JANOME SCARA/GANTRY ROBOT
JS/JSR4400N/JSG Series
JANOME DESKTOP ROBOT
JR2000N Series

Operation Manual
Features II
(Variables/Commands/Functions)

Thank you for purchasing a Janome Robot.

- **Read this manual thoroughly in order to ensure proper use of this robot. Be sure to read "For Your Safety" before you use the robot. The information will help you protect yourself and others from possible dangers during operation.**
- **After having read this manual, keep it in a handy place so that you or the operator can refer to it whenever necessary.**
- **This manual is written according to IEC 62079.**

JANOME

FOR YOUR SAFETY

Safety Precautions

The precautions in this manual are provided for the customer to make the best use of this product safely, and to provide preventive measures against injury to the customer or damage to property.

Please follow these instructions

Various symbols are used in this manual. Please read the following explanations of each symbol.

- **Symbols Indicating the Degree of Potential Damage or Danger**

  The following symbols indicate the degree of damage or danger which may be incurred if the safety notes are ignored.

Warning	The Warning symbol indicates the possibility of death or serious injury.
Caution	The Caution symbol indicates the possibility of accidental injury or damage to property.

- **Symbols Indicating the Type of Danger and Preventive Measures**

  The following symbols indicate the type of safety measures that should be taken.

	Indicates the safety measures that should be taken.
	Be careful. (General caution)
	Indicates a prohibition.
	Never do this. (General prohibition)
	Do not disassemble, modify or repair.
	Do not touch.
	Indicates a necessity.
	Be sure to follow instructions.
	Be sure to unplug power cord from wall outlet.
	Be sure to check grounding.

FOR YOUR SAFETY

JR2000N Series

⚠ Warning

Do not use the unit where flammable or corrosive gas is present.
Leaked gas accumulated around the unit can cause fire or explosion.

Use the robot in a condensation-free environment between 0 and 40 degrees centigrade with a humidity level of 20 to 95 percent.
Failure to do so may result in malfunction.
IP Protection Rating: IP30 (IP40 for CE specification models)

Use the robot in an electric noise-free environment.
Failure to do so may result in malfunction or breakdown.

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

Install the robot in a place which can endure its weight and conditions while running.
Placing the unit in an insufficient or unstable surface may cause the unit to fall or overturn, thereby causing damage or injury.
Leave a space of at least 30cm between the back of the robot (the side with a cooling fan) and the wall. Insufficient space can lead to overheating or fire.

Be sure to use within the voltage range indicated on the unit.
Failure to do so may cause electric shock or fire.

If your robot has the I/O-S circuit, be sure to install a safeguard such as an area sensor when installing the robot.
Without a safeguard, going near the robot while it is operating may cause injury.

Plug the power cord into the wall outlet firmly.
Failure to do so can cause the plug to heat up and may result in fire.

FOR YOUR SAFETY

JR2000N Series

⚠ Warning

Wipe the power plug with a clean, dry cloth periodically to eliminate dust.
Dust accumulation can undermine the electrical insulation and cause fire.

Ensure that the robot is properly grounded before use. Do not use without grounding.
Insufficient grounding can cause electric shock and/or fire.

Be sure to use within the voltage range indicated on the unit.
Failure to do so may cause fire or unit malfunction.

Keep water and oil away from the robot and power cable.
Contact with water or oil can cause electric shock, fire, or unit malfunction.
IP Protection Rating: IP30 (IP40 for CE specification models)

If tools have been attached (such as the electric screwdriver), make sure they are properly connected.
Failure to do so may result in injury or breakdown.

During prolonged use, be sure to regularly check that the mounting screws are sufficiently tightened.
Loose screws may cause injury or breakdown.

Be sure to check the wiring to the main unit.
Improper wiring may result in malfunction or breakdown.

Be sure to secure the movable parts of the robot before transportation.
Failure to do so may result in injury or breakdown.

FOR YOUR SAFETY

JR2000N Series

⚠ Warning

Do not bump or jar the unit while it is being transported or installed.
Doing so can cause breakdown.

Be sure to check that no danger is around the operator before starting the robot.
Careful attention will protect the operator from injury.

Do not attempt to disassemble or modify the unit.
Disassembly or modification may cause electric shock or malfunction.

Be sure to unplug the power cord from the wall outlet when examining or lubricating the machine.
Failure to do so may cause electric shock or injury.

If anything unusual (e.g. a burning smell or an abnormal sound) occurs, stop operation and unplug the cable immediately. Contact the dealer from whom you purchased the robot or the office listed on the last page of this manual.
Continuous use without repair can cause electric shock, fire, or unit breakdown.

Do not leave the unit plugged in (power cord and connectors) when it is not in use for long periods of time. Accumulated dust can cause fire.
Be sure to shut off the power supply before removing the power cord.

Be sure to turn off the robot before inserting and removing all cables.
Failure to do so may result in electric shock, fire, data loss or unit malfunction.

Keep the emergency stop switch within reach of an operator while teaching or running the robot.
Failure to do so is dangerous since it may not be possible to stop the robot immediately and safely.

FOR YOUR SAFETY

JR2000N Series

⚠ Warning

Regularly check that the emergency stop switch works properly.
For models with I/O-S circuits, also check that they work properly.
Failure to do so is dangerous since it may not be possible to stop the robot immediately and safely.

FOR YOUR SAFETY

JSR4400N Series

⚠ Warning

Do not leave the unit plugged in (power cord and connectors) when it is not in use for long periods of time. Accumulated dust can cause fire.
Be sure to shut off the power supply before removing the power cord.

Keep the emergency stop switch within reach of an operator while teaching or running the robot.
Failure to do so is dangerous since it may not be possible to stop the robot immediately and safely.

Regularly check that the emergency stop switch works properly.
For models with I/O-S circuits, also check that they work properly.
Failure to do so is dangerous since it may not be possible to stop the robot immediately and safely.

During prolonged use, check the mounting screws etc, regularly and ensure that they are firmly tightened.
Loose screws may cause injury or breakdown.

Be sure to use within the voltage range indicated on the unit.
Failure to do so may cause fire or unit malfunction.

Keep water and oil away from the robot and power cable.
Contact with water or oil can cause electric shock, fire, or unit malfunction.
IP Protection Rating: IP20

Wipe the power plug with a clean, dry cloth periodically to eliminate dust.
Dust accumulation can undermine the electrical insulation and cause fire.

FOR YOUR SAFETY

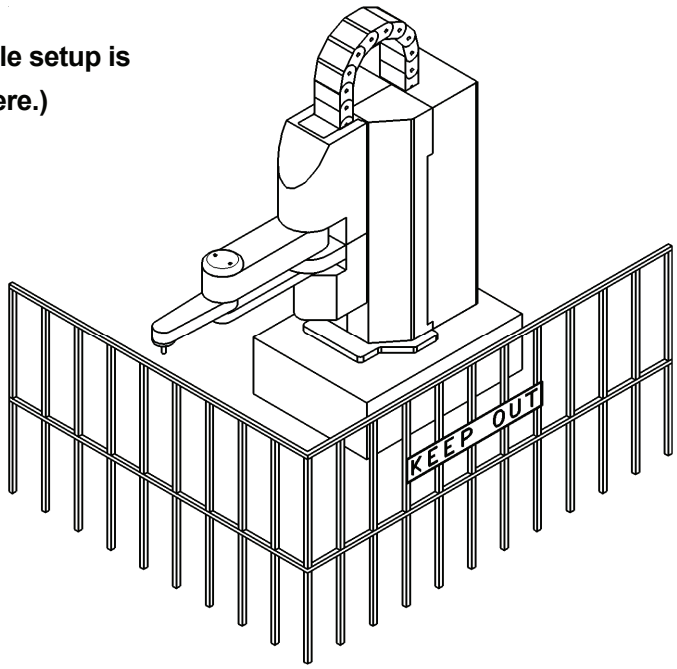
JSR4400N Series

INSTALLATION

⚠ Warning

Always set up a safety barrier.
Anyone entering the operating range of the robot may be injured.
Set up an emergency stop interlock system that is triggered when the entrance to the safety barrier is opened.
Use the I/O-S connector included in the package. Ensure there is no other way of entering the restricted area.
Furthermore, **put up a "No Entry" or "No Operating" warning sign** in a clearly visible place.

(An example setup is pictured here.)



Make sure the safety barrier is strong enough to protect the operator from flying debris caused during robot operation.
Always wear protective gear (helmet, protective gloves, protective glasses, and protective footwear) when going inside the safety barrier.

If the object the robot is holding is in danger of flying off or falling, make sure **safety precautions adequate for the object's size, weight, temperature and chemical composition have been taken.**

FOR YOUR SAFETY

JSR4400N Series

INSTALLATION

⚠ Warning

Be sure to check grounding before using the unit.
Improper grounding can cause electric shock, fire, unit malfunction, or breakdown.

Install the robot in a place which can endure its weight and conditions while running.
Placing the unit in an insufficient or unstable surface may cause the unit to fall or overturn, and may cause injury or damage.

There is an air intake vent on the lower back of the robot (18mm above the ground). Do not block this vent, or overheating or fire may result.

Do not use the unit in the presence of flammable or corrosive gas.
Leaked gas accumulated around the unit can cause fire or explosion.

For the health and safety of the operator, place the unit in a well-ventilated area.

Use the robot in a condensation-free environment between 0 and 40 degrees centigrade with a humidity level of 20 to 90 percent.
Failure to do so may result in malfunction.
IP Protection Rating: IP20

Use the robot in an electric noise-free environment.
Failure to do so may result in malfunction or breakdown.

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

Be sure to use within the voltage range indicated on the unit.
Failure to do so may cause electric shock or fire.

FOR YOUR SAFETY

JSR4400N Series

INSTALLATION

⚠ Warning

Do not attempt to disassemble or modify the unit.
Disassembly or modification may cause electric shock or malfunction.

Be sure to secure the movable parts of the robot before transportation.
Failure to do so may result in injury or breakdown.

Do not bump or jar the unit while it is being transported or installed.
Doing so can cause breakdown.

Be sure to confirm that all the air tubes are connected correctly and firmly.

If tools have been attached (such as the electric screwdriver), make sure they are properly connected.
Failure to do so may result in injury or breakdown.

Be sure to check the wiring to the main unit.
Improper wiring may result in malfunction or breakdown.

Keep the emergency stop switch within reach of an operator while teaching or running the robot.
Failure to do so is dangerous since it may not be possible to stop the robot immediately and safely.

Turn off the unit before inserting and removing cables.
Failure to do so may result in electric shock, fire, or unit malfunction.

Be sure to turn off the robot before plugging it in.

FOR YOUR SAFETY

JSR4400N Series

INSTALLATION

⚠Warning	
	**Plug the power cord firmly into the wall outlet.** Incomplete insertion into the wall outlet causes the plug to heat up and may result in fire. Be sure to turn off the robot y before connecting the power cable.
	Place the operation box on a flat surface more than 80 cm above the floor so that it is easy to operate.
	**Use the unit in an environment that is not dusty or damp.** Dust or dampness may lead to breakdown or malfunction. IP Protection Rating: IP20

FOR YOUR SAFETY

JSR4400N Series

DURING OPERATION

⚠ Warning

When lubricating or inspecting the unit, unplug the power cord from the robot.
Failure to do so may result in electric shock or injury.
Be sure to shut off the power supply before removing the power cord from the robot.

Put up a **"Do Not Operate"** sign on the start switch when people are entering the safety barrier.

Keep the emergency stop switch within reach of an operator while teaching or running the robot.
Failure to do so is dangerous since it may not be possible to stop the robot immediately and safely.

Make sure the safety barrier is strong enough to protect the operator from flying debris caused during robot operation.
Always wear protective gear (helmet, protective gloves, protective glasses, and protective footwear) when going inside the safety barrier.

Make sure all the air tubes are connected correctly and firmly.

Always be aware of the robot's movement, even in the Teaching mode.
Careful attention will protect the operator from injury.

FOR YOUR SAFETY

JSR4400N Series
DURING OPERATION

⚠ Warning

When entering the safety barrier, **do not enter the operating range of the robot.**

If you must go inside the safety barrier, make sure you **push the emergency stop switch** and **put a "Do Not Operate" sign** on the start switch.

When activating the robot, check that **no one is inside the safety barrier and that no object will interfere with the robot's operation.**

Under no circumstances should you go inside the safety barrier or place your hands or head inside the safety barrier while the robot is operating.

If anything unusual (e.g. a burning smell or an abnormal sound) occurs, stop operation and turn off the robot. Contact the dealer from whom you purchased the robot or the office listed on the last page of this manual.
Continuous use without repair can cause electric shock, fire, or breakdown of the unit.

Keep the emergency stop switch within reach of an operator while teaching or running the robot.
Failure to do so is dangerous since it may not be possible to stop the robot immediately and safely.

FOR YOUR SAFETY

JS Series and JSG Series

⚠ Warning

When not in use for extended periods of time, unplug the power cord from the power source. Accumulated dust leads to fires.
When unplugging the cord, make sure the power is off.

Change the robot battery periodically (approximately every three years) to prevent unit malfunction or breakdown.

Keep the emergency stop switch within reach of an operator while teaching or running the robot.
Failure to do so is dangerous since the robot may not be able to be stopped immediately and safely.

Regularly check that the I/O-S circuits and emergency stop switch work properly.
Failure to do so is dangerous since the robot may not be able to be stopped immediately and safely.

FOR YOUR SAFETY

JS Series and JSG Series

⚠ Warning

During prolonged use, check the mounting screws etc, regularly and ensure that they are firmly tightened.
Loose screws may cause injury or breakdown.

Power the unit only with the rated voltage.
Excessive voltage can cause fire or unit malfunction.

Keep water or oil away from the robot, control box and power cable.
Contact with water or oil can cause electric shock, fire, or unit malfunction.
IP Protection Rating: IP40

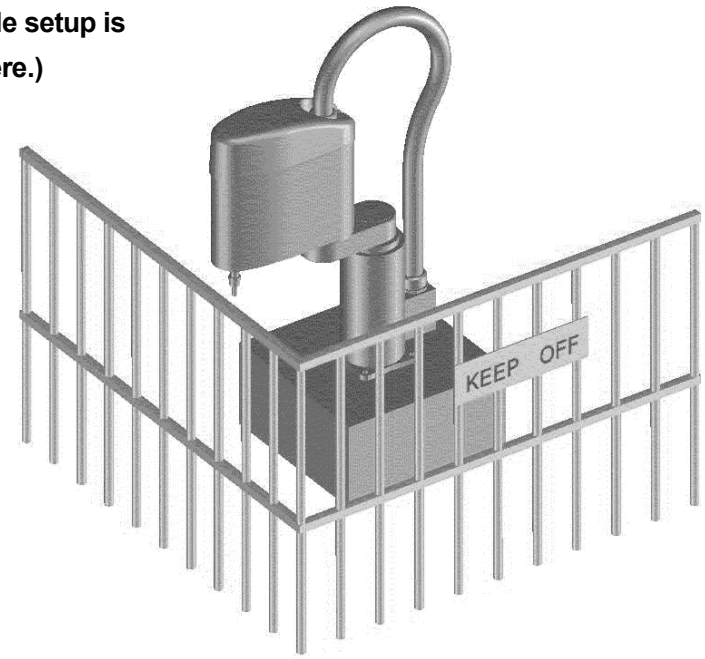
FOR YOUR SAFETY

JS Series and JSG Series
INSTALLATION

⚠ Warning

Always set up a safety barrier.
Anyone entering the operating range of the robot may be injured.
Set up an emergency stop interlock system that is triggered when the entrance to the safety barrier is opened.
Use the I/O-S connector included in the package. Ensure there is no other way of entering the restricted area.
Furthermore, **put up a "No Entry" or "No Operating" warning sign** in a clearly visible place.

(An example setup is pictured here.)



Make sure the safety barrier is strong enough to protect the operator from flying debris caused during robot operation.
Always wear protective gear (helmet, protective gloves, protective glasses, and protective footwear) when going inside the safety barrier.

If the object the robot is holding is in danger of flying off or falling, make sure **safety precautions adequate for the object's size, weight, temperature and chemical composition have been taken.**

FOR YOUR SAFETY

JS Series and JSG Series
INSTALLATION

⚠ Warning

Be sure to check grounding before you use the unit.
Improper grounding can cause electric shock, fire, unit malfunction, or breakdown.

Plug the power cord firmly into the wall outlet and check for dust.
Incomplete insertion into the wall outlet causes the plug to heat up and may result in fire.
Be sure to shut off the power supply before connecting the power cable.

Install the robot in a place which can endure its weight and conditions while running.
Placing the unit on an insufficient or unstable surface may cause the unit to fall or overturn, thereby causing damage or injury.
Be sure to leave a space of at least 30cm between the back of the robot (the side with a cooling fan) and the wall. Insufficient space can lead to overheating or fire.

Do not attempt to disassemble or modify the unit.
Disassembly or modification may cause electric shock or malfunction.

Be sure to use within the voltage range indicated on the unit.
Failure to do so may cause electric shock or fire.

Do not use the unit where flammable or corrosive gas is present.
Leaked gas accumulated around the unit can cause fire or explosion.

For the health and safety of the operator, place the unit in a well-ventilated area.

Turn off the unit before inserting and removing cables.
Failure to do so may result in electric shock, fire, or unit malfunction.

FOR YOUR SAFETY

JS Series and JSG Series

INSTALLATION

⚠ Warning

Be sure to confirm that all the air tubes are connected correctly and firmly.

Use the robot in a condensation-free environment between 0 and 40 degrees centigrade with a humidity level of 20 to 90 percent.
Failure to do so may result in malfunction.
IP Protection Rating: IP40

Use the robot in an electric noise-free environment.
Failure to do so may result in malfunction or breakdown.

Be sure to secure the movable parts of the robot before transportation.
Failure to do so may result in injury or breakdown.

Do not bump or jar the unit while it is being transported or installed.
Doing so can cause breakdown.

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

If tools have been attached (such as the electric screwdriver unit), make sure they are properly connected.
Failure to do so may result in injury or breakdown.

Check the wiring to the main unit.
Improper wiring may result in malfunction or breakdown.

Keep the emergency stop switch within reach of an operator while teaching or running the robot.
Failure to do so is dangerous since it may not be possible to stop the robot immediately and safely.

FOR YOUR SAFETY

JS Series and JSG Series

INSTALLATION

⚠ Warning

Be sure to turn off both the robot and the power source before plugging in the power cable.

Be sure to remove the eyebolt after installing the robot.
If it is not removed, the Arm may bump into it, thereby causing damage or injury.

Place the control box on a flat surface more than 80 cm above the floor so that it is easier to operate it.

Use the unit in an environment that is not dusty or damp.
Dust or dampness may lead to breakdown or malfunction.
IP Protection Rating: IP40

Use the Clean Room type model in an environment of Clean Class 10* or less.
*: Federal Standard 209D (FED-STD209D)

FOR YOUR SAFETY

JS Series and JSG Series

DURING OPERATION

⚠ Warning

When lubricating or inspecting the unit, unplug the power cord from the robot.
Failure to do so may result in electric shock or injury.
Be sure to shut off the power supply before removing the power cord from the robot.

Put **a "Do Not Operate" sign** on the start switch when people are entering the safety barrier.

Keep the emergency stop switch within reach of an operator while teaching or running the robot.
Failure to do so is dangerous since it may not be possible to stop the robot immediately and safely.

Make sure the safety barrier is strong enough to protect the operator from flying debris caused during robot operation.
Always wear protective gear (helmet, protective gloves, protective glasses, and protective footwear) when going inside the safety barrier.

Make sure all the air tubes are connected correctly and firmly.

Always be aware of the robot's movement, even in the Teaching mode.
Careful attention will protect the operator from injury.

FOR YOUR SAFETY

JS Series and JSG Series
DURING OPERATION

⚠ Warning

When entering the safety barrier, **do not enter the operating range of the robot.**

If you must go inside the safety barrier, make sure you **push the emergency stop switch** and **put a "Do Not Operate" sign** on the start switch.

When starting the robot, check that **no one is within the safety barrier and that no object will interfere with the robot's operation.**

Under no circumstances should you go inside the safety barrier or place your hands or head inside the safety barrier while the robot is operating.

If anything unusual (e.g. a burning smell or an abnormal sound) occurs, stop operation and turn off the robot. Contact the dealer from whom you purchased the robot or the office listed on the last page of this manual.
Continuous use without repair can cause electric shock, fire, or breakdown of the unit.

Keep the emergency stop switch within reach of an operator while teaching or running the robot.
Failure to do so is dangerous since it may not be possible to stop the robot immediately and safely.

PREFACE

The Janome Desktop Robot JR2000N Series and pulse motor driven Janome SCARA Robot JSR4400N Series are a new style of low-cost, high-performance robots. We have succeeded in reducing the price while maintaining functionality. Energy- and space-saving qualities have been made possible through the combined use of stepping motors and specialized micro step driving circuits.

The servomotor driven Janome SCARA Robot JS Series and servomotor driven Gantry Robot JSG Series feature diverse applications, high speed, rigidity and precision, and can accommodate a wide variety of requirements.

There are several manuals pertaining to these robots.

Setup	Explains how to set up the robot. ● **Be sure to read this manual before you operate the robot.** ● **This manual is designed for people who receive safety and installation instructions regarding the robot.**
Maintenance	Explains maintenance procedures for the robot. ● **Be sure to read this manual before you operate the robot.** ● **This manual is designed for people who receive safety and installation instructions regarding the robot.**
Basic Instructions	Provides part names, data structures, and the basic knowledge necessary to operate the robot.
Quick Start	Explains the actual operation of the robot with simple running samples.
Teaching Pendant Operation	Explains how to operate the robot via the teaching pendant.
PC Operation	Explains how to operate the robot from a computer (using the JR C-Points software).
Features I	Explains point teaching.
Features II	Explains commands, variables, and functions.
Features III	Explains features such as run mode parameters, sequencer programs, and other functions used during operation.
Features IV	Explains features in the Customizing mode.
External Control I (I/O-SYS)	Explains the I/O-SYS control.
External Control II (COM Communication)	Explains the COM communication control system (COM1 – COM3).
Specifications	Outlines general specifications such as the robot's operating range, weight, etc.

Note: These products are regularly updated, therefore the product specification outlines in this manual may differ from those of the robot in your possession.
These manuals are based on the standard specifications. Menu items may vary depending on the model.

Please be sure to follow the instructions described in these manuals. Proper use of the robot will ensure continued functionality and high performance.

Be sure to turn off the robot and check that the power to the robot is shut off before plugging in the power cord.

BE SURE TO PROPERLY GROUND THE MACHINE WHEN INSTALLING.

Be sure to save data whenever it is added or modified.
Any unsaved data will be lost when the robot is turned off.

CONTENTS

Features II

EXPRESSION STRUCTURE

Expression

An expression is fixed numbers (string type and numeric type), variables, functions, and operators combined.

Fixed Number

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

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

e.g. eoutCOM port2,"%0D%0A" : Output CR・LF code.

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.

e.g. eoutCOM port2,"%G01" : Output *%G01*.

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

e.g. eoutCOM port2,"%%300" : Output *%300*.

Variable

A variable is a container into which numeric and string values are placed.

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

User-defined variables other than local variables (variables effective only in defined point job data which are defined by the *declare* command) are defined in the Customizing mode. (See the operation manual *Features IV* for details of the Customizing mode.)

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

Function

A function returns a converted value if values or strings are given.

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

The user-defined functions are defined in the Customizing mode. (See the operation manual *Features IV* for details of the Customizing mode.)

Operator

Operator	Description	Value
+	Adds the left and right values.	num
-	Deducts the right value from the left value.	num
*	Multiplies the left and right values.	num
/	Divides the left value by the right value.	num
&	Combines the left and right values. *e.g.* "A" & "B" → "AB"	str
=	Assigns the right value to a left value.	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

The priority of operators is as follows:

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

COMMAND LIST

If you assign point job data that includes any of the highlighted ( ) commands to a CP passing point, the commands will be ignored.

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	Output ON pulses of predetermined length.
	invPulse	Output Destination, Pulse Width	Output OFF pulses of predetermined length.
	delaySet	Output Destination, Delay Time	Output ON after the predetermined delay time.
	delayReset	Output Destination, Delay Time	Output OFF after the predetermined delay time.
	onoffBZ	ON Time, OFF Time	Sound the buzzer on and off.
	onoffGLED	ON Time, OFF Time	Flash the green LED on the front of the JR2000N Series robot (or on the operation box for the JSR4400N robot).
	onoffRLED	ON Time, OFF Time	Flash the red LED on the front of the JR2000N Series robot (or on the operation box for the JSR4400N robot).
	dataOut	Output Data, Output Bit Number, Output Destination	Output numeric data or a tag code assigned to a point to the I/O.
	dataOutBCD	Output Value, Output Bit Number, Output Destination	Output numeric data or a tag code assigned to a point to the I/O in BCD (binary-coded decimal).
	motorPowerON	–	
	servoON	Axis	Turn on the designated Axis servomotor. (Available only for the JS and JSG Series)
	servoOFF	Axis	Turn off the designated Axis servomotor. (Available only for the JS and JSG Series)
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.

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	Input serial ON.
	ani	Boolean variable or Expression	Input serial OFF.
	or	Boolean variable or Expression	Input parallel ON.
	ori	Boolean variable or Expression	Input parallel OFF.
	anb	–	Serial connection of blocks
	orb	–	Parallel connection of blocks
Delay	delay	Delay Time	Stop for a specified time.
	dataIn	Numeric Variable Name, Input Bit Number, Input Source	Read out numeric data from the I/O.
	dataInBCD	Numeric Variable Name, Input Bit Number, Input Source	Read numeric data in BCD from the I/O.
	waitStart	–	Wait for a start signal.
	waitStartBZ	–	Wait for a start signal while acknowledging an error with an alarm buzzer.
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)
Execute Flow Control	callBase	–	Call a user-defined point job from a point to which a point job number is set.
	callJob	Point Job Number	Call a point job data subroutine specified by number.
	callPoints	Variable Name (Identifier)	Perform a specified point string (defined in the Customizing mode).
	returnJob	–	End of point job
	returnFunc	Return Value (Expression)	Terminate the function by assigning the value of the specified expression as a return value. (This command is valid in functions only.)
	callProg	Program Number	Call a program subroutine specified by number.
	endProg	–	End of program
	goPoint	PTP Condition Number, go Point Number	Jump to a specified point.
	goRPoint	PTP Condition Number, Relative go Point Number	Jump to a relatively-specified point.
	goCRPoint	PTP Condition Number, Relative go Point Number	Jump to a selected destination point while running in the CP drive.
	jump	Label Number	Jump to a specified label.
	Label	Label Number	Label

Category	Command	Necessary Parameter	Description
For, do-loop	for	Variable Name, 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.
Move	upZ	Speed, Distance	Z Up
	downZ	Speed, Distance	Z Down
	movetoZ	Speed, Distance	Z Move
	lineMove	Line Speed, (X, Y, Z) Distance, R Rotate Angle	Make an Axis move a specified distance (relative distance) at a specified speed in the CP line drive. (Relative move command) Entering this command will display the specified shifting speed and distance of each Axis as follows: **e.g.** lineMoveSpeed 20 lineMoveX 10 lineMoveY 20 lineMoveZ 0 lineMoveR 0
	lineMoveStopIf	–	Terminate the movement of an Axis made by *lineMove* if the conditions are met.
	endLineMove	–	End of *lineMoveStopIf* condition statements.
	initMec	Axis	Return the specified Axis to its initial position. (Perform mechanical initialization.) (Available only for the JR2000N and JSR4400N Series)
	checkPos	–	Detect a position error. (Available only for the JR2000N Series)
LCD Control	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 result of the string expression on the LCD display.
	sys7SLED	–	Returns the 7 segment LED display changed by *out7SLED* to the previous program number. (Available only for the JR2000N Series and JSR4400N Series)
	out7SLED	Type, Output Value	Output 7 segment LED. (Available only for the JR2000N Series and JSR4400N Series)

Category	Command	Necessary Parameter	Description
COM Input/Output	outCOM	Input/Output, Output Data	Output the string from the COM.
	eoutCOM	Input/Output, Output Data	Output the result of the string expression from the COM.
	setWTCOM	Input/Output, Wait Time	Set [Wait Time] (time-out period) for receiving data from the COM.
	inCOM	Variable Name, Input/Output, Character Length	Assign the receive data from the COM to the specified variable.
	cmpCOM	Input/Output, Compare Data	Compare the receive data and string. The result is entered into the system flags (sysFlag(1) – sysFlag(20)).
	ecmpCOM	Input/Output, Compare Data	Compare the receive data and string expression. The result is entered into the system flags (sysFlag(1) – sysFlag(20)).
	clrCOM	Input/Output	Clear the COM receive buffer.
	shiftCOM	Input/Output, Shift Number	Shift data received from the COM. Delete data from the top by the specified [Shift Number].
	stopPC	–	Stop the PC communication of the COM1.
	startPC	–	Start the PC communication of the COM1.
Variable, Comment, System Control	declare	Variable Type, Variable Name	Local variable declaration
	let	Expression	Assign the l-value to the r-value. The symbols +, -, *, /, =, (,), & can be used.
	rem	Output Data	One line comment
	crem	Output Data	Comment at the end of a command line
	setProgNum	Program Number	Change the program number. ● **Do not carry out this command while the robot is running. Use the command *callProgram* if you change the program to be run while the robot is running.**
	setSeqNum	Sequencer Number	Change the sequencer number in the system data.
Camera, Z Adjustment	cameraWadj	Work Adjustment Number	Take an image with a camera and calculate the offset from the data gained according to the [Workpiece Adjustment] setting.
	wCameraWadj	Work Adjustment Number, Shot Number	Use this command when calculating offset using two camera images according to the [Workpiece Adjustment].
	cameraTool	Tool Number	Take an image with a camera and calculate [TCP-X] and [TCP-Y] from the data gained according to the [Point Tool Data Settings].
	cameraPallet	Pallet Routine Number	Take an image with a camera and set the number of gained marks and the coordinates as the number and the coordinates of the [Pallet Routine] to be performed.
	takeZWadj	Work Adjustment Number	Calculate the Z offset from the data gained by the distance or touch-sensitive sensor according to the [Workpiece Adjustment] settings.

- For the [Camera, Z Adjustment] command category, refer to the *Camera & Sensor Functions* operation manual.

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	Input serial ON.
	ani	Boolean variable or expression	Input serial OFF.
	or	Boolean variable or expression	Input parallel ON.
	ori	Boolean variable or expression	Input parallel OFF.
	anb	–	Serial connection of blocks
	orb	–	Parallel connection of blocks

Sequencer

Category	Command	Necessary Parameter	Description
Calculate	ld	Boolean variable	Input ON.
	ldi	Boolean variable	Input OFF.
	and	Boolean variable	Input serial ON.
	ani	Boolean variable	Input serial OFF.
	or	Boolean variable	Input parallel ON.
	ori	Boolean variable	Input parallel OFF.
Coil	out	Output Destination	Coil drive
	set	Output Destination	Set the coil drive hold command.
	reset	Output Destination	Reset the coil drive hold command.
	pls	Output Destination	Output the rising edge of pulse.
	plf	Output Destination	Output the falling edge of pulse.
Connection	anb	–	Parallel connection of serial circuit block
	orb	–	Serial connection of parallel circuit block
	mps	–	Store data in process of calculation.
	mrd	–	Read out data in process of calculation.
	mpp	–	Read out and reset data in process of calculation.
Others	nop	–	No operation

VARIABLE LIST

You can use the built-in variables (which are built into the robot as a function), and the user-defined variables (which can be freely defined by the user).
User-defined variables other than local variables (variables effective only in defined point job data which are defined by the *declare* command) are defined in the Customizing mode. (See the operation manual *Features IV* for details of the Customizing mode.)

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

Category	Type	Identifier	Description	
Free Variable	boo	#mv (1 – 99)	Boolean variable	
	boo	#mkv (1 – 99)	Boolean variable (Keeping variable)*	
	num	#nv (1 – 99)	Numerical variable	
	num	#nkv (1 – 99)	Numerical variable (Keeping variable)*	
	str	#sv (1 – 99)	String variable	
	str	#skv (1 – 99)	String variable (Keeping variable)*	
Input Variable	boo	#sysIn1 – 15	(JS and JSG Series)	I/O-SYS
		#sysIn1 – 16	(JR2000N and JR4400N Series)	
	boo	#genIn1 – 18	(JS and JSG Series)	I/O-1
		#genIn1 – 8	(JR2000N and JR4400N Series)	
	boo	#handIn1 – 4	(Available only for JS Series)	I/O-H
Output Variable	boo	#sysOut1 – 14	(JS and JSG Series)	I/O-SYS
		#sysOut1 – 16	(JR2000N and JR4400N Series)	
	boo	#genOut1 – 22	(JS and JSG Series)	I/O-1
		#genOut1 – 8	(JR2000N and JR4400N Series)	
	boo	#handOut1 – 4	(Available only for JS Series)	I/O-H
System Flag	boo	#sysFlag(1) – #sysFlag(999)	Refer to “SYSTEM FLAG LIST” on Page 14.	
Buzzer	boo	#FBZ	set #FBZ : Sound the buzzer. reset #FBZ : Stop the buzzer. (onoffBZ : Turn the buzzer on and off.)	

*: Variables which hold their values even if the robot is turned off are collectively referred to as *keeping variables* in the operation manuals.

Variables

Category	Type	Identifier	Description
Special Variable	num	#downTimer1 – 10	The assigned value will be decreased automatically (by msec).
	num	#jobStartHight	Start a point job from a position above the Z-coordinate determined by the assigned value. (Invalid in the CP drive)
	num	#jobStartX	Start a point job from a position at a distance from the X-coordinate determined by the assigned value. (Invalid in the CP drive)
	num	#jobStartY	Start a point job from a position at a distance from the Y-coordinate determined by the assigned value. (Invalid in the CP drive)
	num	#jobStartR	Start a point job from a position at a distance from the R-coordinate determined by the assigned value. (Invalid in the CP drive)
Pallet Routine	boo	#palletFlag (1 – 100)	Pallet flag (Corresponds to Pallet Routine Nos. 1 – 100.)
	num	#palletCount (1 – 100)	Pallet counter (Corresponds to Pallet Routine Nos. 1 – 100.)
Workpiece Adjustment	num	#workAdj_X (1 – 100) #workAdj_Y (1 – 100) #workAdj_Z (1 – 100) #workAdj_R (1 – 100) #workAdj_Rotation (1 – 100)	Adjustment value of each Axis in [Workpiece Adjustment] settings (Corresponds to Work Adjustment Nos. 1 – 100.)
Tool Data	num	#tool_X (1 – 100) #tool_Y (1 – 100) #tool_Z (1 – 100) #tool_R (1 – 100)	TCP value of each Axis in [Point Tool Data Settings] settings. (Corresponds to Tool Nos. 1 – 100.)
PTP Condition	num	#priorityPTPCondNum	PTP condition number The PTP condition number set by using this variable has priority over other PTP condition numbers in the PTP drive (even at the points to which the additional function data [PTP Condition] is set).
Sequencer	boo	#seqT (1 – 99)	Add 1 when #seqTCount reaches the specified value or greater.
	num	#seqTCount (1 – 50): Integrating timer #seqTCoun (51 – 99): Unintegrating timer	One counter can count 0.001 – 2,147,483,647 seconds (by 0.001 second).
	boo	#seqC (1 – 99)	Add 1 when #seqCCount reaches the specified value or greater.
	num	#seqCCount (1 – 99)	One counter can count 1 – 2,147,483,647.

Variables

Category	Type	Identifier	Description
Current Point Coordinates	num	#point_X	X-coordinate value of the point currently performed
	num	#point_Y	Y-coordinate value of the point currently performed
	num	#point_Z	Z-coordinate value of the point currently performed
	num	#point_R	R-coordinate value of the point currently performed
	num	#point_TagCode	Tag code value of the point currently performed
Specified Point Coordinates	num	#P_X (1 – last point number)	X-coordinate value of the specified point
	num	#P_Y (1 – last point number)	Y-coordinate value of the specified point
	num	#P_Z (1 – last point number)	Z-coordinate value of the specified point
	num	#P_R (1 – last point number)	R-coordinate value of the specified point
	num	#P_TagCode (1 – last point number)	Tag code value of the specified point
Specified Program, Specified Point Coordinates	num	#prog_P_X (1 – 255, 1 – last point number)	X-coordinate value of the specified point in the specified program
	num	#prog_P_Y (1 – 255, 1 – last point number)	Y-coordinate value of the specified point in the specified program
	num	#prog_P_Z (1 – 255, 1 – last point number)	Z-coordinate value of the specified point in the specified program
	num	#prog_P_R (1 – 255, 1 – last point number)	R-coordinate value of the specified point in the specified program
	num	#prog_P_TagCode (1 – 255, 1 – last point number)	Tag code value of the specified point in the specified program
Condition Number	num	#point_CondNum	Condition setting variable number set to the point currently performed
	num	#P_CondNum (1 – last point number)	Condition setting variable number set to the specified point
	num	#prog_P_CondNum (1 – 255, 1 – last point number)	Condition setting variable number set to the specified point in the specified program

FUNCTION LIST

You can use the built-in functions (which are built into the robot as a function) and the user-defined functions (which can be freely defined by the user).
The user-defined functions are defined in the Customizing mode. (See the operation manual *Features IV* for details of the Customizing mode.)

x, y: Numerical value or numerical variable
n, m: Numeric value becomes larger than a certain value through rounding or truncation
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	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	isConditionData (n)	Display whether the specified condition data number is available (1) or not (0).
	str	strCenterLCD (a)	Adjust the strings on the teaching pendant LCD (centering).
	str	strRightLCD (a)	Adjust 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: 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 drive [sec]
	num	getSystemPTPrestTime ()	Valid only for [Job while Moving]. Time left before the current PTP drive ends (reaching the destination) [sec]

x, y: Numerical value or numerical variable
n, m: Numeric value becomes larger than a certain value through rounding or truncation
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	degrade (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
	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)	Return a string (1 character) with the given character code.
	num	ord (a)	Return the top character code. Other codes are ignored.
	num	len (a)	Return the string length (non-multibyte).
	num	strPos (a,b)	Return the first part string position in a matching b.
	str	strMid (a,n,m)	Return the strings n – m counted from the top of the given string a.
	str	str (x)	Convert a numeric value to a decimal digit string.
	str	strBin (n,m)	Convert a numeric value to a binary string. m: Number of binary string digits
	str	strHex (n,m)	Convert a numeric value to a hexadecimal string. m: Number of hexadecimal string digits
	str	str1SI (x)	Round a numeric value to a 1-byte signed integer to convert it to a 1-byte string. (1-byte Signed Integer)
	str	str2SIBE (x)	Round 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)	Round 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)	Round 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)	Round 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)

x, y: Numerical value or numerical variable
n, m: Numeric value becomes larger than a certain value through rounding or truncation
a, b: String or string variable

Category	Type	Identifier	Description
String System	str	str4FBE (x)	Regard a numeric value as a float to convert it to a 4-byte string using the Big Endian byte order. (4-byte Signed Float Big Endian)
	str	str4FLE (x)	Regard a numeric value as a float to convert it to a 4-byte string using the Little Endian byte order. (4-byte Signed Float Big Endian)
	str	str8DBE (x)	Regard a numeric value as a float to convert it to an 8-byte string using the Big Endian byte order. (8-byte Signed Float Big Endian)
	str	str8DLE (x)	Regard a numeric value as a float to convert it to an 8-byte string using the Little Endian byte order. (8-byte Signed Float Little Endian)
	num	val (a)	Regard a string as a decimal digit string to convert it to a numeric value.
	num	valBin (a)	Regard a string as a binary string (sequence of “0”, “1”) to convert it to a numeric value.
	num	valHex (a)	Regard a string as a hexadecimal string (sequence of “0” – “1”, “A” – “F”, or “a” – “f”) to convert it to a numeric value.
	num	val1SI (a)	Convert the top character to a 1-byte signed integer. (1-byte Signed Integer)
	num	val2SIBE (a)	Convert 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)	Convert 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)	Convert 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)	Convert 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)	Convert the top 4 characters to a float using the Big Endian byte order. (4-byte Float Big Endian)
	num	val4FLE (a)	Convert the top 4 characters to a float using the Little Endian byte order. (4-byte Float Little Endian)
	num	val8DBE (a)	Convert the top 8 characters to a double-precision float using the Big Endian byte order. (8-byte Double Big Endian)
	num	val8DLE (a)	Convert the top 8 characters to a double-precision float using the Little Endian byte order. (8-byte Little Big Endian)
	num	valSum (a)	Return the sum of a string code from top to bottom.
	num	valCRC (a)	Remainder of dividing a string (bit string) by a generator polynomial $X^{16}+X^{12}+X^5+1$
	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

SYSTEM FLAG LIST

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

JR2000N Series

No.	Identifier	Description	Condition “1” (True)
01	#FisCOM1	COM1 receive data existence	Exists
02	#FltCOM1	Result of COM1 receive data compare command (cmpCOM)	Constant > Receive data
03	#FeqCOM1	Result of COM1 receive data compare command (cmpCOM)	Constant = Receive data
04	#FgtCOM1	Result of COM1 receive data compare command (cmpCOM)	Constant < Receive data
05	#FtimeOutCOM1	COM1 receive data compare command (cmpCOM) timeout	Timeout
06	#FisCOM2	COM2 receive data existence	Exists
07	#FltCOM2	Result of COM2 receive data compare command (cmpCOM)	Constant > Receive data
08	#FeqCOM2	Result of COM2 receive data compare command (cmpCOM)	Constant = Receive data
09	#FgtCOM2	Result of COM2 receive data compare command (cmpCOM)	Constant < Receive data
10	#FtimeOutCOM2	COM2 receive data compare command (cmpCOM) timeout	Timeout
11	#FisCOM3	COM3 receive data existence	Exists
12	#FltCOM3	Result of COM3 receive data compare command (cmpCOM)	Constant > Receive data
13	#FeqCOM3	Result of COM3 receive data compare command (cmpCOM)	Constant = Receive data
14	#FgtCOM3	Result of COM3 receive data compare command (cmpCOM)	Constant < Receive data
15	#FtimeOutCOM3	COM3 receive data compare command (cmpCOM) timeout	Timeout
16	#FisCOM4	COM4 receive data existence	Exists
17	#FltCOM4	Result of COM4 receive data compare command (cmpCOM)	Constant > Receive data
18	#FeqCOM4	Result of COM4 receive data compare command (cmpCOM)	Constant = Receive data
19	#FgtCOM4	Result of COM4 receive data compare command (cmpCOM)	Constant < Receive data
20	#FtimeOutCOM4	COM4 receive data compare command (cmpCOM) timeout	Timeout
30	#FinitMecError	State of mechanical initialization command error	Mechanical initialization error
31	#FcameraError	State of camera data error	Error
32	#FtakeZError	State of Z height data (takeZWadj) error	Error
33	#FlMoveOutRange	Relative move command range	Out of range
34	#FlMoveStop	Condition stop state of relative move command	Stopped by the stop condition
35	#FcheckPosError	Result of the position error detect command	Position error
36	#FdataInBCDError	State of dataInBCD command error	Error

No.	Identifier	Description	Condition "1" (True)
60	#FstartSW	Start 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	Initial X position sensor	ON
72	#Fsensor2	Initial Y position sensor	ON
73	#Fsensor3	Initial Z position sensor	ON
74	#Fsensor4	Initial R position sensor	ON
76	#Fdrvoz1	X driver 0-phase	Close
77	#Fdrvoz2	Y driver 0-phase	Close
78	#Fdrvoz3	Z driver 0-phase	Close
79	#Fdrvoz4	R driver 0-phase	Close
91	#FenableSW	Enable switch	ON (Pressed)
92	#FspmodeSW	Special mode switch	ON
93	#FspareSW	Spare switch	ON
94	#FmotorPower	State of the power to the motor	ON

JR4400N Series

No.	Identifier	Description	Condition "1" (True)
01	#FisCOM1	COM1 receive data existence	Exists
02	#FltCOM1	Result of COM1 receive data compare command (cmpCOM)	Constant > Receive data
03	#FeqCOM1	Result of COM1 receive data compare command (cmpCOM)	Constant = Receive data
04	#FgtCOM1	Result of COM1 receive data compare command (cmpCOM)	Constant < Receive data
05	#FtimeOutCOM1	COM1 receive data compare command (cmpCOM) timeout	Timeout
06	#FisCOM2	COM2 receive data existence	Exists
07	#FltCOM2	Result of COM2 receive data compare command (cmpCOM)	Constant > Receive data
08	#FeqCOM2	Result of COM2 receive data compare command (cmpCOM)	Constant = Receive data
09	#FgtCOM2	Result of COM2 receive data compare command (cmpCOM)	Constant < Receive data
10	#FtimeOutCOM2	COM2 receive data compare command (cmpCOM) timeout	Timeout
11	#FisCOM3	COM3 receive data existence	Exists
12	#FltCOM3	Result of COM3 receive data compare command (cmpCOM)	Constant > Receive data
13	#FeqCOM3	Result of COM3 receive data compare command (cmpCOM)	Constant = Receive data
14	#FgtCOM3	Result of COM3 receive data compare command (cmpCOM)	Constant < Receive data
15	#FtimeOutCOM3	COM3 receive data compare command (cmpCOM) timeout	Timeout
16	#FisCOM4	COM4 receive data existence	Exists
17	#FltCOM4	Result of COM4 receive data compare command (cmpCOM)	Constant > Receive data
18	#FeqCOM4	Result of COM4 receive data compare command (cmpCOM)	Constant = Receive data

No.	Identifier	Description	Condition "1" (True)
19	#FgtCOM4	Result of COM4 receive data compare command (cmpCOM)	Constant < Receive data
20	#FtimeOutCOM4	COM4 receive data compare command (cmpCOM) timeout	Timeout
30	#FinitMecError	State of mechanical initialization command error	Mechanical initialization error
31	#FcameraError	State of camera data error	Error
32	#FtakeZError	State of Z height data (takeZWadj) error	Error
33	#FlMoveOutRange	Relative move command range	Out of range
34	#FlMoveStop	Condition stop state of relative move command	Stopped by the stop condition
35	#FcheckPosError	Result of the position error detect command	Position error
36	#FdataInBCDError	State of dataInBCD command error	Error
60	#FstartSW	Start 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)
66	#FmponSW	Power ON switch	ON (Pressed)
68	#FmdSW1	Select key switch 1	ON
69	#FmdSW2	Select key switch 2	ON
71	#Fsensor1	Initial X position sensor	ON
72	#Fsensor2	Initial Y position sensor	ON
73	#Fsensor3	Initial Z position sensor	ON
74	#Fsensor4	Initial R position sensor	ON
76	#Fdrvoz1	X driver 0-phase	Close
77	#Fdrvoz2	Y driver 0-phase	Close
78	#Fdrvoz3	Z driver 0-phase	Close
79	#Fdrvoz4	R driver 0-phase	Close
91	#FenableSW	Enable switch	ON (Pressed)
92	#FspmodeSW	Special mode switch	ON
93	#FspareSW	Spare switch	ON
94	#FmotorPower	State of the power to the motor	ON

JS Series and JSG Series

No.	Identifier	Description	Condition "1" (True)
01	#FisCOM1	COM1 receive data existence	Exists
02	#FltCOM1	Result of COM1 receive data compare command (cmpCOM)	Constant > Receive data
03	#FeqCOM1	Result of COM1 receive data compare command (cmpCOM)	Constant = Receive data
04	#FgtCOM1	Result of COM1 receive data compare command (cmpCOM)	Constant < Receive data
05	#FtimeOutCOM1	COM1 receive data compare command (cmpCOM) timeout	Timeout
06	#FisCOM2	COM2 receive data existence	Exists
07	#FltCOM2	Result of COM2 receive data compare command (cmpCOM)	Constant > Receive data
08	#FeqCOM2	Result of COM2 receive data compare command (cmpCOM)	Constant = Receive data
09	#FgtCOM2	Result of COM2 receive data compare command (cmpCOM)	Constant < Receive data
10	#FtimeOutCOM2	COM2 receive data compare command (cmpCOM) timeout	Timeout

No.	Identifier	Description	Condition “1” (True)
11	#FisCOM3	COM3 receive data existence	Exists
12	#FltCOM3	Result of COM3 receive data compare command (cmpCOM)	Constant > Receive data
13	#FeqCOM3	Result of COM3 receive data compare command (cmpCOM)	Constant = Receive data
14	#FgtCOM3	Result of COM3 receive data compare command (cmpCOM)	Constant < Receive data
15	#FtimeOutCOM3	COM3 receive data compare command (cmpCOM) timeout	Timeout
16	#FisCOM4	COM4 receive data existence	Exists
17	#FltCOM4	Result of COM4 receive data compare command (cmpCOM)	Constant > Receive data
18	#FeqCOM4	Result of COM4 receive data compare command (cmpCOM)	Constant = Receive data
19	#FgtCOM4	Result of COM4 receive data compare command (cmpCOM)	Constant < Receive data
20	#FtimeOutCOM4	COM4 receive data compare command (cmpCOM) timeout	Timeout
30	#FinitMecError	State of mechanical initialization command error	Mechanical initialization error
31	#FcameraError	State of camera data error	Error
32	#FtakeZError	State of Z height data (takeZWadj) error	Error
33	#FlMoveOutRange	Relative move command range	Out of range
34	#FlMoveStop	Condition stop state of relative move command	Stopped by the stop condition
35	#FcheckPosError	Result of the position error detect command	Position error
36	#FdataInBCDError	State of dataInBCD command error	Error
63	#FemgSW	EMG direct input	ON (Pressed)
64	#Fios	IOS direct input	Circuit open (Disconnected)
66	#FmponSW	Power ON switch	ON (Pressed)
68	#FmdSW1	Select key switch 1	ON
69	#FmdSW2	Select key switch 2	ON
91	#FenableSW	Enable switch	ON (Pressed)
92	#FspmodeSW	Special mode switch	ON
93	#FspareSW	Spare switch	ON
94	#FmotorPower	Motor power state	ON
121	#FsvReady1	J1/X servo ready	Ready
122	#FsvReady2	J2/Y servo ready	Ready
123	#FsvReady3	J3/Z servo ready	Ready
124	#FsvReady4	J4/R servo ready	Ready
126	#FsvAlarm1	J1/X servo alarm	Servodriver error
127	#FsvAlarm2	J2/Y servo alarm	Servodriver error
128	#FsvAlarm3	J3/Z servo alarm	Servodriver error
129	#FsvAlarm4	J4/R servo alarm	Servodriver error
131	#FsvPos1	J1/X servo finish positioning	Positioning finished
132	#FsvPos2	J2/Y servo finish positioning	Positioning finished
133	#FsvPos3	J3/Z servo finish positioning	Positioning finished
134	#FsvPos4	J4/R servo finish positioning	Positioning finished
136	#FencOz1	J1/X encoder zero phase	Close
137	#FencOz2	J2/Y encoder zero phase	Close
138	#FencOz3	J3/Z encoder zero phase	Close
139	#FencOz4	J4/R encoder zero phase	Close
141	#FencBattery1	J1/X battery warning	Battery run out
142	#FencBattery2	J2/Y battery warning	Battery run out
143	#FencBattery3	J3/Z battery warning	Battery run out
144	#FencBattery4	J4/R battery warning	Battery run out

VARIABLES

■ Free Variables: #mv, #mkv, #nv, #nkv, #sv, #skv

A variable is a container into which numeric and strings values are placed.
You can use the built-in variables listed below freely. Variable declaration is unnecessary when using these variables.

	Identifier	Description
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)	String variable
	#skv (1 – 99)	String variable (Keeping variable)*

*: Variables which hold their values even if the robot is turned off are collectively referred to as *keeping variables* in the operation manuals.

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

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

- Boolean type free variables, #mv (1 – 99) and #mv (1 – 99), can also be used in sequencer programs

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

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

■ #sv (1 – 99) and #skv (1 – 99): String variable

These can hold up to 255 bytes. When used as assignment expression (let) parameters, assignment by = and connection by *&* are possible.

■ Input Variables: #sysIn1..., #genIn1..., #handIn1...

An input variable is a Boolean variable that can only be referred to. You cannot enter a value into it.
It corresponds to the I/O-SYS, I/O-1, and I/O-H input pins. When an ON signal comes, the input variable becomes “1” (true).

Category	Identifier (JS and JSG Series)	Identifier (JR2000N and JSR4400N Series)	Connector	Description
Input Variable	#sysIn1 – #sysIn15	#sysIn1 – #sysIn16	I/O-SYS	Boolean variable for reference only
	#genIn1 – #genIn18	#genIn1 – #genIn8	I/O-1	Boolean variable for reference only
	#handIn1 – #handIn4	—	I/O-H	Boolean variable for reference only

Some of the #sysIn1 – #sysIn15, and 16 (I/O-SYS) pins have pre-assigned functions.
e.g. #sysIn1: Start (When this signal is turned on, the robot starts operation.)

If you wish to use the #sysIn1 – #sysIn15, and 16 (I/O-SYS) pins for functions other than the pre-assigned ones, switch the function to [Free] in the [IO-SYS Function Assignment] settings ([Run Mode Parameter] menu).

- The JSG, JR2000N, and JSR4400N Series are not equipped with I/O-H. Identifiers #handIn1 – #handIn4 are activated for the JS Series only. Note that if you are using I/O-U (option) for the JS Series, any commands from the robot to I/O-H are deactivated.

- For details of the I/O-SYS pre-assigned functions, see the *External Control I (I/O-SYS)* operation manual.

■ Output Variables: #sysOut1..., #genOut1..., #handOut1...

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

Category	Identifier (JS and JSG Series)	Identifier (JR2000N and JSR4400N Series)	Connector	Description
Output Variable	#sysOut1 – #sysOut14	#sysOut1 – #sysOut16	I/O-SYS	Boolean variable
	#genOut1 – #genOut18	#genOut1 – #genOut8	I/O-1	Boolean variable
	#handOut1 – #handOut4	—	I/O-H	Boolean variable

Some of the #sysOut1 – #sysOut14, 15 and 16 (I/O-SYS) pins have pre-assigned functions.
e.g. #sysIn1: Ready for Start (When this signal is turned on, the robot can start operation.)

If you wish to use the #sysOut1 – #sysOut14, 15 and 16 (I/O-SYS) pins for functions other than the pre-assigned ones, switch the function to [Free] in [IO-SYS Function Assignment] settings ([Run Mode Parameter] menu).

- The JSG, JR2000N, and JSR4400N Series are not equipped with I/O-H. Identifiers #handOut1 – #handOut4 are activated for the JS Series only. Note that if you are using I/O-U (option) for the JS Series, any commands from the robot to I/O-H are deactivated.

- For details of the I/O-SYS pre-assigned functions, see the *External Control I (I/O-SYS)* operation manual.

■ Down Timer: #downTimer1 – #downTimer10

A numeric variable: The assigned value (using a *let* command) is decreased automatically (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 is decreased automatically (by msec).

For example, 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 it moves in the CP drive.

```
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.

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

■ Point Job Start Height: #jobStartHight

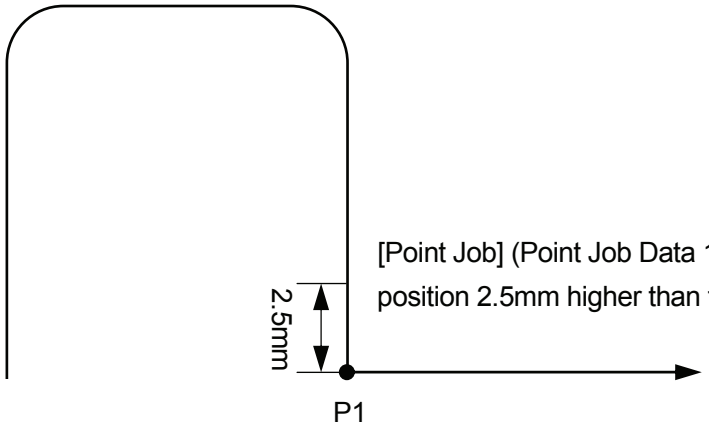
When a value is assigned to the variable "#jobStartHight" (using a *let* command) and the variable is set as [Job before Moving] or [Job while Moving], the point job starts from a position that is higher than the set Z-coordinate by the assigned value.
Do not set point job data that includes *#jobStartHight* as the [Point Job] because the robot Axis or Arm has already reached the point job start position. Also, since this variable acts only on the set point, the point job start position of the next point does not change.

e.g.

P01
Type: CP Start Point
Job before Moving: Point Job Data 3
Point Job: Point Job Data 12

Point Job Data 3

#jobStartHight 25



[Point Job] (Point Job Data 12) is started from a position 2.5mm higher than the preset height.

Category	Identifier	Description
Special Variable	#jobStartHight	The point job is started from a position that is higher than the set Z coordinate by the assigned value. (Deactivated during the CP drive)

■ Pallet: #palletFlag (1 – 100), #palletCount (1 – 100)

#palletCount (1 – 100) is a numeric variable and #palletFlag (1 – 100) is a Boolean variable.
Each variable retains the value of the corresponding pallet counter and pallet flag (1 (true) when the pallet counter is full) 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.

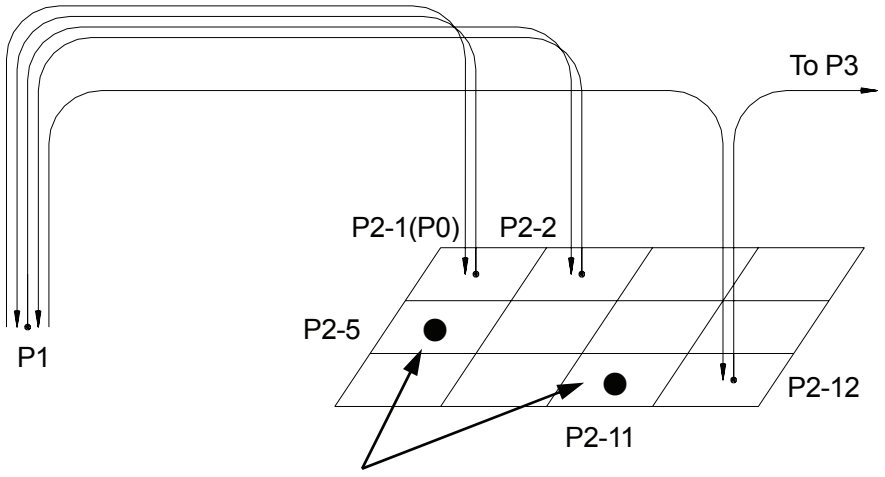
Category	Identifier	Description
Pallet	#palletFlag (1 – 100)	Pallet flag (Corresponds to Pallet 1 – 100.)
	#palletCount (1 – 100)	Pallet counter (Corresponds to Pallet 1 – 100.)

- #palletFlag (1 – 100) does not become "1" (true) even if a value which fills the counter is assigned to #palletCount (1 – 100).

For example, you can skip a designated pallet during a pallet job.
The robot picks up a workpiece at P1, places it on a pallet (set at P2) and moves to the next point (P3) when the pallet becomes full. However, there are two points (P2-5 and P2-11) on the pallet where a workpiece is not placed.

In this example, the [Pallet Routine Number] is [3] and the tool unit is connected according to the following settings:

Picking up a workpiece: #handOut1 is ON.
Placing a workpiece: #handOut1 is OFF.



Points where a workpiece is not placed

Point job data set to P1

set #handOut1	Pick up workpiece.

Point job data set to P2

if
  ld #palletCount(3) == 5
  or #palletCount(3) == 11
else
  reset #handOut1
endIf
	If
	#palletCount (3) is
	
	other than 5 (P2-5) and 11 (P2-11),
	Place (release) a workpiece.

loopPallet 3,1	Add 1 to the counter of Pallet 3. If the counter reaches maximum, go to the next command. (In this example, the point job is over because there are no more commands.) If not, shift to Point 01 (P1).

■ Workpiece Adjustment: #workAdj_X, #workAdj_Y, #workAdj_Z, #workAdj_R, #workAdj_Rotation

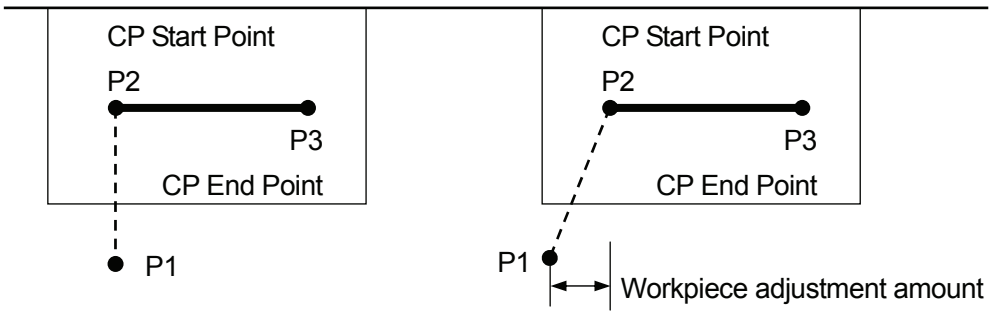
These numeric variables hold the adjustment amount and rotation adjustment amount of each Axis in additional function data [Workpiece Adjustment].

Category	Identifier	Description
Workpiece Adjustment	#workAdj_X (1 – 100)	Workpiece adjustment amount in the X direction (Corresponds to Workpiece Adjustment 1 – 100.)
	#workAdj_Y (1 – 100)	Workpiece adjustment amount in the Y direction (Corresponds to Workpiece Adjustment 1 – 100.)
	#workAdj_Z (1 – 100)	Workpiece adjustment amount in the Z direction (Corresponds to Workpiece Adjustment 1 – 100.)
	#workAdj_R (1 – 100)	Workpiece adjustment amount in the R direction (Corresponds to Workpiece Adjustment 1 – 100.)
	#workAdj_Rotation (1 – 100)	Workpiece adjustment amount by the rotating angle (Corresponds to Workpiece Adjustment 1 – 100.)

For example, you can perform a line dispensing between P2 – P3.
At P1, the workpiece adjustment amount (workpiece offset value) is received from the sensor connected to COM.

In this example, the [Workpiece Adjustment] is [6] and the tool unit is connected according to the following settings:

Starting dispensing: #handOut1 is ON.
Stopping dispensing: #handOut1 is OFF.



Point job data set to P1

```
declare str hosei
inCom hosei,port1,10
#workAdj_X(6) = hosei
```

Declaration of a string type local variable *hosei*
Receive a work adjustment amount from COM1 to *hosei*.
Assign the value in *hosei* to #workAdj_X(6).
(*#workAdj_X(6)* is the X direction adjustment amount of Workpiece Adjustment 6)

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

set #handOut1	Start dispensing.

Point job data set in P3

reset #handOut1	Stop dispensing.

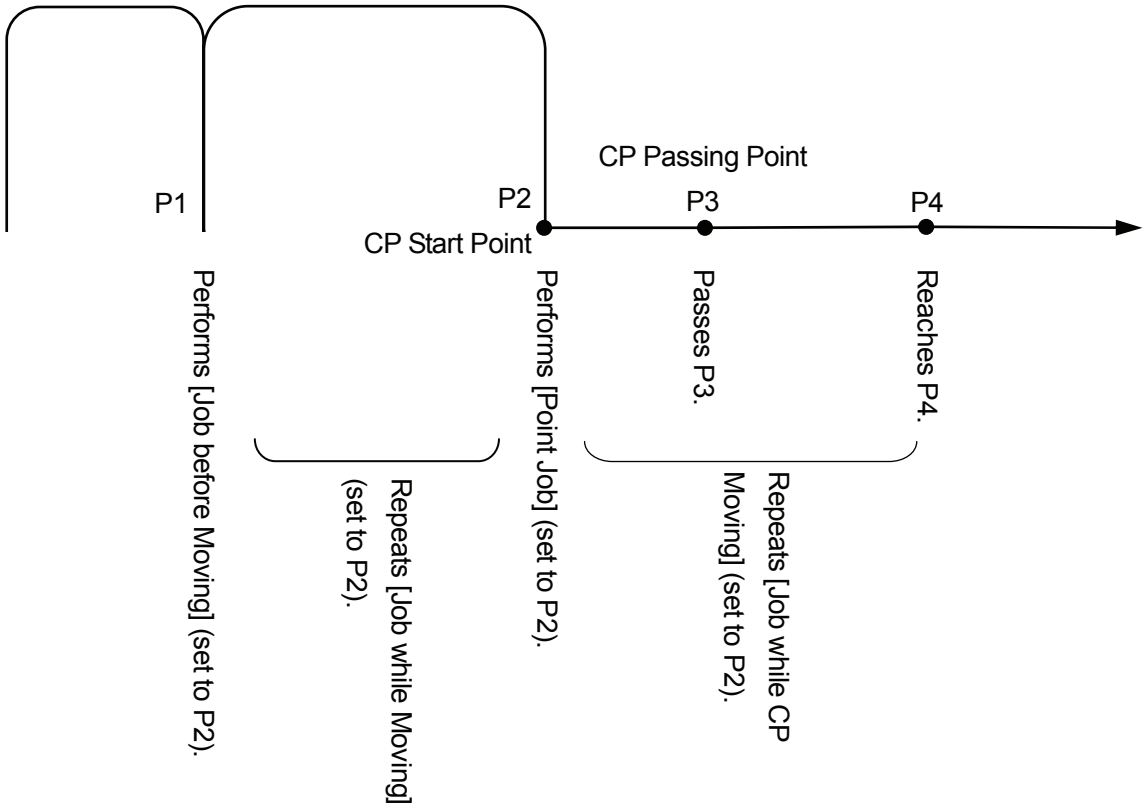
- The [Workpiece Adjustment] set to a [CP Start Point] point is activated until the tool unit reaches a [CP End Point] point.

■ Point Coordinates: #point_X,#point_Y,#point_Z, #point_R,#point_TagCode

These variables hold the coordinates and tag code values of the running point. A *running point* is the point to which point job data including this variable is set. When point job data including this variable 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 the [Job before Moving] includes these variables, the P2 coordinates are retained even at P1.

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

■ Designated Point Coordinates: #P_X, #P_Y, #P_Z, #P_R, #P_TagCode

These variables hold the coordinates and tag code values of the designated point in the current program.

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

Category	Identifier	Description
Given 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

■ Designated Point Coordinates in Designated Programs: #prog_P_X, #prog_P_Y, #prog_P_Z, #prog_P_R, #prog_P_TagCode

These variables hold the coordinates and tag code values of the designated point in the designated 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
Designated Point Coordinates in Designated Program	#prog_P_X (1 – 255, 1 – Last point number)	X coordinate value of the designated point in the designated program
	#prog_P_Y (1 – 255, 1 – Last point number)	Y coordinate value of the designated point in the designated program
	#prog_P_Z (1 – 255, 1 – Last point number)	Z coordinate value of the designated point in the designated program
	#prog_P_R (1 – 255, 1 – Last point number)	R coordinate value of the designated point in the designated program
	#prog_P_TagCode (1 – 255, 1 – Last point number)	Tag code value of the designated point in the designated program

FUNCTIONS

■ Robot System Functions

You can use the built-in variables (which are built into the robot system) and the user-defined variables (which can be freely defined by the user).
The user-defined variables other than local variables (variables effective only in defined point job data which are defined by the *declare* command) are defined in the Customizing mode. (See the operation manual *Features IV* for details of the Customizing mode.)
The functions built into the robot system are as follows:

Type	Identifier	Description
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	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	isConditionData (n)	Display whether the specified condition data number is available (1) or not (0).
str	strCenterLCD (a)	Adjust the strings on the teaching pendant LCD (centering).
str	strRightLCD (a)	Adjust 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: 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 drive [sec]
num	getSystemPTPrestTime ()	Valid only for [Job while Moving]. Time left before the current PTP drive ends (reaching the destination) [sec]

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

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

- **currentPointNumber()**
  This variable holds the currently performed point number. The point number of the work home position is 0.

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

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

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

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

- **numCOM(port#)**
  This variable holds the data byte count of COM receiving port.

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

- **strCenterLCD(string s)**
  This variable adjusts the strings on the teaching pendant LCD (centering).

- **strRightLCD(string s)**
  This variable adjusts the strings on the teaching pendant LCD (right justification). (Normally left-justified)

- **strPlusRLCD(string a, string b)**

  This variable adjusts the 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 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 drive (in seconds).
  Valid only for [Job while Moving].

- **getSystemPTPrestTime()**

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

■ Arithmetic System Functions

The following built-in arithmetic functions can be used:

x, y : Numeric value

n, m : Rounded integer value

Type	Identifier	Description
num	abs (x)	Absolute value
num	max (x,y)	Maximum value
num	min (x,y)	Minimum value
num	degrade (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
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 Functions

The following string built-in functions can be used:

x, y: Numerical value or numerical variable

n, m: Numeric value becomes larger than a certain value through rounding or truncation

a, b: String or string variable

Type	Identifier	Description
str	chr (x)	Return a string (1 character) with the given character code.
num	ord (a)	Return the top character code. Other codes are ignored.
num	len (a)	Return the string length (non-multibyte).
num	strPos (a,b)	Return the first part string position in a matching b.
str	strMid (a,n,m)	Return the strings n – m counted from the top of the given string a.
str	str (x)	Convert a numeric value to a decimal digit string.
str	strBin (n,m)	Convert a numeric value to a binary string. m: Number of binary string digits
str	strHex (n,m)	Convert a numeric value to a hexadecimal string. m: Number of hexadecimal string digits
str	str1SI (x)	Round a numeric value to a 1-byte signed integer to convert it to a 1-byte string. (1-byte Signed Integer)
str	str2SIBE (x)	Round 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)	Round 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)	Round 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)	Round 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)	Regard a numeric value as a float to convert it to a 4-byte string using the Big Endian byte order. (4-byte Signed Float Big Endian)
str	str4FLE (x)	Regard a numeric value as a float to convert it to a 4-byte string using the Little Endian byte order. (4-byte Signed Float Big Endian)
str	str8DBE (x)	Regard a numeric value as a float to convert it to an 8-byte string using the Big Endian byte order. (8-byte Signed Float Big Endian)
str	str8DLE (x)	Regard a numeric value as a float to convert it to an 8-byte string using the Little Endian byte order. (8-byte Signed Float Little Endian)
num	val (a)	Regard a string as a decimal digit string to convert it to a numeric value.
num	valBin (a)	Regard a string as a binary string (sequence of “0”, “1”) to convert it to a numeric value.

x, y: Numerical value or numerical variable

n, m: Numeric value becomes larger than a certain value through rounding or truncation

a, b: String or string variable

Type	Identifier	Description
num	valHex (a)	Regard a string as a hexadecimal string (sequence of “0” – “1”, “A” – “F”, or “a” – “f”) to convert it to a numeric value.
num	val1SI (a)	Convert the top character to a 1-byte signed integer. (1-byte Signed Integer)
num	val2SIBE (a)	Convert 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)	Convert 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)	Convert 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)	Convert 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)	Convert the top 4 characters to a float using the Big Endian byte order. (4-byte Float Big Endian)
num	val4FLE (a)	Convert the top 4 characters to a float using the Little Endian byte order. (4-byte Float Little Endian)
num	val8DBE (a)	Convert the top 8 characters to a double-precision float using the Big Endian byte order. (8-byte Double Big Endian)
num	val8DLE (a)	Convert the top 8 characters to a double-precision float using the Little Endian byte order. (8-byte Little Big Endian)
num	valSum (a)	Return the sum of a string code from top to bottom.
num	valCRC (a)	Remainder of dividing a string (bit string) by a generator polynomial $X^{16}+X^{12}+X^5+1$
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

ON/OFF OUTPUT CONTROL

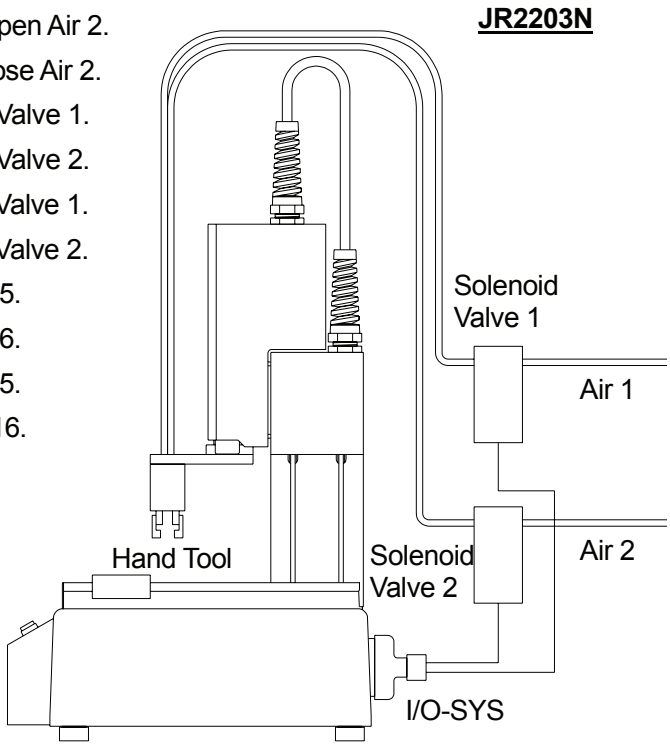
■ Output to I/O: set, reset, pulse, invPulse

This section explains the commands to be output to the tool unit (I/O). 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 designated output destination.
	reset	Output Destination		Output OFF to a designated output destination.
	pulse	Output Destination	Pulse Width	Output ON pulse of a designated width to a designated output destination.
	invPulse	Output Destination	Pulse Width	Output OFF pulse (inverting pulse) of a specified width to a specified output destination.

For example, connect the hand tool 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,

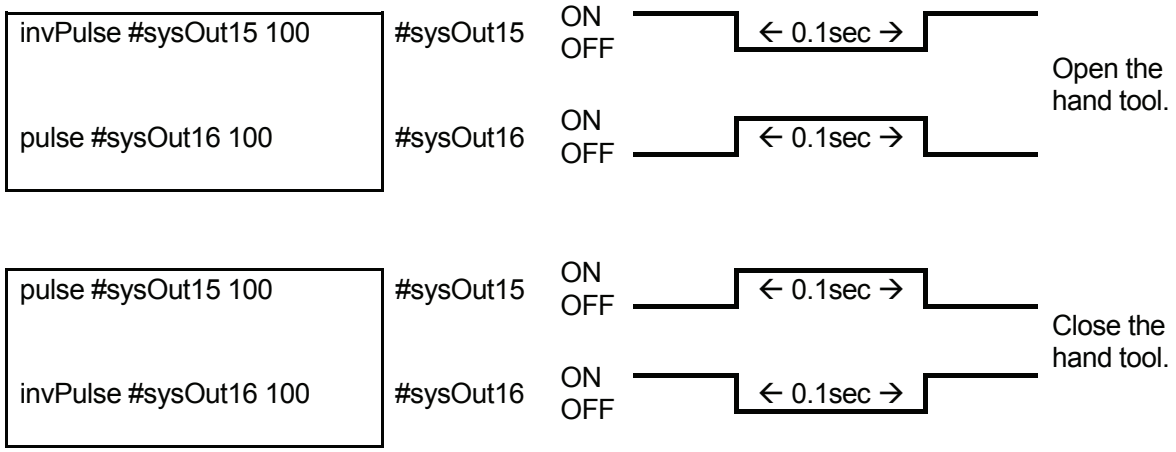
- The hand tool opens. ← Turn off #sysOut15 and on #sysOut16.
- The hand tool closes. ← Turn on #sysOut15 and off #sysOut16.

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

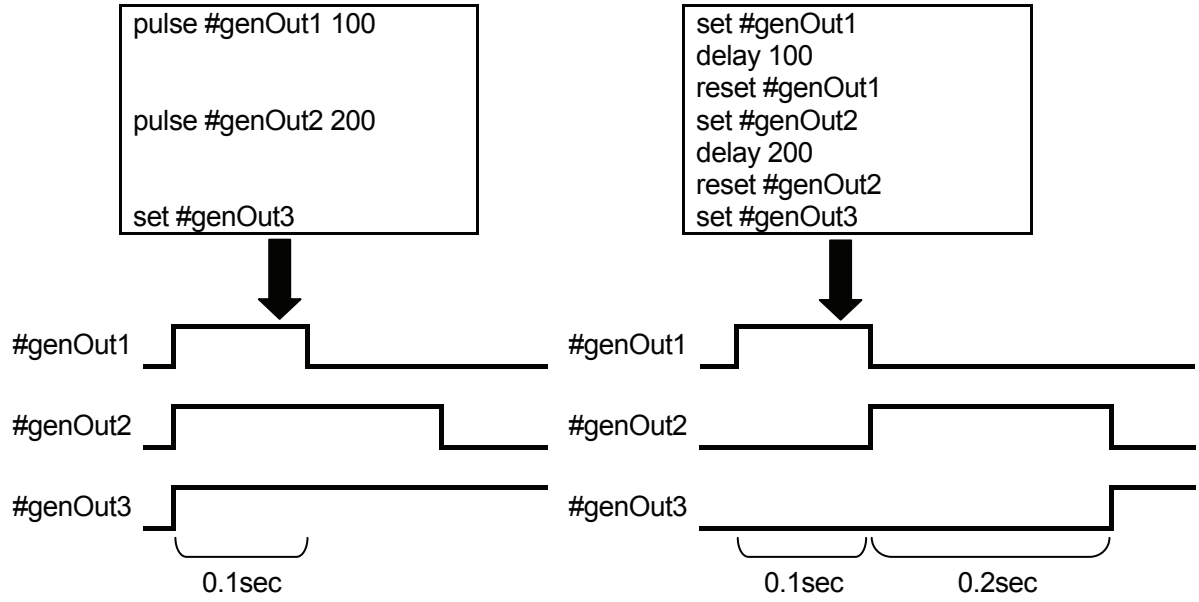
reset #sysOut15 set #sysOut16	Output #sysOut15 OFF. Output #sysOut16 ON.	→ Open the hand tool.
set #sysOut15 reset #sysOut16	Output #sysOut15 ON. Output #sysOut16 OFF.	→ Close the hand tool.

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

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



- The *pulse* and *invPulse* commands move on to the next command before completing output.
  For example, the following two kinds of point job data have different results:



  delay100 means “Stand by for 0.1 second at that point”.

- You can set the pulse width for the *pulse* and *invPulse* commands using variables or expressions.

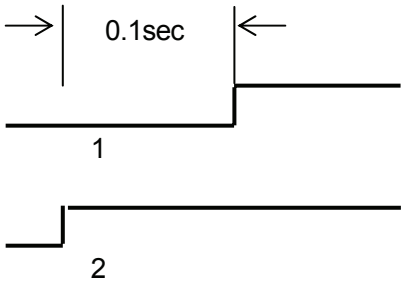
■ Output after X Seconds: delaySet, delayReset

The *delaySet* and *delayReset* commands are used to output ON/OFF signals to a designated output destination after a designated time.
The delay time can be set 0.001sec – 9999.999sec.

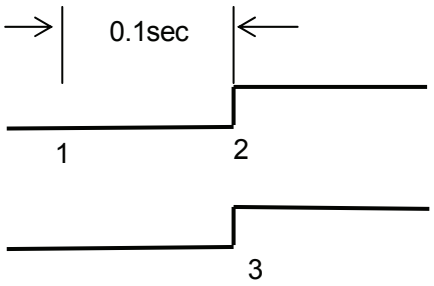
Command Category	Command	Parameter		Job
ON/OFF Output Control	delaySet	Output Destination	Delay Time	ON output after specified delay time
	delayReset	Output Destination	Delay Time	OFF output after specified delay time

The *delaySet* and *delayReset* commands move on to the next command before completing output. If signals are output by *set* or *reset* commands after *waitCondTime*, the next command execution timing is different as follows:

e.g. delaySet
1. delaySet #sysOut2 100
2. set #sysOut1
3. …….



e.g. waitCondTime/set
1. waitCondTime 100
2. set #sysOut1
3. set #sysOut2
4. …….



- You can set the delay time using variables or expressions.

■ Sound a Buzzer: onoffBZ

You can sound a buzzer using a point job command.

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

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

- You can set [ON Time] and [OFF Time] for the *onoffBZ* command using variables or expressions.

■ Blink the LED (Green): onoffGLED

- The commands below are available only for the JR2000N and JSR4400N Series. The JS and JSG Series are not equipped with these commands.

You can turn on and off, or blink the LED light on the front body (JR2000N) or the operation box (JSR4400N) using a point job command.

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).

If the *set* or *onoffGLED* commands are executed, the green LED is on or blinking until the *reset* command is executed.

- You can set [ON Time] and [OFF Time] for the *onoffGLED* command using variables or expressions.

■ Blink the LED (Red): onoffRLED

- The commands below are available only for the JR2000N and JSR4400N Series. The JS and JSG Series are not equipped with these commands.

You can turn on and off, or blink the LED light on the front body (JR2000N) or the operation box (JSR4400N) using a point job command.

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).

If the *set* or *onoffRLED* commands are executed, the red LED is on or blinking until the *reset* command is executed.

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

■ Output Values from I/O: dataOut, dataOutBCD

Any numeric values 1 – 999,999,999 or tag codes 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 Data	Output Destination	Output Bit Number	Output values from the I/O.
	dataOutBCD	Output Data	Output Destination	Output Bit Number	Output values in BCD from the I/O.

- 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 widths can be set using variables or expressions.

You need to set the three parameters: the output value (value to be output), the output width (the number of I/O pins to be used for output; also referred to as [Output Bit No.]), and the output destination (the smallest number between I/Os to be used for output: for example, if you use #genOut8 – #genOut10, the output destination is [8]), for the *dataOut* and *dataOutBCD* commands.

- The *dataOut* and *dataOutBCD* commands require **serial** I/O pins for output.

Example:

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

- If the output value does not fall within the set output width, the upper digit will be 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)

- The output width ([Output Bit No.]) can be set up to [31]. However, different types of I/O pins cannot be combined.

■ Motor Power ON, Servo Motor ON and OFF: motorPowerON, servoON, servoOFF

- The commands below are available only for the JS and JSGN Series. The JR2000N Series is not equipped with these commands. The JSR4400N Series is equipped with only *motorPowerON*.

You can turn on the power to the robot's motor or turn on and off the designated Axis servomotor by using a point job command. If an Axis servomotor is off, the Axis cannot be controlled by the robot. When the X-, Y- and R-Axes servomoters are off, they can be moved manually.

- You cannot turn off the motor power using commands.

Command Category	Command	Parameter	Job
ON/OFF Output Control	motorPowerON	–	Turn on the power to the robot's motor.
	servoON	Axis	Turn on the designated Axis servomoter.
	servoOFF	Axis	Turn off the designated Axis servomoter.

IF BRANCH, WAIT CONDITION

■ if Branch: if, then, else, endIf

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

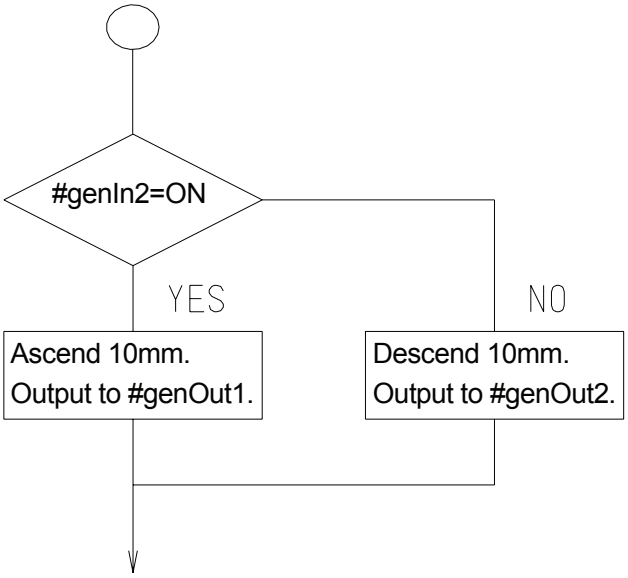
Command Category	Command	Parameter	Job
if Branch, Wait Condition	if	–	if Branch
	then	–	Execute the following command if true:
	else	–	Execute the following command if false:
	endIf	–	End of if Branch

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

For example, perform the following point jobs using the *if*, *then*, *else* and *endIf* commands:

Example 1

If #genIn2 is on, raise the Z-Axis by 10mm and output a pulse to #genOut1.
If #genIn2 is not on, lower the Z-Axis by 10mm and output a pulse to #genOut2.



The Example 1 above will be performed using the following point job commands:

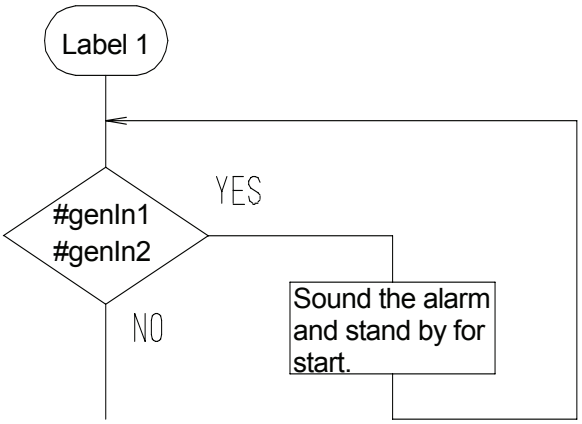
```
if
   ld #genIn2
then
   upZ 10,20
   pulse #genOut1,200
else
   downZ 10,20
   pulse #genOut2,200
endIf
```

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-Axis by 10mm at 20mm/sec, and
    Output ON pulse to #genOut1. (in 0.2sec widths).
If the condition is false, execute the following commands.
    Lower the Z-Axis by 10mm at 20mm/sec, and
    Output ON pulse to #genOut2 (in 0.2sec widths).
End of if Branch

Example 2

If both #genIn1 and #genIn2 are on, sound a buzzer and stand by until the start instructions come.

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



The Example 2 above will be performed using the following point job commands:

```
 Label 1
if
   ld #genIn1
   and #genIn2
then
   waitStartBZ
   jump L1
endIf
```

(A destination mark for the *jump* command)
If the following conditions are true, go to *then*. If false, go to the point next to *endIf*.
  #genIn1=ON (Condition 1)
  And #genIn2=ON (Condition 2)
If the conditions are true, execute the following commands:
  Sound the buzzer and stand by at the point until the start instructions come.
  Jump to [Label 1] when start instructions come.
End of If Branch

- 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.

- The *waitCondTime*, *timeUp … endWait*, and *if … endIf* command lines will be indented as follows:

```
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, it will be recognized as an error and the error message [Error on Point Job] will be displayed.

If *timeUp* or *endWait* precedes *waitCondTime* or if *then*, *else* or *endIf* proceeds *if*, it will also be recognized as an error and the message [Error on point job] will be displayed.

■ 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].

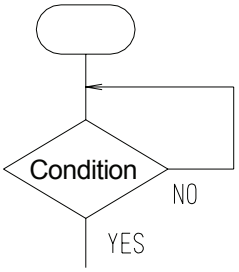
Command Category	Command	Parameter	Job
Wait Condition	waitCondTime	Wait Time	Wait the condition for a set period.
	timeUp	–	Execute when time is up.
	endWait	–	End of *wait* command
	waitCond	–	Wait the condition.

- The wait condition commands are deactivated 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.

Following are some examples where the wait condition commands are used:

■ ***waitCond … endWait*: Wait until conditions are met.**

e.g. A workpiece exists. → Sensor (#genIn2) ON
A workpiece does not exist. → 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*: Wait for the specified period of time until conditions are met.**

e.g. If the workpieces do not come within 30 seconds, it is recognized as an error, an external lamp (connected to #genOut2) comes on, and the robot stands by until a start signal comes.
To restart operation, fix the problem and press the start button.

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

Wait for 3 seconds until the following condition is met:
#genIn2=ON (Condition)
If the condition is not met within 3 seconds,
Output ON signal to #genOut2,
Stand by in place until a start signal comes.
Output OFF signal to #genOut2 when a start signal comes.
End of the line for commands if the condition is not met for another 3 seconds.

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

Example:

declare num wtime	Declare a local variable *wtime*.
if	If
ld #genIn3	#genIn3=ON
then	then
wtime = 3000	Assign 3000 to *wtime*.
else	If not
wtime = 1000	Assign 1000 to *wtime*.
endIf	
waitCondTime wtime	Wait for 3 or 1sec until the following condition is met:
ld #genIn2	#genIn2=ON (Condition)
timeUp	If the condition is not met within 3 or 1sec,
set #genOut2	Output ON signal to #genOut2,
waitStartBZ	Stand by in place until a start signal comes.
reset #genOut2	When a start signal comes, output an OFF signal to #genOut2.
endWait	End of the command line if the condition is not met within 3 or 1sec.

CONDITION

■ Condition Settings: ld, ldi, and, ani, or, ori, anb, orb

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

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

I/O-SYS input (#sysIn), I/O-SYS output (#sysOut), I/O-1 input (#genIn), I/O-1 output (#genOut), I/O-H input (#handIn), I/O-H output (#handOut), system flag (#sysFlag), internal relay (#mv), keep relay (#mkv), and pallet flag can be given as command parameters.
Comparison operation expressions can also be used. Variables and functions can also be used in comparison operation expressions as well as the above parameters.

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

Comparison 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
```

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

■ ***ldi*: OFF input**

```
waitCond
   ldi #genIn2
endWait
```

Wait 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
```

Wait 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

- *count* is a variable.

■ ***ani*: Series OFF input**

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

Wait 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
```

Wait 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
```

Wait 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
```

Wait in place until the following conditions are met:

- Count is 10 or greater or flag is ON — Condition 1
- #genIn1 is OFF and #genIn2 is also OFF — Condition 2

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
```

Waits in place until the following conditions are met.

- Count is 10 or greater or flag is ON — Condition 1
- #genIn2 is OFF and #genIn2 is also OFF — Condition 2

Either Condition 1 or 2 is true,
End of condition line

- When there is no *ld* or *ldi* corresponding *anb* or *orb*, the error message [Error on Point Job] is displayed.

DELAY, DATA IN, WAIT START

■ 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.

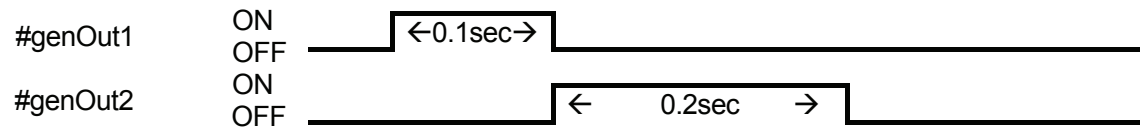
- The *delay* command is deactivated at points where the (base) point type [CP Passing Point] is set.

■ *delay*: Delay for the 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 num wtime	Declare the local variable *wtime*.
if	If
ld #genIn1	#genIn1=ON
then	then
wtime = 100	Assign 100 to *wtime*.
else	If not
wtime = 200	Assign 200 to *wtime*.
endIf	
set #genOut1	Output ON signal to #genOut1.
delay wtime	Delay for 0.1 or 0.2sec.
reset #genOut1	Output OFF signal to #genOut1.

■ Waiting for a Start Signal: waitStart, waitStartBZ

This section explains the point job data commands to stop the robot until a start signal comes.

Command Category	Command	Parameter	Job
Delay, Data In, Wait Start	waitStart	–	Stand by in place until a start signal comes.
	waitStartBZ	–	Stand by in place while sounding a buzzer until a start signal comes.

- The *waitStart* and *waitStartBZ* commands are deactivated at points where the (base) point type [CP Passing Point] is set.

■ *waitStart*: Wait for start

Example:

set #genOut1	Output an ON signal to #genOut1.
waitStart	Stand by in place until a start signal comes.
reset #genOut1	Output an OFF signal to #genOut1 (if a start signal comes).

■ *waitStartBZ*: Wait for start (with buzzer)

Example: If #genIn1 does not come on within 2 seconds, it is recognized as an error, #genOut2 (connected to an external alarm or alarm lamp) comes on, and the robot **stands by for a start signal while sounding a buzzer**.
When the operator fixes the problem and sends a start signal, #genOut2 goes off and the operation will restart from Point 05.

waitCondTime 2000 ld #genIn1	Wait until #genIn1 comes on for 2 seconds.
timeUp upZ 50,20 set #genOut2 waitStartBZ reset #genOut2 goPoint PTP3,5 endWait	If #genIn1 does not come on within 2 seconds, Raise the Z-Axis 50mm (at 20mm/sec), Output ON signal to #genOut2, Sound a buzzer and stand by in place until a start signal comes. Output OFF signal to #genOut2 (when a start signal comes), Go to Point 05. End of the command if #genIn1 does not come on within 2 seconds

- If the *waitCondTime*, *timeUp* … *endWait* and *if* … *endIf* commands are combined, the command lines are indented as shown 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, it will be recognized as an error and the error message [Error on Point Job] will be displayed.

When *timeUp* or *endWait* comes before *waitCondTime,* or if *then*, *else* or *endIf* comes before *if*, it will also be recognized as an error and the error message [Error on Point Job] will be displayed.

■ Input from I/O: dataIn, dataInBCD

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

Command Category	Command	Parameter			Job
Delay, Data In, Wait Start	dataIn	Numeric Variable Name	Input Source	Input Bit No.	Read out numeric data from I/O.
	dataInBCD	Numeric Variable Name	Input Source	Input Bit No.	Read numeric data from I/O in BCD.

- BCD: Binary-Coded decimal
- Read out width can be set using variables or expressions.

The *dataIn* and *dataInBCD* commands require the following three parameters: a variable to which a loaded value is assigned, an input width (number of I/Os to be used for input), and an input width (the smallest number between I/Os to be used for input: If #genIn3 – #genIn10 are used, the input source is [3]).

- The *dataIn* and *dataInBCD* commands require the **serial** I/O pins for input.

Example:

```
declare numeric code
dataIn code,#genIn3,8
```

Declare a local variable *code*.
Read out data #genIn3 – #genIn10 (I/O-1) as a numeric value and assign it to *code*.

```
declare numeric code
dataInBCD code,#genIn3,8
```

Declare the local variable of “code.”
Read out data #genIn3 – #genIn10 (I/O-1) in BCD and assign it to *code*.

Status of I/O-1

#genIn3	#genIn4	#genIn5	#genIn6	#genIn7	#genIn8	#genIn9	#genIn10
OFF	OFF	OFF	ON	OFF	OFF	ON	OFF

Input width: 8 →

In the above I/O status, the following result will be returned:

For the *dataIn* command, the value of *code* is **18**.

For the *dataInBCD* command, the value of *code* is **12**.

Input Width can be set up to “31.” However, it cannot be extended to a different I/O.

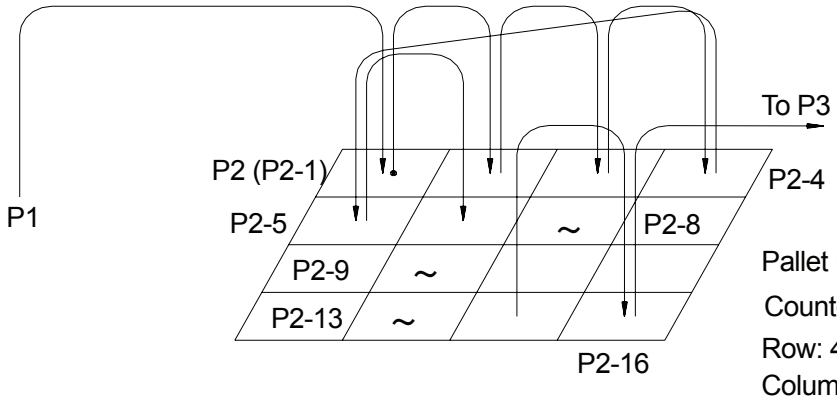
- The input width ([Input Bit No.]) can be set up to [31]. However, different types of I/O pins cannot be combined.

PALLET CONTROL

■ Pallet Command: loopPallet, resPallet, incPallet

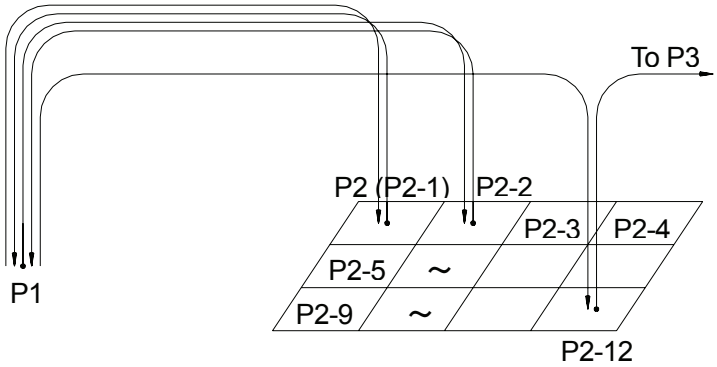
There are two types of additional function data [Pallet Routine]: One is [Auto Increment], which increases the counter automatically (the tool unit will move to the next point on the pallet sequentially). The other is [Increment by Point Job], which will not increase the counter (that is, the tool unit will not move to the next position on the pallet) unless you set the point job data to update the counter.
If you select [Auto Increment], you do not need to set a point job command to control the pallet operation. The tool unit will automatically move to the next point and update the pallet counter. However, on the [Auto Increment] pallet, the tool unit can only move in serial order P2-1, P2-2, P2-3 … as shown below.

Example: [Auto Increment] Pallet



Pallet Routine Type: Plane Pallet
Counter Control: Auto Increment
Row: 4
Column: 4

Example: [Increment by Point Job] Pallet



Pallet Routine Type: Plane Pallet
Counter Control: Increment by Point Job
Row: 4
Column: 3

On the [Increment by Point Job] pallet, the tool unit can move randomly, as shown on the previous page. For example, the tool unit returns to P1 each time before it moves to the next point. (P1 → P2 (P2-1) → P1 → P2-2 → P1 → P2-3…)

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

Command Category	Command	Parameter	Job
Pallet Control	loopPallet	Pallet Number, go Point Number	Add 1 to the pallet counter and if the counter reaches maximum, the tool unit will move to the specified point.
	resPallet	Pallet Number	Reset the counter to 0.
	incPallet	Pallet Number	Add 1 to the pallet counter.

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) reaches maximum = ON (true)
- The Pallet Counter (No. n) does not reach maximum = OFF (false)

[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 workpiece 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 workpiece.

Point Job Data (to be set to P2)

reset #genOut1 loopPallet 10,1	Releases (places) the workpiece. Add 1 to the counter of Pallet No. 10. If the counter reaches 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.

- The tool unit shifts (to P1 in the point job data for P2 above) using the *loopPallet* command according to the program data [PTP Condition].

If the *incPallet* command (Add 1 to the specified pallet counter) is used instead of the *loopPallet* command, the pallet control command will be as follows:

e.g. *incPallet* is used instead of *loopPallet*.

```
reset #genOut1
incPallet 10
if
   ld #palletFlag(10)
else
   goPoint   PTP3,1
endif
```

Release (Place) the workpiece.
Add 1 to the counter of Pallet No. 10.
If
The counter of Pallet 10 does not reach maximum,

Go to P1 (according to PTP Condition 03).

- If you use the *loopPallet* command, the tool unit shifts to the specified point according to the program data [PTP Condition]. If you use the *incPallet* command, you can use the *goPoint* or *goRPoint* command together, and you can also select the additional function data [PTP Condition].

If you use the *incPallet* command, another job (e.g. pulse output) can be performed each time the tool unit shifts to P1.

```
reset #genOut1
incPallet 10
if
   ld #palletFlag(10)
else
   pulse #genOut5,200
   goPoint   PTP0,1
endif
```

Release (Place) the workpiece.
Add 1 to the counter of Pallet No. 10.
if
The counter of Pallet 10 does not reach maximum,

Output a pulse and
Shift to P1.

The pallet number (and the go Point number for the *loopPallet* command) can be specified using expressions.

Example:

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

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

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

EXECUTION FLOW CONTROL

■ Subroutine Call for Jobs according to Point Types: callBase

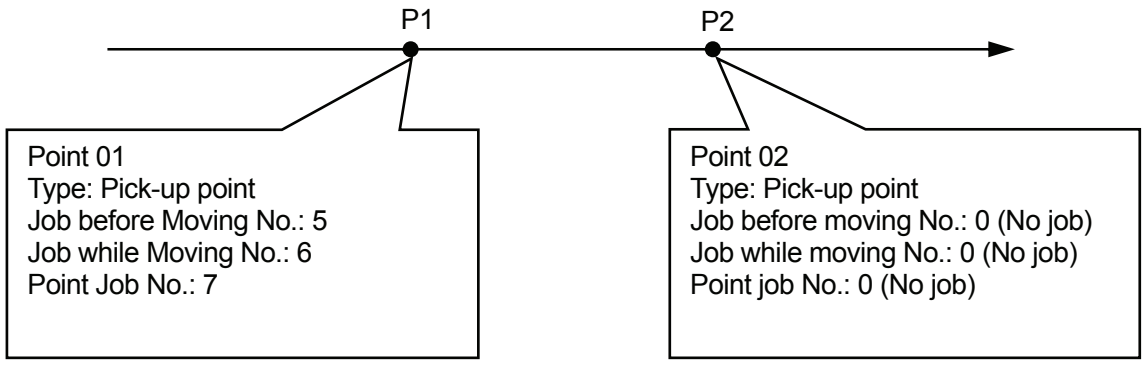
If you set point job data to a point where the user-defined point type created in the Customizing mode is already set, the point job set under the user-defined point type will not be performed.
For example, If you add a new point job to a point where the point type [Wait Start Point] is already set, the job originally set to the point will be ignored (the tool unit does not stand by until the start switch is pressed or a start signal comes on at the [Wait Start Point]) and the newly added job will be performed instead.

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

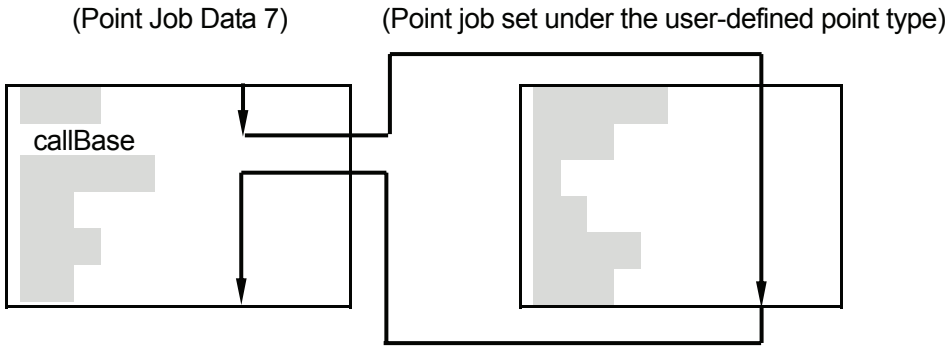
Title	:Pick-up point
Base type	:PTP Point
Job before Moving	:Yes
Job while Moving	:Yes
Point Job	:Yes

(P1) Job before Moving : Point job data 5
Job while Moving : Point job data 6
Point Job : 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
Point Job : [Point Job] set under the user-defined point type



In this case, if you execute the *callBase* command in a newly added job, the original job set under the user-defined point type can be called as a subroutine and be performed.
For the example on the previous page, if you execute the *callBase* command in Point Job Data 7, the original job set under the user-defined point type can be called as a subroutine when a new point job is performed at P1.



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

- The *callBase* command is deactivated at points where the (base) point type [CP Passing Point] is set.

The job command string set under the user-defined point type can be called as a subroutine by the *callBase* command. Therefore, if the *callBase* command is executed in a [Job before Moving], [Job while Moving], and [Job while CP Moving], the command string for each job will be called as a subroutine.
In the example on the previous page, if the *callBase* command is used in Point Job Data 5, the command string for [Job before Moving] set under the user-defined point type will be called when [Job before Moving] set to P1 is executed.

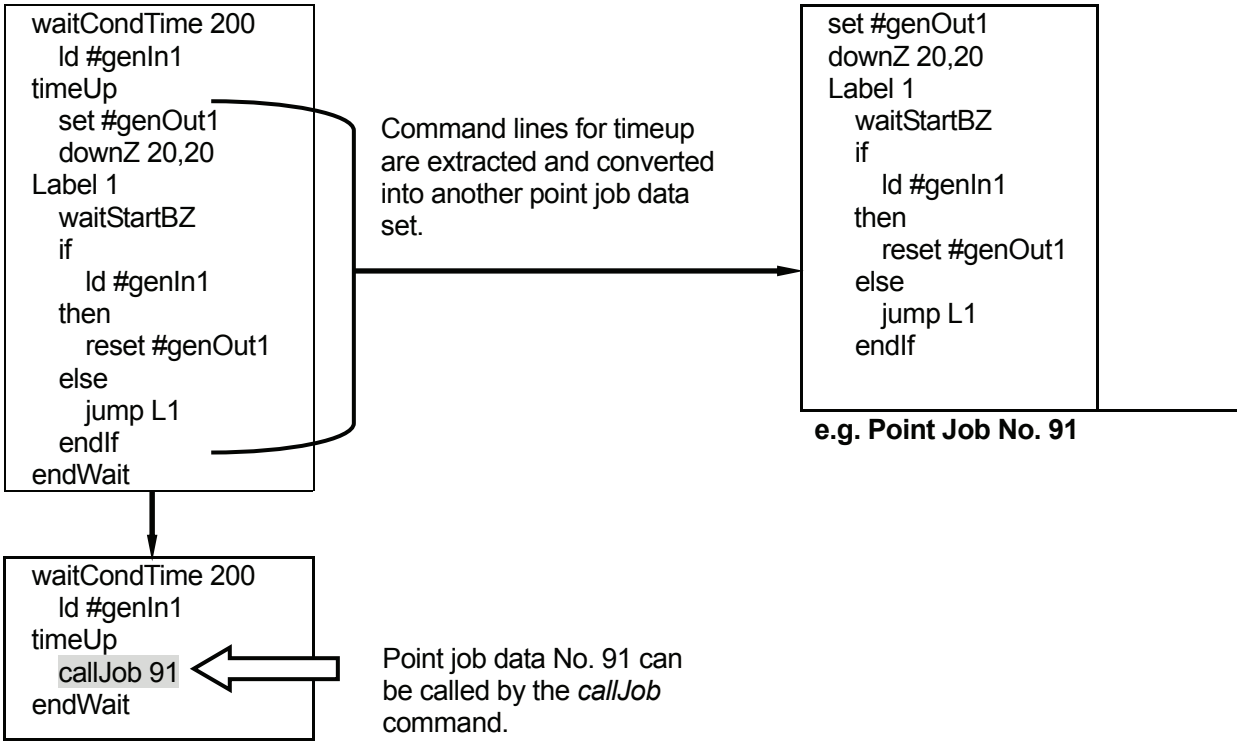
■ Subroutine Call for Point Job Data: callJob

While performing a point job, another point job data can be called and executed as a subroutine.

A point job can be clearer and easier if you create a specific job common to multiple point job data sets (e.g. error handling) as a point job data set and call it when necessary.
Also, if you extract command lines from a point job data set and create them as another point job data set, you can check a part of the point job data.

Command Category	Command	Parameter	Job
Execute Flow Control	callJob	Point Job Number	Call a subroutine of the point job data for the given number.

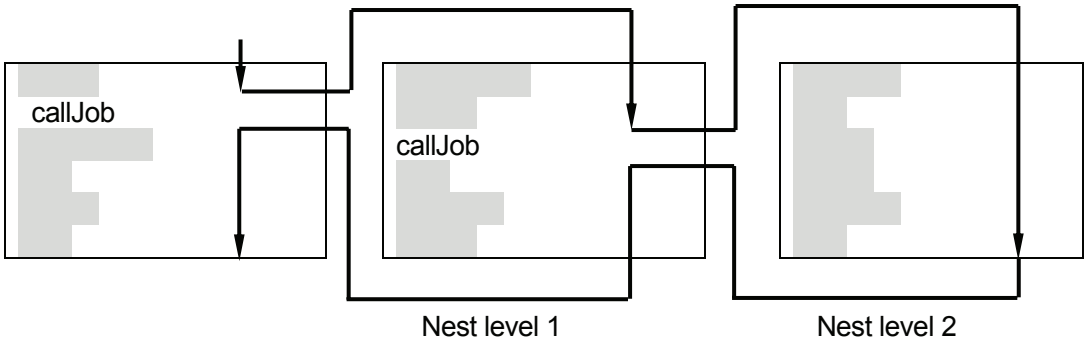
- The *callJob* command is deactivated at points where the (base) point type [CP Passing Point] is set.



After the called point job data (No. 91 in this example) is complete, the next command line of the *callJob* command (*endWait* in this example) in the calling point job data is performed.

- When the point job data called by the *callJob* command contains a *callJob* command, an error (No. 042: [Job for callJob doesn't exist]) is returned if the nest level exceeds Level 10. (The following example shows the nest level 2.)

Command Execution Flow



Point job data numbers can also be given using expressions.

Example:

```
declare num ejob
waitCondTime 200
   ld #genIn1
timeUp
if
   ld #genIn2
then
   ejob = 9
else
   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
Assigns 9 to *ejob*.
If not,
Assigns 10 to *ejob*.

Call the point job data No. 9 or 10 as a subroutine.

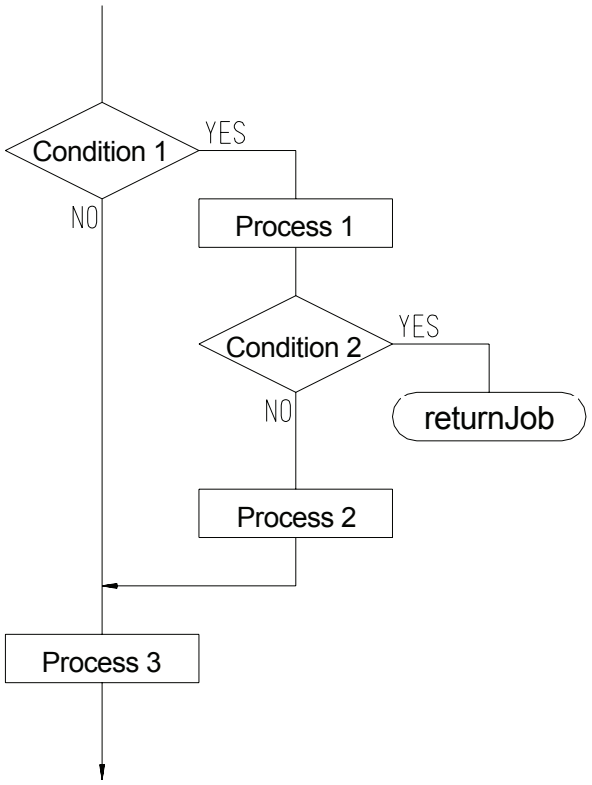
■ End the Point Job: returnJob

If a condition in the point job data is complex and there is no process to meet the condition, the point job can be ended by the *returnJob* command.

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

The following example shows point job data using the *returnJob* command:

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



Note: Once registered, if the *returnJob* command is removed from the point job data, Process 3 will be performed even if Condition 2 is ON (YES).

■ Subroutine Call for 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 the specified program number as a subroutine.

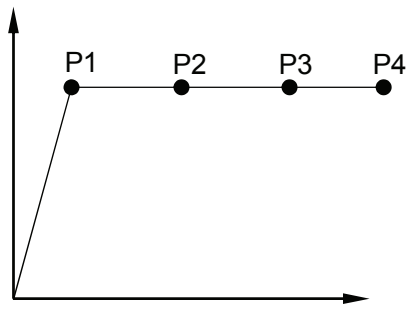
- The *callProg* command is deactivated at points where the (base) point type [CP Passing Point] is set.

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

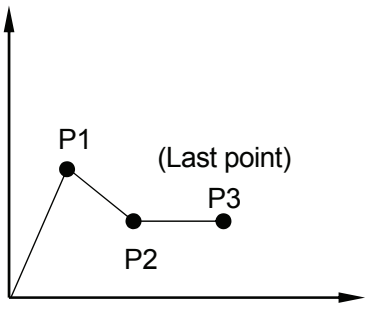
Program No. 17 can be called by the *callProg* command.

After the called program (No. 17 in this example) is complete, the next command line of the *callProg* command (*endWait* in this example) in the calling point job data is performed.

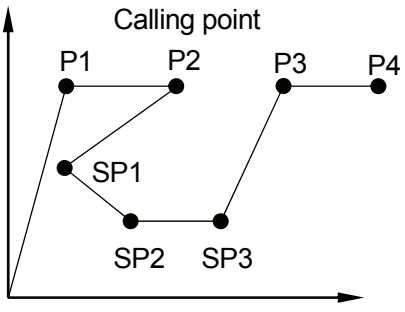
- The called program (hereinafter referred to as *subprogram* in this manual) will be performed as a [1 Cycle Playback] program even if [Continuous Playback] is selected as [Cycle Mode] settings in the [Program Data Settings] menu. The tool unit does not return to the work home position after running the last point of the called program, as shown below.
  (SP1: Subprogram Point 1)



Program 5



Program 12



Program 12 was called from P2 in Program 5.

Program numbers can also be given using expressions.

Example:

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

Declare a 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
  Assign 9 to *eprg*.
If not,
  Assign 10 to *eprg*.

Call the program No. 9 or 10 as a subroutine.

■ Position Data Settings

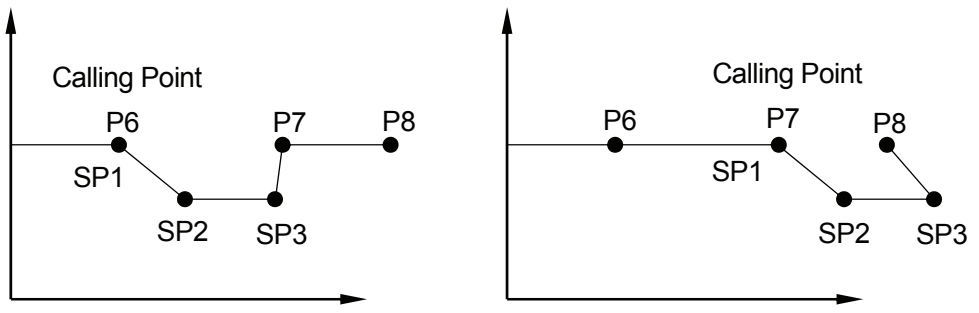
Point coordinates definition (position data settings) in the point data can be set in [Position Data Types] in the [Program Data Settings] menu. There are the following three position data types:

- Absolute: The position data value is equal to **the robot's absolute coordinates**. (Default)
- Relative: The position data is equal to **the distance from the program start coordinates to the current position coordinates**.
  (If the start coordinate is (0,0), it is equal to the [Absolute] setting.)
- Moving Amount: The position data value is equal to **the distance to the next point**.

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

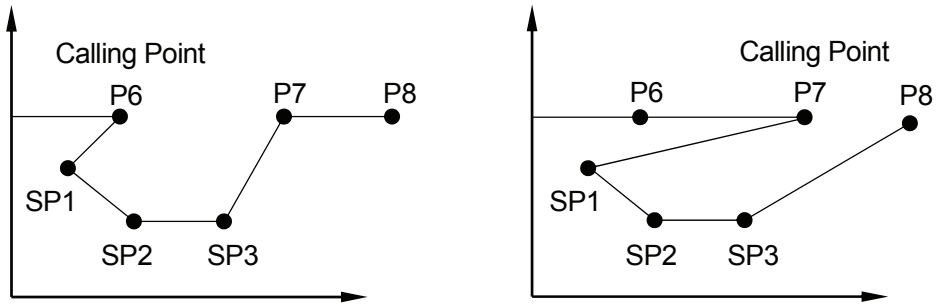
Example: The subprogram is set to [Relative] or [Moving Amount].

The current point (calling point) is handled as SP1 (Subprogram Point 1); but the position data of SP1 is not used. The work home position is not called.



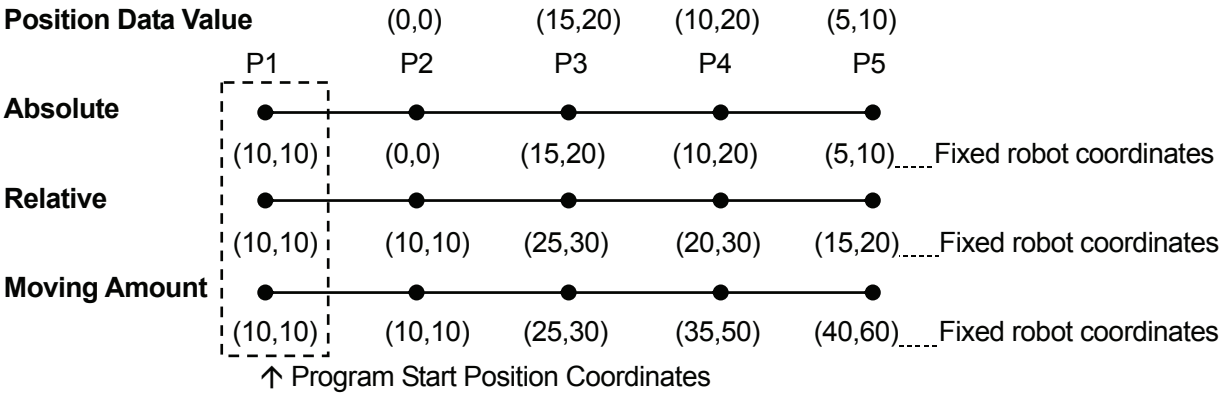
Example: The subprogram is set to [Absolute].

The tool unit runs on the coordinates of point data of the called point regardless of the position of the calling point. At the current point (calling point), the tool unit performs the [Job on Start of Cycle] (set to the work home position of the subprogram, and then shifts to SP1 (Subprogram Point 1).



- When a program called by the *callProg* command contains a *callProg* command, an error (No. 044: [Program for callProg doesn't exist]) is returned if the nest level exceeds Level 10.

Depending on the position data settings, the next point coordinates vary even if the position data values are the same. (See below)



- If you run a program as a subprogram, the robot Axis or Arm will not return to the work home. If [Relative] or [Moving Amount] is set in a program, the robot Axis or Arm will also not return to the work home.
  The robot Axis or Arm returns to the work home only when [Absolute] is set in the program and is performed independently, not by a *CallProgram* command.

■ How to Register a [Relative] Program

When registering any point in the JOG mode, the position data setting must be set to [Absolute].

When creating a program which is set to [Relative], shift (offset) all the points so as to match the coordinates of the first point to the position (0,0,0) after registering the desired point.

■ How to Register a [Moving Amount] Program

You cannot convert the registered coordinates into [Moving Amount]. Register any points under the [Moving Amount] setting in the MDI mode.

■ Subroutine Call for a Point String: callPoints

A point string (also referred to as an *array of points* or *series of points*, defined in the Customizing mode) with an identifier can be called and executed as a subroutine.

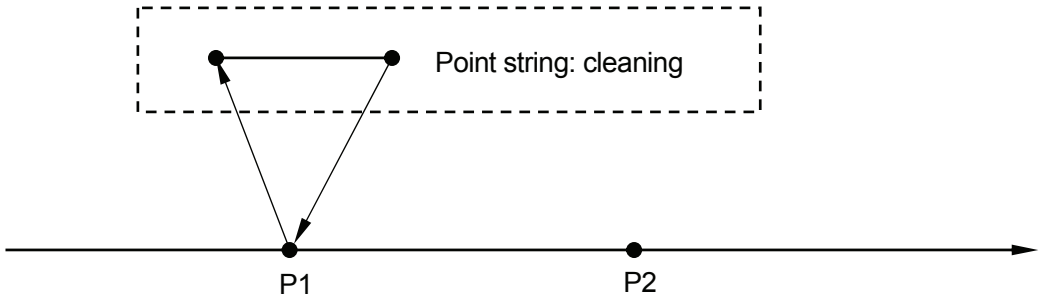
Command Category	Command	Parameter	Job
Execute Flow Control	callPoints	Variable Name (Identifier)	Call the specified point string as a subroutine.

- The *callPoints* command is deactivated at points where the (base) point type [CP Passing Point] is set.

For example, set the point job data shown below to P1.

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

A point string cleaning can be called by the *callPoints* command.



If #genIn1 is ON, go to the point string *cleaning* and execute point job data and additional function data set to the point string *cleaning*. Then go to P1, lower the Z-Axis by 10mm and output the ON signal to #genOut1.
If #genIn1 is OFF, lower the Z-Axis by 10mm and then output the ON signal to #genOut1.

■ End a Program: endProg

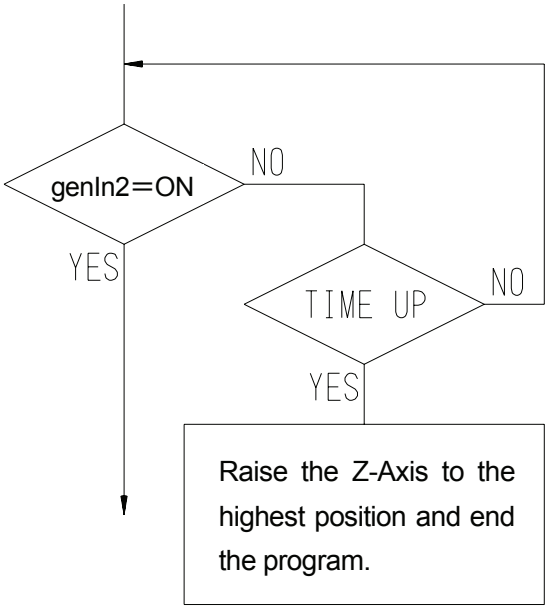
A program (operation) can be terminated at the current point by the *endProg* command. The robot Arm or Axis will not return to the work home position

Command Category	Command	Parameter	Job
Execute Flow Control	endProg	–	End a program run at the current point.

- The *endProg* command is deactivated at points where the (base) point type [CP Passing Point] is set.

The following example shows point job data using the *endProg* command:

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



By the *endProg* command, a program is terminated at the current point and the robot Arm or Axis will not return to the work home position. You cannot restart the program from that position. Start the program from the first point.

- If you wish to return to the work home position before terminating the program, use the *goPoint* command with a destination number [0] (work home position). (Refer to Page 72)

■ Assigning the Returned Value of a Function: returnFunc

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

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

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

Point Job Data

```
outLCD 7,4,radians(x)
```

⇦ The function can be called.

Function (Identifier: radians)

```
returnFunc 0.017453*x
```

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

■ Jump to the Specified Point: goPoint, goRPoint, goCRPoint

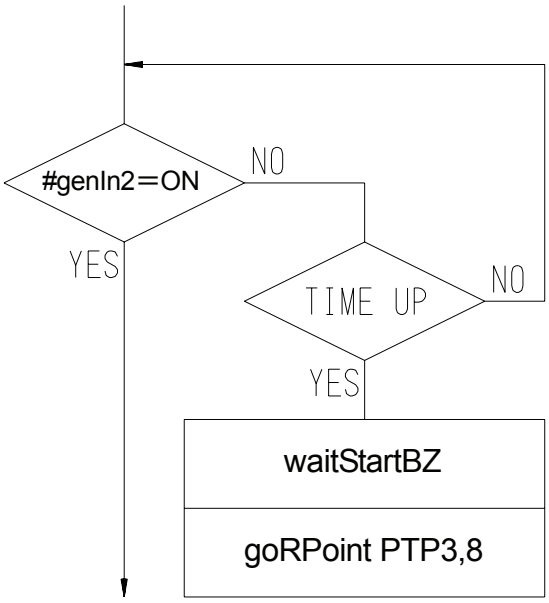
The following explains how to jump to a specified point after carrying out a point job instead of going to the next point.

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

- The *goPoint*, *goRPoint* and *goCRPoint* commands are deactivated at points where the (base) point type [CP Passing Point] is set.
- The point number and relative point number for the goPoint and goRPoint can be given using variables or expressions. If you wish to set the destination for the *goCRPoint* command using variables or expressions, the value must be either [0] or [1].

The following example shows point job data using the *goRPoint* command:

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



If #genIn2 is not turned on within 0.5 seconds, sound the buzzer and wait until a start signal comes. After receiving a start signal, restart operation from **8 points ahead (plus 8 points) of the current point**.

Example

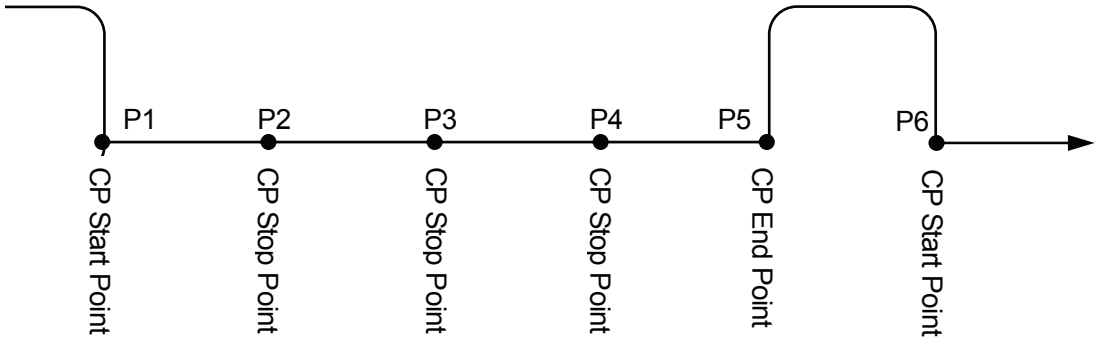
[goPoint PTP3,25]: Jump to Point 25 (according to PTP Condition 03).
If you set [0] as the [PTP Condition Number], the tool unit will move according to the program data [PTP Condition].
If you set [0] as the [Point Number], the tool unit will return to the work home position. (Jumps to a specified point number.)

[goRPoint PTP3,-4]: Jump to 4 points behind (minus 4 points from) the current point (according to PTP Condition 03).
If you set [0] as [PTP Condition Number], the tool unit will move according to the program data [PTP Condition].
If you set [0] as [Relative Point Number], the tool unit will restart operation from the same point. (Jumps to a relatively specified point number.)

[goCRPoint PTP3,1]: This command is used to jump to the specified point during the CP drive.
After a cycle of operation, from [CP Start Point] to [CP End Point], the tool unit will return to the point where the current CP drive is started ([CP Start Point] if the destination number is set to [0]. If [1] is set, the tool unit will move to the next point of the [CP End Point] (according to PTP Condition 03).
If you set [0] as [PTP Condition Number], the tool unit will move according to the program data [PTP Condition].

For example, if this command is executed between P1 and P5, the tool unit will move according to the destination number as follows:

Destination 0: Shifts to P1.
Destination 1: Shifts to P6.

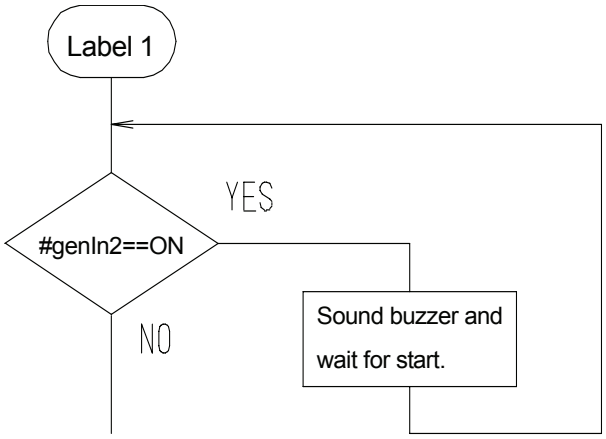


■ Jumping to a Specified Command Line: jump, Label

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

Example

If #genIn2 is ON, sound the buzzer and stand by until a start signal comes.
If #genIn2 is not ON, go to the next job.




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

(Destination mark)
If the following condition is true, go to *then*. If not, go to the next command after *endIf*.
genIn2=ON
If the above condition is true, execute the following commands:
Sound a buzzer and stand by in place until a start signal comes.
Jump to [Label 1] (when a start signal comes).
End of if Branch

- The *Label* command cannot be set between *if … endIf* or *waitCondTime … endWait* command lines.

- The label number can be set from [Label 1] up to [Label 99].

FOR, DO-LOOP

■ 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* loop.
	do	–	Repeat commands between *do* and *loop*.
	loop	–	
	exitDo	–	Break from *do* loop.

■ *for … exitFor … next*

The *for* command specifies the number of repetitions.

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

Declare a local variable *ival*.
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 the *ival* becomes 8.

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

Declare a local variable *ival*.
The exitFor command breaks out of the *for … next* loop and go to the command after *next*.

Condition: If #genIn1=1, break out of the *for … next* loop even if the ival does not become 8 and go to the command after *next*.

The *for* command parameters: initial value, end value and step value, can be given using variables or expressions.

```
declare num loop
declare num ival
if
   ld #genIn1
then
   loop = 5
else
   loop = 10
endIf
for ival=1 to loop step 1
   (Contents to be repeated)
next
```

Declare a local variable *loop*.
Declare a local variable *ival*.
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 the *ival* becomes the same value (5 or 10) as the variable *loop*.

■ do … exitDo … loop

The *do … exitDo … loop* command lines are repeated until the *exitDo* command exists.

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

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

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

- The contents to be repeated can be put both before and after Condition.

Condition:
If #genIn1=1, break out of the *do … loop* loop and go to the next command of the *loop*.

- When the [for, do-loop] commands are used, an error (No. 046: [for, do Nesting Error]) is returned if the nest level exceeds Level 10.
- If the [for, do-loop] commands are set to the (base) point type [CP Passing Point] points as point jobs, the robot may be stopped due to there being too many loops.

MOVE

■ Move the Z-Axis Alone: upZ, downZ, movetoZ

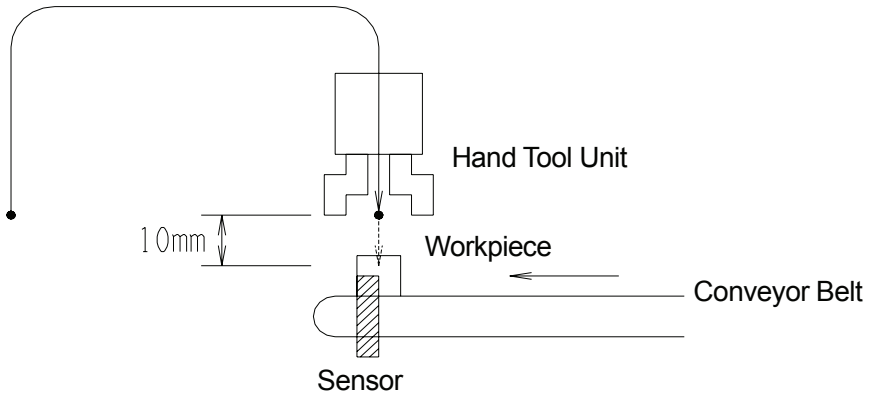
Only the Z-Axis can be raised or lowered using point job data. These commands belong to the [Move] command category.

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

- The [Move] commands are deactivated at points where the (base) point type [CP Passing Point] is set.

Example

In the PTP drive, stop the hand tool unit 10mm above the workpiece, check whether or not the workpiece is in place using a sensor, and then lower the tool slowly to hold the workpiece.



waitCond ld #genIn2 endWait downZ 10,20	Wait in place until the following condition is met: #genIn2=ON (Condition) End of condition Lower only the Z-Axis by 10mm (at 20mm/sec).

The distance and speed can be given by variables and expressions.

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

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

#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

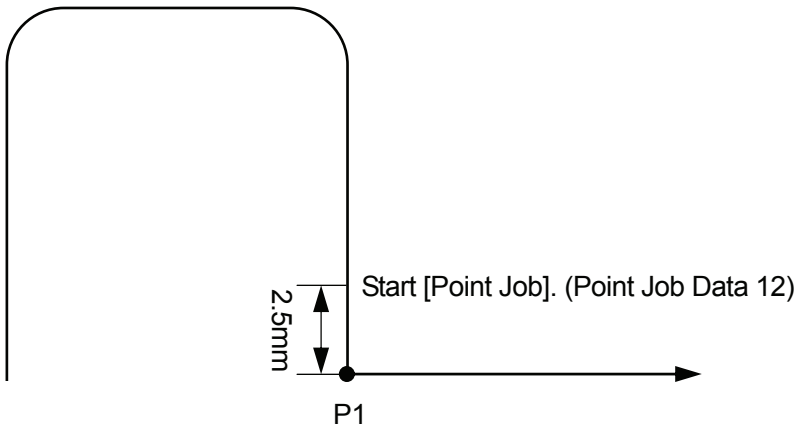
- If you assign a value to the variable *#jobStartHight* to point job data as [Job before Moving] or [Job while Moving] (using a *let* command), the robot starts the point job from a position higher than the set Z-coordinate determined by the assigned value.

Example:

P01
Type: CP Start Point
Job before Moving: Point Job Data 3
Point Job: Point Job Data 12

Point Job Data 3

```
#jobStartHight 25
```



■ Linear Movement in CP Drive by Point Job: lineMove, lineMoveStopIf

The robot Axis or Arm can move linearly in the CP drive using point job data commands.
The moving speed (referred to as *CP speed*) and the moving amount in each Axis direction can be set.
The CP drive can be terminated in the middle of the movement by setting conditions.

Command Category	Command	Parameter	Job
Move	lineMove	(CP) Line Speed X Distance Y Distance Z Distance R Rotate Angle	Move by the entered distance in CP drive.

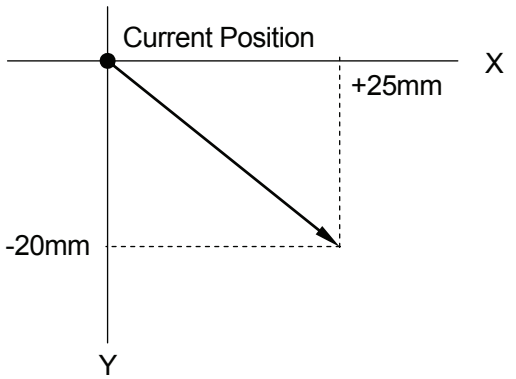
- The [Move] commands are deactivated at points where the (base) point type [CP Passing Point] is set.

To set each Axis distance, enter the distance from the current point to the destination point you wish to shift to. Enter [0] as the distance if you wish to move the Axis in that direction.
The distance can also be entered using a variable or an expression.

The point job data shown to the right is an example of linear movement in the CP drive using the *lineMove* commands.

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

Commands for each Axis are displayed separately as shown above; however, all of the Axes will move together at one time.



■ How to Stop the Movement in the Middle of the CP Drive using Conditions

Example:

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

Condition to stop in the middle of the movement

If #sysIn1 comes on, the robot Axis stops moving and goes to the next command (callJob11), even before the Z-Axis moves 20mm in the plus direction.

If the movement in the CP drive is terminated by conditions, you can check whether or not the robot Axis has moved the specified distance before it was stopped by referring to the system flag (#sysFlag34).

0: The robot Axis has completed moving the specified distance in that direction.

1: The robot Axis has completed moving in that direction with conditions.

■ If the Robot Axis Exceeds the [Move Area Limit] defined by the *lineMove* Command

If the Arm exceeds the [Move Area Limit] by the lineMove command, it will stop at the position where it reaches the move area limit and then go to the next command.

By referring the system flag (#sysFlag33), you can check whether the robot Axis has completed moving the specified distance before reaching the [Move Area Limit].

Complete: 0

Not complete: 1

[Wait Start] command: If the robot Axis reaches the [Move Area Limit] before moving the specified distance, sound a buzzer and stop until a start signal comes.

```
lineMoveSpeed 3
   lineMoveX 20
   lineMoveY 0
   lineMoveZ 0
   lineMoveR 0
endLineMove
if
   ld #sysFlag33
then
   waitStartBZ
endIf
```

■ Mechanical Initialization by Point Job: initMec

- The command explained below is available for the JR2000N and JSR4400N Series only. It is not available for the JS and JSG Series. If you are using the JSR4400N Series, this command is activated only when [Axis] (specifications) is set to [ALL] (all Axes).

Mechanical initialization is performed when the robot is turned on. You can initialize only the desired Axis using a point job command. With the *initMec* command, the robot Axes are able to return to the absolute coordinates (x: 0, y: 0, z: 0, r: 0) even if a position error has occurred.

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

- The [Move] commands are deactivated at points where the (base) point type [CP Passing Point] is set.

Axis Specification	Contents
all	Initialize all Axes.
x	Initialize the X-Axis.
y	Initialize the Y-Axis.
z	Initialize the Z-Axis.
r	Initialize the R-Axis.

- Mechanical initialization is performed at low speed.

■ Position Error Detection: checkPos

- The command explained below is available for the JR2000N Series only. It is not available for the JSR4400N, JS, and JSG Series.

Position errors can be detected using point job commands.
When the *checkPos* command is executed, the robot Axis 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 Axis goes to the next point.

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

- The [Move] commands are deactivated at points where the (base) point type [CP Passing Point] is set.

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
```

Perform a position error check.
If
 a position error is detected,

 sound a buzzer and wait until a start signal comes.

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

LCD, 7SLED

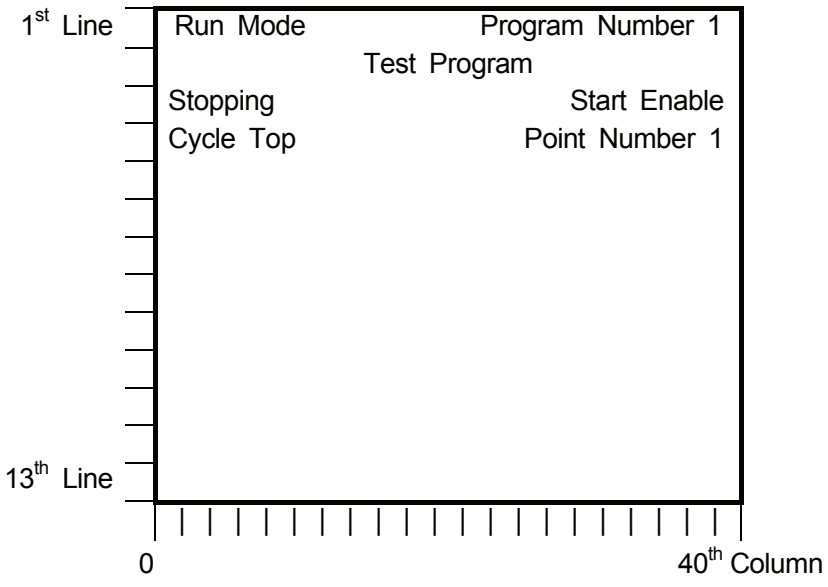
■ Display the Specified Strings on the Teaching Pendant: clrLCD, clrLineLCD, outLCD, eoutLCD

The following explains how to display/not display entered items on the teaching pendant LCD.

Command Category	Command	Parameter	Job
LCD Control	clrLCD	–	Clear the LCD display.
	clrLineLCD	Clear Line	Clear the specified line on the LCD display.
	outLCD	Display Line, Display Column, Display Data	Display the entered strings at the specified position on the LCD display.
	eoutLCD	Display Line, Display Column, Display Data	Display the evaluation of the entered string expression at the specified position on the LCD display.

- Rows or columns can be specified using variables or expressions. For the *eoutLCD* command, the displayed strings can be specified using string expressions.
  outLCD 7,4,"PULSE": Display the string *PULSE* on the teaching pendant LCD.
  eoutLCD 7,4,#sv(24) + #sv(25): Display the combined value of the string 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 row.



■ Display the Desired Number on the 7SLED: sys7SLED, out7SLED

- The command explained below is available for the JR2000N and JSR4400N Series only. It is not available for the JS and JSG Series.

Using the *out7SLED* command, you can display the desired number on the 7-segment LED (program number display) on the front of the robot (JR2000N) or the operation box (JSR4400N).
The number display will return to the program number specified by the *sys7SLED* command.

Command Category	Command	Parameter	Job
LCD Control, 7Seg LED	sys7SLED	–	Switch the number displayed by the *out7SLED* command to the program number previously displayed.
	out7SLED	Output Type Output Value	Output the specified number to 7-segment LED.

Select from the following four output types:

Output Type	Description
Num	Display the specified number on the 7-segment LED.
Er	Display the code *Er* and the specified number alternately on the 7-segment LED.
St	Display the code *St* and the specified number alternately on the 7-segment LED.
Ur	Display the code *Ur* and the specified number alternately on the 7-segment LED.

The output value can be specified using variables and expressions.

Example:

out7SLED Num,10:	Display a numeric value *10*.
out7SLED Er,20:	Display the code *Er* and a numeric value *20* alternately.
out7SLED St,nMyNum:	Display the code *St* and a value of the variable *nMyNum* alternately.

COM INPUT/OUTPUT

■ COM Input/Output: outCOM, eoutCOM, inCOM, setWTCOM, cmpCOM, ecmpCOM, clrCOM, shiftCOM

By point job data commands, data can be input or output from COM.

Command Category	Command	Parameter	Job
COM Input/Output	outCOM	Input/Output, Output Data	Output a character string from COM.
	eoutCOM	Input/Output, Output Data	Output an evaluation of the string expression from COM.
	inCOM	Variable Name, Input/Output, Character Length	Assign the data received on COM to a specified variable.
	setWTCOM	Input/Output, Wait Time	Set [Wait Time] (timeout period) for receiving data on COM.
	cmpCOM	Input/Output, Compare Data	Compare the received data on COM with a character string. The result is entered into the system flag (sysFlag1 – 20).
	ecmpCOM	Input/Output, Compare Data	Compare the received data on COM with a string expression. The result is entered into the system flag (sysFlag1 – 20).
	clrCOM	Input/Output	Clear the COM port receive buffer.
	shiftCOM	Input/Output, Shift Number	Shift the data received on COM. Delete data from the top by [Shift Number].

■ COM Output: *outCOM, eoutCOM*

Up to a 255-character strings can be output from COM.

Select [outCOM] or [eoutCOM] and then select the desired COM port number on the Input/Output selection screen. The Output Data entry screen will appear. Enter the character string you wish to output and press the ESC key. (For the key operations on Character entry screen, see “Entering Characters and Formulas” in the *Teaching Pendant Operation* manual.) Enclose the character strings to be output in double quotes (“”) (See e.g.1) When outputting variables and formulas, do not use double quotes (“”) (See e.g.2)

e.g.1: eoutCOM port2,"ERROR" : Output the character string *ERROR*.

e.g.2: eoutCOM port2,#sv(24) & #sv(25) : Output a value combining character string variables #sv(24) and #sv(25).

For the *eoutCOM* command, characters can be specified in hexadecimal code using the % symbol (See e.g.3) However, if any character other than 0 – 9, A – F, or % comes after the % symbol, the % symbol is output as a character. (See e.g.4) If you wish to output the % symbol as a character when any of w0 – 9, A – F come after it, enter two % symbols (%%). (See e.g.5)

e.g.3: eoutCOM port2,"%0D%0A" : Output CR LF codes.
e.g.4: eoutCOM port2,"%G01" : Output a character string *%G01*.
e.g.5: eoutCOM port2,"%%300" : Output a character string *%300*.

■ **COM Input: *inCOM***

Received data from COM is assigned to a variable by a specified number of characters. If the received data is larger than the specified number of characters, characters counted from the top according to the specified number are assigned.

If the received data is less than the specified number of characters, the robot stands by for the time specified by the *setWTCOM* command, and then assigns the received data to a variable. If the *setWTCOM* command is not set, the robot stands by for 0.1sec.

- If point job data including any COM input command is set at a [CP Passing Point], the robot stands by for 0sec to receive data.

■ **COM Receive Data Comparison: cmpCOM, ecmpCOM**

The COM receive buffer (a place where received data is stored) and a specified character string are compared one by one starting from the top character. The comparison result is indicated by a system flag.

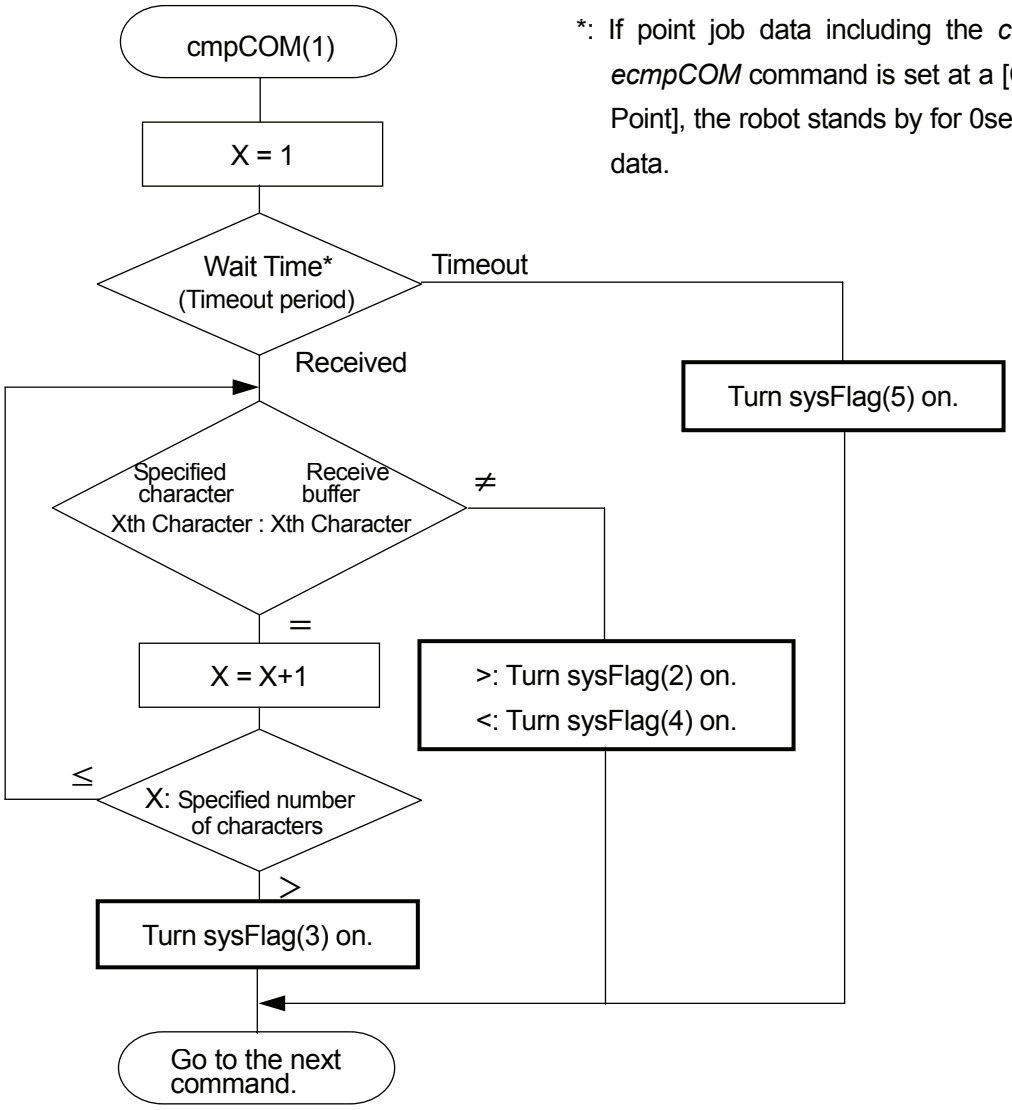
In cases when the data and the string are not equal, the comparison is completed, and data is not received after the specified wait time, the corresponding system flags will be turned on. (See table below)

You can set a wait time for receiving data using the *setWTCOM* command. If the *setWTCOM* command is not set, the robot stands by for 0.1sec.

When using the *ecmpCOM* command, a character string compared with a receive buffer can be specified using a string expression.

System Flags

	COM1	COM2	COM3	COM4 (TPU)
Specified Character > Receive Buffer	sysFlag(2)	sysFlag(7)	sysFlag(12)	sysFlag(17)
Specified Character = Receive Buffer	sysFlag(3)	sysFlag(8)	sysFlag(13)	sysFlag(18)
Specified Character < Receive Buffer	sysFlag(4)	sysFlag(9)	sysFlag(14)	sysFlag(19)
Timeout	sysFlag(5)	sysFlag(10)	sysFlag(15)	sysFlag(20)



*: If point job data including the *cmpCOM* or *ecmpCOM* command is set at a [CP Passing Point], the robot stands by for 0sec to receive data.

■ **COM Receive Wait Time Settings: *setWTCOM***

You can set the wait time (timeout period) for receiving data for the *cmpCOM* or *ecmpCOM* commands. If no data is received within the specified wait time, it will time out and the corresponding system flag will be turned on. The default is set to 0.1sec.

■ **COM 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 will not overwrite the existing data, but will be written after the existing data.

A receive buffer will be cleared by turning off the power or executing the *clrCOM* command.

■ **Shifting COM Receive Data: *shiftCOM***

A specified data byte in the receive buffer is deleted.

e.g. 2-byte Shift

Receive Buffer

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

	COM1	COM2	COM3	COM4 (TPU)
With received data	sysFlag(1)	sysFlag(6)	sysFlag(11)	sysFlag(16)

■ PC Communication: stopPC, startPC

■ Stopping and Starting COM1 Communication: *stopPC*, *startPC*

COM1 is normally used to communicate with a PC. If you wish to connect COM1 to devices to control the robot using point job commands, instead of communicating with a PC (for C & T data transmission), it is necessary to stop the PC communication transaction operated by the system.

If the *stopPC* command is executed, PC communication will not be established unless the power is turned off or the *startPC* command is executed.

- The *stopPC* command disables the C & T data transmission. To connect control devices, we recommend that you use alternative connectors rather than COM1.

VARIABLE, COMMENT, SYSTEM CONTROL

■ Variable Declaration and Assignment: declare, let

A variable that is activated only in point job data containing a *declare* command and a user-defined function (defined in the Customizing mode) is referred to as a *local variable*.
When a local variable is declared, it is necessary to set the variable type and the identifier. The identifier is used as a variable name and the variable type can be selected from either the numeric type or the string type.
A local variable can also be declared as an up to three-dimensional array.

The *let* command assigns the right-hand operand (numeric value, variable value, or evaluation of string expression) to the left-hand operand. When this command is input, only an expression is displayed.

Command Category	Command	Parameter	Job
Variable, Comment, System Control	declare	Variable Type, Variable Name	Local variable declaration
	let	Expression	Assign right-hand operand to the left-hand operand.

e.g. *declare* command

declare numeric abc declare string def	Numeric variable *abc* declaration String variable *def* declaration

e.g. *let* command

count = 0	Assign 0 to the variable *count*.
count = count + 1	Add 1 to the variable *count*.
count = in - out	Assigns the difference of the value of *out* subtracted from the value of *in* to the variable *count*.
total = nin * 365	Assigns the product of 365 multiplied by the value of *nin* to the variable *total*.
tsuki = total / 12	Assigns the quotient of the value of *total* divided by 12 to the variable *tsuki*.
fullname=name1 & name2	Assigns the string composed of *name1* and *name2* to the variable *fullname*.

Both of the following point job data items use a local variable *count*, but the two variables do not interfere with each other since a local variable is activated only in point job data containing a declare command. For example, if 0 is assigned to the variable *count* in point job data 24 (or 05), the value of the variable *count* in point job data 05 (or 24) will not change.

e.g. Point Job Data 05

```
declare numeric count
count=0
do
   count=count+1
   callJob 24
   if
      ld count>=10
   then
      exitDo
   endIf
loop
```

Numeric local variable *count* declaration
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.

e.g. Point Job Data 24

```
declare count
count=0
 Label 1
pulse #genOut11,250
count=count+1
if
   ld count<=3
then
   jump L1
endIf
```

Local variable *count* declaration
Set the initial value 0 to the variable *count*.
Label 1 (jump destination mark)
Output (Turn on) a pulse signal to #genOut11.
Add 1 to the variable *count*.
If
    the value of the variable *count* is less than 3,

    jump to Label 1.

■ Comment Insertion: rem, crem

You can add comments to point job data and sequencer data commands.

Command Category	Command	Parameters	Job
Variable, Comment, System Control	rem	Output Data	1 line comment
	crem	Output Data	End-of-line comment

e.g.

```
if
   ld #genIn1
   rem #genIn1: Obstacle sensor
then
   waitStartBZ
```

If
#genIn1 is true,
(#genIn1: Obstacle sensor): Comment

Sound a buzzer and stand by until a start signal comes.

e.g.

```
if
   ld #genIn1 crem #genIn1: Obstacle
   sensor
then
   waitStartBZ
```

If
#genIn1 is true, **(#genIn1:Obstacle sensor): Comment**

Sound a buzzer and stand by until a start signal comes.

- If you are using the teaching pendant, an end-of-line comment made by the *crem* command will be displayed in multiple lines as shown above if the comment does not fit into one line.

■ Change a Program Number by Point Job: setProgNum

The program number being selected can be changed using point job data commands. This function is useful in the following cases:

- If you set a *setProgNum* command to the point job data performed when the power is turned on, the same program number will always be activated every time the power is turned on.
- If you set a *setProgNum* command to the point job data performed after a point job at the work home position, the program number will automatically be changed.
  For example, if you wish to repeat Program 1 → Program 2 → Program 3 as a cycle of operation, set the *setProgNum2* command to the point job data performed after a point job at the work home position of Program 1. The program number will automatically change to Program 2 after running Program 1. Accordingly, you can change the program numbers, from 2 to 3 and from 3 to 1.
- If you set a *setProgNum* command to the point job data performed when the robot is standing by for a start signal, you can change the program number according to inputs from COM. For example, if you connect the barcode reader to COM, you can change the program number according to the barcode.

If you change a program number while running a program, the running program will not be changed immediately. After the robot runs the current program, the robot stands by for a start signal. Then the program number is changed when the robot restarts.
Use the *callProg* command if you wish to execute another program while running a program.

Command Category	Command	Parameter	Job
Variable, Comment, System Control	setProgNum	Program Number	Change the program number when the robot restarts running after standing by.

- Program numbers can be specified by variables or expressions.

■ Change a Sequencer Number by Point Job: setSeqNum

The selected sequencer number can be changed using point job data commands.
A complicated sequencer data set cannot be created because the number of commands for sequencer data is limited to up to 100 steps. However, you can create multiple sets of sequencer data that are performed such as when the power is turned on, the robot is standing by or running programs, and change the number using the *setSeqNum* command.
For example, if you set a *setSeqNum02* command to [Job on Start of Cycle] ([Job and Sequencer on Run Mode] menu) and a *setSeqNum01* command to [Job on End of Cycle] ([Job and Sequencer on Run Mode] menu), the sequencer data No. 2 will be executed during operation and the sequencer data No. 1 will be executed during standby.

If you change the sequencer number while running a program, the running sequencer data will not be changed immediately. After the robot runs the current program, the robot stands by for a start signal. The sequencer number is then changed when the robot restarts.

Command Category	Command	Parameter	Job
Variable, Comment, System Control	setSeqNum	Sequencer Number	Change the sequencer number when the robot restarts running after standing by.

- Sequencer numbers can be specified using variables or expressions.

Janome Sewing Machine Co., Ltd.

Industrial Equipment Sales Department

Postal Code: 193-0941
1463 Hazama-machi, Hachioji-shi, Tokyo, Japan
Tel: +81-42-661-6301
Fax: +81-42-661-6302

970803108 as of 2007-09

8 January 2009