Refer to the operation manual *Functions IV*.)

Built-in Functions: In the character and expression entry screen, when [BFunc] is displayed at the very bottom of the LCD screen (above the F2 key), press the F2 key to view a list of the built-in functions.

x, y: Numerical value or numerical variable

n, m: Round the numeric value up or off to the specified digit(s)

a, b: String or string variable

Category	Type	Identifier	Description
Robot System	num	currentMainProgNumber()	Currently performed main program number
	num	currentSubProgNumber()	Currently performed sub program number
	num	currentPointNumber()	Currently performed point number
	num	currentArmX()	Current X coordinate [mm]
	num	currentArmY()	Current Y coordinate [mm]
	num	currentArmZ()	Current Z coordinate [mm]
	num	currentArmR()	Current R coordinate [deg]
	num	currentArmH()	Current arm coordinate system (1: righty − 1: lefty) NOTE: This is fixed as 1:righty for JR3000/JC-3 series.
	num	currentCmdArmX()	Current command X coordinate [mm]
	num	currentCmdArmY()	Current command Y coordinate [mm]
	num	currentCmdArmZ()	Current command Z coordinate [mm]
	num	currentCmdArmR()	Current command R coordinate [deg]
	num	numCOM (COM port number)	Data byte count of COM receiving port
	num	moveAPTP (num a, num b, num X, num Y, num Z, num R)	Function of PTP movement to a designated absolute position. The robot makes a PTP movement to a specified position.
	num	moveRPTP (num a, num b, num X, num Y, num Z, num R)	Function of PTP movement to a designated relative position. The robot moves by a PTP movement from the current position to a remote position by exactly the specified distance.
	num	isConditionData(n)	Displays whether the specified condition data number is available (1) or not (0).
	str	strCenterLCD(a)	Adjusts the strings on the teaching pendant LCD (centering).

x, y: Numerical value or numerical variable

n, m: Round the numeric value up or off to the specified digit(s)

a, b: String or string variable

Category	Type	Identifier	Description
Robot System	str	strRightLCD(a)	Adjusts the strings on the teaching pendant LCD (right justification).
	str	strPlusRLCD(a, b)	Teaching pendant LCD: Right priority; items on the right are displayed in full if there is an overlap.
	str	strPlusLLCD(a, b)	Teaching pendant LCD: Right priority; items on the right are displayed in full if there is an overlap.
	num	getSystemPTPmoveTime()	Valid only for [Job while Moving]. Time required for the current PTP movement [sec]
	num	getSystemPTPrestTime()	Valid only for [Job while Moving]. Time left before the current PTP movement ends (time until arriving at the destination) [sec]
	num	Pause(X)	Pause cannot be performed halfway through a movement. The argument (x) in the brackets is the *pause* number for when executing Reference Value.
	str	getUserMessage(num x)	Acquire the message character string defined by number and specified by x.
	num	addPointSkip()	Register a specified point to skip.
	num	delPointSkip()	Clear the skip operation for a specified point.
	num	clearPointSkip()	Clear all skip operations for the specified points.
	num	addPalletSkip()	Register a specified pallet routine to skip.
	num	delPalletSkip()	Clear the skip operation for a specified pallet routine.
	num	clearPalletSkip()	Clear all skip operations for the specified pallet routines.
	num	qCameraWadj(num X, num Y)	Calculate the workpiece adjustment using 4 camera points.

x, y: Numerical value or numerical variable

n, m: Round the numeric value up or off to the specified digit(s)

a, b: String or string variable

Category	Type	Identifier	Description
Arithmetic System	num	abs(x)	Absolute value
	num	max(x, y)	Maximum value
	num	min(x, y)	Minimum value
	num	degrad(x)	Conversion from degree to radian (x* π /180)
	num	raddeg(x)	Conversion from radian to degree (x*180/ π)
	num	sqrt(x)	Square root
	num	sin(x)	Sine
	num	cos(x)	Cosine
	num	tan(x)	Tangent
	num	atan(x)	Arctangent
	num	atan2(x, y)	Arctangent of the value of *y* divided by *x* (y/x)
	num	int(x)	Maximum integer that does not exceed *x*. e.g. int (1.3) → 1, int (-1.3) → -2
	num	ip(x)	Integer part of *x: sgn (x)*int (abs(x))* *(If x is a negative number, sgn (x) becomes -1. If x is a positive number, sgn (x) becomes +1.)* *e.g. ip (1.3) → 1, ip (-1.3) → -1*
	num	fp(x)	Decimal part of *x: x-ip (x)* e.g. fp (1.3) → 0.3, fp (-1.3) → -0.3
	num	mod(x, y)	Value of *x* modulo *y*: x-y*int (x/y)
	num	remainder(x, y)	Remainder of dividing x by y: x − y*ip (x/y)
	num	pow(x, y)	x to the power of y
String System	str	chr(x)	Returns a string (1 character) with the given character code.
	num	ord(a)	Returns the top character code. Other codes are ignored. Returns 0 when the number of characters for the string set to *a* is 0.
	num	len(a)	Returns the string length (byte length). Not compatible with multi-bytes.
	num	strPos(a, b)	Returns the first part of the string position in *a* that matches *b*.
	str	strMid(a, n, m)	Returns the string from *n* to the amount of *m* counted from the start of string *a*.
	str	str(x)	Converts a numeric value to a decimal digit string.
	str	strBin(n, m)	Converts a numeric value to a binary string. *m*: Number of binary string digits
	str	strHex(n, m)	Converts a numeric value to a hexadecimal string. *m*: Number of hexadecimal string digits

x, y: Numerical value or numerical variable

n, m: Round the numeric value up or off to the specified digit(s)

a, b: String or string variable

Category	Type	Identifier	Description
String System	str	str1SI(x)	Rounds a numeric value to a 1-byte signed integer to convert it to a 1-byte string. (1-byte Signed Integer)
	str	str2SIBE(x)	Rounds a numeric value to a 2-byte signed integer to convert it to a 2-byte string using the Big Endian byte order. (2-byte Signed Integer Big Endian)
	str	str2SILE(x)	Rounds a numeric value to a 2-byte signed integer to convert it to a 2-byte string using the Little Endian byte order. (2-byte Signed Integer Little Endian)
	str	str4SIBE(x)	Rounds a numeric value to a 4-byte signed integer to convert it to a 4-byte string using the Big Endian byte order. (4-byte Signed Integer Big Endian)
	str	str4SILE(x)	Rounds a numeric value to a 4-byte signed integer to convert it to a 4-byte string using the Little Endian byte order. (4-byte Signed Integer Little Endian)
	str	str4FBE(x)	Regards a numeric value as a decimal float to convert it to a 4-byte string using the Big Endian byte order. (4-byte Float Big Endian)
	str	str4FLE(x)	Regards a numeric value as a decimal float to convert it to a 4-byte string using the Little Endian byte order. (4-byte Float Big Endian)
	str	str8DBE(x)	Regards a numeric value as a decimal float to convert it to an 8-byte string using the Big Endian byte order. (8-byte Double Float Big Endian)
	str	str8DLE(x)	Regards a numeric value as a decimal float to convert it to an 8-byte string using the Little Endian byte order. (8-byte Double Float Little Endian)
	num	val(a)	Regards a character string as a decimal digit string to convert it to a numeric value (integer type with no symbol). Returns 0 if the head of the character string is a minus sign.
	num	valBin(a)	Regards a character string as a binary string (sequence of “0”, “1”) to convert it to a numeric value.
	num	valHex(a)	Regards a character string as a hexadecimal string (sequence of “0” – “9”, “A” – “F”, or “a” – “f”) to convert it to a numeric value.
	num	val1SI(a)	Converts the top character to a 1-byte signed integer. (1-byte Signed Integer)

x, y: Numerical value or numerical variable

n, m: Round the numeric value up or off to the specified digit(s)

a, b: String or string variable

Category	Type	Identifier	Description
String System	num	val2SIBE(a)	Converts the top 2 characters to a 2-byte signed integer using the Big Endian byte order. (2-byte Signed Integer Big Endian)
	num	val2SILE(a)	Converts the top 2 characters to a 2-byte signed integer using the Little Endian byte order. (2-byte Signed Integer Little Endian)
	num	val4SIBE(a)	Converts the top 4 characters to a 4-byte signed integer using the Big Endian byte order. (4-byte Signed Integer Big Endian)
	num	val4SILE(a)	Converts the top 4 characters to a 4-byte signed integer using the Little Endian byte order. (4-byte Signed Integer Little Endian)
	num	val4FBE(a)	Converts the top 4 characters to a decimal float using the Big Endian byte order. (4-byte Float Big Endian)
	num	val4FLE(a)	Converts the top 4 characters to a decimal float using the Little Endian byte order. (4-byte Float Little Endian)
	num	val8DBE(a)	Converts the top 8 characters to a double-precision decimal float using the Big Endian byte order. (8-byte Double Big Endian)
	num	val8DLE(a)	Converts the top 8 characters to a double-precision decimal float using the Little Endian byte order. (8-byte Double Little Endian)
	num	valSum(a)	Returns the sum of a string code from top to bottom.
	num	valCRC(a)	The remainder of dividing a string (bit string) by a generator polynomial X16+X12+X5+1
	str	bitNot(a)	Bit invert
	str	bitAnd(a, b)	Bit logical conjunction
	str	bitOr(a, b)	Bit logical add
	str	bitXor(a, b)	Bit exclusive disjunction

6. SYSTEM FLAG LIST

You can use system flags as Boolean valuables. If the conditions are met, “1” (true) is automatically assigned to a system flag. If the conditions are not met, “0” (false) is automatically assigned. You can refer to the assigned values whenever necessary.

6.1 JR3000/JC-3 Series

No.	Identifier	Description	Condition “1” (True)
01	#FisCOM1	Existence of COM1 received data	Exists
02	#FltCOM1	Comparison command (cmpCOM) result of COM1 received data	Constant > Receive data
03	#FeqCOM1	Comparison command (cmpCOM) result of COM1 received data	Constant = Receive data
04	#FgtCOM1	Comparison command (cmpCOM) result of COM1 received data	Constant < Receive data
05	#FtimeOutCOM1	Comparison command (cmpCOM) timeout of COM1 received data	Timeout
06	#FisCOM2	Existence of COM2 received data	Exists
07	#FltCOM2	Comparison command (cmpCOM) result of COM2 received data	Constant > Receive data
08	#FeqCOM2	Comparison command (cmpCOM) result of COM2 received data	Constant = Receive data
09	#FgtCOM2	Comparison command (cmpCOM) result of COM2 received data	Constant < Receive data
10	#FtimeOutCOM2	Comparison command (cmpCOM) timeout of COM2 received data	Timeout
11	#FisCOM3	Existence of COM3 received data	Exists
12	#FltCOM3	Comparison command (cmpCOM) result of COM3 received data	Constant > Receive data
13	#FeqCOM3	Comparison command (cmpCOM) result of COM3 received data	Constant = Receive data
14	#FgtCOM3	Comparison command (cmpCOM) result of COM3 received data	Constant < Receive data
15	#FtimeOutCOM3	Comparison command (cmpCOM) timeout of COM3 received data	Timeout
30	#FinitMecError	State of mechanical initialization command error	Mechanical initialization error
31	#FcameraError	State of camera data acquisition error	Error

No.	Identifier	Description	Condition “1” (True)
32	#FtakeZError	State of Z height adjustment (takeZWadj) acquisition error	Error
33	#FlMoveOutRange	Range status of relative move command	Out of range
34	#FlMoveStop	Conditional stop status of relative move command	Stopped by the stop condition
35	#FcheckPosError	Result of the position discrepancy detection command	Position discrepancy error
36	#FdataInBCDError	Error status of *dataInBCD* command	Error
37	#FmultiWadjValError	Shows whether there was an error or not when obtaining the workpiece adjustment amount with the readout counter.	Error
60	#FstartSW	Start/Stop switch	ON (Pressed)
61	#FincSW	Program number selection key (+)	ON (Pressed)
62	#FdecSW	Program number selection key (–)	ON (Pressed)
63	#FemgSW	EMG direct input	ON (The emergency stop switch is pressed.)
64	#Fios	I/O-S direct input	Circuit open (Disconnected)
71	#Fsensor1	X initialization position detection sensor	Blocked
72	#Fsensor2	Y initialization position detection sensor	Blocked
73	#Fsensor3	Z initialization position detection sensor	Blocked
74	#Fsensor4	R initialization position detection sensor	Blocked
76	#Fdrvoz1	X driver 0-phase	ON
77	#Fdrvoz2	Y driver 0-phase	ON
78	#Fdrvoz3	Z driver 0-phase	ON
79	#Fdrvoz4	R driver 0-phase	ON
80	#FpurgeSW	Purge switch	ON (pressed)
81	#FdspRunning	The robot is dispensing	The robot is dispensing
82	#FdspDevRespError	Error status of dispenser response signal	Error
91	#FenableSW	Enable switch	ON (Pressed)
94	#FmotorPower	Motor power status	ON
95	#Finitialize1*	X axis mechanical initialization status	Complete
96	#Finitialize2*	Y axis mechanical initialization status	Complete
97	#Finitialize3*	Z axis mechanical initialization status	Complete
98	#Finitialize4*	R axis mechanical initialization status	Complete
111	#FoptionSW1	Optional switch 1	ON (Pressed)
112	#FoptionSW2	Optional switch 2	ON (Pressed)
113	#FoptionSW3	Optional switch 3	ON (Pressed)
300	#FisEther1	Existence of Ether1 received data	Data exists

No.	Identifier	Description	Condition “1” (True)
301	#FltEther1	Ether1 Receive data comparative results	Constant > Receive data
302	#FeqEther1	Ether1 Receive data comparative results	Constant = Receive data
303	#FgtEther1	Ether1 Receive data comparative results	Constant < Receive data
304	#FtimeOutEther1	Ether1 Receive data comparative results	Timeout
305	#FconnectEther1	Ether1 Connection state	Connected
306	#FisEther2	Existence of Ether2 received data	Data exists
307	#FltEther2	Ether2 Receive data comparative results	Constant > Receive data
308	#FeqEther2	Ether2 Receive data comparative results	Constant = Receive data
309	#FgtEther2	Ether2 Receive data comparative results	Constant < Receive data
310	#FtimeOutEther2	Ether2 Receive data comparative results	Timeout
311	#FconnectEther2	Ether2 Connection state	Connected
312	#FisEther3	Existence of Ether3 received data	Data exists
313	#FltEther3	Ether3 Receive data comparative results	Constant > Receive data
314	#FeqEther3	Ether3 Receive data comparative results	Constant = Receive data
315	#FgtEther3	Ether3 Receive data comparative results	Constant < Receive data
316	#FtimeOutEther3	Ether3 Receive data comparative results	Timeout
317	#FconnectEther3	Ether3 Connection state	Connected

* Precautions Regarding System Flags #Finitialize1 to 4
  - Flags for all axes are false (0) immediately after the power is turned ON.
  - Flags for all axes are false (0) when an emergency stop occurs.
  - If the robot already performed a mechanical initialization, the flag is true (1), and a mechanical initialization is performed again, the flag is false (0) immediately before the mechanical initialization starts.
  - The flags for each of the Z and R axes are true (1) when a mechanical initialization completes.
  - X and Y axes:
    - If [Order of Init.] is set to [Simultaneous], flags 95 and 96 are true (1) when the mechanical initialization for both the X and Y axes completes. For example, if the initialization of the X axis completes and the Y axis is still performing the initialization, both flags 95 and 96 are false (0).

- If [Order of Init.] is set to [X before Y] or [Y before X], the system flag for the respective axis is true (1) when the mechanical initialization completes for that axis.
- If mechanical initialization is performed for 1 axis using the point job command *initMec*, only the flag for that axis changes. For example, if mechanical initialization is performed for only the X axis, only the X axis flag status changes (the Y axis flag does not change) regardless of the [Order of Init.] settings.

- These flags all react in the same way regardless of the mechanical initialization method or type.
  - Method: switchbox, I/O-SYS, Fieldbus, communication command, point job command, etc.
  - Type: mechanical initialization at power ON, mechanical initialization while standing by, mechanical initialization in Teaching Mode, etc.

6.2 JS3 Series

No.	Identifier	Description	Condition “1” (True)
01	#FisCOM1	Existence of COM1 received data	Exists
02	#FltCOM1	Comparison command (cmpCOM) result of COM1 received data	Constant > Receive data
03	#FeqCOM1	Comparison command (cmpCOM) result of COM1 received data	Constant = Receive data
04	#FgtCOM1	Comparison command (cmpCOM) result of COM1 received data	Constant < Receive data
05	#FtimeOutCOM1	Comparison command (cmpCOM) timeout of COM1 received data	Timeout
06	#FisCOM2	Existence of COM2 received data	Exists
07	#FltCOM2	Comparison command (cmpCOM) result of COM2 received data	Constant > Receive data
08	#FeqCOM2	Comparison command (cmpCOM) result of COM2 received data	Constant = Receive data
09	#FgtCOM2	Comparison command (cmpCOM) result of COM2 received data	Constant < Receive data
10	#FtimeOutCOM2	Comparison command (cmpCOM) timeout of COM2 received data	Timeout
31	#FcameraError	State of camera data acquisition error	Error
32	#FtakeZError	State of Z height adjustment (takeZWadj) acquisition error	Error
33	#FlMoveOutRange	Range status of relative move command	Out of range
34	#FlMoveStop	Conditional stop status of relative move command	Stopped by the stop condition
36	#FdataInBCDError	Error status of *dataInBCD* command	Error
37	#FmultiWadjValError	Shows whether there was an error or not when obtaining the workpiece adjustment amount with the readout counter.	Error
63	#FemgSW	EMG direct input	ON (The emergency stop switch is pressed.)
66	#FmponSW	Motor power ON switch	ON (Pressed)
68	#FmdSW1	Select switch	ON
91	#FenableSW	Enable switch	ON (Pressed)
94	#FmotorPower	Motor power status	ON
121	#FsvReady1	J1/X servomotor is ready	The servomotor is ready
122	#FsvReady2	J2/Y servomotor is ready	The servomotor is ready
123	#FsvReady3	J3/Z servomotor is ready	The servomotor is ready
124	#FsvReady4	J4/R servomotor is ready	The servomotor is ready

No.	Identifier	Description	Condition “1” (True)
126	#FsvAlarm1	J1/X servomotor alarm	Servo driver error
127	#FsvAlarm2	J2/Y servomotor alarm	Servo driver error
128	#FsvAlarm3	J3/Z servomotor alarm	Servo driver error
129	#FsvAlarm4	J4/R servomotor alarm	Servo driver error
131	#FsvPos1	J1/X servomotor positioning complete	Positioning complete
132	#FsvPos2	J2/Y servomotor positioning complete	Positioning complete
133	#FsvPos3	J3/Z servomotor positioning complete	Positioning complete
134	#FsvPos4	J4/R servomotor positioning complete	Positioning complete
136	#FencOz1	J1/X encoder zero phase	Sensor is blocked
137	#FencOz2	J2/Y encoder zero phase	Sensor is blocked
138	#FencOz3	J3/Z encoder zero phase	Sensor is blocked
139	#FencOz4	J4/R encoder zero phase	Sensor is blocked
300	#FisEther1	Existence of Ether1 received data	Data exists
301	#FltEther1	Ether1 Receive data comparative results	Constant > Receive data
302	#FeqEther1	Ether1 Receive data comparative results	Constant = Receive data
303	#FgtEther1	Ether1 Receive data comparative results	Constant < Receive data
304	#FtimeOutEther1	Ether1 Receive data comparative results	Timeout
305	#FconnectEther1	Ether1 Connection state	Connected
306	#FisEther2	Existence of Ether2 received data	Data exists
307	#FltEther2	Ether2 Receive data comparative results	Constant > Receive data
308	#FeqEther2	Ether2 Receive data comparative results	Constant = Receive data
309	#FgtEther2	Ether2 Receive data comparative results	Constant < Receive data
310	#FtimeOutEther2	Ether2 Receive data comparative results	Timeout
311	#FconnectEther2	Ether2 Connection state	Connected
312	#FisEther3	Existence of Ether3 received data	Data exists
313	#FltEther3	Ether3 Receive data comparative results	Constant > Receive data
314	#FeqEther3	Ether3 Receive data comparative results	Constant = Receive data
315	#FgtEther3	Ether3 Receive data comparative results	Constant < Receive data
316	#FtimeOutEther3	Ether3 Receive data comparative results	Timeout
317	#FconnectEther3	Ether3 Connection state	Connected

No.	Identifier	Description	Condition “1” (True)
400	#FwarningBattery	Robot Maintenance Function : Battery Warning Y/N	Yes
401	#FwarningGrease1	Robot Maintenance Function : J1 Axis Grease Warning Y/N	Yes
402	#FwarningGrease2	Robot Maintenance Function : J2 Axis Grease Warning Y/N	Yes
403	#FwarningGrease3	Robot Maintenance Function : J3 Axis Grease Warning Y/N	Yes
404	#FwarningGrease4	Robot Maintenance Function : J4 Axis Grease Warning Y/N	Yes
405	#FwarningBelt1	Robot Maintenance Function : J1 Axis Belt Warning Y/N	Yes
406	#FwarningBelt2	Robot Maintenance Function : J2 Axis Belt Warning Y/N	Yes
407	#FwarningBelt3	Robot Maintenance Function : J3 Axis Belt Warning Y/N	Yes
408	#FwarningBelt4	Robot Maintenance Function : J4 Axis Belt Warning Y/N	Yes

7. VARIABLES

7.1 Built-In Variables

7.1.1 Free Variables

■ #mv, #mkv, #nv, #nkv, #sv, #skv

A variable is a receptacle into which numeric values are placed.

You can use the built-in variables listed below as free variables. Variable declaration is unnecessary when using these variables.

	Identifier	
Free Variable	#mv (1 – 99)	Boolean variable
	#mkv (1 – 99)	Boolean variable (Keeping variable)
	#nv (1 – 99)	Numerical variable
	#nkv (1 – 99)	Numerical variable (Keeping variable)
	#sv (1 – 99)	Character string type variable
	#skv (1 – 99)	Character string type variable (Keeping variable)

NOTE: Variables which hold their values even if the robot is turned OFF are keeping variables.

■ #mv (1 – 99) and #mkv (1 – 99): Boolean variable

A Boolean variable is a variable that can hold a bit's 0/1 value. It can be used as a condition operation expression (ld, ldi) or assignment expression (let) parameter.

NOTE: Boolean type free variables, #mv (1 – 99) and #mv (1 – 99), can also be used in PLC programs

■ #nv (1 – 99) and #nkv (1 – 99): Numeric variable

These are 8 byte real data type (double type) numeric variables that can be used as assignment expression (let) parameters.

■ #sv (1 – 99) and #skv (1 – 99): Character string type variable

These are string type constant variables that can hold up to 255 bytes. When used as assignment expression (let) parameters, assignment by "=" and connection by "&" are possible.

7.1.2 Input Variables

■ #sysIn1..., #genIn1..., #handIn1⋯, #fbIn1...

An input variable is a variable that can be referred to only. You cannot enter values into them. Input variables correspond to the I/O-SYS, I/O-1, I/O-H, and Fieldbus input pins. When an ON signal is received, the input variable becomes “1” (true).

Category	Identifier (JR3000/JC-3 Series)	Identifier (JS3 Series)	Connector
Input Variable	#sysIn1 – 16, #sysIn(1 – 16)	#sysIn1 – 15, #sysIn(1 – 15)	I/O-SYS
	#genIn1 – 8, #genIn(1 – 8)	#genIn1 – 18, #genIn(1 – 18)	I/O-1
	-	#handIn 1 – 8, #handIn(1 – 8)	I/O-H
	#fbIn(0x1000 – 0x17FF, 1 – 32)		Fieldbus

Some of the #sysIn1 – 16 or #sysIn1 – 15 (I/O-SYS) pins have pre-assigned functions.
Example: #sysIn1: Start (When this signal is turned ON, the robot starts operation.)

If you want to use #sysIn1 – 16 or #sysIn1 – 15 (I/O-SYS) for functions other than the pre-assigned ones, switch the settings for all program common settings to: [Free] ([All Program Common Settings] → [I/O Settings] → [I/O-SYS Function Assignment]).

If you want to use #fbIn1000 – 17FF (Fieldbus) for functions other than the pre-assigned ones, switch the settings for all program common settings to: [Free] ([All Program Common Settings] → [I/O Settings] → [Fieldbus Function Assignment]).

NOTE:

- #handIn1 to 8 are only valid for JS3 Series. The JR3000/JC-3 Series do not have I/O-H variables.
- For details of the I/O-SYS and Fieldbus pre-assigned functions, refer to the operation manual *External Control*.

The Fieldbus variables: #fbIn are described as follows:

- #fbIn (I/O address, bit width)

NOTE: Specify the I/O address as 4 hexadecimal digits continuing on from 0x.

You can specify a bit width of 1 – 32 and the maximum you can specify is 2 words (4 bytes). However, if the address specified exceeds the input area due to the Fieldbus settings, the exceeded bits are not included.

7.1.3 Output Variables

■ #sysOut1 – , #genOut1 – , #handOut1 – , #fbOut1 –

An output variable is a Boolean variable (you can also use Fieldbus as numeric variables). Output variables correspond to the I/O-SYS, I/O-1, I/O-H, and Fieldbus output pins. When an ON signal is output, the output variables become “1” (true).

Category	Identifier (JR3000/JC-3 Series)	Identifier (JS3 Series)	Connector
Output Variable	#sysOut1 – 16, #sysOut(1 – 16)	#sysOut1 – 14, #sysOut(1 – 14)	I/O-SYS
	#genOut1 – 8, #genOut(1 – 8)	#genOut1 – 18, #genOut(1 – 18)	I/O-1
	-	#handOut1 – 8, #handOut(1 – 8)	I/O-H
	#fbOut(0x1800 – 0x1FFF, 1 – 32)		Fieldbus

Some of the #sysOut1 – #sysOut 16 or #sysOut1 – 15 (I/O-SYS) pins have pre-assigned functions.
Example: #sysOut1: Ready for Start (When this signal is turned ON, the robot can start operation.)

If you want to use #sysOut1 – 16 or #sysOut1 – 15 (I/O-SYS) for functions other than the pre-assigned ones, switch the settings for all program common settings to: [Free] ([All Program Common Settings] → [I/O Settings] → [I/O-SYS Function Assignment]).

If you want to use #fbOut1800 – 1FFF (Fieldbus) for functions other than the pre-assigned ones, switch the settings for all program common settings to: [Free] ([All Program Common Settings] → [I/O Settings] → [Fieldbus Function Assignment])

NOTE:

- #handOut1 to 8 are only valid for JS3 Series. The JR3000/JC-3 Series do not have I/O-H variables.
- For details of the I/O-SYS and Fieldbus pre-assigned functions, refer to the operation manual *External Control*.

Fieldbus variables: #fbOut are described as follows:

- #fbOut (I/O address, bit width)

NOTE: Specify the I/O address as 4 hexadecimal digits continuing on from 0x.
You can specify a bit width of 1 – 32 and the maximum you can specify is 2 words (4 bytes). However, if the address specified exceeds the output area due to Fieldbus settings, the exceeded bits are not included.

7.1.4 Down Timer

■ #downTimer1 – #downTimer10

A numeric variable: The assigned value (using a *let* command) automatically starts counting down (by msec). You can assign another value during the countdown.

The maximum value that can be assigned is 2, 147, 483, 647 (msec).

Category	Identifier	Description
Special Variable	#downTimer1 – #downTimer 10	The assigned value automatically starts counting down (by msec).

For example, if you create the following point job data and set it to a point as [Job while CP Moving], the hexadecimal CR code is output to COM2 every 0.5 seconds while making the CP movement.

```
if
   ld #downTimer1 == 0
then
   eoutCOM port2, “%0D”
   #downTimer1 = 500
```

If

#downTimer1 = 0

Then

Output a hexadecimal code “CR” from COM2 and assign 500 (0.5sec) to #downTimer1.

NOTE: In this case, you need to assign a value to #downTimer1 in advance (e.g. during a point job).

7.1.5 Job After Moving Start Height

■ #jobStartHight

This is a variable to modify the Z (J3) axis height at the end of a PTP movement. You can use this to modify the actual Z (J3) axis height at the end of a PTP movement from the height set for the Z (J3) coordinate in point data. If you set the Z (J3) axis height to a higher position, a job after moving set at the end of the PTP movement can be started earlier.

If you assign a value to the variable #jobStartHight using the *let* command with a job before moving, the Z axis height is modified at the end of the PTP movement performed after the job before moving.

Values assigned to the variable #jobStartHight at a job after moving or job while moving are ignored and do not affect the movement or operation.

A positive value assigned to #jobStartHight sets the Z coordinate to a position higher than that set in the point data, and a negative value sets the Z coordinate to a position lower than that set in the point data.

The robot performs the job after moving at the Z (J3) axis height modified for the end of the PTP movement and then proceeds to the next point.

Category	Identifier	Description
Specialized Variable	#jobStartHight	This is a variable to modify the Z (J3) axis height at the end of a PTP movement.

■ Example

The robot makes a PTP movement to P1. The Z coordinate at the end of the PTP movement is modified to be exactly 2.5 mm higher than that which is set for P1.

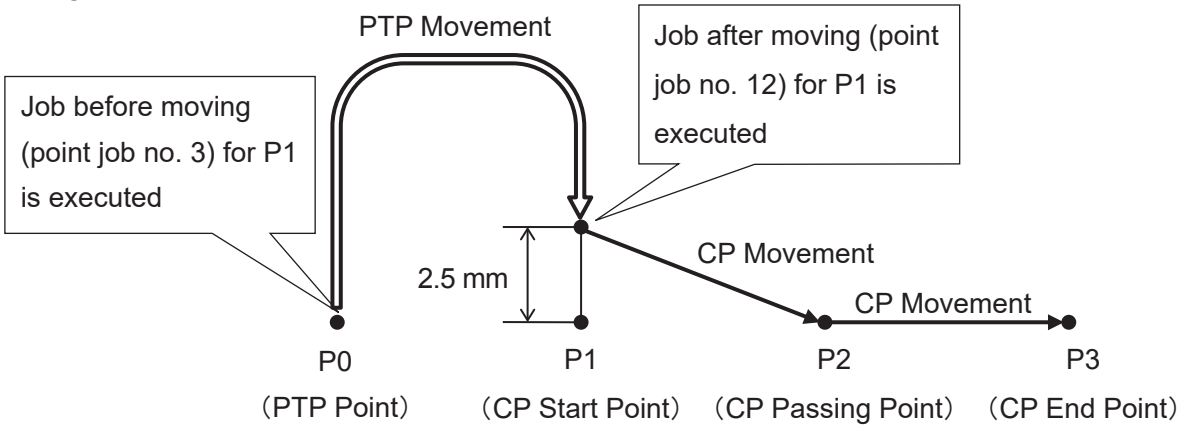
Teaching Example

	P0 (Work Home)	**P1**	**P2**	**P3**
Type	PTP Point	CP Start Point	CP Passing Point	CP End Point
Job before Moving No.	-	3	-	-
Job after Moving No.	-	12	-	-

Point Job No. 3
let #jobStartHight = 2.5

Point Job No. 12
Arbitrary job content

Run Content



The robot performs the job after moving for P1 at a height 2.5 mm higher than the Z coordinate set in P1, and then moves to P2 while still at this offset height.

If you want to return the Z (J3) axis to the original height set for P1, use a movement command such as the *downZ* command with the P1 job after movement.

7.1.6 Pallet Routine

■ #palletFlag (1 – 100), #palletCount (1 – 100)
#palletCount (1 – 100) is a numeric variable and #palletFlag (1 – 100) is a Boolean variable. The values of the corresponding pallet counter and pallet flag (1 (true) when the pallet counter is at its maximum) are retained in additional function data [Pallet Routine].
By using these variables, you can move to the next point during a pallet job or skip the designated pallet.

Category	Identifier	Description
Pallet	#palletFlag (1 – 100)	Pallet flag (Corresponds to Pallet 1 – 100.)
	#palletCount (1 – 100)	Pallet counter (Corresponds to Pallet 1 – 100.)

NOTE:

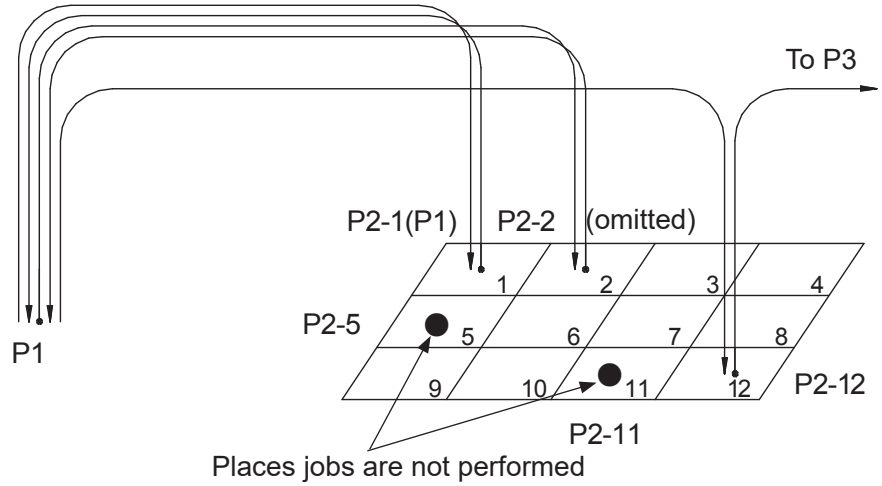
- #palletFlag (1 – 100) does not become “1” (true) even if a value which maxes out the counter is assigned to #palletCount (1 – 100).
- If using a JS3 Series pick and place application model, pallet fl ags and pallet counters 41 to 100 are reserved by the internal system software. Do not use these fl ags or counters.

Example: Skip designated places during a pallet job (when using the JS3 Series)

Pallet Type:	Plane pallet
Count Control Type:	Point job control
No. of Pallet Rows:	4
No. of Pallet Columns:	3
P0, Pa, Pb:	Arbitrary values

The robot picks up an object at P1, places it on a pallet (set at P2) and moves to the next point (P3) when the pallet reaches its maximum. However, there are two places (P2-5 and P2-11) on the pallet where an object is not placed.

In the diagram below, the pallet number is “3” and the tool unit is connected according to the following settings:
Pick: #handOut1 is ON.
Place: #handOut1 is OFF.



Point job data set to P1

```
set #handOut1
```

Pick

Point job data set to P2

```
if
   ld #palletCount （3） == 5
   or #palletCount （3） == 11
then
   loopPallet 3,2
else
   reset #handOut1
   loopPallet 3,1
endIf
```

If
#palletCount (3) is 5 (P2-5) or 11 (P2-11),
Add 1 to the Pallet 3 counter, and move to the next P2 work place.
Anything else, Place the object.
If the counter reaches its maximum, go to the next command. (In this example, the point job is over because there are no more commands.)
If the counter is not at maximum, move to P1.

7.1.7 Workpiece Adjustment

■ #workAdj_X, #workAdj_Y, #workAdj_Z, #workAdj_R, #workAdj_Rotation, #mulWorkAdj_Wrt_Cam, #mulWorkAdj_Wrt_Zadj, #mulWorkAdj_Read, #mulWorkAdj_Num
These numeric variables hold the additional function data [Workpiece Adjustment] offsets and rotation offsets for each axis direction.

Category	Identifier	Description
Workpiece Adjustment	#workAdj_X (1 – 3000)	Workpiece adjustment amount in the X direction (Compatible with workpiece adjustment 1 – 3000.)
	#workAdj_Y (1 – 3000)	Workpiece adjustment amount in the Y direction (Compatible with workpiece adjustment 1 – 3000.)
	#workAdj_Z (1 – 3000)	Workpiece adjustment amount in the Z direction (Compatible with workpiece adjustment 1 – 3000.)
	#workAdj_R (1 – 3000)	Workpiece adjustment amount in the R direction (Compatible with workpiece adjustment 1 – 3000.)
	#workAdj_Rotation (1 – 3000)	Workpiece adjustment amount by the rotation angle (Compatible with workpiece adjustment 1 – 3000.)
Workpiece Adjustment (CCD Camera Adjust with Counter)	#mulWorkAdj_Wrt_Cam	The write value of the workpiece adjustment counter for a CCD camera adjustment with counter.
	#mulWorkAdj_Wrt_Zadj	The write value of the Z adjustment counter for a CCD camera adjustment with counter.
	#mulWorkAdj _Read	The readout value of the workpiece adjustment counter for a CCD camera adjustment with counter.
	#mulWorkAdj_Num	The workpiece adjustment number of the workpiece adjustment counter for a CCD camera adjustment with counter.

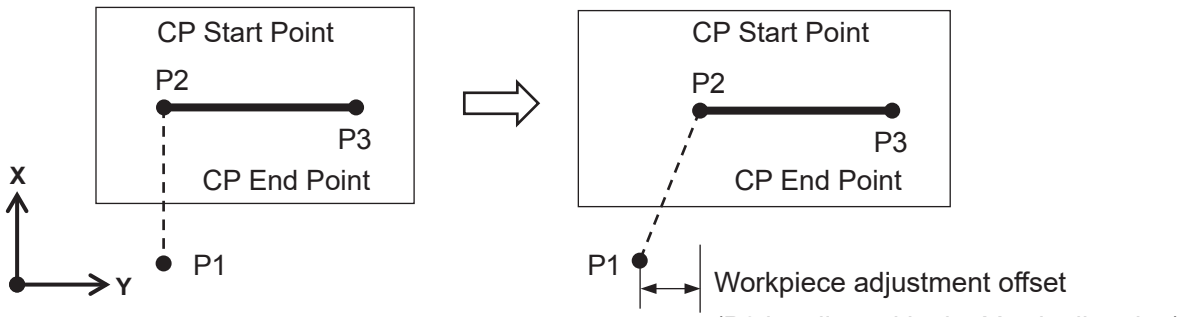
Example: Line dispensing between P2 – P3.

At P1, the workpiece adjustment amount (workpiece offset value) is received from the sensor connected to COM.

In the diagram below, the [Workpiece Adjustment] is “6” and the tool unit is connected according to the following settings:

Starting dispensing: #genOut1 is ON.

Stopping dispensing: #genOut1 is OFF.



Point job data set to P1

```
declare string adjust
setWTCOM port1,1000
inCOM hosei,port1,10
let #workAdj_Y(6) = val(adjust)
```

Declaration of the string type local variable *adjust*.
Transfer standby 1.0 sec (0.1 sec standby when omitted)
Transfer workpiece adjustment amount from COM1 to *adjust*.
Assign the value in *adjust* to #workAdj_Y(6).
(#workAdj_Y(6) is the Y direction adjustment offset of Workpiece Adjustment 6)

Point job data set to P2 ([Workpiece Adjustment] is set to this point.)

```
set #genOut1
```

Start dispensing.

Point job data set to P3

```
reset #genOut1
```

Stop dispensing.

NOTE: The [Workpiece Adjustment] set to a [CP Start Point] point is enabled until the tool unit reaches a [CP End Point] point.

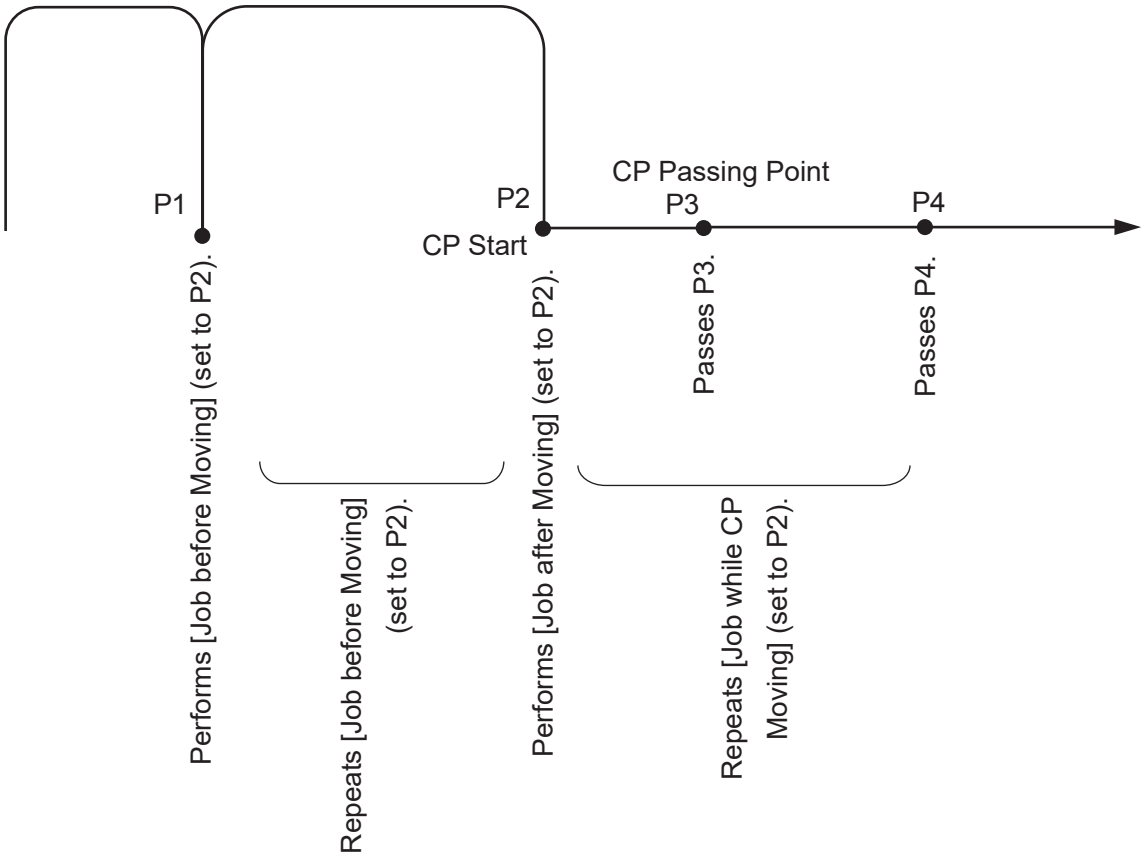
7.1.8 Point Coordinates

■ #point_X, #point_Y, #point_Z, #point_R, #point_TagCode

These variables hold the coordinate values and tag code values of a running point. A running point is a point for which point job data including these variables are set. When point job data including these variables is set to [Job before Moving], [Job while Moving], or [Job while CP Moving], the current tool center point position is different from the value in this variable.

In the figure below, a [Job before Moving] set to P2 is performed at P1, but when point job data set in [Job before Moving] includes these variables, the coordinate values for P2 are retained.

These variables hold the original coordinate values of a point. The values do not change even when the additional function data [Workpiece Adjustment] and the variable #jobStartHight are used.



Category	Identifier	Description
Current Point Coordinates	#point_X	X coordinate value of the running point
	#point_Y	Y coordinate value of the running point
	#point_Z	Z coordinate value of the running point
	#point_R	R coordinate value of the running point
	#point_TagCode	Tag code value of the running point

7.1.9 Specified Point Coordinates

■ #P_X, #P_Y, #P_Z, #P_R, #P_TagCode

These variables hold the coordinate values and tag code values of a specified point in the current program.

These variables hold the original coordinate values of a point. The values do not change even when the additional function data [Workpiece Adjustment] and the variable #jobStartHight are used.

Category	Identifier	Description
Specified Point Coordinates	#P_X (1 – Last point number)	X coordinate value of given point in current program
	#P_Y (1 – Last point number)	Y coordinate value of given point in current program
	#P_Z (1 – Last point number)	Z coordinate value of given point in current program
	#P_R (1 – Last point number)	R coordinate value of given point in current program
	#P_TagCode (1 – Last point number)	Tag code value of given point in current program

7.1.10 Specified Point Coordinates in a Specified Program

■ #prog_P_X, #prog_P_Y, #prog_P_Z, #prog_P_R, #prog_P_TagCode

These variables hold the coordinate values and tag code values for specified points in a specified program. These variables hold the original coordinate values of a point. The values do not change even when the additional function data [Workpiece Adjustment] and the variable #jobStartHight are used.

Category	Identifier	Description
Specified Point Coordinates in Specified Program	#prog_P_X (1 – 999, 1 – Last point number)	X coordinate value of the specified point in the specified program
	#prog_P_Y (1 – 999, 1 – Last point number)	Y coordinate value of the specified point in the specified program
	#prog_P_Z (1 – 999, 1 – Last point number)	Z coordinate value of the specified point in the specified program
	#prog_P_R (1 – 999, 1 – Last point number)	R coordinate value of the specified point in the specified program
	#prog_P_TagCode (1 – 999, 1 – Last point number)	Tag code value of the specified point in the specified program

7.1.11 Specified Program Tool Data

■ #prog_ToolData_X, #prog_ToolData_Y, #prog_ToolData_DeltaZ, #prog_ToolData_EachCommon
These are variables that hold tool data distance and common/individual settings for the specified program. If the argument within the brackets is specified as “0”, tool data for all program common settings is specified.

Category	Identifier	Description
Specified Program Tool Data	#prog_ToolData_X(0 – 999)	TCP-X tool data for the specified program
	#prog_ToolData_Y(0 – 999)	TCP-Y tool data for the specified program
	#prog_ToolData_DeltaZ (0 – 999)	TCP-ΔZ tool data for the specified program
	#prog_ToolData_EachCommon (1 – 999)	Common/individual tool data for the specified program (common = 0, individual = 1)

7.1.12 Additional Function Data Numbers

■ #point_WorkAdjNum, #point_ToolNum, #point_PalletNum
These are variables that hold the workpiece adjustment number, tool data number, pallet routine number for the point being executed.

Category	Identifier	Description
Additional Function Data Number	#point_WorkAdjNum	Workpiece adjustment number registered to the point being executed
	#point_ToolNum	Tool data number registered to the point being executed
	#point_PalletNum	Pallet routine number registered to the point being executed

7.2 User Defined Variables

7.2.1 Global Variables and Keeping Variables

You can choose between numeric type and string type, and set up to three dimensions in the variable array.
As opposed to local variables, variables which can be seen from any program and any point are called “global variables”. All variables, other than the local variables, declared and used in point job data are global variables. Keeping variables are variables that maintain their values even when the power is turned OFF. Keeping values defined here are global variables that maintain their values.

TP If there is an owner, login into the account in Customizing Mode
[Variable Definition]
[Global Variables Definition]
[Keeping Variables Definition]
If there is no owner, make the selection from Teaching Mode
[Variable, Function, Alias Settings]
[Global Variables Definition]
[Keeping Variables Definition]

PC [Data] → [Global Variables]
→ [Keeping Variables]

Item	Type	Content
Identifier	Identifier String	The identifiers stipulated here are used in point job data expressions. Specify the identifier when creating a new variable. The specified identifier cannot be changed.
Protect Mode	Selection	Consists of the following four levels to protect data: (1) No Limit, (2) Public, (3) Protected, (4) Private
Variable Type	Selection	Select the type of variable from the following: (1) Numeric (8-byte real number type (double type)) (2) String (Up to 255 byte strings can be stored.)
Dimension	Selection	At maximum you can make the variable a 3 dimensional array. Select the variable dimension from the following 4 selections: (1) Simple Element (3) 2 Dimension (2) 1 Dimension (4) 3 Dimension
Number of Element 1	Numeric	Number of array elements when the variable array is 1 dimension or higher.
Number of Element 2	Numeric	Number of array elements when the variable array is 2 dimensions or higher.
Number of Element 3	Numeric	Number of array elements when the variable array is 3 dimensions.

8. FUNCTIONS

With this robot, you can use functions built into the robot system and user-defined functions which are freely defined by the user.

8.1 Built-In Functions

8.1.1 Robot System Functions

The functions built into the robot system are as follows:

Type	Identifier	Description
num	currentMainProgNumber()	Currently running main program number
num	currentSubProgNumber()	Currently running sub program number
num	currentPointNumber()	Currently running point number
num	currentArmX()	Current X coordinate [mm]
num	currentArmY()	Current Y coordinate [mm]
num	currentArmZ()	Current Z coordinate [mm]
num	currentArmR()	Current R coordinate [deg]
num	currentArmH()	Current coordinate system (1: Right, − 1: Lefty) NOTE: This is fixed as 1 (righty) for JR3000/JC-3 series.
num	currentCmdArmX()	Current command X coordinate [mm]
num	currentCmdArmY()	Current command Y coordinate [mm]
num	currentCmdArmZ()	Current command Z coordinate [mm]
num	currentCmdArmR()	Current command R coordinate [deg]
num	numCOM(port#)	Data byte count of COM receiving port
num	moveAPTP (num a, num b, num X, num Y, num Z, num R)	Function of PTP movement to a specified absolute position. The robot makes a PTP movement to a specified position.
num	moveRPTP (num a, num b, num X, num Y, num Z, num R)	Function of PTP movement to a specified relative position. The robot makes a PTP movement from the current position to a remote position by exactly the specified distance.
num	isConditionData(num n)	Display whether the specified condition data number is available (1) or not (0).
str	strCenterLCD(string s)	Adjust the strings on the teaching pendant LCD (centering).
str	strRightLCD(string s)	Adjust the strings on the teaching pendant LCD (right justification).
str	strPlusRLCD(string a, string b)	Teaching pendant LCD: Right priority; items on the right are displayed in full if there is an overlap.

Type	Identifier	Description
str	strPlusLLCD(string a, string b)	Teaching pendant LCD: Left priority; Items on the left are displayed in full if there is an overlap.
num	getSystemPTPmoveTime()	Valid only for [Job while Moving]. Time required for the current PTP movement [sec]
num	getSystemPTPrestTime()	Valid only for [Job while Moving]. Time left before the current PTP movement ends (reaching the destination) [sec]
num	Pause()	Temporary Stop. To do this, you need to enable [Change Pause] in JR C-Points II. The argument in the brackets () is the *pause* number for when executing Reference Value.
str	getUserMessage(num x)	Acquire the message character string defined by number and specified by x.
num	addPointSkip()	Register a specified point to skip.
num	delPointSkip()	Clear the skip operation for a specified point.
num	clearPointSkip()	Clear all skip operations for the specified points.
num	addPalletSkip()	Register a specified pallet routine to skip.
num	delPalletSkip()	Clear the skip operation for a specified pallet routine.
num	clearPalletSkip()	Clear all skip operations for the specified pallet routines.
num	qCameraWadj(num X, num Y)	Calculate the workpiece adjustment using 4 camera points.

- currentMainProgNumber()
  This variable holds the currently running main program number.

- currentSubProgNumber()
  This variable holds the currently running subprogram number. When a subprogram is not being performed, it holds the currently running main program number.

- currentPointNumber()
  This variable holds the currently running point number. When this is the work home position, this number is “0”.

- currentArmX(), currentArmY(), currentArmZ()
  This variable holds the current coordinate position. (Absolute coordinates, in millimeters)

- currentArmR()
  This variable holds the current R (J4) axis rotation angle (R-axis coordinate). (Absolute coordinates, in degrees)

- currentCmdArmX(), currentCmdArmY(), currentCmdArmZ()
  This variable holds the current coordinate position. (Absolute coordinates, in millimeters)

- currentCmdArmR()
  This variable holds the current R (J4) axis rotation angle (R-Axis coordinate). (Absolute coordinates, in degrees)

- moveAPTP(num a, num b, num X, num Y, num Z, num R)
  The robot moves by PTP movement to a specified position.
  However, this is invalid for CP Passing Points and point types which are based on CP Passing Points.

num a : PTP Condition Number
If 0 is specified, the movement is performed according to the program data PTP conditions.
If you are not executing a program, the movement is performed according to all program common settings.

num b : Coordinate system = 1: Righty, − 1: Lefty
For the JR3000/JC-3 series make sure to specify 1.
num X : X (J1) axis position (absolute coordinates, unit [mm])
num Y: Y (J2) axis position (absolute coordinates, unit [mm])
num Z: Z (J3) axis position (absolute coordinates, unit [mm])
num R: R (J4) axis position (absolute coordinates, unit [deg])
Return values: 0 = movement successful
-1 = movement failed
Return values: Causes for − 1 (movement failed) are as follows:

- The PTP condition for the number specified by a does not exist.
- A CP Passing Point or a point type based on a CP Passing Point was executed.
- Executed during either a Job While Moving or a Job While CP Moving.

For example, if moveAPTP(3,1,100,110,50,90) is specified, move to X=100 mm, Y=110 mm, Z=50 mm, R=90° position according to the PTP condition number 3 (Righty with the JS3 series).

- moveRPTP(num a, num b, num X, num Y, num Z, num R)
  The robot makes a PTP movement from the current position to a remote position by exactly the specified distance.
  However, this is invalid for CP Passing Points and point types which are based on CP Passing Points.

num a : PTP Condition Number
If 0 is specified, the movement is performed according to the program data PTP conditions. If you are not executing a program, the movement is performed according to all program common settings.
num b : Coordinate system = 1: Righty, − 1: Lefty
For the JR3000/JC-3 series make sure to specify 1.
num X : X (J1) axis distance (relative coordinates, unit [mm])
num Y: Y (J2) axis distance (relative coordinates, unit [mm])
num Z: Z (J3) axis distance (relative coordinates, unit [mm])
num R: R (J4) axis distance (relative coordinates, unit [deg])
Return values: 0 = movement successful
− 1 = movement failed
Return values: Causes for − 1 (movement failed) are as follows:

- The PTP condition for the number specified by *a* does not exist.
- A CP Passing Point or a point type based on a CP Passing Point was executed.
- Executed during either a Job While Moving or a Job While CP Moving.

For example, if the robot's current positions are X = 50mm, Y = 50mm, Z = 10mm and R = 30° , and you execute moveRPTP (3, 1, 100, 110, 50, 90), the robot moves to X = 150mm, Y = 160mm, Z = 60mm, R = 120° position.

- numCOM(port#)
  This is the data byte count of the COM receiving port.

- isConditionData(num n)
  This is the presence (1) or absence (0) of the specified condition data number.

- strCenterLCD(string s)
  This centers the character strings on the teaching pendant LCD.

- strRightLCD(string s)
  This adjusts the character strings for right-side alignment on the teaching pendant LCD. (Normally left-justified).

- strPlusRLCD(string a, string b)
  This variable adjusts the character strings on the teaching pendant LCD (right priority). Items on the right are displayed in full if there is an overlap.

- strPlusLLCD(string a, string b)
  This variable adjusts the character strings on the teaching pendant LCD (left priority). Items on the left are displayed in full if there is an overlap.

- getSystemPTPmoveTime()
  This variable holds the time required for the current PTP movement (in seconds).
  Valid only for [Job while Moving].

- getSystemPTPrestTime()
  This variable holds the time left before the current PTP movement ends (reaching the destination) (in seconds). Valid only for [Job while Moving].

- Pause ()
  Temporary Stop. The specified argument (0 – 100) is used as the “pause number” when [Reference Value] is executed. If there is no argument, an expression evaluation error occurs. For this to work, [Change Pause] needs to be set to [Valid] in JR C-Points II. However, this will have no effect even if [Change Pause] is set to [Valid] while the robot is moving. For example, pushing [Pause] during a job with CP movement does not pause the operation.
  To release the pause function, you need to do this through JR C-Points II.

- getUserMessage (num x)
  Acquire the message character string defined by number and specified by x. Message character strings acquired are displayed according to the language settings made in [Teaching Environment Settings]. You need to register translated message character strings for each individual language to the user defined messages in advance.
  num x: user defined message number
  return value: message character string

- addPointSkip (num a, num b)
  Register a specified point to skip.
  num a: specify a program number (1 to 999).
  num b: specify a point number (1 to maximum point number per program)

- delPointSkip (num a, num b)
  Clear the skip operation for a specified point.
  num a: specify a program number (1 to 999).
  num b: specify a point number (1 to maximum point number per program).
  You can specify “-1” to delete all data for the program number specified with *num a*.

- clearPointSkip()
  Clear all skip operations for the specified points.

- addPalletSkip (num a, num b)
  Register a specified pallet routine to skip.
  num a: specify a pallet routine number (1 to 100)
  num b: specify a pallet count (0+)

- delPalletSkip(num a, num b)
  Clear the skip operation for a specified pallet routine.
  num a: specify a pallet routine number (1 to 100).
  num b: specify a pallet count (0+).
  You can specify “-1” to delete all data for the pallet routine number specified with *num a*.

- clearPalletSkip()
  Clear all skip operations for the specified pallet routines.

- qCameraWadj (num x, num y)
  Calculate the adjustment value for the workpiece adjustment using 4 camera points.
  For details on how to use 4 camera points and make adjustments, refer to the operation manual *Camera & Sensor Functions*.
  num x: workpiece adjustment number (the first number in a consecutive series).
  num y: shot number (1 to 4)
  return value: 0 = normal
  -1 = failed (argument is incorrect)
  -2 = failed (settings related to the camera are incorrect)

8.1.2 Arithmetic System Functions

The following built-in arithmetic functions can be used:

Identifier	num abs(num x)
Details	Requests absolute value
Argument	x: numerical value
Returned Value	x absolute value

Identifier	num max(num x, num y)
Details	Requests maximum value. Compares the given numerical values *x* and *y* and returns the larger numerical value.
Argument	x: Comparative numerical value y: Comparative numerical value
Returned Value	Maximum value

Identifier	num min(num x, num y)
Details	Requests minimum value. Compares the given numerical values *x* and *y* and returns the smaller numerical value.
Argument	x: Comparative numerical value y: Comparative numerical value
Returned Value	Minimum value

Identifier	num degrad(num x)
Details	Coverts degrees (deg) to radians (rad). Makes a request using the formula below. x * π / 180
Argument	x: numerical value to convert to radian (rad)
Returned Value	Radian value

Identifier	num raddeg(num x)
Details	Coverts radians (rad) to degrees (deg). Makes a request using the formula below. x * 180 / π
Argument	x: numerical value to convert to degrees (deg)
Returned Value	Numerical value of degree (deg) units

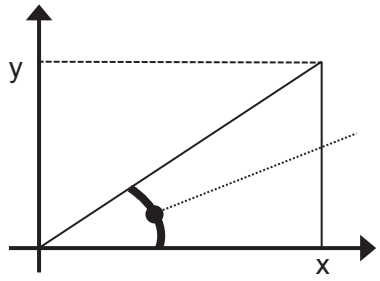
Identifier	num sqrt(num x)
Details	Requests the square root.
Argument	x: Requests numerical values of the square root. Do not assign *x* values which are less than zero (x<0).
Returned Value	Square root value of x

Identifier	num sin(num x)
Details	Requests a sine value. Assign a radian value.
Argument	x: radian value for the requested sine value.
Returned Value	x sine value

Identifier	num cos(num x)
Details	Requests a cosine value. Assign a radian value.
Argument	x: radian value for the requested cosine value.
Returned Value	x cosine value

Identifier	num tan(num x)
Details	Requests a tangent value. Assign a radian value.
Argument	x: radian value for the requested tangent value.
Returned Value	x tangent value

Identifier	num atan(num x)
Details	Requests inverse tangent.
Argument	x: value of the requested inverse tangent.
Returned Value	x inverse tangent value Radian value of the range of −π /2 ~ π /2.

Identifier	num atan2(num x, num y)
Details	Requests inverse tangent. Inverse tangent of the value of *y* divided by *x* (y/x).  atan2 (x, y) When x=0 and y>0, the return value is π /2. When x=0 and y<0, the return value is $-\pi$ /2. When x>0 and y=0, the return value is 0. When x<0 and y=0, the return value is π .
Argument	x: divisor (number used for division) y: dividend (number to be divided) Do not assign values x=0 and y=0
Returned Value	x inverse tangent value Radian value of the range of $-\pi$ - π

Identifier	num int(num x)
Details	Requests the maximum integer that does not exceed x.
Argument	x: numerical value
Returned Value	Maximum integer that does not exceed x.
Example	Examples of execution results: Example 1: int (1.3) becomes 1. Example 2: int (− 1.3) becomes 2.

Identifier	num ip(num x)
Details	Returns the integer part of x. Equivalent to the equation below: sgn(x) * int (abs(x)) (If x is a negative value, sgn(x) is expressed as -1, if x is a positive value, sgn(x) is expressed as +1.)
Argument	x: numerical value
Returned Value	Integer part of x.
Example	Examples of execution results: Example 1: ip (1.3) becomes 1. Example 2: ip (− 1.3) becomes − 1.

Identifier	num fp(num x)
Details	Returns the decimal fraction part of x. Equivalent to the equation below: x − ip (x)
Argument	x: numerical value
Returned Value	Decimal fraction part of x
Example	Examples of execution results: Example 1: fp (1.3) becomes 0.3. Example 2: fp (− 1.3) becomes − 0.3.

Identifier	num mod(num x, num y)
Details	Requests the value of x modulo y. Equivalent to the equation below: x − y * int (x / y)
Argument	x: divisor (number used for division) y: dividend (number to be divided) Do not assign y=0
Returned Value	The value of x modulo y

Identifier	num remainder(num x, num y)
Details	Requests the remainder of dividing x by y. Equivalent to the equation below: x − y * ip(a / b)
Argument	x: divisor (number used for division) y: dividend (number to be divided) Do not assign y=0
Returned Value	Remainder of dividing x by y

Identifier	num pow(num x, num y)
Details	Returns x to the power of y.
Argument	x: (the base number) y: (the exponent) Do not assign x=0 and b=0. Do not assign x<0 and numbers where y is a decimal fraction.
Returned Value	Value of x to the power of y

8.1.3 String System Functions

The following string built-in functions can be used:

x, y: Numerical value or numerical variable

n, m: Round the numeric value up or off to the specified digit(s)

a, b: String or string variable

Category	Type	Identifier	Description
String System	str	chr(x)	Returns a string (1 character) with the given character code.
	num	ord(a)	Returns the top character code. Other codes are ignored. Returns 0 when the number of characters for the string set to *a* is 0.
	num	len(a)	Returns the string length (non-multibyte).
	num	strPos(a ,b)	Returns the first part string position in *a* that matches *b*.
	str	strMid(a, n, m)	Returns the string from *n* to the amount of *m* counted from the start of string *a*.
	str	str(x)	Converts a numeric value to a binary string.
	str	strBin(n, m)	*m*: Number of binary string digits
	str	strHex(n, m)	Converts a numeric value to a hexadecimal string. *m*: Number of hexadecimal string digits
	str	str1SI(x)	Rounds a numeric value to a 1-byte signed integer to convert it to a 1-byte string. (1-byte Signed Integer)
	str	str2SIBE(x)	Rounds a numeric value to a 2-byte signed integer to convert it to a 2-byte string using the Big Endian byte order. (2-byte Signed Integer Big Endian)
	str	str2SILE(x)	Rounds a numeric value to a 2-byte signed integer to convert it to a 2-byte string using the Little Endian byte order. (2-byte Signed Integer Little Endian)
	str	str4SIBE(x)	Rounds a numeric value to a 4-byte signed integer to convert it to a 4-byte string using the Big Endian byte order. (4-byte Signed Integer Big Endian)
	str	str4SILE(x)	Rounds a numeric value to a 4-byte signed integer to convert it to a 4-byte string using the Little Endian byte order. (4-byte Signed Integer Little Endian)
	str	str4FBE(x)	Regards a numeric value as a decimal float to convert it to a 4-byte string using the Big Endian byte order. (4-byte Float Big Endian)
	str	str4FLE(x)	Regards a numeric value as a decimal float to convert it to a 4-byte string using the Big Endian byte order. (4-byte Float Big Endian)
	str	str8DBE(x)	Regards a numeric value as a decimal float to convert it to a 4-byte string using the Little Endian byte order. (4-byte Float Big Endian)

x, y: Numerical value or numerical variable
n, m: Round the numeric value up or off to the specified digit(s)
a, b: String or string variable

Category	Type	Identifier	Description
String System	str	str8DLE(x)	Regards a numeric value as a double precision decimal float to convert it to an 8-byte string using the Little Endian byte order. (8-byte Double Little Endian)
	num	val(a)	Regards a string as a decimal digit string to convert it to a numeric value (integer type with no symbol). Returns 0 if the head of the character string is a minus sign.
	num	valBin(a)	Regards a string as a binary string (sequence of “0”, “1”) to convert it to a numeric value.
	num	valHex(a)	Regards a string as a hexadecimal string (sequence of “0” – “9”, “A” – “F”, or “a” – “f”) to convert it to a numeric value.
	num	val1SI(a)	Converts the top character to a 1-byte signed integer. (1-byte Signed Integer)
	num	val2SIBE(a)	Converts the top 2 characters to a 2-byte signed integer using the Big Endian byte order. (2-byte Signed Integer Big Endian)
	num	val2SILE(a)	Converts the top 2 characters to a 2-byte signed integer using the Little Endian byte order. (2-byte Signed Integer Little Endian)
	num	val4SIBE(a)	Converts the top 4 characters to a 4-byte signed integer using the Big Endian byte order. (4-byte Signed Integer Big Endian)
	num	val4SILE(a)	Converts the top 4 characters to a 4-byte signed integer using the Little Endian byte order. (4-byte Signed Integer Little Endian)
	num	val4FBE(a)	Converts the top 4 characters to a decimal float using the Big Endian byte order. (4-byte Float Big Endian)
	num	val4FLE(a)	Converts the top 4 characters to a decimal float using the Little Endian byte order. (4-byte Float Little Endian)
	num	val8DBE(a)	Converts the top 8 characters to a double-precision decimal float using the Big Endian byte order. (8-byte Double Big Endian)
	num	val8DLE(a)	Converts the top 8 characters to a double-precision decimal float using the Little Endian byte order. (8-byte Double Little Endian)
	num	valSum(a)	Returns the sum of a string code from top to bottom.
	num	valCRC(a)	Remainder of dividing a string (bit string) by a generator polynomial X16+X12+X5+1
	str	bitNot(a)	Bit invert
	str	bitAnd(a, b)	Bit logical conjunction
	str	bitOr(a, b)	Bit logical add
	str	bitXor(a, b)	Bit exclusive disjunction

8.2 User Defined Functions

It is convenient to define frequently used command lines as functions. You can select either the numeric type or the string type and you can set up to three dimensions for the argument in the function array.

You can set arguments to user functions. An argument can be set with a character string type and a numeric type. If executing the user function from a point job and you want to specify the argument using a variable name, you need to make sure the defined type of variable matches the variable name.

Definition example:
UserFunction (“character string type”, “numeric type”)

Example:
let Result = UserFunction(str(UserStr), int(SelectValue))
Convert the variables as shown below to match the argument and the variable type. If you want to specify a numeric type argument with a “selection type” variable name, you need to convert the variables as follows:

- The character string type is a built-in function: use *str ()* to convert the variable to a character string type.
- The numeric type is a built-in function: use *int ()* to convert the variable to a numeric type.

If you execute the point job without converting the variable type, an expression error occurs.

TP If there is an owner, log into the account in Customizing Mode
[User Function Definition]
If there is no owner, select the following from Teaching Mode
[Variable, Function, Alias Settings]
[User Function Definition]

PC [Data] → [User Function]

Item	Type	Content
Identifier	Identifier String	The identifiers stipulated here are used in point job data expressions. By writing the identifiers in parentheses you can call up the function. Specify the identifier when creating a new function. The specified identifier cannot be changed.
Protect Mode	Selection	Consists of the following four levels to protect data: (1) No Limit, (2) Public, (3) Protected, (4) Private
Function Type	Selection	Select the type of function values from the following: (1) Numeric, (2) String
Argument Identifier	Identifier String	Dummy argument identifier. Function argument values are referred to and assigned in the function body using the dummy argument identifier.
Passing Type	Selection	(1) Call by Value: An actual argument is evaluated before evaluating the function and the argument value is passed to the dummy argument. The value will not change even if the actual argument is a variable. (2) Call by Reference: Only variables can be actual arguments. The dummy argument reference is evaluated as the actual argument variable reference. Assigning a value to the dummy argument is also evaluated as assigning a value to the actual argument variable.
Argument Type	Selection	Select the type of argument from the following: (1) Numeric, (2) String
Dimension	Selection	(1) Simple Element (2) 1 Dimension (3) 2 Dimensions (4) 3 Dimensions
Number of Element	Numeric	Number of array elements for each dimension when the variable array is one dimension or higher.
Function Body	Operation	Command lines that are executed after being called.

9. ALIAS DEFINITIONS

9.1 I/O Alias

I/O aliases are functions used to give alternative names to I/O inputs and outputs. Alias definitions define the I/O-SYS and I/O-1 input/output including the data width. You can select the [I/O-Alias] as the input source/output destination for the *set*, *reset*, *dataIn*, and *dataOut* commands (select [I/O-Alias] when selecting the input source/output destination during point job command teaching). For example, if a dispenser is connected to #sysOut16, set #sysOut16 to [DISPENSE] using the alias settings. If you then register the point job command as [set DISPENSE], this setting is not only easy to understand and work with, but if you change the dispenser connection destination later, you only need to change the alias settings without having to rewrite the point job command. (To set the point job command to [set DISPENSE], select [Alias] when selecting the *set* command output destination.)
Note that if you want to define the subject I/O as an alias, check that it is set to [Free] in All Program Common Settings.

Example: set #sysOut16 → set DISPENSE
(I/O alias has not been set) (#sysOut16 is set to [DISPENSE])

TP If there is an owner, login into the account in Customizing Mode
[Alias Definition]
[I/O Alias Definition]
If there is no owner, make the selection from Teaching Mode
[Variable, Function, Alias Settings]
[Alias Definition]
[I/O Alias Definition]

PC [Data] → [Alias]

Item	Type	Content
Identifier	Identifier String	This is a character string used as a point job command or a PLC program command parameter. It is a kind of variable identifier however it varies from variables in that it is not used as an expression element. Specify the identifier when creating a new alias. Once specified, the identifier cannot be modified.
Protect Mode	Selection	Consists of the following four levels to protect data: (1) No Limit, (2) Public, (3) Protected, (4) Private
Caption	Multilingual String	A character string displayed as an item name in Teaching Mode
Default	IOM	The default value. The value before modification. Or, the value set when deleting all data or resetting the system defaults. Specify the type (sysIn/genIn/fbIn/sysOut/genOut/fbOut/mv/mkv) and the number.
Default Data Width	Numeric	Default value. The width is set using numeric values. Set the data width for use with the *dataOut* and *dataIn* commands. The data width is not necessary for other commands such as *ld*, *set*, and *reset*.
Direction	Selection	This swaps the selections when selecting the type in Teaching Mode. If you select output, this limits the options so that those which are only for input are not displayed. (1) Input: sysIn/genIn/fbIn/sysOut/genOut/fbOut/mv/mkv (2) Output: sysOut/genOut/fbOut/mv/mkv
Data Width Setting	Selection	Select whether or not to enter the data width in Teaching Mode. Select [Invalid] if the data width has no meaning (is fixed to 1) or you do not want to change the data width. (1) Valid: You can enter the data width when making settings in Teaching Mode (editable). (2) Invalid: You cannot enter the data width when making settings in Teaching Mode.

9.2 COM Alias

The COM alias is used to provide alternative names to COM. Define the COM ports using identifiers. You can use the COM alias to specify the COM port for the COM input/output commands (*outCOM* etc.) during teaching (select [COM-Alias] when selecting input source/output destination during point job command teaching).
For example, if COM1 is connected to a PC, define COM1 as [PC] using the alias settings. If you then register the point job command [eoutCOM PC,"ERROR"], this is not only easy to understand and work with, but later on you can change the PC connection port without needing to rewrite the point job command.
(To set the point job data command [eoutCOM PC,"ERROR"], select [Alias] when selecting the *eoutCOM* command output destination.)

Example: eoutCOM port1,"ERROR" → eoutCOM PC,"ERROR"
(COM alias has not been set.) (COM1 has been set to [PC].)

TP If there is an owner, login into the account in Customizing Mode
[Alias Definition]
[COM Alias Definition]

If there is no owner, make the selection from Teaching Mode
[Variable, Function, Alias Settings]
[Alias Definition]
[COM Alias Definition]

PC [Data] → [Alias]

Item	Type	Content
Identifier	Identifier String	A character string which uses the stipulated identifier as a point job command parameter. This is a kind of variable identifier but it cannot be used as an expression element. The identifier is specified when creating a new alias. Once specified, the identifier cannot be modified.
Protect Mode	Selection	Consists of the following four levels to protect data: (1) No Limit, (2) Public, (3) Protected, (4) Private
Caption	Multilingual String	A character string displayed as an item name when making settings in Teaching Mode
Default	Selection	The default value. The value before modification. Or the value set when deleting all data or resetting the system defaults (1) COM1 (2) COM2 (3) COM3 (JR3000/JC-3 Series only)

10. ON/OFF OUTPUT CONTROL

10.1 Output to I/O

■ set, reset, pulse, invPulse

This section explains output to the tool unit (output to the I/O) commands. These commands belong to the [ON/OFF Output Control] command category.

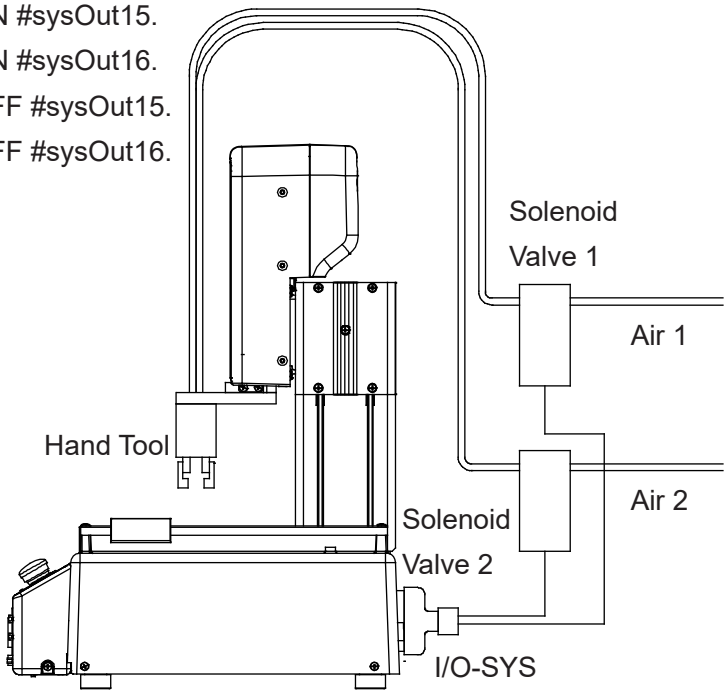
Command Category	Command	Parameter		Job
ON/OFF Output Control	set	Output Destination		Output ON to a specified output destination.
	reset	Output Destination		Output OFF to a specified output destination.
	pulse	Output Destination	Pulse Width	Output ON pulse of a specified width to a specified output destination.
	invPulse	Output Destination	Pulse Width	Output OFF pulse (inverted pulse) of a specified width to a specified output destination.

- For example, this hand tool is connected to the robot according to the following settings:
- The hand tool opens ← Close Air 1 and Open Air 2.
- The hand tool closes ← Open Air 1 and Close Air 2.
- Air 1 opens ← Turn ON Solenoid Valve 1.
- Air 2 opens ← Turn ON Solenoid Valve 2.
- Air 1 closes ← Turn OFF Solenoid Valve 1.
- Air 2 closes ← Turn OFF Solenoid Valve 2.
- Solenoid Valve 1 is ON ← Turn ON #sysOut15.
- Solenoid Valve 2 is ON ← Turn ON #sysOut16.
- Solenoid Valve 1 is OFF ← Turn OFF #sysOut15.
- Solenoid Valve 2 is OFF ← Turn OFF #sysOut16.

Accordingly,

- The hand tool opens
  ← #sysOut15 OFF, #sysOut16 ON.
- The hand tool closes
  ← #sysOut15 ON, #sysOut16 OFF.

Example: JR3203N-AC

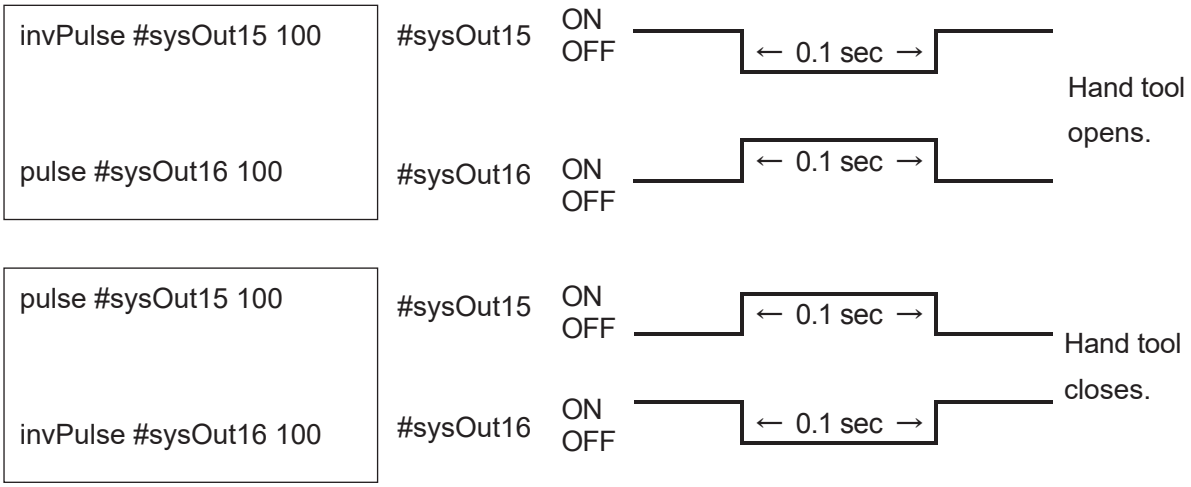


■ The output commands to open and close the hand tool are as follows:

reset #sysOut15 set #sysOut16	#Output #sysOut15 OFF. Output #sysOut16 ON.	> Open the hand tool.
set #sysOut15 reset #sysOut16	Output #sysOut15 ON. Output #sysOut16 OFF.	> Close the hand tool.

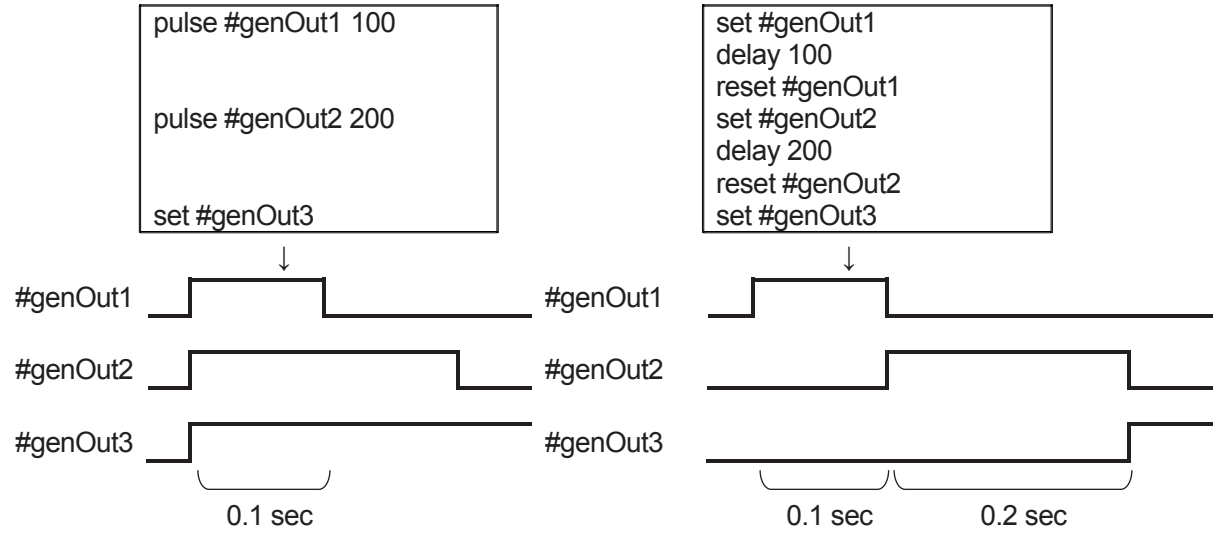
NOTE: The *set* command continues to output an ON signal unless the *reset* command comes.

■ The pulse output commands to open and close the hand tool are as follows:



NOTE: The *pulse* and *invPulse* commands move on to the next command without waiting to finish output.

For example, the following two kinds of point job data have different results:



delay 100 means “Stand by for 0.1 second at that point”.

NOTE: You can specify the pulse width for the pulse and *invPulse* commands using variables or expressions.

10.2 Output after X Seconds

■ delaySet, delayReset

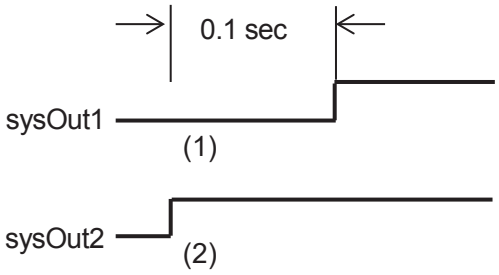
The *delaySet* and *delayReset* commands are used to output ON/OFF signals to a specified output destination after a specified time.

The delay time can be set 1 msec to 999,999,999 msec.

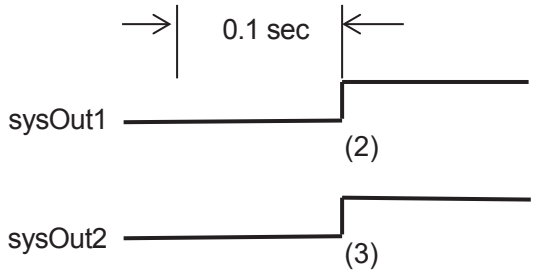
Command Category	Command	Parameter		Job
ON/OFF Output Control	delaySet	Output Destination	Delay Time	ON output after specified delay time
	delayReset	Output Destination	Delay Time	OFF output after specified delay time

The *delaySet* and *delayReset* commands move on to carry out the next command without waiting for output. If signals are output by *set* or *reset* commands after waiting due to the *delay* command, the execution timing of the command after that differs as follows:

delaySet example:
(1) delaySet #sysOut1 100
(2) set #sysOut2
(3) ・・・・・・・・



delay/set example:
(1) delay 100
(2) set #sysOut1
(3) set #sysOut2
(4) ・・・・・・



NOTE: You can specify the delay time using variables or expressions.

10.3 Sound the Buzzer

■ onoffBZ

You can sound the buzzer using a point job command.

Command Category	Command	Parameter	Job
ON/OFF Output Control	set	Output Destination (BZ)	Sound buzzer.
	reset	Output Destination (BZ)	Stop buzzer.
	onoffBZ	ON Time, OFF Time	Sound buzzer intermittently.

If the set or *onoffBZ* commands are executed, the buzzer continues to sound until the *reset* command is executed.

NOTE: You can specify [ON Time] and [OFF Time] for the *onoffBZ* command using variables or expressions.

10.4 Make the Green LED Blink

■ onoffGLED

The LED on the front panel of the robot or on the switchbox/operation box can be turned ON and OFF, or made to blink using point job commands.

Command Category	Command	Parameter	Job
ON/OFF Output Control	set	Output Destination (GLED)	Turn ON the LED (Green).
	reset	Output Destination (GLED)	Turn OFF the LED (Green).
	onoffGLED	ON Time, OFF Time	Blink the LED (Green).

The robot turns ON/blinks the LED if the commands (*set* or *onoffGLED*) are executed, the green LED is on or blinking until the turn OFF LED command (*reset*) is executed.

NOTE: You can specify [ON Time] and [OFF Time] for the *onoffGLED* command using variables or expressions.

10.5 Make the Red LED Blink

■ onoffRLED

The LED on the front panel of the robot or on the switchbox/operation box can be turned ON and OFF, or made to blink using point job commands.

Command Category	Command	Parameter	Job
ON/OFF Output Control	set	Output Destination (RLED)	Turn ON the LED (Red).
	reset	Output Destination (RLED)	Turn OFF the LED (Red).
	onoffRLED	ON Time, OFF Time	Blink the LED (Red).

The robot turns ON/blinks the LED if the commands (*set* or *onoffRLED*) are executed, the red LED is on or blinking until the turn OFF LED command (*reset*) is executed.

NOTE: You can set [ON Time] and [OFF Time] for the *onoffRLED* command using variables or expressions.

10.6 Output Values from I/O

■ dataOut, dataOutBCD
Any given numeric value 0 – 999,999,999, or tag code, can be output to the I/O or the Boolean free variables #mv (1 – 99) and #mkv (1 – 99).

Command Category	Command	Parameter			Job
ON/OFF Output Control	dataOut	Output Value	Output Destination	Output Bit Number	Output values from I/O.
	dataOutBCD	Output Value	Output Destination	Output Bit Number	Output values in BCD from I/O.

NOTE:
- Using tag code output, you can output different values using the same point job data if you set different values as tag codes to multiple points.
- The output values and bit numbers can be set using variables or expressions.

Other than setting the output values for the commands *dataOut* and *dataOutBCD*, you need to set the parameters; “output bit number” (the number of I/O pins used for output), and “output destination” (the smallest I/O pin number used for output) for example, if you use #genOut8 – #genOut10, the output destination is [8].

NOTE: With the *dataOut* and *dataOutBCD* commands, I/O output bit number from output numbers use serial numbers. You cannot use these commands if the I/O numbers you are using are not consecutive.

Example:

(Settings)	(Command)	(Output) 6=110 (binary)
Output Value: 6		#genOut8: 0 (OFF)
Output Bit Number: 3	dataOut 6, #genOut8, 3	#genOut9: 1 (ON)
Output Destination: #genOut8		#genOut10: 1 (ON)

NOTE: If the output values do not fall within the set output bit number, the upper digits are truncated.

Example:

(Settings)	(Command)	(Output) 14=1110 (binary)
Output Value: 14		#genOut8: 0 (OFF)
Output Width: 3	dataOut 14, #genOut8, 3	#genOut9: 1 (ON)
Output Destination: #genOut8		#genOut10: 1 (ON)
		: 1 (truncation)

NOTE: The output bit number can be set up to “32”. However, you cannot set the width to extend over different I/O types.

If you use Fieldbus as the output destination, the output is as follows:

Example:

(Settings)	(Command)	(Output)
Output Value: 100000		100000=186A0 (hex)
Output Bit Number: 32	dataOut 100000, fbOut(1820),32	82h : 86A0h (hex)
Output Destination: #fbOut(1820)		183h : 0001h (hex)

NOTE:

- If the output values do not fall within the set output bit number, the upper digits are truncated.
- The output bit number can be set up to “32”. However, if the values exceed the Fieldbus output area due to the Fieldbus settings, the values outside of the output area are truncated.

10.7 Motor Power ON, Servomotor ON/OFF

■ motorPowerON, servoON, servoOFF

These commands are valid with the JS3 Series only. The JR3000/JC-3 Series do not have these commands.

You can use these commands to turn the robot motor power ON, and turn the servomotor ON/OFF for a given axis. When a servomotor is switched OFF, the robot cannot control the respective axis. You can manually move X (J1), Y (J2), R (J4) axis when the corresponding servomotor is switched OFF.

NOTE: Some commands may prevent you from turning OFF the motor power.

Command Category	Command	Necessary Parameter	Action
ON/OFF Output Control	motorPowerON	-	Turns the motor power ON
	servoON	Specified axis	Turns the servomotor ON for a specified axis
	servoOFF	Specified axis	Turns the servomotor OFF for a specified axis

11. IF BRANCH, WAIT CONDITION

11.1 if Branch

■ if, then, else, endIf

This section explains point job data commands for performing different jobs according to the conditions. These belong to the *if* Branch, *Wait Condition* command category.

Command Category	Command	Parameter	Job
if Branch, Wait Condition	if	–	*if* Branch
	then	–	Execute the following command if true:
	else	–	Execute the following command if false:
	endIf	–	End of *if* Branch

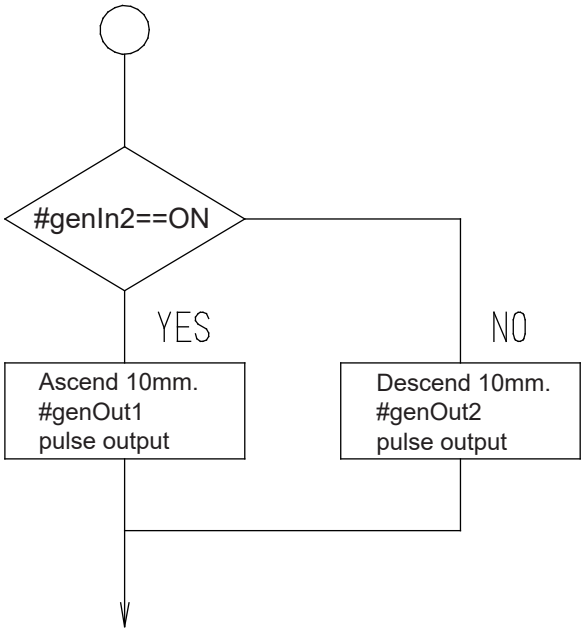
NOTE: Be sure to put a conditional command after the *if* command.

■ Using i*f, then, else* and *endIf* commands:

Example1:

If #genIn2 is ON, the Z (J3) axis raises by 10mm and a pulse is output to #genOut1.

If #genIn2 is not ON, the Z (J3) axis lowers by 10mm and a pulse is output to #genOut2.



The commands for Example 1 look like this:

```
if
  ld #genIn2
then
  upZ 20,10
  pulse #genOut1,200
else
  downZ 20,10
  pulse #genOut2,200
endIf
```

If the following condition is true, go to *then*. If false, go to *else*.
  #genIn2 == ON (Condition)

If the condition is true, execute the following commands:
  Raise the Z (J3) axis by 10 mm at 20 mm/sec, and
  Output ON pulse to #genOut1. (in 0.2 sec widths).

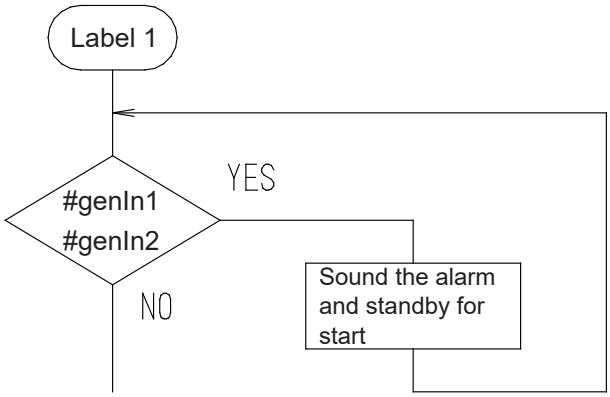
If the condition is false, execute the following commands:
  Lower the Z (J3) axis by 10 mm at 20 mm/sec, and
  Output ON pulse to #genOut2 (in 0.2 sec widths).

End of if Branch

Example2:

If both #genIn1 and #genIn2 are ON, the buzzer sounds and the robot is on standby for a start instruction.

If both #genIn1 and #genIn2 are not ON, advance to the next job.



The commands for example 2 look like this:

Label 1	(A destination mark for the jump command)
if	If the conditions below are true, go to *then*. If false, go to the item following *endIf*.
ld #genIn1	#genIn1=ON (Condition 1)
and #genIn2	And #genIn2=ON (Condition 2)
then	If the conditions are true, the following commands are executed:
waitStartBZ	Sound the buzzer and stand by at the point for a start instruction.
jump L1	Jump to [Label 1] (if there was a start instruction).
endIf	End of If Branch

NOTE:

- It is not necessary for both the *then* and *else* commands to exist at the same time. However, the *if* command without the *endIf* command is recognized as an error.
- When you use the *waitCondTime*, *timeUp* to *endWait*, and *if* to *endIf* command lines, they are indented (refer to the diagram below).

```
waitCondTime 200
  ld #genIn2
timeUp
  set #genOut2
  if
      ld #genIn1
  then
      downZ 20,20
      waitCondTime 200
          ld #genIn4
      timeUp
          waitStartBZ
      endWait
  endIf
endWait
```

3rd indent

2nd indent

1st indent

Be sure not to use more than 9 indents.

If the point job data includes more than 9 indents, when the point job data is run it is recognized as an error and the error message [Error on Point Job] is displayed.

If *timeUp* or *endWait* come before *waitCondTime* or if *then*, *else* or *endIf* come before *if*, an error occurs and the message [Error on point job] is displayed.

11.2 Wait Condition

■ waitCond, waitCondTime, timeUp, endWait

This section explains the point job data commands for waiting until the sensor (connected to #genIn2) is turned ON. These commands belong to the category [Wait Condition]. [Wait Condition] has the following commands:

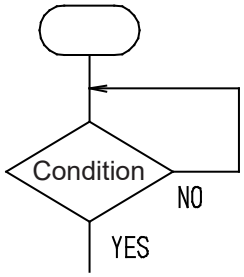
Command Category	Command	Parameter	Job
Wait Condition	waitCondTime	Wait Time	Wait for the condition for a set period.
	timeUp	–	Execute when time is up.
	endWait	–	End of wait command
	waitCond	–	Wait for the condition.

NOTE:

- The wait condition commands are disabled at the points whose point type or base type is set to [CP Passing Point].
- Be sure to put a conditional command after the *waitCond* or *waitCondTime* command.

■ waitCond – endWait: the robot waits until the conditions are met.

Example: There is a workpiece → Sensor (#genIn2) ON
There is no workpiece → Sensor (#genIn2) OFF




```
waitCond
  ld #genIn2
endWait
```

Stand by at the point until the following conditions are met:
#genIn2=ON (Condition)
End of the condition line

■ waitCondtime – timeUp – endWait: the robot waits for the specified period of time until the conditions are met.

Example: If a workpiece does not arrive within 3 seconds, an error occurs, an external lamp (connected to #genOut2) comes ON, and the robot stands by for a start instruction. Once you have resolved the problem, press the start switch to restart operation.

```
waitCondTime 3000
  ld #genIn2
timeUp
  set #genOut2
  waitStartBZ
  reset #genOut2
endWait
```

Wait for 3 seconds until the following conditions are met:
#genIn2=ON (Condition)
If the condition is not met within 3 seconds,
ON output to #genOut2,
Stand by in place for a start instruction.
OFF output to #genOut2 if a start instruction is received.
End of the command line if the condition is not met within 3 seconds.

NOTE:

- *endWait* and *timeUp* cannot be used independently.
- For *waitCondTime*, the wait time can be specified using variables and expressions.

Example:

```
declare numeric wtime
if
   ld #genIn3
then
   let wtime = 3000
else
   let wtime = 1000
endIf
waitCondTime wtime
   ld #genIn2
timeUp
   set #genOut2
   waitStartBZ
   reset #genOut2
endWait
```

Declare the local variable *wtime*.
If
   #genIn3=ON
then
   Assign 3000 to *wtime*.
If not
   Assign 1000 to *wtime*.

Wait for 3 or 1sec until the following condition is met:
   #genIn2=ON (Condition)
If the condition is not met within 3 or 1sec,
   Output ON signal to #genOut2,
   Stand by in place for a start instruction.
   OFF output to #genOut2 if a start instruction is received.
   End of the command line if the condition is not met within 3 or 1sec.

12. CONDITIONS

12.1 Condition Settings

■ ld, ldi, and, ani, or, ori, anb, orb

This chapter explains the conditional operation commands that come after the *if Branch* and *Wait Condition* commands (*if, waitCond, waitCondTime*). These belong to the [Condition] command category.

Command Category	Command	Parameter	Job
Condition	ld	Boolean variable or expression	ON input
	ldi	Boolean variable or expression	OFF input
	and	Boolean variable or expression	Serial ON input
	ani	Boolean variable or expression	Serial OFF input
	or	Boolean variable or expression	Parallel ON input
	ori	Boolean variable or expression	Parallel OFF input
	anb	–	Block serial connection
	orb	–	Block parallel connection

NOTE: Boolean variables are handled as follows: ON (true) when not 0; OFF (false) when 0.

In addition to I/O-SYS input (#sysIn), I/O-1 input (#genIn), I/O-H input (#handIn), and I/O-FB input (#fbIn), you can also specify I/O-SYS output (#sysOut), I/O-1 output (#genOut), I/O-H output (#handOut), I/O-FB output (#fbOut), system flag (#sysflag), internal relay (#mv), keep relay (#mkv), pallet flag, I/O alias, PLC timer (#seqT), and PLC counter (#seqC) as command parameters. Comparative operation expressions can also be used. Variables and functions can also be used in comparative operation expressions in addition to the above parameters.

Comparative operation expression	Meaning
○ == □	□ is equal to ○ .
○ < □	□ is greater than ○ .
○ > □	□ is less than ○ .

Comparative operation expression	Meaning
○ <= □ ○ =< □	□ is greater than or equal to ○ .
○ >= □ ○ => □	□ is less than or equal to ○ .
○ <> □ ○ >< □	Not equal.

A [Condition] command must always start from an *ld* or *ldi* command line. If the command includes only an independent ON (true) or OFF (false) condition, it needs 1 line, but when multiple conditions are connected with *and, or*, etc., multiple lines are required. Expressions can also be used in the condition commands. In this case, the result of the expression is judged as 0 (false) or nonzero (true).

■ ld: ON input

```
waitCond
  ld #genIn2
endWait
```

Standby in place until the following condition is met:
#genIn2=ON (Condition)
End of condition line

■ ldi: OFF input

```
waitCond
  ldi #genIn2
endWait
```

Standby in place until the following condition is met:
#genIn2=OFF (Condition)
End of condition line

■ and: Series ON input

```
waitCond
  ld #genIn1
  and count>=10
endWait
```

Standby in place until the following conditions are met:
#genIn1 is ON (Condition 1)
and count value is 10 or greater (Condition 2)
End of condition line

■ ani: Series OFF input

```
waitCond
  ldi #genIn1
  ani count>=10
endWait
```

Standby in place until the following conditions are met:
#genIn1 is OFF (Condition 1)
and *count* value is 10 or less (Condition 2)
End of condition line

■ or: Parallel ON input

```
waitCond
  ld #genIn1
  or #genIn2
endWait
```

Standby in place until the following conditions are met:
#genIn1 is ON (Condition 1)
or #genIn2 is ON (Condition 2).
End of condition line

■ ori: Parallel OFF input

```
waitCond
  ldi #genIn1
  ori #genIn2
endWait
```

Standby in place until the following conditions are met:
#genIn1 is OFF (Condition 1)
or #genIn2 is OFF (Condition 2)
End of condition line

■ anb: Block serial connection

```
waitCond
   ld count>=10
   or flag
   ldi #genIn1
   ani #genIn2
   anb
endWait
```

Standby in place until the following conditions are met:

count is 10 or greater
or if *flag* is ON — Condition 1

#genIn1 is OFF
and #genIn2 is also OFF — Condition 2

if both conditions 1 and 2 are true,
End of condition line

■ orb: Block parallel connection

```
waitCond
   ld count>=10
   or flag
   ldi #genIn1
   ani #genIn2
   orb
endWait
```

Standby in place until the following conditions are met.

count is 10 or greater
or if *flag* is ON — Condition 1

#genIn2 is OFF
and #genIn2 is also OFF — Condition 2

if either condition 1 or 2 is true,
End of condition line

NOTE:

- When there is no *ld* or *ldi* corresponding to *anb* or *orb*, an error occurs and the error message “Error on Point Job” is displayed.
- In these setting examples *count* and *flag* are arbitrary variables.
- When you set an assignment expression that uses a variable, if a declaration is not set, an error occurs. For further details refer to “20. VARIABLE, COMMENT, SYSTEM CONTROL.”

13. DELAY, DATA IN, WAIT START

13.1 Time Delay

■ delay

This section explains the point job data command for controlling time delay.

Command Category	Command	Parameter	Job
Delay, Data In, Wait Start	delay	Delay Time	Stand by in place for the specified delay time.

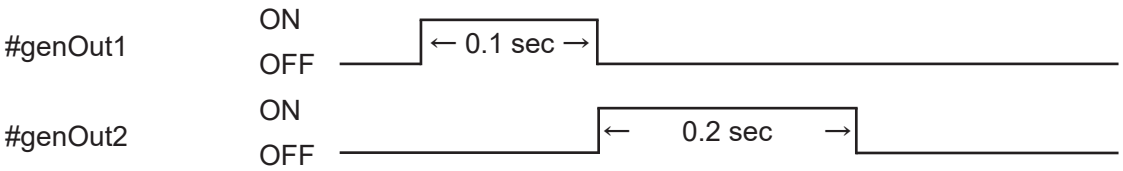
NOTE: The *delay* command is disabled when the point is a CP passing point as well as when a CP passing point is the point type used as a base type.

■ delay: Delay for a specified period of time

Example:

```
set #genOut1
 delay 100
 reset #genOut1
 set #genOut2
 delay 200
 reset #genOut2
```

Output ON signal to #genOut1,
Delay for 0.1sec.
Output OFF signal to #genOut1.
Output ON signal to #genOut2.
Delay for 0.2sec.
Output OFF signal to #genOut2.



The delay time can be set using variables or expressions as well as numeric values.

Example:

```
declare numeric wtime
if
   ld #genIn1
then
   let wtime = 100
else
   let wtime = 200
endIf
set #genOut1
delay wtime
reset #genOut1
```

Declare the local variable *wtime*.
If
    #genIn1=ON
then
    Assign 100 to *wtime*.
If not
    Assign 200 to *wtime*.

ON output to #genOut1.
Delay for 0.1 or 0.2sec.
OFF output to #genOut1.

13.2 Waiting for a Start Signal

■ waitStart, waitStartBZ

This section explains point job data commands that stop the robot until there is a start instruction.

Command Category	Command	Parameter	Job
Delay, Data In, Wait Start	waitStart	-	Stand by in place until start instruction.
	waitStartBZ	-	Stand by in place while sounding the buzzer until start instruction.

NOTE: The *waitStart* and *waitStartBZ* commands are disabled at CP passing points, as well as at points where the base point type is set as [CP Passing Point].

■ waitStart: Wait for start

Exapmle:

```
set #genOut1
waitStart
reset #genOut1
```

ON output to #genOut1.
Stand by in place until start instruction.
OFF output to #genOut1 (if a start instruction is received).

■ waitStartBZ: Wait for start (with buzzer)

Example: If #genIn1 does not come ON within 2 seconds, it is recognized as an error and #genOut2 (connected to an external alarm or alarm lamp) comes ON, and the robot "stands by for a start signal while the buzzer sounds".
Once you have resolved the problem and pressed the start switch again, OFF output is sent to #genOut2 and the operation restarts from point 05.

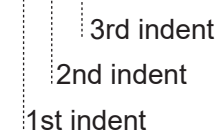
```
waitCondTime 2000
  ld #genIn1
timeUp
  upZ 20, 50
  set #genOut2
  waitStartBZ
  reset #genOut2
  goPoint PTP3, 5
endWait
```

Wait for 2 seconds until #genIn1 comes ON.

If #genIn1 does not come ON within 2 seconds,
Raise the Z (J3) axis 50 mm (at 20 mm/sec),
ON output to #genOut2,
Sound the buzzer and stand by in place until start instruction.
OFF output to #genOut2 (when a start instruction is received),
Go to Point 05.
End of the command if #genIn1 does not come on within 2 seconds.

NOTE: If the *waitCondTime, timeUp – endWait* and *if – endIf* commands are used, the command lines are indented (as shown on the next page).

```
waitCondTime 200
  ld #genIn2
timeUp
  set #genOut2
  if
    ld #genIn1
  then
    downZ 20,20
    waitCondTime 200
      ld #genIn4
    timeUp
      waitStartBZ
    endWait
  endIf
endWait
```



Be sure not to use more than 9 indents.

If the point job data includes more than 9 indents, an error occurs and the error message [Error on Point Job] is displayed.

When *timeUp* or *endWait* comes before *waitCondTime*, or if *then, else* or *endIf* comes before *if*, an error occurs and the error message [Error on Point Job] is displayed.

13.3 Input from I/O

■ dataIn, dataInBCD

Read out a numeric value from I/O, Boolean variables #mv (1 – 99) or #mkv (1 – 99) and assign the value to a specified variable.

Command Category	Command	Parameter			Job
Delay, Data In, Wait Start	dataIn	Destination Variable	Read Source	Read Width	Read numeric data from I/O.
	dataInBCD	Destination Variable	Read Source	Read Width	Read numeric data from I/O in BCD.

BCD: binary-coded decimal

Read out width can be set using variables or expressions.

Other than assigning variables to the readout values, *dataIn* and *dataInBCD* commands also need the parameters; “read width” (the number of I/O pins used for input), and “read destination” (the smallest I/O pin number used for input, for example, if you use #genIn3 – 10, the smallest number is “3”).

NOTE: The I/O width from the read source uses serial numbers. You cannot use these commands if the I/O numbers you are using are not consecutive.

Example:

declare numeric code dataIn code, #genIn1, 8	Declare the local variable *code*. Read data #genIn1 (I/O-1) – #genIn8 as a numeric value and assign it to *code*.

declare numeric code dataInBCD code, #genIn1, 8	Declare the local variable *code*. Read data #genIn1 (I/O-1) – #genIn8 in BCD and assign it to *code*.

IStatus of I/O-1

#genIn8	#genIn7	#genIn6	#genIn5	#genIn4	#genIn3	#genIn2	#genIn1
OFF	OFF	OFF	ON	OFF	OFF	ON	OFF

Input width: 8

If the read status is 00010010 as it is above, the following result is generated:
When using the *dataIn* command, the value of *code* is 18.
When using the *dataInBCD* command, the value of *code* is 12.

NOTE: The maximum input width you can set is “32.” However, you cannot set the read width to extend into different types of I/O.

If using Fieldbus as the read source, the input is as follows:

Example:

declare numeric code dataIn code, #fbIn (1020), 32	Declare the local variable *code*. Read data #fbIn(1020) – #fbIn(103F) as a numeric value and assign it to *code*.

declare numeric code dataInBCD code, #fbIn (1020), 32	Declare the local variable *code*. Read data #fbIn(1020) – #fbIn(103F) in BCD and assign it to *code*.

NOTE: Set the Fieldbus input area 102h – 103h from an external device (PLC) with the following:
102h : 0000h (hexadecimal)
103h : 0010h (hexadecimal)

If the read status is set as per the previous page, the following result is generated:
If using the *dataIn* command, *code* value is 1048576.
If using the *dataInBCD* command, *code* value is 100000.

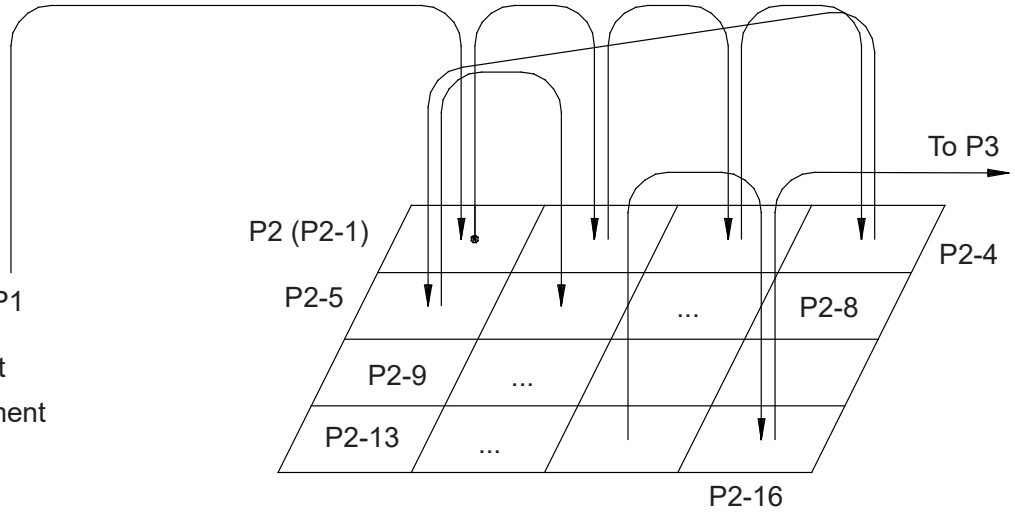
14. PALLET CONTROL

14.1 Pallet Command

There are two types of additional function data [Pallet Routine]: one is [Auto Increment], which increases the counter automatically (the tool unit moves to the next point on the pallet sequentially), and the other is [Point Job Control], which does not increase the counter (the tool unit does not move to the next position on the pallet) unless you set point job data so the counter is updated. You can switch between these.

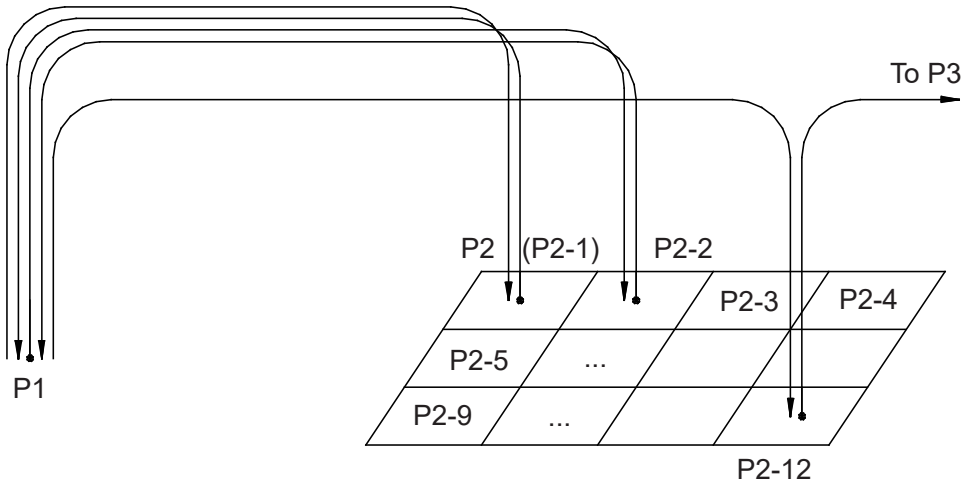
If you select Auto Increment, you do not need to set a point job command to control the pallet operation. The tool unit moves to the next point and automatically updates the pallet counter. However, with Auto Increment pallets, the tool unit can only move in a serial order P2-1 → P2-2 → P2-3 ... as shown below.

Example: Auto Increment Pallet



Plane Pallet
Auto Increment
Row: 4
Column: 4

Example: [Point Job Control] Pallet



Plane Pallet
Point Job Control
Row: 4
Column: 3

On the Point Job Control pallet, the robot can move randomly, as shown in the diagram on the previous page. For example, the robot returns to P1 each time before continuing (P1 → P2 (P2-1) → P1 → P2-2 → P1 → P2-3...)

The following three commands are used for [Point Job Control]:

Command Category	Command	Parameter	Job
Pallet Control	loopPallet	Pallet Number, go Point Number	Add 1 to the pallet counter and if the counter has not reached its maximum value, the tool unit will move to the specified point.
	resPallet	Pallet Number	Reset the counter to 0. If the pallet flag is 1 (true), the pallet flag goes OFF (false). (system software version 5 and later)*
	incPallet	Pallet Number	Add 1 to the pallet counter.

* The *resPallet* command functions differently depending on the system software version:

■ Version 4 or Older

- The pallet counter is reset to 0.
- When the pallet flag is ON (true), it does not go OFF (false).

■ Version 5 or Newer

- The pallet counter is reset to 0.
- When the pallet flag is ON (true), it goes OFF (false).

The following two variables can also be used to control the pallet:

[palletFlag(n)]: A Boolean variable which has the following content:
The Pallet Counter (No. n) is at maximum = ON (true)
The Pallet Counter (No. n) is not at maximum = OFF (false)
Resetting the Pallet Flag
Reset the pallet flag by either power cycling the robot/controller or starting a run.
After the pallet flag comes ON (true) it holds this value until it is reset.
[palletCount(n)]: A numeric variable which has the value of Pallet Counter (No. n)

In the following example of point job data, the tool unit picks up the object from P1 (set #genOut1) and places it at P2 (reset #genOut1) on the [Increment by Point Job] pallet shown on the previous page:

Point Job Data (to be set to P1)

set #genOut1

Picks up the object.

Point Job Data (to be set to P2)

reset #genOut1 loopPallet 10,1	Releases (places) the object. Add 1 to the Pallet No. 10 counter. If the counter is at maximum, go to the next command. (In this example, the point job is over because there are no more commands.) If the counter is not at maximum, move to P1.

NOTE: The robot moves (to P1 if using point job data set to P2 above) according to the program data [PTP Condition].

If the incPallet command (1 is added to the specified pallet counter) is used instead of the *loopPallet* command, the pallet control commands are as follows:

Example: *incPallet* is used instead of *loopPallet*.

reset #genOut1 incPallet 10 if ldi #palletFlag （10） then goPoint PTP3,1 endIf	Release (Place) the object. Add 1 to the Pallet No. 10. counter If If the counter of Pallet 10 is not at maximum, Go to P1 (according to PTP Condition 03).

NOTE: If you use the *loopPallet* command, the robot moves to the specified point according to the program data [PTP Condition]. However, if you use the *incPallet* command, you can use it together with the *goPoint* or *goRPoint* commands, and you can select the additional function data [PTP Condition].

If you use the *incPallet* command, you can also pulse output, etc., every time the robot moves to P1.

reset #genOut1 incPallet 10 if ldi #palletFlag （10） then pulse #genOut5,200 goPoint PTP0,1 endIf	Release (Place) the workpiece. Add 1 to the Pallet No. 10. counter if if the counter of Pallet 10 is not at maximum, Pulse output and move to P1.

The pallet number (if using the *loopPallet* command, the point destination number as well) can be specified using expressions.

Example:

```
declare numeric pal
if
   ld #genIn3
then
   let pal = 5
else
   let pal = 6
endIf
reset #genOut1
loopPallet pal,1
```

Declare a local variable pal.
If
    #genIn3=ON
then
    5 is assigned to *pal*.
If not
    6 is assigned to *pal*.

Release (Place) the object.
Add 1 to the Pallet 5 or 6 counter,
if the counter is at maximum, go to the next command. (In this this example, the point job is over because there are no more commands.)
If the counter is not at maximum, move to P1.

15. EXECUTION FLOW CONTROL

15.1 Subroutine Calling Jobs Set to Point Types

■ callBase

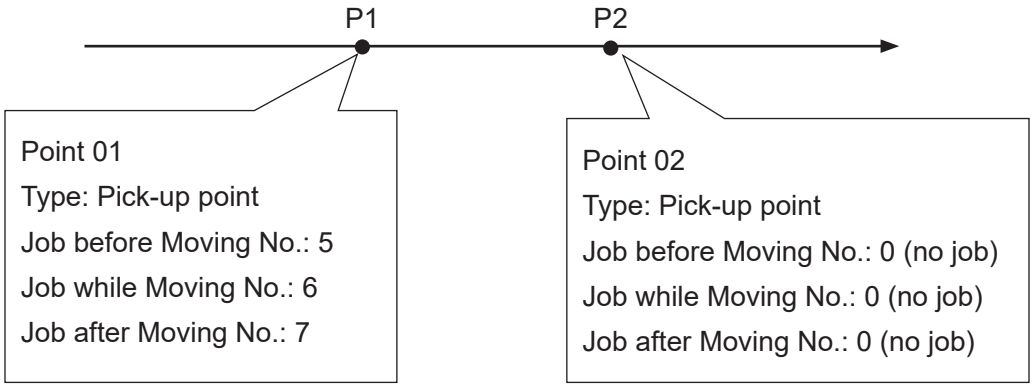
If you set a point job, etc., to a user-defined point type created in Customizing Mode, you cannot perform the point job attached to the point type. For example, with [Wait Start] the robot waits until the start/stop switch is pressed or a start signal is received, but if an optional [Point Job] is set, the robot will not wait at this point.

Example: At Points P1 and P2 where the user-defined point type shown to the right is set, the following point job data is performed at each point:

Title:	Pick-up point
Base type:	PTP Point
Job before Moving:	Yes
Job while Moving:	Yes
Job after Moving:	Yes

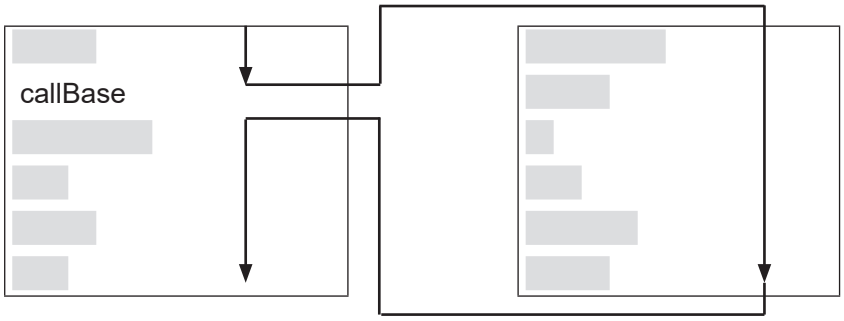
(P1) Job before Moving: Point job data 5
Job while Moving: Point job data 6
Job after Moving: Point job data 7

(P2) Job before Moving: [Job before Moving] set under the user-defined point type
Job while Moving: [Job while Moving] set under the user-defined point type
Job after Moving: [Job after Moving] set under the user-defined point type



In cases like this, if you use the *callBase* command with the point job data set to a user-defined point, the original job attached to that point type can be made into a subroutine call and performed. With the example on the previous page, if you use the *callBase* command for Point Job Data 7, the point job attached to the user-defined point type in Point Job Data 7 can be made into a subroutine call in the command string when executing the point job at P1.

(Point Job Data 7) (Point job attached to the user-defined point type)



Command Category	Command	Parameter	Job
Execute Flow Control	callBase	–	Call and execute the job command string attached to the point type at the point where the user-defined point type is set.

The job command string attached to the point type can be made into a subroutine call by the *callBase* command. Therefore, if the *callBase* command is executed in a [Job before Moving], [Job while Moving], or [Job while CP Moving] operation, the command string for these jobs are made into a subroutine call.
In the example on the previous page, if the *callBase* command is used in Point Job Data 5, when [Job before Moving] set to P1 is executed, the command string attached to the point type [Job before Moving] is made into a subroutine call.

15.2 Subroutine Calling Point Job Data

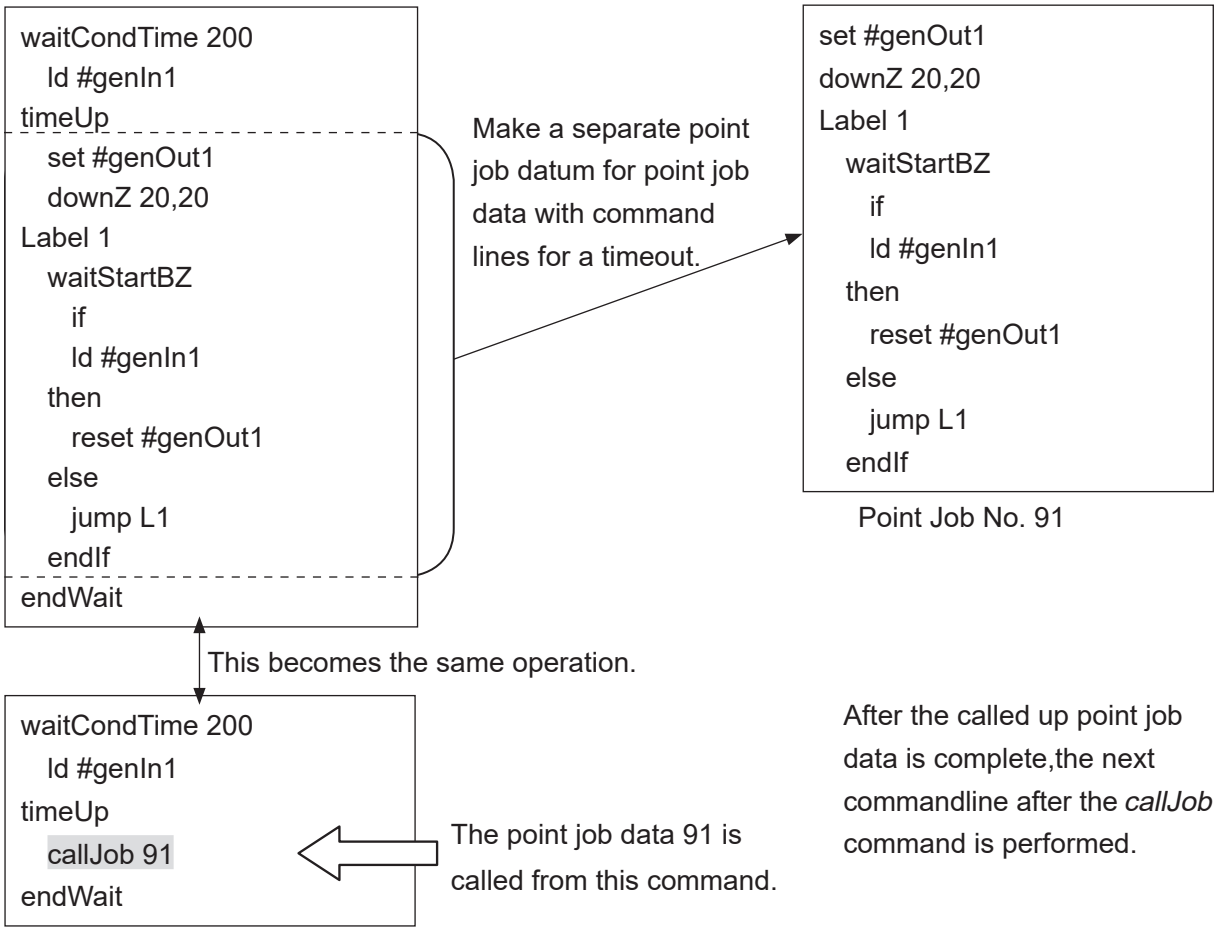
■ callJob

While performing a point job, other point job data can be called up and executed (as a subroutine).

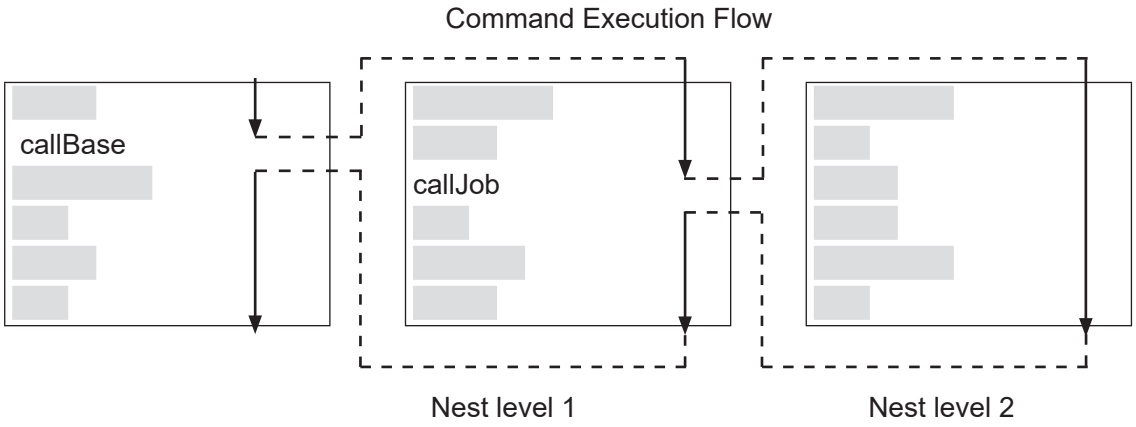
To handle errors, etc., point job data can be made short and easy if you take common parts of multiple point jobs and make it into one point job datum to call up and use from other point job data.
Also, if you take a single command line which is part of point job data and make it into a single point job datum, you can test that individual part.

Command Category	Command	Parameter	Job
Execute Flow Control	callJob	Point Job Data Number	Make a subroutine call for the specified point job number.

NOTE: The *callJob* command is disabled at CP passing and points where a CP passing point is set as the base point type.



NOTE: When point job data called by the *callJob* command contains a *callJob* command, if the nest level exceeds Level 30 (No. 043), an error occurs. (The following example shows nest level 2.)



Point job data numbers can also be specified using expressions.

Example:

```
declare numeric ejob
waitCondTime 200
   ld #genIn1
timeUp
   if
      ld #genIn2
   then
      let ejob = 9
   else
      let ejob = 10
   endIf
   callJob ejob
endWait
```

Local variable *ejob* declaration
Wait for 0.2 seconds until the following condition is met:
    #genIn1=ON (Condition)
If the condition is not met within 0.2 seconds,
if
    #genIn2=ON
then
9 is assigned to *ejob*.
If not,
10 is assigned to *ejob*.

Point job data No. 9/10 is called by subroutine.

15.3 End the Point Job

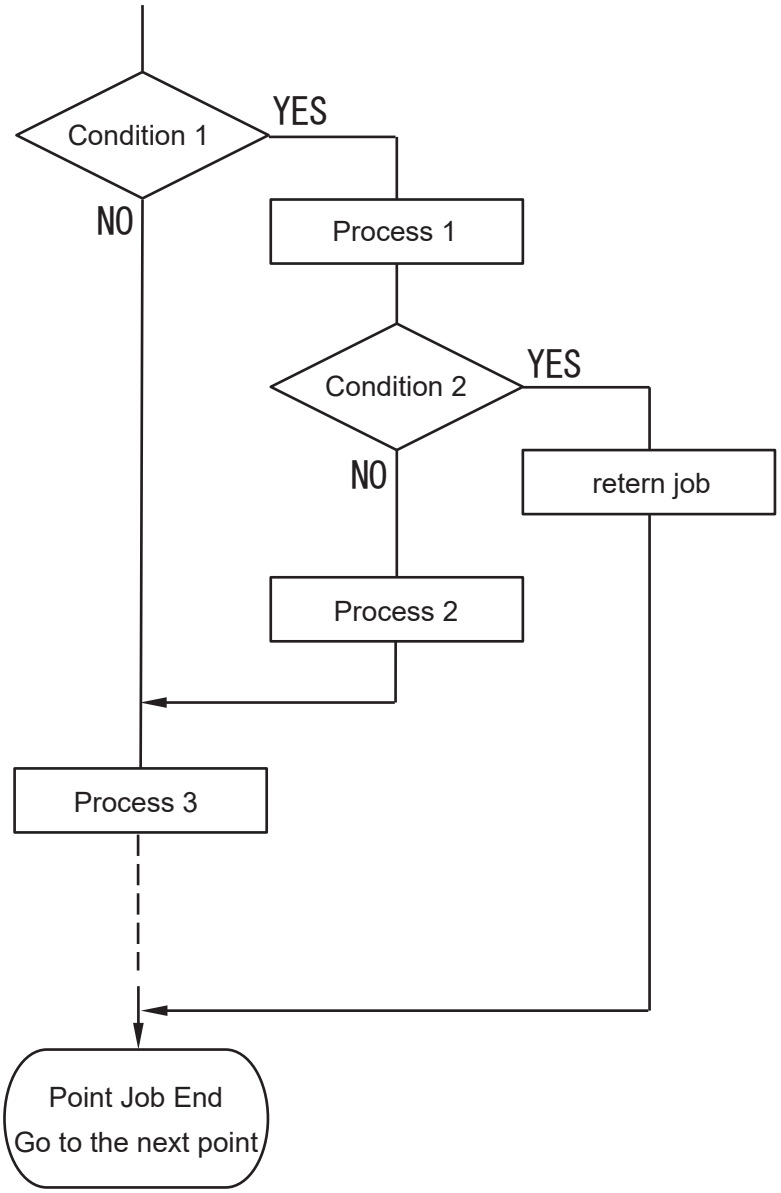
■ returnJob

If there is a complex condition that requires handling and there is no process in place to handle it, the point job can be ended by the *returnJob* command.

Command Category	Command	Parameter	Job
Execute Flow Control	returnJob	–	End point job.

Example: If you register a point job as such on the right, the point job data commands are like these on the left.

```
if
  Condition 1
then
  Process 1
  if
    Condition 2
  then
    returnJob
  else
    Process 2
  endIf
endIf
Process 3
```



If the *returnJob* command is removed from the point job data, Process 3 is performed even if Condition 2 is ON (YES).

15.4 Subroutine Calling a Program

■ callProg

While performing a point job, another program can be called and executed (as a subroutine).

Command Category	Command	Parameter	Job
Execute Flow Control	callProg	Program Number	Call up the specified program number as a subroutine.

NOTE: The *callProg* command is disabled at CP passing points and points where a CP passing point is set as the base point type.

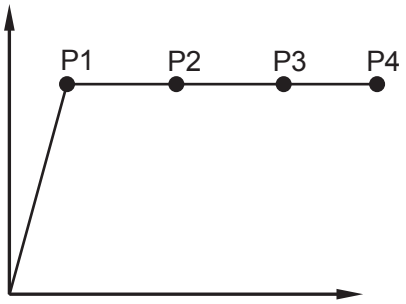
```
waitCondTime 200
  Id #genIn1
timeUp
  callProg 17
endWait
```

Program No. 17 is called by the *callProg* command.

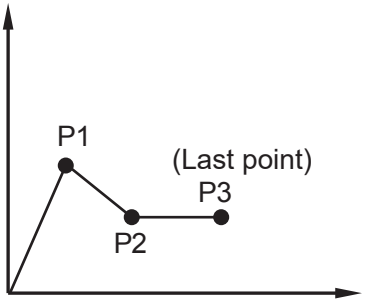
After the called program is complete, the next point job data command line (on the side that called the program) after the *callProg* command (*endWait* in this example) is performed.

Below, the program called up is referred to as “subprogram” and subprogram point 1 is referred to as “SP1”.

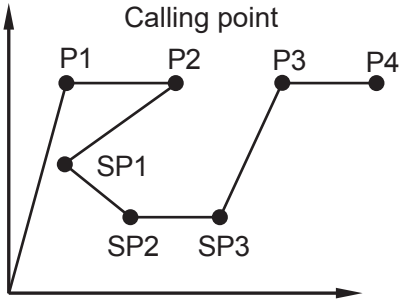
NOTE: If [Cycle Mode] in the subprogram is set to [Continuous Playback], the subprogram is continuously run.



Program 5



Program 12



Program 12 was called from P2 in Program 5.

Program numbers can also be specified using expressions.

Exapmle:

```
declare numeric eprg
waitCondTime 200
  ld #genIn1
timeUp
  if
    ld #genIn2
  then
    let eprg = 9
  else
    let eprg = 10
  endIf
  callProg eprg
endWait
```

Declare the local variable *eprg*.
Wait for 0.2 seconds until the following condition is met:
    #genIn1=ON (Condition)
If the condition is not met within 0.2 seconds,
If
    #genIn2=ON
then
    9 is assigned to *eprg*.
If not,
    10 is assigned to *eprg*.

Program No. 9/10 is called as a subroutine.

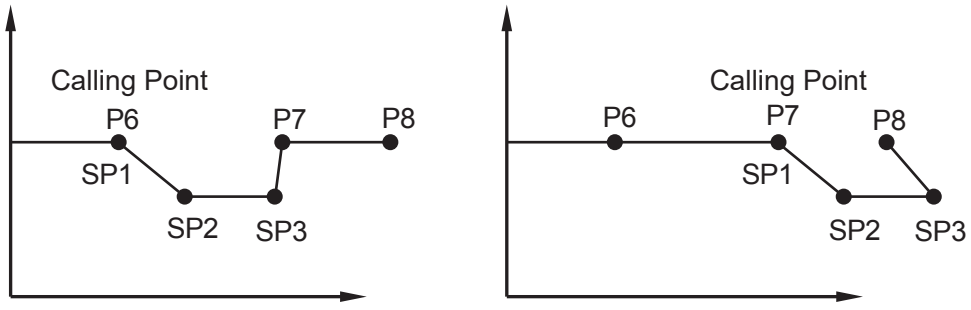
■ Position Data Settings (Program Data Item Names)

You can set the handling method for the coordinates (position data) included in the point data with the program data [Position Data]. There are 3 handling methods:

- Absolute: The position data values are treated as the robot's absolute coordinates.
- Relative: The position data values are treated as the distance from the program start coordinates.
- Moving Amount: The position data values are treated as the distance to the next point.

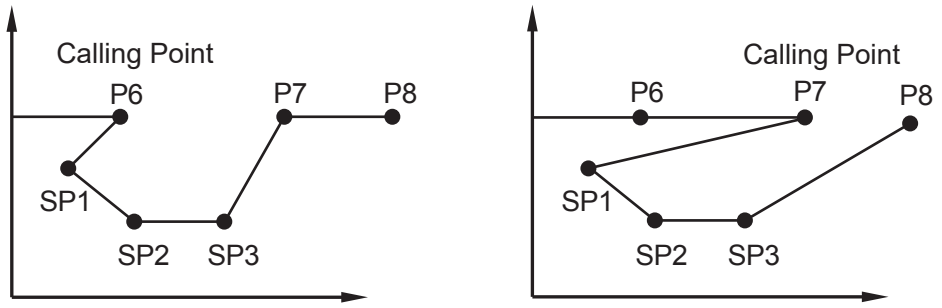
If you set a subprogram to [Relative] or [Moving Amount], the robot always runs at an equal distance to the calling point (where point job data including the *callProg* command is set.)

Example: The subprogram is set to [Relative] or [Moving Amount].
The current point (calling point) is handled as SP1 (Subprogram Point 1); (but the point coordinate for P1 of the subprogram is ignored). The work home is ignored.



Exapmle: The subprogram is set to [Absolute].

The robot runs on the point data coordinates regardless of the position of the calling point. At the current point (calling point), the robot performs the subprogram work home starting point job (without moving), and then moves to P1 (SP1).

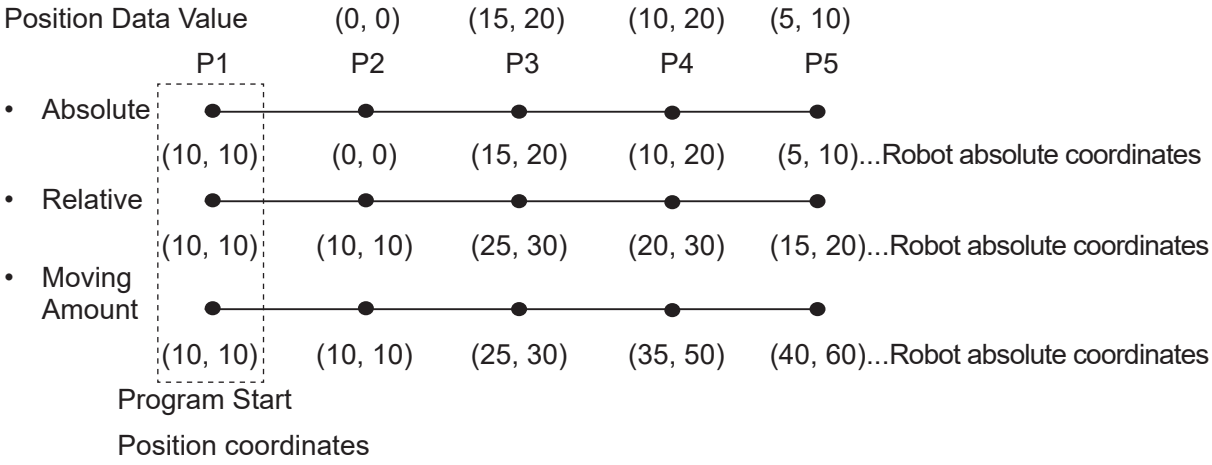


NOTE: When a program called by the *callProg* command contains a *callProg* command, an error occurs if the nest level exceeds Level 30.

Handling methods of coordinates (position data) included in point data can be selected from [Absolute], [Relative], or [Moving Amount]. The coordinate type is set to [Absolute] by default.

- Absolute: The position data values are treated as the robot's fixed coordinates.
- Relative: The position data values are treated as the distance from the program start coordinates. (If the start coordinates are (0, 0), the result is the same as Absolute coordinates.)
- Moving Amount: The position data values are treated as the distance to the next point.

Depending on the handling method of the position data, the position of the points may vary even if the values are the same. (See below)



NOTE: If you run a program as a subprogram, the robot will ignore the work home. Also, it the specified program has [Relative] or [Moving Amount] coordinates, the robot will not return to the work home.
The robot returns to the work home only when the specified program has [Absolute] coordinates and is performed independently (not run by a *callProg* command).

■ [Relative] Program Teaching

When teaching a point in JOG mode, teaching is done in absolute coordinates, regardless of the position data settings.
When creating a program with relative coordinates, after registering the points, move (offset) all the points so the coordinates of the first point are (0, 0, 0).

■ [Moving Amount] Program Teaching

You cannot convert registered coordinates to Moving Amount coordinates. Point teaching for all points in [Moving Amount] must be done in MDI Mode.

15.5 Subroutine Call for a Point String

■ callPoints

A point string (defined in Customizing Mode) with an identifier can be called up and executed.

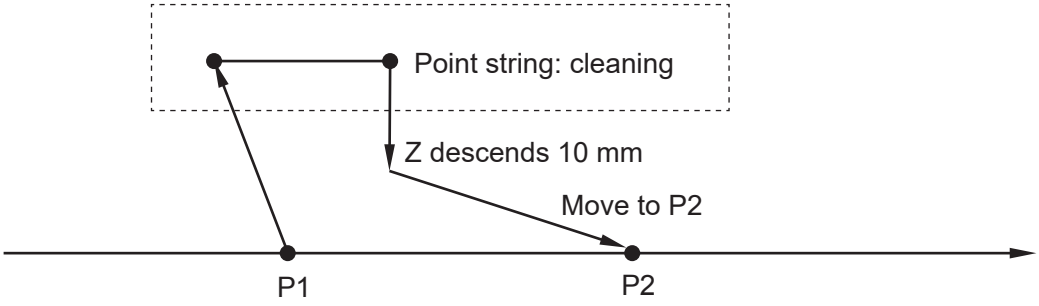
Command Category	Command	Parameter	Job
Execute Flow Control	callPoints	Point String Identifier	Call up the specified point string as a subroutine.

NOTE: The *callPoints* command is disabled at CP passing points and at points where CP passing points are set as the base point type.

Example: When the point job data below is set to P1.

```
if
  ld #genIn1
then
  callPoints cleaning
endIf
downZ 10,10
set #genOut1
```

This command calls the “cleaning” point string.



If #genIn1 is ON, the robot calls up the “cleaning” point string and executes the point job data and additional function data set to the “cleaning” point string. After executing the cleaning point string, the robot lowers the Z (J3) axis by 10mm and sends an ON output to #genOut1. The robot then moves to P2. If #genIn1 is OFF, the robot simply lowers the Z (J3) axis by 10 mm and ON output is sent to #genOut1.

15.6 Forced Program Termination

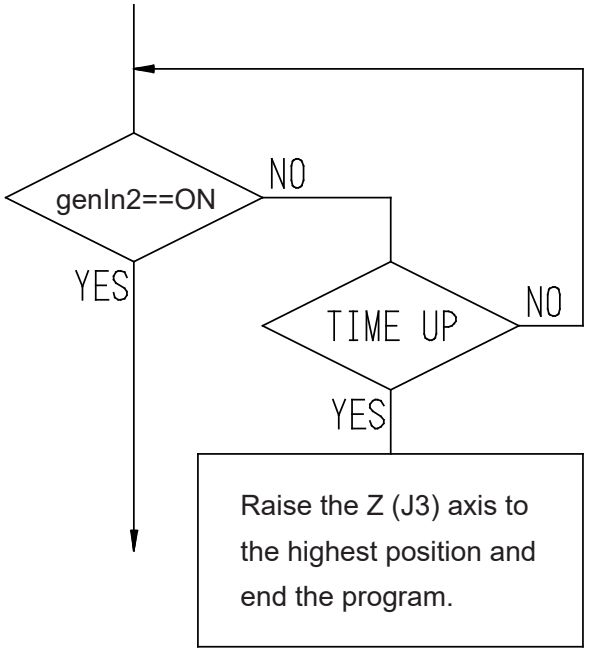
■ endProg

A program (run) can be terminated on the spot with the *endProg* command. The robot does not return to the work home position.

Command Category	Command	Parameter	Job
Execute Flow Control	endProg	–	End a program run on the spot.

Example: If you register a point job as such on the right, the point job data commands are like those on the left.

```
waitCondTime 500
   ld #genIn2
timeUp
   movetoZ 10,0
   endProg
endWait
```



With the *endProg* command, the program is terminated on the spot and the robot does not return to the work home position. As this is not a temporary stop, you cannot restart and continue the run. Restarting means the program is started from the start.

NOTE: If you want to terminate the program after returning to the work home position, set and use the *goPoint* command with the destination number [0].

15.7 Assigning the Return Value of a Function

■ returnFunc

Assign the value of a specified expression as the return value and end the function.

Command Category	Command	Parameter	Job
Execute Flow Control	returnFunc	Expression	Assign the specified expression as a return value and end the function.

NOTE: The *returnFunc* command cannot be used in point job data.

Point Job Data

```
eoutLCD 7,4,str(radians(x))
```

The function is called.

Function (Identifier: radians)

```
returnFunc 0.017453*x
```

* str() converts numerical values into decimal character strings.

The return value of the function *radians* for an argument x is displayed on the teaching pendant LCD.

15.8 Jump to a Specified Point

■ goPoint, goRPoint, goCRPoint

With these commands, after completing a point job at a given point, instead of going to the next point, you can jump to a specified point.

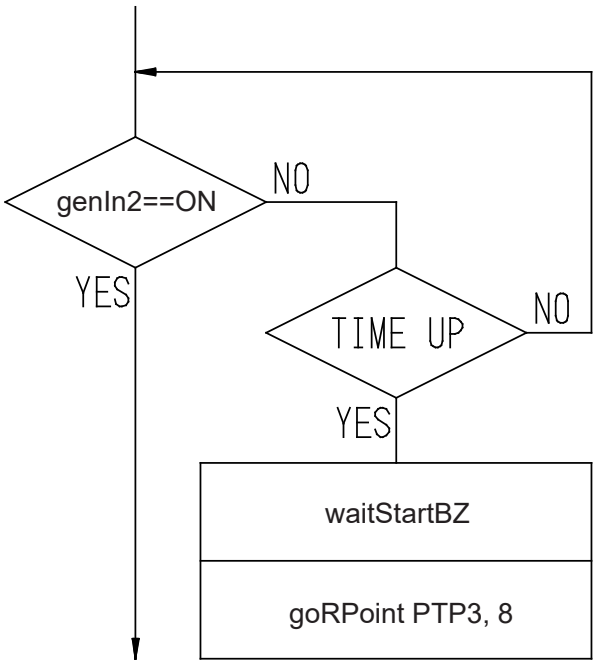
Command Category	Command	Parameter	Job
Execute Flow Control	goPoint	PTP Condition Number, Point Number	Jump to the specified point.
	goRPoint	PTP Condition Number, Relative Point Number	Jump to the specified relative point.
	goCRPoint	PTP Condition Number, Select destination	Jump to the specified destination during a CP movement.

NOTE:

- The *goPoint, goRPoint* and *goCRPoint* commands are disabled at CP passing points and points where a CP Passing Point is set at the base point type.
- The point number and relative point number for the *goPoint* and *goRPoint* commands can be specified using variables or expressions.
  You can set the destination selection for the *goCRPoint* command using variables or expressions, but the value must be either [0] or [1].

Example: If you register a point job as such on the right, the point job data commands are like those on the left.

```
waitCondTime 500
   ld #genIn2
timeUp
   waitStartBZ
   goRPoint PTP3, 8
endWait
```



What this point job means:
If #genIn2 does not come ON within 0.5 seconds, the buzzer sounds and the robot will wait for a start signal. After receiving the start signal, the robot restarts operation from 8 points ahead (plus 8 points) of the current point number.

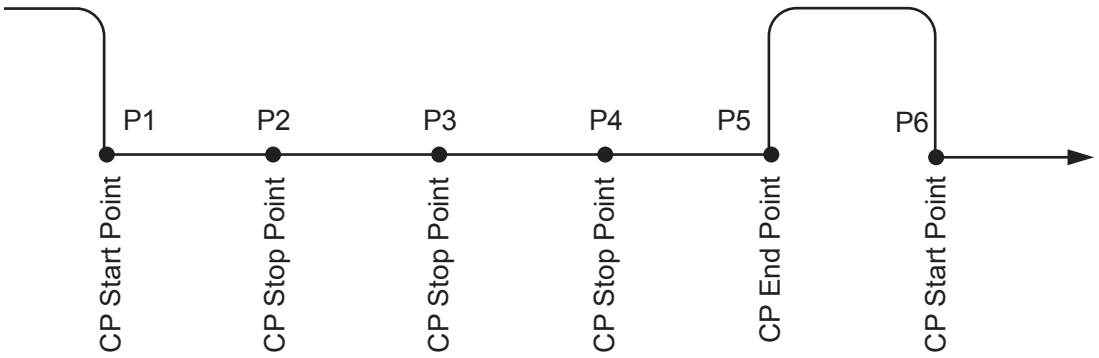
[goPoint PTP3,25]: Jump to Point 25 (the robot moves according to PTP Condition 03).
If you set 0 as the PTP Condition Number, the robot moves according to the PTP Condition set in program data.
If you set 0 as the Point Number, the robot returns to the work home position. (The robot jumps to a specified point number.)

[goRPoint PTP3,-4]: Jump to 4 points behind (minus 4 points) the current point (the robot moves according to PTP Condition 03).
If you set 0 as the PTP Condition Number, the robot moves according to the PTP Condition set in program data.
If you set 0 as the Relative Point Number, the robot restarts operation from the same point. (The robot jumps to a relatively specified point number.)

[goCRPoint PTP3,1]: This command is used to jump to a specified point while in the middle of making a CP movement.
CP Start Point to CP End Point is performed as one operation, and if the destination number is set to 0, the robot returns to the point where the current CP movement started (CP Start Point) (the robot moves according to PTP Condition 03). If 1 is set, the robot moves to the next point after the CP End Point (the robot moves according to PTP Condition 03).
If you set 0 as the PTP Condition Number, the robot moves according to the PTP Conditions set in program data.

Example: if this command is executed between P1 and P5, the robot moves according to the destination number as follows:

Destination 0: The robot moves to P1
Destination 1: The robot moves to P6

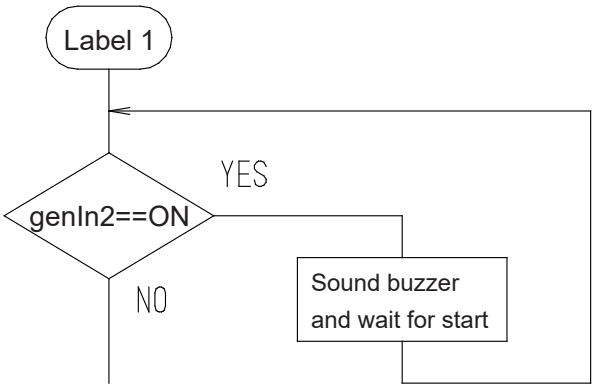


15.9 Jumping to a Specified Command

■ jump, Label

Command Category	Command	Parameter	Job
Execute Flow Control	jump	Label Number	Jump to the specified *label* number.
	Label	Label Number	A destination mark for the *jump* command

Example:
If #genIn2 is ON, the buzzer sounds and the robot is on standby for a start instruction. If #genIn2 is not ON, the robot goes to the next job as is.




```
Label 1
 if
  ld #genIn2
 then
  waitStartBZ
  jump L1
 endIf
```

(Destination mark)
If the following condition is true, go to *then*. If not true, go to the next command after *endIf*.
genIn2=ON
If the above condition is true, the following commands are executed:
Sound the buzzer and stand by for a start instruction.
Jump to [Label 1] (if a start instruction was received).
End of *if Branch*

NOTE:

- The *label* command cannot be set between *if* – *endIf* or *waitCond* – *endWait* command lines.
- The label number can be set from [Label 1] – [Label 99].

16. FOR, DO-LOOP

■ for, next, exitFor, do,loop, exitDo

Command Category	Command	Parameter	Job
for, do-loop	for	Variable Name, Initial Value, End Value, Step Value	Repeats commands between *for* and *next* until the specified variable changes from the initial value to the end value.
	next	-	
	exitFor	-	Break from *for*.
	do	-	Repeat commands between *do* and *loop*.
	loop	-	
	exitDo	-	Break from *do*.

■ for – exitFor – next

The *for* command specifies the number of repetitions.

```
for ival=1 to 8 step 1
    (Contents to be repeated)
next
```

The initial value of the variable *ival* is 1. Add 1 to the variable for every loop and repeat the commands between *for* and *next* until *ival* becomes 8.

```
for ival=1 to 8 step 1
    (Contents to be repeated)
    if
        ld #genIn1
    then
        exitFor
    endIf
next
```

The *exitFor* command breaks out of the *for* – *next* loop and goes to read the command after *next*.

Condition: If #genIn1=ON, break out of the *for* – *next* loop even if *ival* is not 8 and go to read the command after *next*.

If the conditions come ON during the loop, the loop content is After the loop content has executed for 1 cycle, the robot goes to the next command.
(only if the set condition is registered in between the loop commands *for* and *next*)

The *for* command parameters (initial value, end value and step value) can be specified using variables or expressions.

```
declare numeric loop
    if
    ld #genIn1
then
    let loop = 5
else
    let loop = 10
endIf
for ival=1 to loop step 1
    (loop content)
next
```

Declare the local variable *loop*.
If
#genIn1=ON
then
  Assign 5 to *loop*.
If not
  Assign 10 to *loop*.

The initial value of the variable *ival* is 1. Add 1 to the variable for every loop and repeat the commands between *for* and *next* until *ival* is the same value (5 or 10) as *loop*.

NOTE: If you want to finish the *for* – *next* loop while the robot is still performing the loop, you can execute the *exitFor* command to finish the loop even if the *ival* value is not the same as the *loop* variable (5 or 10). Use and specify the conditions for the *exitFor* command in the *for* – *next* "loop content" using an *if* expression, etc.

■ do – exitDo – loop

The *do* – *loop* command lines are repeated until *exitDo* breaks the loop.

```
do
    (Contents to be repeated)
loop
```

Without a condition to exit from the *do* – *loop* commands, the robot goes into an infinite loop.

```
do
    (Contents to be repeated)
    if
        ld #genIn1
    then
        exitDo
    endIf
    (Contents to be repeated)
loop
```

NOTE: The "contents to be repeated" can be put both before and after the condition.

Condition:
If #genIn1=ON, break out of the *do* – *loop* and go to read the next command of the loop.

NOTE:

- If the nest level of the repetitive commands exceeds Level 10, an error occurs.
- If the repetitive commands are set as a point job to a CP passing point, or to a point where a CP passing point is set as the base point type, the robot may stop depending on the number of loops.

17. MOVE

17.1 Raising/Lowering Only the Z (J3) Axis

■ upZ, downZ, movetoZ

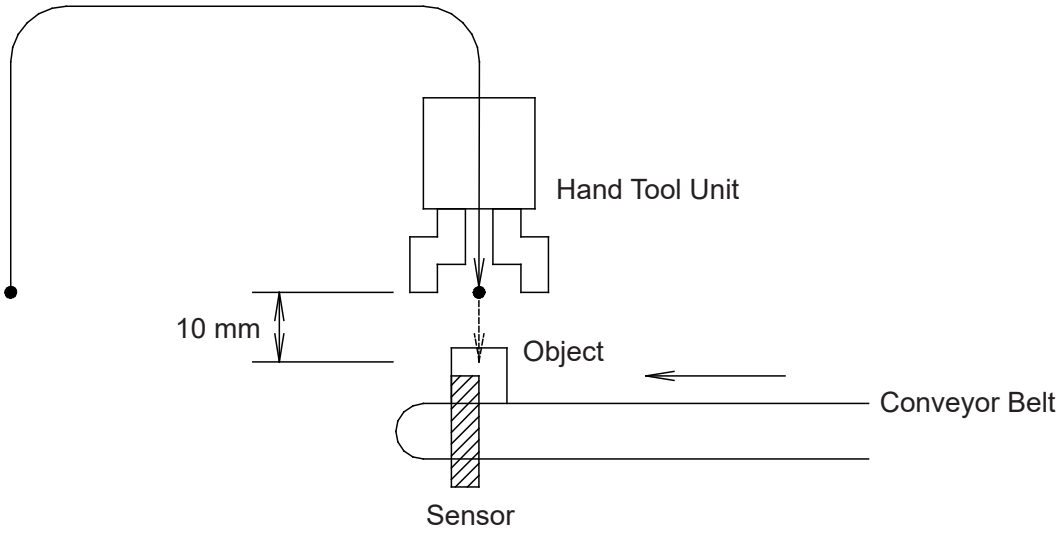
Only the Z (J3) axis can be raised or lowered using point job data. These commands belong to the *Move* command category.

Command Category	Command	Parameter	Job
Move	upZ	Speed, Distance	Raise only the Z (J3) axis by the specified distance.
	downZ	Speed, Distance	Lower only the Z (J3) axis by the specified distance.
	movetoZ	Speed, Z movement pos.	Raise or lower the Z-axis to the specified Z- coordinates (absolute coordinates).

NOTE:

- The *Move* commands are disabled at CP Passing Points and points where CP Passing Point is set as the base point type.
- These commands are invalid even if set to [Job while Moving] or [Job while CP Moving] regardless of the point type.

Example:
Picking up an object with the hand tool; the robot makes a PTP movement at high speed and instead of moving to the object pick up point, it stops short, pauses for the sensor to check whether or not the object is present, and then slowly lowers the axis to pick up the object.




```
waitCond
   ld #genIn2
endWait
downZ 20, 10
```

Wait in place until the following condition is met:
#genIn2=ON (Condition)
End of condition
Lower only the Z (J3) axis at a speed of 20 mm/sec by 10 mm.

The distance and speed can also be designated by variables and expressions.

```
waitCond
   ld #genIn2
endWait
downZ 20, #P_Z(1) -#point_Z
```

Wait in place until the following condition is met:
#genIn2=ON (Condition)
End of condition
Lower or raise only the Z (J3) axis at 20mm/sec by the distance calculated by deducting the current point Z-coordinates from the P1 Z-coordinates.

#P_Z(1): Variable which has the Z-coordinate value of P1 in the current program
#point_Z: Variable which has the Z-coordinate value of the current point

NOTE: If you assign a value (using the *let* command) to the variable #jobStartHight with a job before moving, the robot starts the job after moving from a position higher than the Z coordinate in the registered point by the amount of the assigned value. In other words, the start timing of the job after moving can be brought forward earlier.

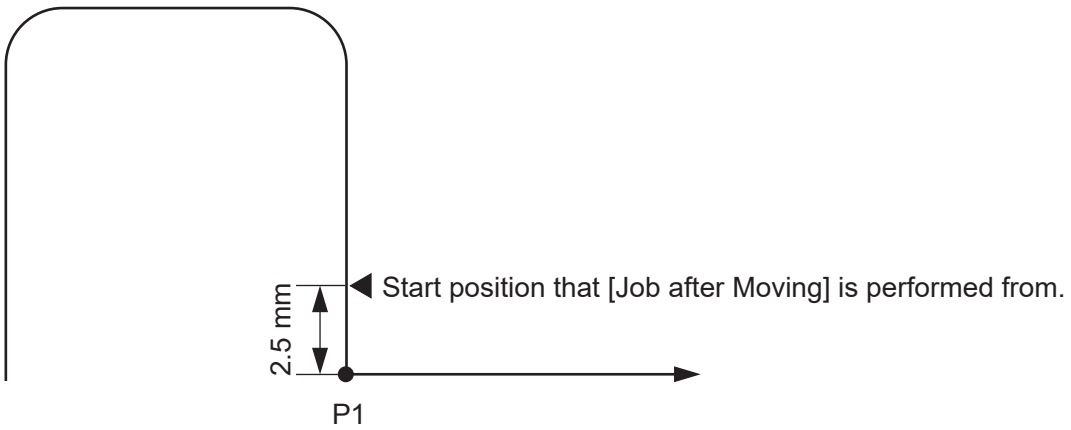
Example:

P1
Type: CP Start Point
Job before Moving: Point Job Data 3
Job after moving: Point Job Data 12

Point Job Data 3

```
let #jobStartHight = 2.5
```

NOTE: Set point job data 12 at your discretion.



17.2 Linear CP Movement in a Point Job

■ lineMove, lineMoveStopIf

The robot can make linear CP movements using point job data commands.
The CP speed and the moving amount for each coordinate axis can be set. Also, you can set conditions to terminate the movement halfway through.

Command Category	Command	Parameter	Job
Move	lineMove	Movement Speed (CP Speed) X Distance Y Distance Z Distance R Rotation Angle	The robot makes a CP movement by the distance entered.

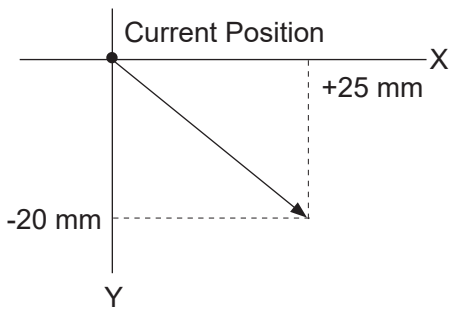
NOTE: The *Move* commands are invalid with CP Passing Points and point types which are based on CP Passing Points.

Enter each distance by the distance you want to move from the current point and not by coordinates. For directions you do not want to move, enter “0”.
Also, distance cannot only be entered by numerical values but also by variables and expressions.

When this command is entered, the commands from *lineMoveSpeed* to *endLineMove* are displayed as shown to the right.

```
lineMoveSpeed 20
  lineMoveX 25
  lineMoveY -20
  lineMoveZ 5
  lineMoveR 0
endLineMove
```

NOTE: If you press [ESC] in the middle of making settings, the settings finish at the item prior to the item which you pressed [ESC] at and you are not able to add items hereafter (you need to set the items from the start again).



Commands for each axis are displayed separately; however, all of the axes move at the same time.

■ How to stop in the middle of a CP movement according to conditions with *lineMove*.

```
lineMoveSpeed 3
   lineMoveX 20
   lineMoveY 0
   lineMoveZ 0
   lineMoveR 0
lineMoveStopIf
   ld #sysIn1
endLineMove
callJob11
```

Condition to stop movement
If #sysIn1 comes ON, even before the robot arrives at the +20 X direction, the robot stops moving and goes to the next command (callJob11).

In this case, you can check whether or not the robot moved and completed the specified distance by referring to the system flag (#sysFlag34).
0: The robot moved the specified distance.
1: The robot stopped in the middle of movement due to the conditions.

■ If the robot exceeds the move area limit with *lineMove*
With this command, if the movement results in the robot exceeding the move area limit, the robot stops moving at the point where it reached the limit and proceeds to the next command.

You can refer to the system flag (#sysFlag33) to check whether or not the robot moved the specified distance.
Normal movement: 0
Short of specified distance: 1

If the robot reaches the move area limit before finishing the specified movement, the buzzer sounds and the robot waits (for a start instruction).

```
lineMoveSpeed 3
   lineMoveX 20
   lineMoveY 0
   lineMoveZ 0
   lineMoveR 0
endLineMove
if
   ld #sysFlag （33）
then
   waitStartBZ
endIf
```

17.3 Mechanical Initialization by Point Job

■ initMec

Mechanical initialization (such as that which is done when the power to the robot is turned ON) can also be done in a point job. By mechanical initializing, it is possible to return to the absolute coordinates (x:0, y:0, z:0, r:0) even if a position error caused by an external force has occurred.

Command Category	Command	Parameter	Job
Move	initMec	Axis	Initialize the specified Axis.

NOTE:

- *Move* commands are invalid at CP passing points or where a CP passing point is set as the base point type.
- Movements and [Job while CP Moving] are invalid with *initMec*.

Specified Axis	Contents	
	JR3000 Series, JC-3 Standard Models	JC-3 Absolute Encoder Models
all	Initialize all axes.	Initialize the auxiliary (MT1, MT2) axes.
x	Initialize the X-axis.	Immediately ends the command without initializing the X-axis.
y	Initialize the Y-axis.	Immediately ends the command without initializing the Y-axis.
z	Initialize the Z-axis.	Immediately ends the command without initializing the Z-axis.
r	Initialize the R-axis.	Immediately ends the command without initializing the R-axis.
MT1	Initialize the MT1-axis.	
MT2	Initialize the MT2-axis.	

JS3 Series	
Specified Axis	Contents
all	Initialize the auxiliary (MT1, MT2) axes.
J1	Immediately end the command without returning the J1-axis to the work home.
J2	Immediately end the command without returning the J2-axis to the work home.
J3	Immediately end the command without returning the J3-axis to the work home.
J4	Immediately end the command without returning the J4-axis to the work home.
MT1	Initialize the MT1-axis.
MT2	Initialize the MT2-axis.

NOTE:

- Mechanical initialization is performed at low speed.
- Do not set *initMec* to a circle start point.
- Auxiliary axes are initialized according to the axis configuration settings. If mechanical initialization is set to [Disable] in the axis configuration settings, the command immediately ends (handled as successfully completed) without executing the initialization.

17.4 Position Error Detection

■ checkPos

This command is only valid for the JR3000/JC-3 Series. The JS3 Series does not have this command.

Position errors can be detected using point jobs.

When the *checkPos* command is executed, the robot goes to the absolute coordinates (x:0, y:0, z:0, r:0), regardless of the current position coordinates. After a position error has been detected, the robot goes to the next point.

Command Category	Command	Parameter	Job
Move	checkPos	–	Detect a position error.

NOTE: The *Move* commands are disabled at CP passing points or where a CP passing point is set as the base point type.

Refer to the system flag (#sysFlag35) for the result of the position error check.
Normal: 0
Position Error: 1

Example:

```
checkPos
if
  ld #sysFlag （35）
then
  waitStartBZ
endIf
```

A position error check is performed and if a position error is detected, the buzzer sounds and the robot waits for a start instruction.

If a position error is detected, the #sysOut8 (Position Error) signal also comes on.

NOTE: With JC-3 absolute encoder models, the checkPos command immediately ends without checking for positioning errors.

17.5 Moving Only the Specified Axis

■ monoMove, endMonoMove, monoMoveStopif Commands

Makes movement for 1 specified axis. You can specify the axis from among the X, Y, Z, R and the auxiliary MT1 and MT2 axes. The distance is specified using the mMoveDistance command.

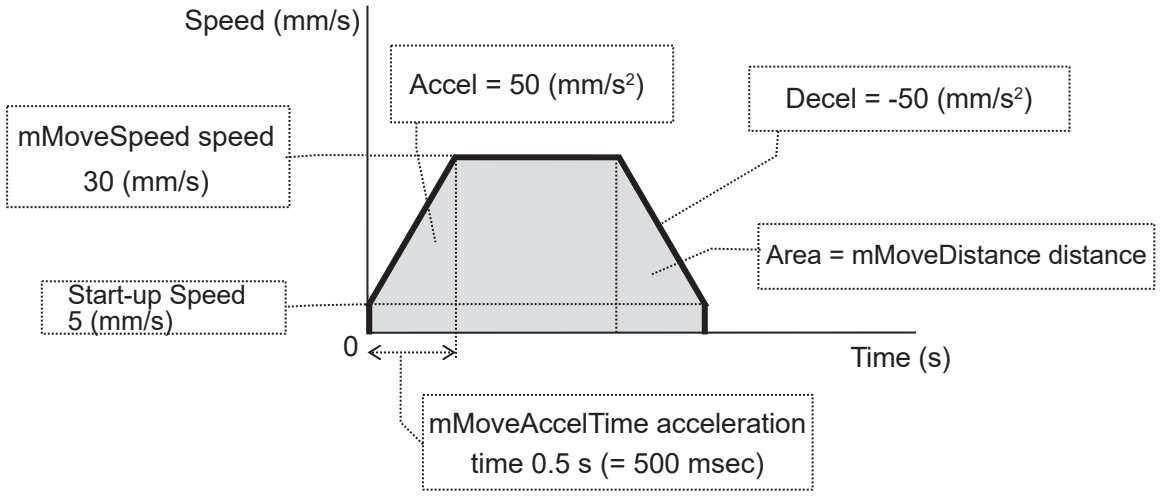
The speed and acceleration are specified using the *mMoveSpeed*, *mMoveAccelRate*, and *mMoveAccelTime* commands. These can also be omitted. For further details regarding the *mMoveSpeed*, *mMoveAccelRate*, and *mMoveAccelTime* commands, refer to the details on the previous page.

Command Category	Command	Parameter	Job
Move	monoMove	Specified axis	Makes movement for 1 specified axis.
	endMonoMove	–	This indicates the end of the movement for the monoMove command.
	monoMoveStopif	–	This ends the movement made by the monoMove command when the conditions are met. You only need to input this command when using conditions to stop the movement.

Example:

monoMove Z	Z axis singular movement
mMoveDistance 50	Distance 50 (mm)
mMoveSpeed 30	Speed 30 (mm/s)
mMoveAccelTime 500	Acceleration Time 500 (msec)
	(acceleration rate = 30 mm/s / 0.5 s = 120 mm/s2)
monoMoveStopIf	Stop Conditions
ld #genIn1	Stop if #genIn1 comes ON.
endMonoMove	

The change that occurs with the speed in this example is shown in the diagram below.



With the mMoveAccelTime command, the acceleration (mm/s^2) is determined according to the following calculation:

Acceleration (mm/s^2) = { specified speed (mm/s) - start-up speed (mm/s) } / acceleration time (s)

Deceleration (mm/s^2) = -1 x acceleration (mm/s2)

There may be discrepancies with the actual speed, acceleration rate, acceleration time, deceleration rate, and deceleration time, as the robot prioritizes and makes adjustments to stop at the specified distance. There is tendency for larger discrepancies with a larger resolution. For the X, Y, Z, and R axes, the [Start-up Speed] is a value specific to each model. For the MT1 and MT2 axes, it is determined by the values set in [Auxiliary Axis Configuration] → [Start-up Speed].

The following system flags can be used to obtain movement result information for movements using the *monoMove* commands:

No.	Identifier	Content	
33	#FlMoveOutRange	True (1)	The movement settings were outside of the operating range. The robot moved until it reached a position at the edge of the operating range.
		False (0)	The robot completed its movement within the operating range or the move area limit.
34	#FlMoveStop	True (1)	The *monoMoveStopIf* conditions were met and the movement was stopped.
		False (0)	The *monoMoveStopIf* conditions were not met.

■ mMoveDistance Command

This specifies the distance for movement using the *monoMove* command.
The distance must be specified for this command, unlike the speed and acceleration commands. If the *mMoveDistance* command is omitted, an expression error occurs and the movement is not made.
The unit parameter varies depending on the axis specified. The unit for the X, Y, and Z axes is “mm”, and “deg” for the R axis. The unit for the auxiliary MT1 and MT2 axes is determined by the settings made in [Auxiliary Axis Configuration] → [Unit Name]. The entry range of the parameter is +9999.999 – -9999.999. The minimum parameter that can be entered is 0.001. When entering this command as an expression, it is possible to enter a value outside of the entry range or a value smaller than the minimum parameter. The robot makes a movement in the positive direction for positive values and it makes a movement in the negative direction for negative values. If a value is entered that exceeds the operating range or the move area limit, the robot moves to a position at the edge of the operating range. If the distance is specified as “0” or the value is smaller than the possible resolution for that axis, the movement is not performed.

Command Category	Command	Parameter	Job
Move	mMoveDistance	Distance	This specifies the distance for movement using the *monoMove* command.

■ mMoveSpeed Command

Specifies the speed for movement using the *monoMove* command.
The *mMoveSpeed* command can be omitted. If this is omitted, the default speed is applied to the movement. For the X, Y, Z, and R axes, the default speed is the maximum speed specific to each model. For the MT1 and MT2 axes, the speed is determined by the value set in [Auxiliary Axis Configuration] → [Limit Speed]. The unit parameter varies depending on the axis specified. The unit for the X, Y, and Z axes is “mm/s”, and “deg/s” for the R axis. The unit for the auxiliary MT1 and MT2 axes is determined by the settings made in [Auxiliary Axis Configuration] → [Unit Name]. The entry range of the parameter is 0 – 9999.99.
The minimum parameter that can be entered is 0.1. When entering this command as an expression, it is possible to enter a value outside of the entry range or a value smaller than the minimum parameter. If a speed is specified that exceeds the default speed, the robot makes a movement using the default speed. The movement is not performed if the speed is specified as “0” or the pulse speed (PPS) is calculated as “0”. The robot also does not perform the movement if a negative value is specified. If you want to move the robot in a negative direction, specify a distance in the negative direction using the *mMoveDistance* command.

Command Category	Command	Parameter	Job
Move	mMoveSpeed	Speed	Specifies the speed for movement using the *monoMove* command.

■ mMoveAccelRate, mMoveAccelTime Commands

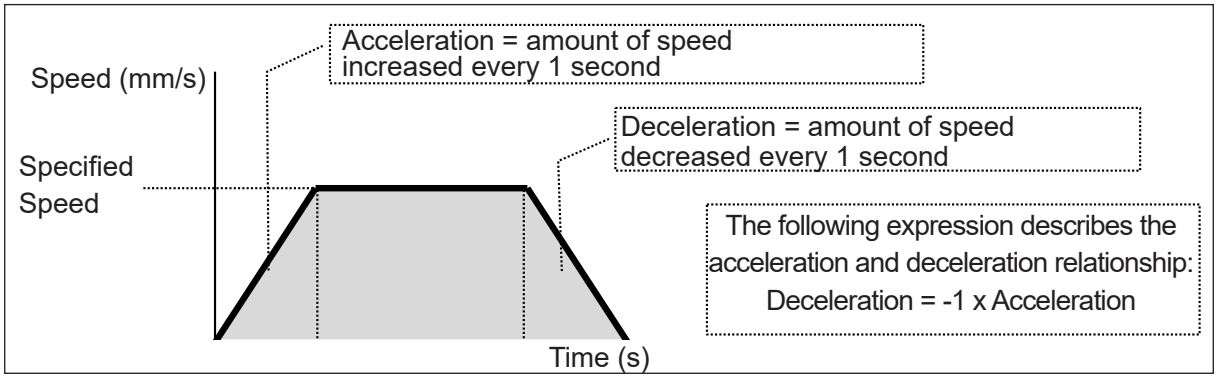
These specify the acceleration for movement using the *monoMove* command.

Specify the acceleration using one of the following methods: use the *mMoveAccelRate* command to specify a rate of acceleration or use the *mMoveAccelTime* command to specify an acceleration time.

If both commands are input, the *mMoveAccelRate* command is prioritized. If both commands are omitted, the movement is applied with the default acceleration. For the X, Y, Z, and R axes, the default acceleration is the maximum acceleration specific to each model. For the MT1 and MT2 axes, the speed is determined by the value set in [Auxiliary Axis Configuration] → [Maximum Acceleration].

Command Category	Command	Parameter	Job
Move	mMoveAccelRate	Acceleration Rate [%]	These specify the acceleration for movement using the *monoMove* command.
	mMoveAccelTime	Acceleration Time [msec]	

With deceleration, speed is decreased at the same rate as it is increased for acceleration. You cannot specify an individual rate of deceleration.

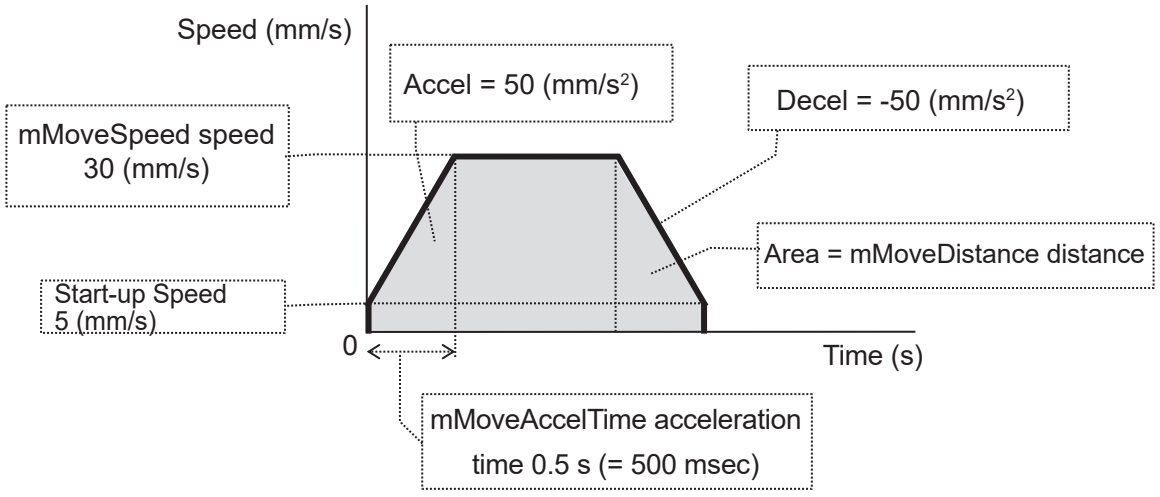


For the *mMoveAccelRate* command (acceleration rate [%]), specify the rate of acceleration as a percentage (%) unit of the default acceleration. The entry range of the parameter is 1 – 100 (%). The minimum parameter that can be entered is 0.1 (%). When entering this command as an expression, it is possible to enter values outside of the operating range or a value smaller than the minimum parameter. If an acceleration rate is specified that exceeds 100 % (the default acceleration), the robot makes a movement using the default 100 % rate of acceleration.
If the acceleration is specified as “0” or it is a negative value, the movement is not performed.

For the *mMoveAccelTime* command (acceleration time [msec]), specify the rate of acceleration as the unit of time (msec) it takes to reach the speed specified with the *mMoveSpeed* command. The entry range of the parameter is 0 – 999,999,999 (msec). The minimum parameter that can be entered is 1 (msec). When entering this command as an expression, it is possible to enter a value outside of the entry range or a value smaller than the minimum parameter. If the acceleration time is specified as a negative value, it is processed as 0.

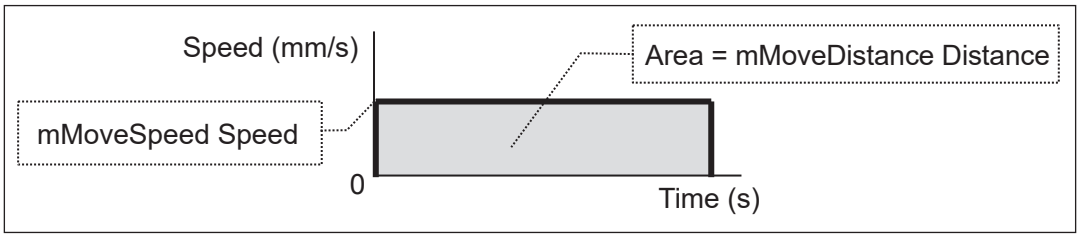
With the mMoveAccelTime command, the acceleration (amount of speed increased every 1 second) is determined from the speed and the specified acceleration time, as shown in the diagram below.

Example: Z axis, mMoveSpeed = 30 (mm/s), mMoveAccelTime = 500 (msec)



With the mMoveAccelTime command, the acceleration (mm/s^2) is determined according to the following calculation:
Acceleration (mm/s^2) = { specified speed (mm/s) - start-up speed (mm/s) } / acceleration time (s)
Deceleration (mm/s^2) = -1 x acceleration (mm/s^2)

An acceleration time of 0 (msec) is unique in that there is no area of acceleration for the movement, as shown in the diagram below.



NOTE: Precautions and Limitations for Exceeding the Default Acceleration (100 % Acceleration)
The maximum acceleration for the X, Y, Z, and R axes is limited to 500 % of the default acceleration. With the JR3000/JC-3 Series, if the specified acceleration exceeds the default acceleration (100 %) or the acceleration time is set to 0 (msec), the speed is limited to the same as that which is used during mechanical initialization. With the auxiliary MT1 and MT2 axes, the acceleration is limited to 500 % of the default acceleration.
If the acceleration time is set as 0 (msec), the robot instantly reaches the specified speed with no acceleration time. Make sure to never set an extreme acceleration that exceeds the specifications or capabilities of the connected device (external motor, etc). When making a movement at high acceleration or making a movement that has no area of acceleration using the *mMoveAccelTime* command, we recommend operations that enable the stepping motor to self-start.

⚠ Caution

With movements using the auxiliary I/O-MT axes, make sure to never set an extreme acceleration that exceeds the specifications or capabilities of the connected device (external motor, etc).
When making a movement at high acceleration or making a movement that has no area of acceleration using the *mMoveAccelTime* command, we recommend operations that enable the stepping motor to self-start.

18. LCD, 7SLED

18.1 Display the Specified Strings on the Teaching Pendant

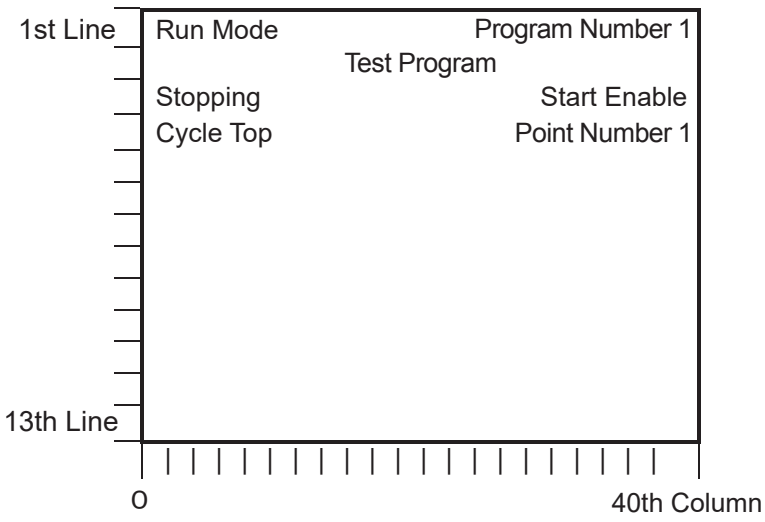
■ clrLCD, clrLineLCD, outLCD, eoutLCD
With these commands you can display and delete characters entered into the Teaching Pendant LCD screen.

Command Category	Command	Parameter	Job
LCD Control	clrLCD	N/A	Clear the entire LCD.
	clrLineLCD	Line	Clear the specified line on the LCD.
	outLCD	Line, Column, Character String	Display the entered strings at the specified position on the LCD.
	eoutLCD	Line, Column, Character String Expression	Display the result of the entered string expression at the specified position on the LCD.

NOTE: The row or column number can also be specified using variables and expressions. The *eoutLCD* command can also provide content for display using string expressions.

outLCD 7,4,"PULSE": Displays the string "PULSE" on the teaching pendant LCD.
eoutLCD 7,4,#sv(24) & #sv(25): Displays the combined value of the character string type variables #sv(24) and #sv(25) on the teaching pendant LCD.

There are 13 lines and 40 columns on the teaching pendant LCD.
Up to 40 one-byte characters (20 double-byte characters) can be displayed in a line.



18.2 Display a Given Number on the 7 Segment LED

■ sys7SLED, out7SLED

Using the *out7SLED* command, you can display a given number on the 7-segment LED (program number display) on the front of the robot (JR3000 Series) or on the switchbox/ operation box.

If you execute the *sys7SLED* command or change program numbers, the LED will revert back to displaying the program number.

Command Category	Command	Parameter	Job
LCD Control, 7Seg LED	sys7SLED	-	Revert to the display before the display change from the *out7SLED* command
	out7SLED	Display Type Display Value	Output the display parameters to the 7-segment LEDs.

You can select from the following four display types for the *out7SLED* command parameters. You can also specify and display the numerical values using variables or expressions.

Output Type	Description
Num	Displays the specified number (0 – 999) on the 7-segment LED. Example: One digit numbers (“001”) are displayed as “1”. Two digit numbers (“011”) are displayed as “11”.
Er	Displays alternates between the code “Er” and the specified number on the 7-segment LED.
St	Displays alternates between the code “St” and the specified number on the 7-segment LED.
Ur	Displays alternates between the code “Ur” and the specified number on the 7-segment LED.

Example:

out7SLED Num,10: Displays the numerical value 10.

out7SLED Er,20: Displays alternates between the code *Er* and the numerical value 20.

out7SLED St,nMyNum: Displays alternates between the code *St* and the value of the variable *nMyNum*

19. COM/ETHERNET INPUT/OUTPUT

■ outCOM, eoutCOM, setWTCOM, inCOM, cmpCOM, ecmpCOM, clrCOM, shiftCOM

You can input and output the data from COM. You can also input/output (send/receive) using Ethernet client port. For details, refer to "21.1 Ethernet Client Functions."

Command Category	Command	Parameter	Job
COM Input/Output	outCOM	Port, Character String	Outputs the characters from COM and Ethernet.
	eoutCOM	Port, Character String Expression	Outputs the string expression result from COM and Ethernet.
	setWTCOM	Port, Wait Time	Sets the COM and Ethernet transfer wait time (timeout).
	inCOM	Variable Name, Port, Character Length	Assigns the data received through COM and Ethernet to a specified variable.
	cmpCOM	Port, Character String	Compare the received data through COM and Ethernet with the character string. The result is entered into the system flag (sysFlag1 to 15).
	ecmpCOM	Port, Character String Expression	Compare the received data through COM and Ethernet with the string expression. The result is entered into the system flag (sysFlag1 to 15).
	clrCOM	Port	Clear the COM and Ethernet receive buffer.
	shiftCOM	Port, Shift Number	Shift the data received through COM and Ethernet. Delete the shift number bytes from the top.

■ COM Output: outCOM, eoutCOM

Character strings with up to 255 characters in length can be output from COM and Ethernet. Select *outCOM* or *eoutCOM*, select the desired COM port number to display the character entry screen. Enter the character string you want to output and press the ENTR key (for the key operations on character entry screens, refer to "3.4 Entering Characters and Expressions" in the operation manual *Teaching Pendant Operation*). Enclose the character strings to output in double quotes ("") (See example 1). You can output variables and formulas with *eoutCOM*, but do not use double quotes ("") for these (See example 2).

Example 1: eoutCOM port2,"ERROR": Outputs the character string "ERROR".

Example 2: eoutCOM port2,#sv(24) & #sv(25): Outputs a combined value of the character string type variables #sv(24) and #sv(25).

Also, with the *eoutCOM* command, characters can be specified in hexadecimal code using the % symbol (see example 3). However, if any character other than 0 to 9, A to F, or % comes after the % symbol, the % symbol is output as a character (see example 4). If you want to output the % symbol as a character when 0 to 9, A to F come after it, enter two % symbols (%%). (See example 5)

Example 3: eoutCOM port2,"%0D%0A": Output CR/LF codes (new line codes).
Example 4: eoutCOM port2,"%G01": Output a character string %G01.
Example 5: eoutCOM port2,"%%300": Output a character string %300.

■ COM Transfer Wait Time: setWTCOM

This sets the transfer wait time for when the robot waits for transfer from *inCOM* and *cmpCOM*. If the transfer is not complete even after the robot waits for the exact time set here, a timeout occurs (the corresponding system flag goes ON). The transfer wait time can be set in 1 msec unit.
The default setting value for *setWTCOM* is 100 msec. If you modify this value, the value is held by the robot until the robot power is turned OFF. After the robot is power cycled, the value returns to the default 100 msec setting.

■ COM Input: inCOM

With this command, data received from COM and Ethernet is assigned to a variable by the exact number of specified characters. If the received data is larger than the specified number of characters, characters counted from the beginning of the string up to the specified number are assigned.
If the received data is less than the specified number of characters, the robot stands by for the exact time specified by the *setWTCOM* command, and then assigns the received data to a variable. If the *setWTCOM* command is not used, the robot stands by for 100 msec.

NOTE: If point job data that includes a COM input command is set to a [CP Passing Point], the robot operates as if the standby time is set for 0 (zero) seconds.

■ COM and Ethernet Received Data Comparison: cmpCOM, ecmpCOM

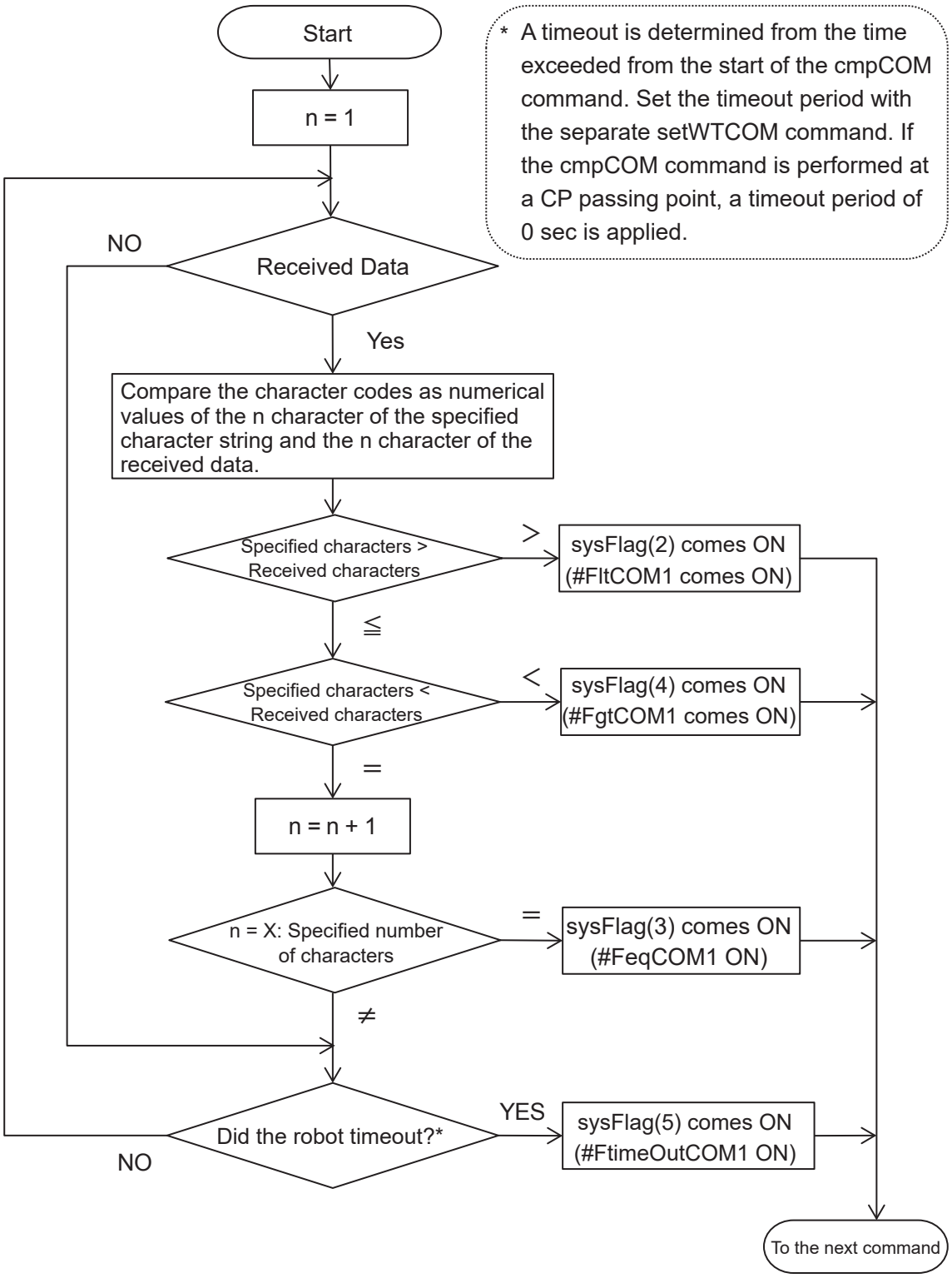
These commands compare the COM and Ethernet receive buffer (a place where received data is stored) and the specified character string. The comparison results are applied to a system flag. Starting from the first character in the string, comparisons are made one character at a time, and if the comparisons do not match, the comparison of the specified number of characters is complete, or if the robot waits for the exact receive standby time but nothing is received, the corresponding system flag comes ON (refer to the table below).
You can set a receive standby time using the *setWTCOM* command. If the *setWTCOM* command is not used, the robot stands by for 100 msec.
With the *ecmpCOM* command, the character string you want to compare with receive buffer can be specified with a character string expression.

Corresponding System Flags

	COM1	COM2	COM3 (JR3000/JC-3 Series Only)
Data Received	sysFlag(1)	sysFlag(6)	sysFlag(11)
Specified Character > Receive Buffer	sysFlag(2)	sysFlag(7)	sysFlag(12)
Specified Character = Receive Buffer	sysFlag(3)	sysFlag(8)	sysFlag(13)
Specified Character < Receive Buffer	sysFlag(4)	sysFlag(9)	sysFlag(14)
Timeout	sysFlag(5)	sysFlag(10)	sysFlag(15)

	Client Port 1	Client Port 2	Client Port 3
Data Received	sysFlag (300)	sysFlag (306)	sysFlag (312)
Specified Character > Receive Buffer	sysFlag (301)	sysFlag (307)	sysFlag (313)
Specified Character = Receive Buffer	sysFlag (302)	sysFlag (308)	sysFlag (314)
Specified Character < Receive Buffer	sysFlag (303)	sysFlag (309)	sysFlag (315)
Timeout	sysFlag (304)	sysFlag (310)	sysFlag (316)

cmpCOM Command Example with COM1



■ COM and Ethernet Receive Buffer Clear: clrCOM

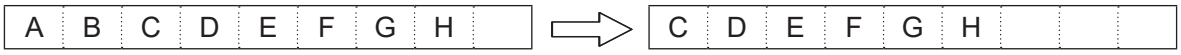
The receive buffer is a place where received data is stored. Each COM port has an 8-kbyte receive buffer. Newly received data does not overwrite existing data, but is written to the buffer after the existing data. A receive buffer is cleared by turning OFF the power or executing the *clrCOM* command.

■ Moving COM and Ethernet Receive Data: shiftCOM

The specified data in the receive buffer is moved.

Example: 2-byte Shift

Receive Buffer



You can tell whether data is stored in each receive buffer by looking at its system flag. If a receive buffer has received data, the corresponding system flag is ON.

■ Precautions for when the COM1 Command Communication Function is Enabled

When the COM1 command communication function is enabled, you cannot use commands such as *inCOM* and *outCOM* to process arbitrary communication via COM1. However, if you perform the following, you can temporarily use the commands such as *inCOM* and *outCOM*:

(1) Disable the command communication function using the command *stopPC* before using the *inCOM* or *outCOM* commands, etc. You can then use *inCOM* and *outCOM* commands, etc., to communicate with no interference from the command communication function.

(2) After using the *inCOM* and *outCOM* commands, etc., use the *startPC* command to re-enable the command communication function.

20. VARIABLE, COMMENT, SYSTEM CONTROL

20.1 Variable Declaration and Assignment

■ declare, let

Point job data which includes *declare* and *let* commands or variables only valid in user-defined functions are referred to as local variables.

When you declare a local variable set the variable type and identifier. The identifier is used as a variable name and the variable type can be selected from either a numeric type or a string type. A local variable can also be declared in an array of up to three-dimensions.

The *let* command assigns the right-hand operand (numeric value, variable value, or evaluation of string expression) to the left-hand operand.

Command Category	Command	Parameter	Job
Variable, Comment, System Control	declare	Variable Type, Identifier	Local variable declaration
	let	Expression	Assign the right-hand operand to the left-hand operand.

Example: *declare* command

```
declare numeric abc
declare string def
```

Numeric variable *abc* declaration
String variable *def* declaration

Example: *let* command

```
let count = 0
let count = count + 1
let count = in - out

let total = nin * 365

let tsuki = total / 12

let fullname=name1 & name2
```

Assign 0 to the variable *count*.
Add 1 to the variable *count*.
Assign the difference of the value of *out* subtracted from the value of *in* to the variable *count*.
Assign the product of 365 multiplied by the value of *people* to the variable *total*.
Assign the quotient of the value of *total* divided by 12 to the variable *month*.
Assign the string composed of *name1* and *name2* to the variable *fullname*.

Both of the following point job data use the local variable *count*, however, the two variables do not interfere with each other since local variables are enabled only within point job data containing a declare command. For example, if 0 is assigned to the variable *count* in point job data 24, the value of the variable *count* in point job data 05 will not change, and vice versa.

Example: Point Job Data 05

```
declare numeric count
let count=0
do
   let count=count+1
   callJob 24
   if
     ld count>=10
   then
     exitDo
   endIf
loop
```

Declare the local variable *count* (numeric type)
Set the initial value 0 to the variable *count*.
Repeat the commands between *do* and *loop*.
  Add 1 to the variable *count*.
  Execute point job data 24.
  If
  the value of the variable *count* is larger than 10,

jump to the next command of the *loop* command.

Return to the *do* command.

Example: Point Job Data 24

```
declare numeric count
let count=0
 Label 1
pulse #genOut1,250
let count=count+1
if
   ld count<=3
then
   jump L1
endIf
```

Declare the local variable *count* (numeric type)
Set the initial value 0 to the variable *count*.
Label 1 (jump destination mark)
Output ON pulse to #genOut1.
Add 1 to the variable *count*.
If
  the value of the variable *count* is less than 3,

  jump to Label 1.

NOTE: Contrary to local variables, variables that can be referenced from any program and any point, are called “global variables”. Global variables are all variables other than local variables declared by the *declare* command.

20.2 Comment Insertion

■ rem, crem

You can add comments to point job data and PLC program commands.

Command Category	Command	Parameters	Job
Variable, Comment, System Control	rem	Character String	1 line comment
	crem		End line comment (only displayed when decompiling)

NOTE: The display on the teaching pendant is the same for *rem* and *crem*.

Example1: *rem* (teaching pendant display)

```
if
    ld #genIn1
    rem #genIn1: obstacle sensor
then
    waitStartBZ
```

If
#genIn1 is true,
(#genIn1: obstacle sensor): Comment Line

Stand by and sound the buzzer until there is a start instruction.

If using *crem*, during the command entry, the details are on separate lines the same as with example 1, however, if you decompile with JR C-Points II, the commands are displayed as they are in example 2.

Example2: *crem* (when decompiling)

```
if
    ld #genIn1 // #genIn1: obstacle sensor
then
    waitStartBZ
```

If
#genIn1 is true, (#genIn1: obstacle sensor):
Comment Line

Stand by and sound the buzzer until there is a start instruction.

20.3 Change a Program Number by Point Job

■ setProgNum

You can change the number for currently selected program using a point job. You can use this command in situations such as the ones below:

- If you set this command to a point job performed when the power is turned ON, the same program number is always activated every time the power is turned ON in Run Mode.
- If you set this command to a point job performed after returning to the work home position, you can change over to the next program number to be performed. This is convenient when you want to carry out a series of programs sequentially.
  For example, you want to repeat Program 1 → Program 2 → Program 3 as a sequential operation process. If you execute the *setProgNum2* command at the point job performed after returning to the work home position of Program 1, the program number changes to Program 2 after running Program 1. Accordingly, you can make the same settings to switch from program 2 to 3 and from program 3 to 1.

- If you set the *setProgNum* command to the point job performed when the robot is standing by, you can change the program number according to the input from COM. You can also connect a barcode reader to COM and change the program number according to the barcode values.

If you change the program number while running a program, the running program does not change immediately. After the number changes, the robot stands by for a start signal (run restart) and then the program is changed.
Use the *callProg* command if you want to execute another program while running a program.

Command Category	Command	Parameter	Job
Variable, Comment, System Control	setProgNum	Program Number	The program number is changed when the robot restarts running after standing by.

NOTE: You can also specify program numbers with variables or expressions.

20.4 Change the PLC Program by Point Job

■ setSeqNum

With this command you can change the currently selected PLC program number by point job. A complicated PLC program cannot be created because the maximum number of commands for a PLC program is 1,000 steps. However, you can create individual PLC programs performed when the power is turned ON, when the robot is standing by, or during a program run and use the *setSeqNum* command to change the PLC number.
For example, if you set *setSeqNum02* to [Common Job on Start of Cycle] (Run Mode Job) and *setSeqNum01* to [Job on End of Cycle] (Run Mode Job), PLC program number 2 is executed during runs and PLC program number 1 is executed during standbys.

If you change the PLC program number while running a program, the PLC program does not change immediately. After the number changes, the robot stands by for a start signal (run restart) and then the PLC is changed.

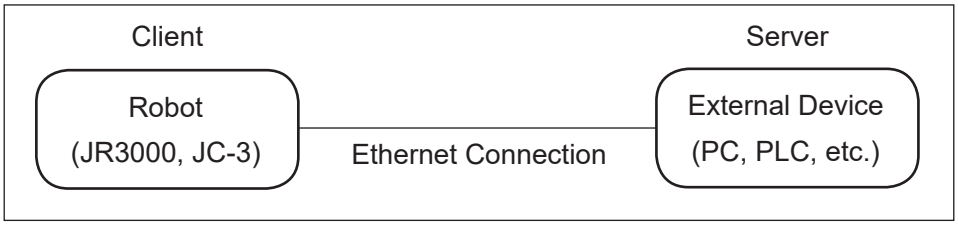
Command Category	Command	Parameter	Job
Variable, Comment, System Control	setSeqNum	PLC Program Number	The PLC program is changed when the robot restarts running after standing by.

NOTE: PLC program numbers can be specified using variables or expressions.

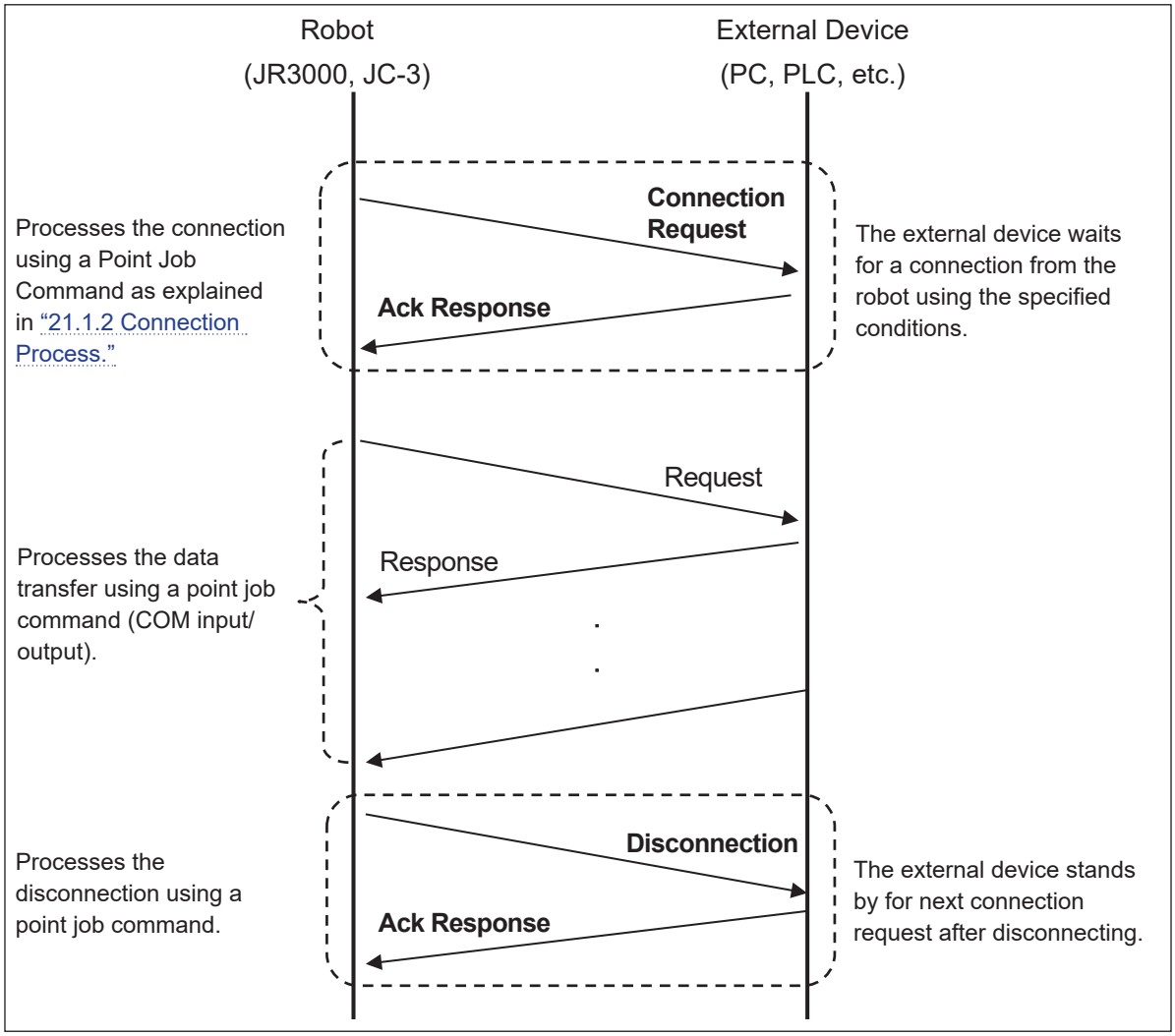
21. APPENDIX

21.1 Ethernet Client Functions

This is a client-side function in the client-server system. The robot is the client, and it can communicate with an external device that is the server. Data send/receive are performed using the inCOM and outCOM point job commands, which are the same command system as for COM communications.



The communication sequence is as shown below.



The robot (JR3000 or JC-3) initiates communication by sending a connection request to the external device. External devices must be in a connection standby state in order to receive the connection request.
The external device cannot receive any data and no data will be transferred if the initial connection is not established, even if the robot attempts to transfer data.
This function provides 3 communication ports for connecting with up to 3 external devices. Refer to “21.1.1 Ethernet Client Port Settings” for further details.

21.1.1 Ethernet Client Port Settings

You need to set the IP address and port number of connection destination (server) in Ethernet client functions.
You can make these settings using either the teaching pendant or the PC software JR C-Points II (limited edition).

Teaching Pendant:

T P MODE [Administration] (JR3000/JC-3 Series)
UTILITY [Change Mode] [Administration] (JS3 Series)
[Administration Settings Mode]
[Ethernet Settings]
[Ethernet Client 1]
[Ethernet Client 2]
[Ethernet Client 3]

Exit Administration Mode after making the settings to automatically power cycle the robot. The network address settings are enabled after the robot has power cycled.

PC Software:

P C [Robot] → [Administration] → [Administration Settings] → [COM/Ethernet Settings]

Click the [OK] button to transfer the settings to the robot. Power cycle the robot to enable any new settings made.

The client port is disabled if the IP address is set to all zero or the port number is set to zero with both the teaching pendant and the PC software. By default, the client ports are disabled, i.e., the IP address is set to “0.0.0.0” and the port number is set to “0”.
If an invalid address (127.0.0.0, etc.) is entered, it is processed as an error and cannot be set.

21.1.2 Connection Process

The robot uses the *connect* command to connect with the external device and the *disconnect* command to disconnect from the device.

Command	Required Parameters	Job
connect	Port number, setting value	Connects to the external device.
disconnect	Port number, setting value	Disconnects from the external device.

■ Connect

Connects to the external device according to the client port specified by the first parameter. Use the second parameter to specify a timeout value (msec) to allow a slow connection process in accordance with the external device or network conditions. A longer timeout value (100 msec or so) is recommended for large scale networks which require a longer connection process.

■ Disconnect

Disconnects from the external device according to the client port specified by the first parameter. Use the second parameter to specify a timeout value (msec) to allow a slow disconnection process in accordance with the external device or network conditions. A longer timeout value (100 msec or so) is recommended for large scale networks which require a longer disconnection process.

The connection and disconnection status can be verified using the corresponding system flag. The system flag turns ON when the connection is established.

Port	Status Confirmation System Flags
Client Port 1	sysFlag (305)
Client Port 2	sysFlag (311)
Client Port 3	sysFlag (317)

When using COM input/output commands (listed in “19. COM/ETHERNET INPUT/OUTPUT”), select either [ether1], [ether2], or [ether3] to specify client ports 1,2, and 3, respectively.

Additionally, unique system flags are provided for saving comparative results using the *cmpCOM* and *ecmpCOM* commands for each client port as shown.

	Client Port 1	Client Port 2	Client Port 3
Received Data Valid	sysFlag(300)	sysFlag(306)	sysFlag(312)
Characters>Receiving Buffer	sysFlag(301)	sysFlag(307)	sysFlag(313)
Characters=Receiving Buffer	sysFlag(302)	sysFlag(308)	sysFlag(314)
Characters<Receiving Buffer	sysFlag(303)	sysFlag(309)	sysFlag(315)
Timeout	sysFlag(304)	sysFlag(310)	sysFlag(316)

■ Example: Connection via Point Job Commands

This is an example of connecting to client port 1 and repeating the connection process until successful.

With this example, connection is attempted every one second without stopping until connection is established.

```
do
  connect ether1, 100        Connects with client port 1.
  if
    ld #sysFlag （305）      Verify connection status of client port 1.
  then
    exitDo                   Exits do-loop if connection is established.
  endIf
  delay 900                  Waits for 900 msec if connection is not established.
loop
```

■ Example: Disconnection via Point Job Commands

This is an example of disconnecting from an external device to which the robot is already connected.

```
if
    ld #sysFlag （305）          When connected to client port 1.
then
    disconnect ether1, 100       Disconnects from client port 1.
endIf
```

NOTE: If frequent connection and disconnection is required, register the above commands as user functions and use them at the point jobs for your convenience. Refer to the operation manual *Function IV* for further details.

21.1.3 Practical Example

This is an example case of notifying a log server on the network with robot movement information. Log data transmissions are made when the program operation completed (when the cycle finished).

■ Log server

The subject log server shall be accessible from the network under the following conditions.

Protocol	TCP/IP, non-procedural communication
Log Server IP Address	192.168.200.20
Log Server Port Number	1234

■ Client port settings (robot-side settings)

In this example, we use the robot's [Client Port 1].

Set [Client Port 1] as shown in the table below, in order to connect with the log server under the above conditions.

Connection Destination IP Address	192.168.200.20
Connection Destination Port Number	1234

TP MODE [Administration] (JR3000/JC-3 Series)
UTILITY [Change Mode] [Administration] (JS3 Series)
[Administration Settings Mode]
[Ethernet Settings]
[Ethernet Client 1]

PC [Robot] → [Administration] → [Administration Settings] → [COM/Ethernet Settings]

Create the following point job and set it to [Job on End of Cycle] (refer to the operation manual *Teaching Pendant Operation* for details on how to make the settings).

```
declare string logstr
declare num pno
let pno = currentMainProgNumber()
let logstr = “Program Finish!! [ProgNo=”
let logstr = logstr & str(pno) & “]”

connect ether1, 100
if
  ld #sysFlag(305)
then
  eoutCOM ether1, logstr
  eoutCOM ether1, “%0D%0A”
  delay 200
  disconnect ether1, 100
endIf
```

Acquires the current program number.
Enters the program character string.
Links to the program number.

Attempts to connect with the log server.

If connection is established,

Transmits the log character string.
Transfers a line feed character (CR+LF).
Waits until transfer complete.
Disconnect from the log server.

After the program run is complete, the following log data is transferred to the external device:

```
Program Finish [ProgNo=10]
Program Finish [ProgNo=10]
Program Finish [ProgNo=10]
Program Finish [ProgNo=10]
```

■ Limitations

- This robot is compatible with Ethernet communication using Internet Protocol Version 4 (IPv4). Internet Protocol Version 6 (IPv6) is not supported. Likewise, the server (an external device such as a PC, PLC, etc.,) connected to the client (JR3000/JC-3/JS3 Series robot) should be compatible with IPv4.

- The port number specified using this function refers to the external device port number (PC, PLC, etc).
  In general, port numbers are defined into the following 3 categories:

Port Number (range)	Description
1 to 1023	Well-known Port
1024 to 49151	Registered Port
49152 to 65535	Dynamic and Private Port

Normally, well-known ports and registered ports are used by the server (an external device such as a PC, PLC etc.), and dynamic and private ports are used by the client (the robot). Using well-known ports is not recommended as server devices likely use them to provide vital functions for the network. It is recommended to use registered ports for this function.

- This function is only compatible with non-procedural communication (protocols that do not have a specific procedure, such as TELNET or HTTP, etc). If a protocol is required, individually specify it for each corresponding point job command.

Janome Sewing Machine Co., Ltd.

Industrial Equipment Sales Division

1463 Hazama-machi, Hachioji-shi, Tokyo, Japan, 193-0941

Tel: +81-42-661-6301

Fax: +81-42-661-6302

E-mail: j-industry@gm.janome.co.jp

170813107 / 202101-E1 (202011-J1)