Kingkong Command Manual - Common Commands

Copyright and Disclaimer

Contents

Introduction

About this manual

This article is a command manual for YiKingSoft software operators. This manual describes in detail the common Kingkong commands and functions used by Bat series, Python series and Mantis series models.

Operation Prerequisites

This manual is revised based on 2.8.9 (RU2). With different software versions, there will be different upgrades to the robot commands.

This manual is intended for individuals with experience in robot programming and who have already read the Kingkong Command Manual - Common Commands. If you have no programming experience, you need to carefully study the contents of the manual before performing programming tests.

Manual Usage

This manual should be read promptly during programming and whenever KingKong commands are needed.

Chapter Structure

This manual consists of the following chapters:

Chapter	Description
Code segment	Detailed descriptions of all code segments, including examples of how to use such code segments.
Logic	Detailed descriptions of all logic statements, including examples of how to use such logic.
Motion	Detailed description of all motion commands, including examples of how to use such functions.
Input/Output	Detailed description of all input and output commands, including examples of how to use such functions.
Settings	Detailed description of some control commands for all setting software, and examples of using such functions.

Chapter 1 Code Segment

1.1 Main Program - Create the main program

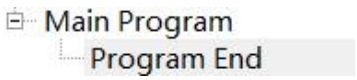
Command Usage

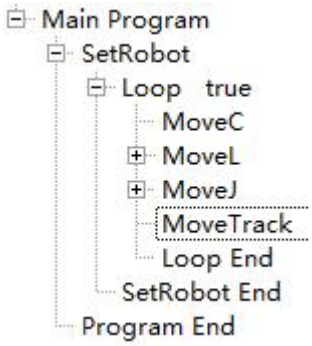
Create the main program and automatically generate the Main Program structure. All program commands must be written between Main Program and Program End. A program framework can only contain one main program, and multiple subprograms can be created in a main program.

Basic Example

The following example introduces the Main Program command:

Example 1





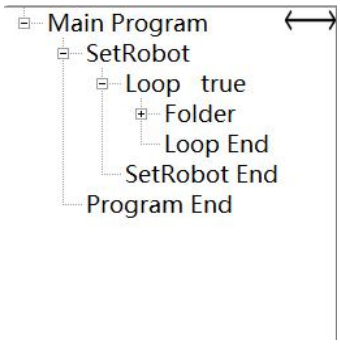
1.2 Folder - Create the folder

Command Usage

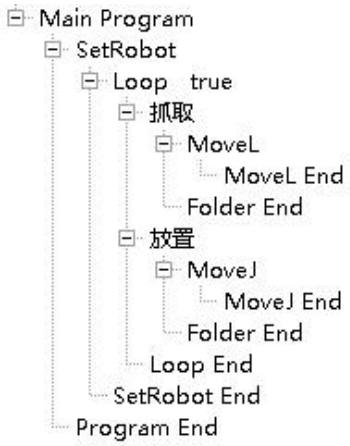
Create the folder. For long programs, you can use folders to hide or open some functional blocks. You can rename the folders to improve readability, which is also convenient for copying, deleting, and shielding functional blocks.

Basic Example

The following example introduces the Folder command :



If there are too many commands and they are complicated, we can use the folder to converge the program, so that the program is clear and organized, programmers can sort it out easily, and problems can be located quickly when they occur.



The Folder command can also be renamed, which makes it more convenient for programmers to use in daily life and also makes it easier for other people to adjust the program later.

1.3 Subprogram - Create the subprogram

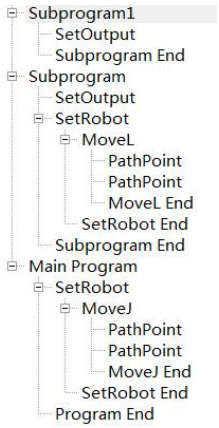
Command Usage

Create a subprogram. You can rename a subprogram, but you must create it outside the main program, that is, outside the Main program and Program End. You cannot create a subprogram within a subprogram, that is, each Subprogram must be created outside the Subprogram and Program End.

Basic Example

The following example introduces the Subprogram command:

Example 1



If you want to create a Subprogram in a program, it can only be created outside the Main Program. If you want to call a Subprogram, you need to use the Call Subprogram command.

Stop_Subprogram - Stop Subprogram

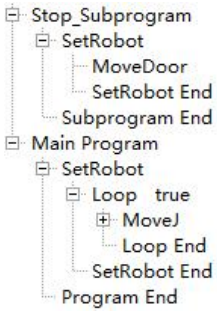
Command Usage

Stop subprogram: If you rename a subprogram to Stop_Subprogram, then this subprogram is a stop subprogram. When the robot stops, Stop_Subprogram is automatically executed without calling it. When the main program ends, if you want the robot to move to a specified stop position, or want to send a signal to stop peripheral equipment, etc., you can write these operation commands in Stop_Subprogram.

Basic Example

The following example introduces the Stop_Subprogram command:

Example 1



If the robot needs to add a stop position, the Subprogram command uses the above writing format. When the main program is finished, if you want the robot to move to the specified stop position, name the Subprogram Stop_Subprogram and write the motion command inside.

1.4 Thread - Create the thread

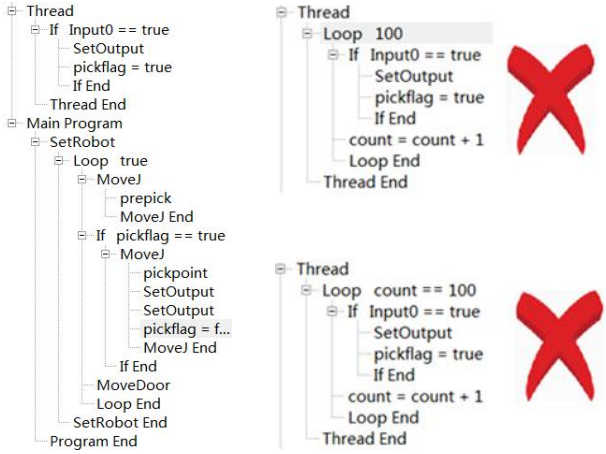
Command Usage

Create a thread. You can rename the thread. The thread and the main program run in parallel. Loop statements and motion statements (i.e., statements in the “Motion” tab) cannot appear in the thread.

Basic Example

The following example introduces the Thread command:

Example 1



As shown in the program above, the Main Program and Thread commands are executed together. No matter where the Main Program runs, Thread will scan whether there is any signal input at the Input. When there is a signal input at the Input, the SetOutput command will output the corresponding signal.

Note:

The Thread command is automatically executed in a loop. When it reaches the Thread End, it automatically returns to the first sentence of the Thread. No loop statement is required. The Thread command is not allowed to contain loop commands.

1.5 SetRobot - Select the robot model

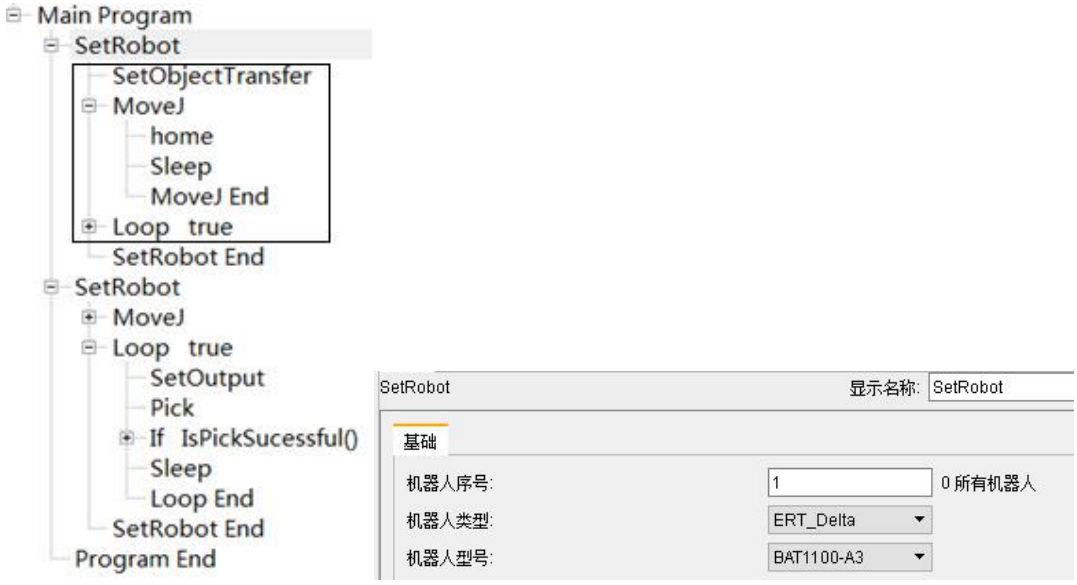
Command Usage

Select the robot model and create a SetRobot structure. The commands in the structure are only executed by the specified robot. The commands not in the SetRobot structure will be executed once by all robots in the system. The motion commands are the commands in the Move command box, which must be included in SetRobot. And the motion commands will only be displayed after the SetRobot command is used.

Basic Example

The following examples introduce the SetRobot command:

Example 1



The statements between SetRobot and SetRobot End are executed by the robot specified in SetRobot. For example, in the above program, if the robot number in the first SetRobot parameter is 1, then the statements in the box are executed by the robot with ID 1.

Example 2

SetRobot parameters:

Robot number: Robot ID number. If this item is set to 0, the statements between SetRobot and SetRobot End are executed by all robots.

Robot type: Select the type of robot to be debugged, for example: ERT_Delta = Bat delta robot. ERT_Scara = Python robot. PumaArm = Mantis six-axis robot.

Robot model: Select the model of the robot to be debugged.

Chapter 2 Logic

2.1 Loop - Loop command

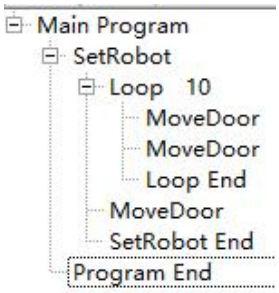
Command Usage

Run a program segment in a loop. When the Loop condition is met, execute the commands in the Loop. When the Loop End is reached, if the Loop condition is still met, execute the commands in the Loop from the beginning again. If the Loop condition is not met when the Loop End is reached at some point, the commands in the Loop will no longer be executed but the commands after the Loop End will be executed.

Basic Example

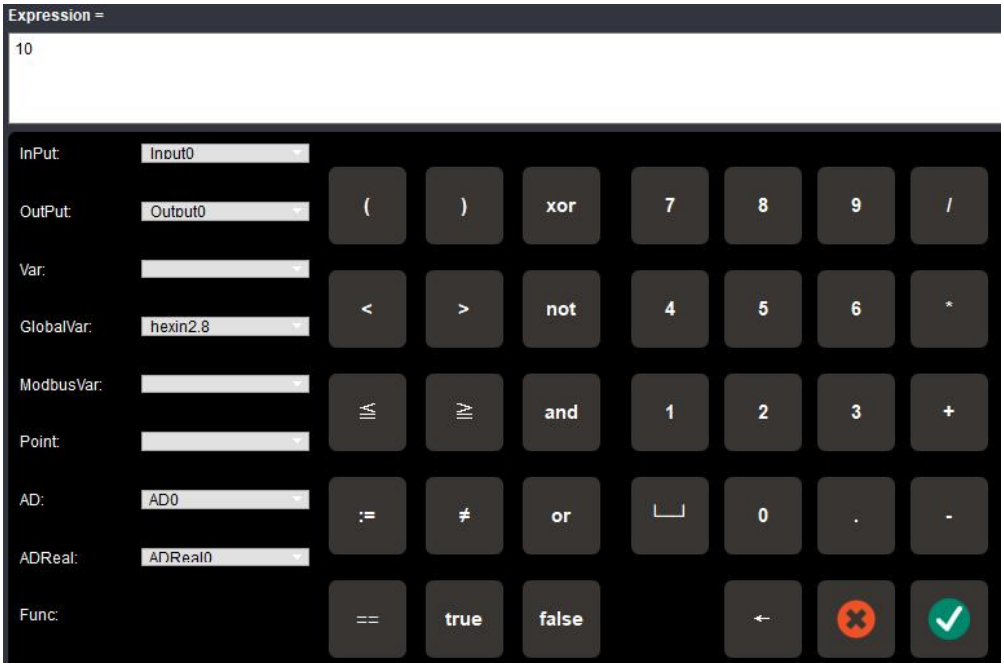
The following examples introduce the Loop command :

Example 1



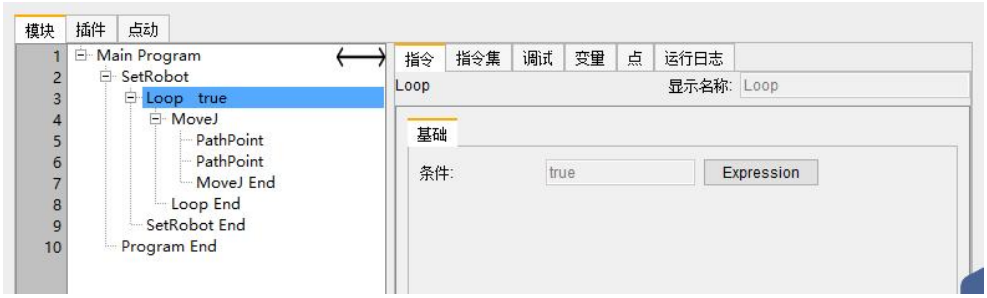
If the Loop condition is an integer, such as 10, then the commands in the Loop will be executed 10 times, and then the MoveDoor command after Loop End will be executed.

Example 2



The conditions are entered using the expression editor. Select Loop with the cursor and click the Expression button under the right tab to enter the expression editor. In the expression editor, you can use the number keys to enter the number of loops or loop conditions.

Example 3



If the Loop condition is true, the commands in the Loop will be executed continuously. If you want to jump out of the Loop command, you should add a break command in the loop command to jump out of the Loop.

2.2 If, Else If, Else - Conditional judgment

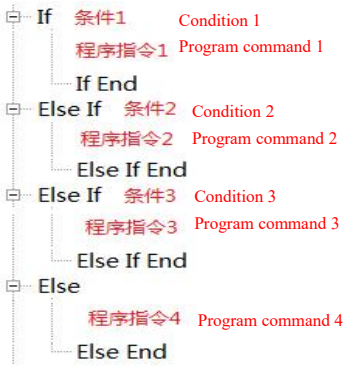
Command Usage

Conditional judgment statements, are used to determine whether a given condition is met and to perform different operations based on the determination results.

Basic Example

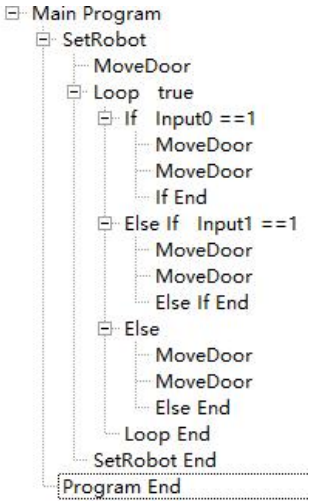
The following examples introduce the If, Else If, Else commands :

Example 1



Conditional judgment, as shown in the above program, if condition 1 is met, program command 1 will be executed, and others will not be executed. If condition 1 is not met, it will determine whether condition 2 is met, and if condition 2 is met, program command 2 will be executed, and others will not be executed. If both condition 1 and condition 2 are not met, it will determine whether condition 3 is met, and if condition 3 is met, program command 3 will be executed, and others will not be executed. If none of the above are met, program command 4 will be executed. The conditions are entered using the expression editor.

Example 2



For example, when the program is actually running, after the MoveDoor command is executed, it enters the Loop statement and starts to judge. If Input0 is equal to 1, only the MoveDoor command in the If statement is executed. If Input0 is equal to 0 and Input1 is equal to 1, only the MoveDoor command in the Else if statement is executed. If Input0 and Input1 are both equal to 0, the MoveDoor command in the Else statement is executed.

2.3 Expression - Expression editor

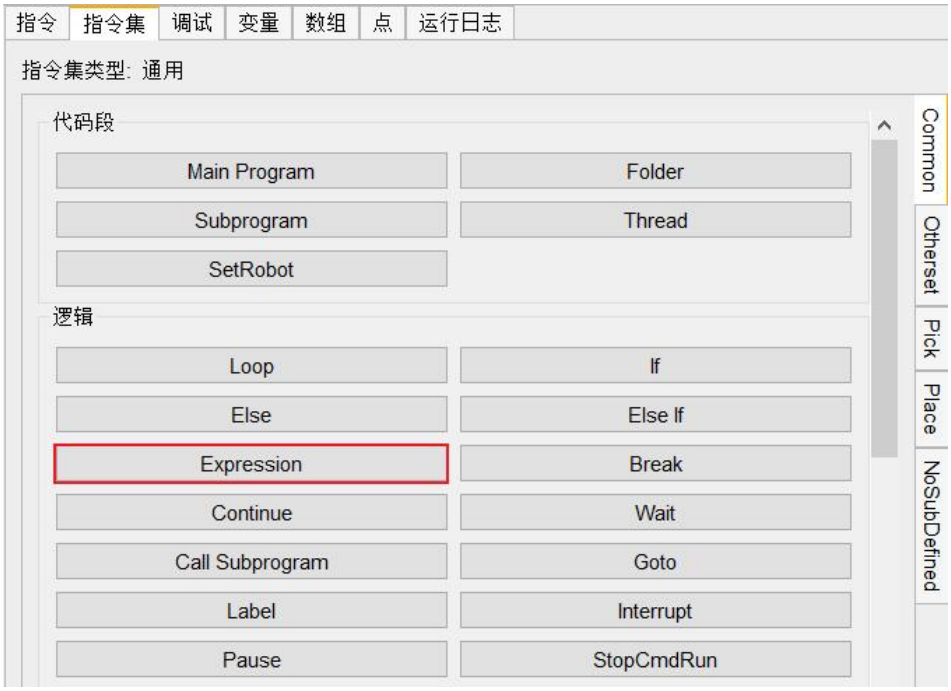
Command Usage

The expression editor, is used to edit and input various logical expressions. You can open the expression editor through the Common tab - Logic command box - Expression button. There is also an Expression button in the Command tab of some commands.

Basic Example

The following example introduces the Expression command :

Example 1

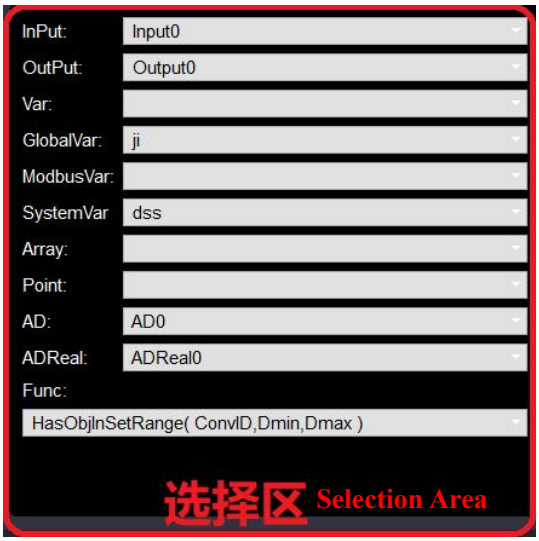


As shown in the figure above, the Expression editor contains three major areas. The functions of each area are described in the following table:

Editing Area:

Area	Function
Editing Area	Display the editing content

The content displayed in the editing area is the written expression statement. For example, the “10” in the “Loop 10” statement means that in the function editor Expression, by clicking the numbers “1” and “0” on the keypad in the button area, they will be written into the editing area. Clicking the “check mark” will make the “Loop 10” loop statement loop 10 times.



Selection Area:

Area	Function
Selection Area	InPut: Input point drop-down list
	OutPut: Output point drop-down list
	Var: Local variable drop-down list
	GlobalVar: Global variable drop-down list
	ModbusVar: Modbus variable drop-down list
	SystemVar: System variable drop-down list
	Array: Array drop-down list
	Point: Point drop-down list
	AD: Not yet open
	ADReal: Not yet open
	Func: Function drop-down list

The selection area contains some variables and functions. The “Loop Input0 == 1” statement is formed by selecting “Input0” from the Input point drop-down list and combining “==” and “1” in the button area. This means that if “Input0” is equal to 1, the “Loop” statement will be entered. If “Input0” is not equal to 1, the “Loop” statement will not be entered. (“==” means whether it is equal to or not, and its judgment effect)

Button Area:

The functions of the button area are as shown above. Each button has its own specific function and is used in conjunction with the editing area and the selection area.

Area	Function	
Button Area	"==": is equal to	
	":=": assigns a value to a variable	
	"≠": is not equal to	
	"≥、≤": is greater than or equal to, is less than or equal to	
	"＞、＜": is greater than, is less than	
	"（）": parentheses	
	"ture": true	
	"false": false	
	"or、and、not、xor": or, and, not, exclusive or	
	Number Area	addition, subtraction, multiplication, division, cancel, confirm, delete, space, brackets

2.4 Expression - Functions

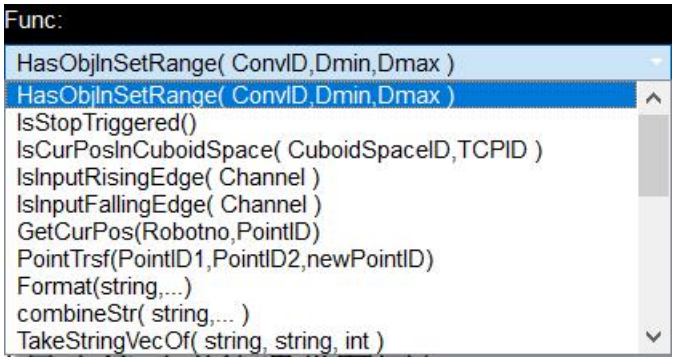
Command Usage

Functions can be selected and used through Func in the Expression selection area.

Basic Example

The following example introduces the function in the Expression command :

Example 1



The function in the expression editor is shown in the figure above, and its specific functions and examples are shown below.

2.4.1 Function - IsStopTriggered()

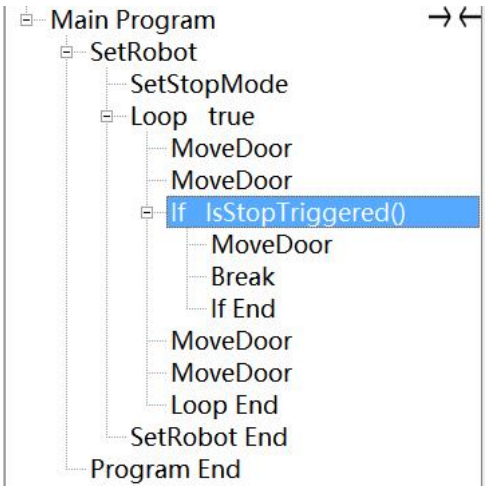
Function Usage

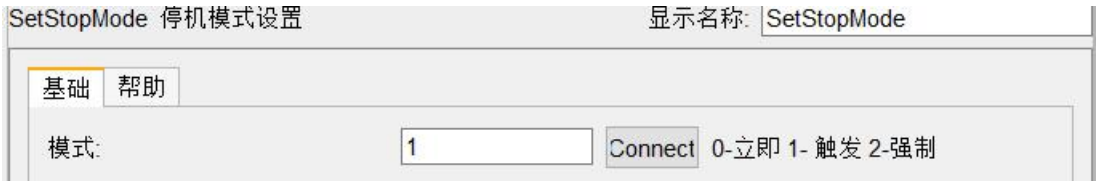
Used with the parameter "Mode=1" in the SetStopMode command. IsStopTriggered() is used to determine whether the END or ENDALL button is pressed. If the button is pressed, the function IsStopTriggered() returns true, otherwise false.

Basic Example

The following example introduces the IsStopTriggered () function:

Example 1





Set the stop mode to 1. If you press the "END" or "ENDALL" button, the if statement is true, the MoveDoor is executed, and then break is executed to exit the Loop, thus ending the entire program. If the if statement in the program above is deleted, the "END" or "ENDALL" button is pressed, the program will still run in the Loop, which is incorrect. If this happens, you can press the "Pause"

button first, and then press the “END” or “ENDALL” button to end the program.

2.4.2 Function - IsCurPosInCuboidSpace（CuboidSpaceID，TCPID）

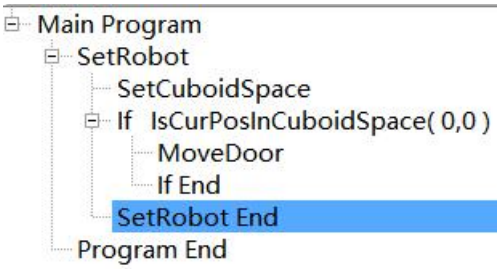
Function Usage

Whether the current TCP is within the space range specified by the SetCuboidSpace command. Parameter CuboidSpaceID: the CuboidSpaceID number of the corresponding SetCuboidSpace command. TCPID: TCP number.

Basic Example

The following example introduces the IsCurPosInCuboidSpace (CuboidSpaceID，TCPID) function:

Example 1



As shown in the figure, use the “SetCuboidSpace” command to set a monitoring area, and the if statement determines whether TCP0 is within the range of monitoring area 0. If TCP0 is within monitoring area 0, the return value is true and the MoveDoor command is executed, otherwise exit the program.

2.4.3 Function - IsInputRisingEdge（Channel）

Function Usage

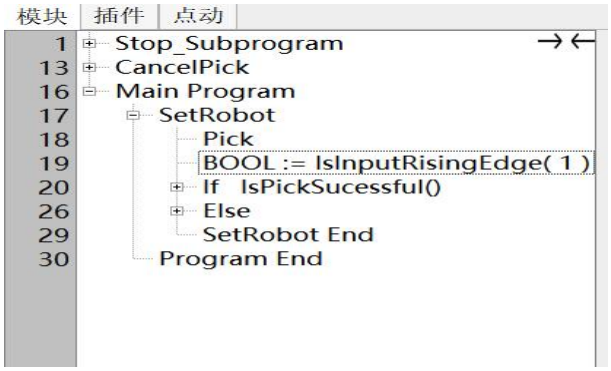
If the input channel is rising edge, the return value is true, otherwise the return value is false. Parameter Channel: Input channel number.

Basic Example

The following example introduces the IsInputRisiingEdge (Channel) function:

Example 1

Add a BOOL variable, and use the BOOL variable to monitor IsInputRisingEdge (1), which returns the value of input channel 1.



The example in the figure is used in conjunction with hardware that can transmit signals, such as photoelectric sensors.

2.4.4 Function - IsInputFallingEdge（Channel）

Function Usage

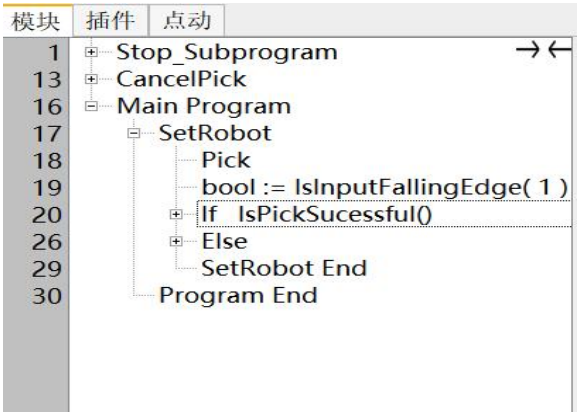
If the input channel is falling edge, the return value is true, otherwise the return value is false. Parameter Channel: Input channel number.

Basic Example

The following example introduces the IsInputFallingEdge (Channel) function:

Example 1

Add a BOOL variable, and use the BOOL variable to monitor IsInputFallingEdge(1), which returns the value of input channel 1.



The example in the figure is used in conjunction with hardware that can transmit signals, such as photoelectric sensors.

2.4.5 Function - GetCurPos（Robotno，Pointid）

Function Usage

Get the current pose of the robot.

Parameter Robotno: Robot ID, indicating the pose of the robot to be picked up.

Pointid: Point variable ID, indicating which point variable the acquired pose is saved in.

Note: This function picks up the pose of the current TCP under the current Base.

Basic Example

The following example introduces the GetCurPos (Robotno, Pointid) function:

Example 1

The example in the figure shows obtaining the current pose of the robot and saving the obtained pose into a point variable.

2.4.6 Function - PointTrsf(PointID1,PointID2,newPointID)

Function Usage

Point Transformation: newPoint = Point1 * Point2, where the corresponding parameters are the point ID numbers.

Basic Example

The following example introduces the PointTrsf(PointID1,PointID2,newPointID) function:

Example 1

Point 1 is a pose matrix, and Point 2 is a transformation matrix. The multiplication of the two means that the Point1 coordinate system is translated and rotated along the x, y, and z axes according to the Point 2 transformation matrix to obtain a new position matrix, which is newPoint.

PointID1: Reference point ID

PointID2: Transformed point ID

NewPointID: New point ID

2.4.7 Function - Format（string，…）

Function Usage

Set the string format. The percent sign (%) followed by a number indicates the position to be replaced. The string 1, string 2, string 3, ... string n after the comma (,) replaces the previous %1, %2, %3, ... %n in sequence.

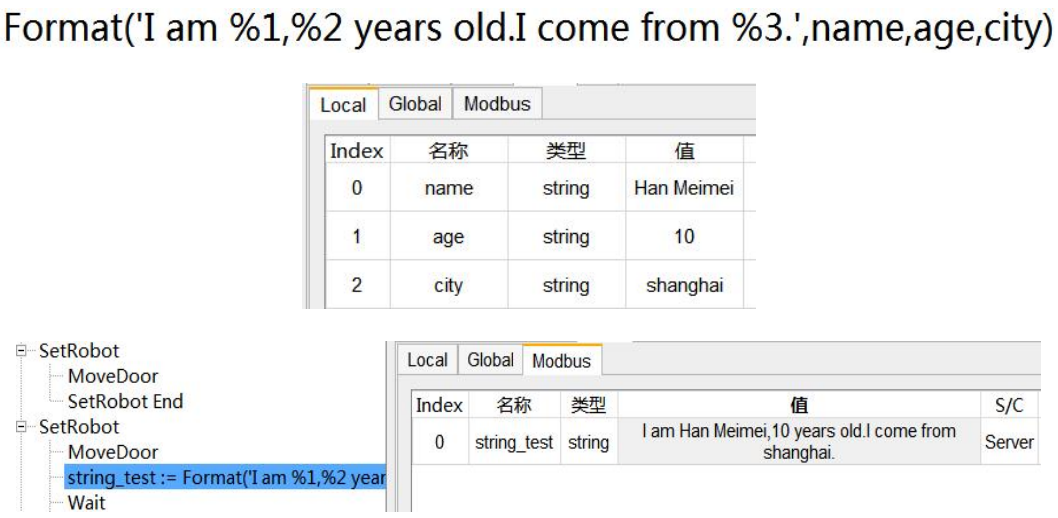
Basic Example

The following examples introduce the ForMat(string,…) function：

Example 1

In this example, Li Lei is replaced at position %1, 12 is replaced at position %2, and beijing is replaced at position %3. The value of Format is assigned to the string variable String_test, and the result is: I am Li Lei, 12 years old. I come from Beijing.

Example 2



You can also use string variable 1, string variable 2, string variable 3, ... string variable n to replace the previous %1, %2, %3, ... %n in sequence. Assign the values in the above table expression to the string variable String_test.

2.4.8 Function - combineStr（string，…）

Function Usage

Combine string function. Combine several short strings into one long string.

Basic Example

The following examples introduce the combineStr(string，…) function：

Example 1

Interleave and combine the separate strings ‘a’, ‘b’, ‘c’ and the string variables name (value is ‘Han Meimei’) and age (value is ‘10’) into a string string_test:=combineStr(‘a’, name, ‘b’, age, ‘c’). The running result is: the value of string_test is aHan Meimeib10c.

2.4.9 Function - TakeStringVelOf（string，string，int）

Function Usage

Extract the string separated by the delimiter from the long string. Parameter string: the name of the long string variable, the delimiter (the string variable that also stores the delimiter), the number of the string to be extracted (starting from 0).

Basic Example

The following examples introduce the TakeStringVelOf() function:

Example 1

Communicate with external device through socket, receive the string information from the external device: ready: 23.45:12.34:57.09:13.45:end and save this string information in string variable recive. The characteristic of this data is that each data is separated by semicolon “:”. To extract 57.09 and store it separately in string variable string_vel, you can use string_vel:=TakeStringVelOf (recive, ‘:’, 3), 57.09 is saved in string_vel (57.09 is a string). To extract ready, you can use string_vel:=TakeStringVelOf(recive, ‘:’, 0).

2.5 Break - Jump command

Command Usage

Jump out of the loop of the Loop statement

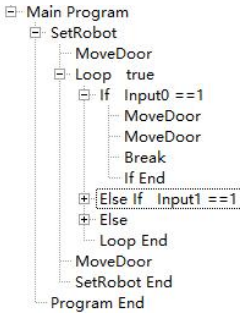
Basic Example

The following examples introduce the Break command :

Example 1

In the above program, when the program runs to the Break command, it will jump out of the current Loop statement and execute the commands after loopEnd. The current loop refers to the loop layer where break is located. Please pay attention to this when multiple loops are nested.

Example 2



For example, in the current program, at the beginning of the program running, the first command MoveDoor runs, then the program enters the “Loop true” statement. If the condition “Input0==1” is met at this time, it will enter the “if” statement to execute the second command MoveDoor and the third command MoveDoor, and then run the Break command. When the Break command is executed, it automatically jumps out of the entire “Loop true” statement, runs the fourth command MoveDoor, and finally the program ends.

Note

The Break command can only be executed within a Loop statement, otherwise an error will occur.

2.6 Continue - Jump statement

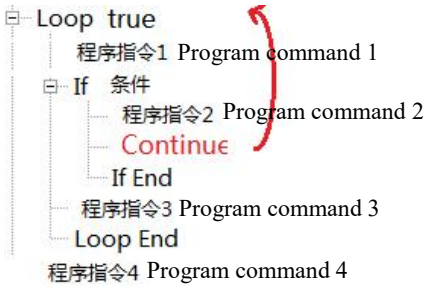
Command Usage

Jump out of the loop of the current Loop statement and perform conditional judgment of the current Loop again.

Basic Example

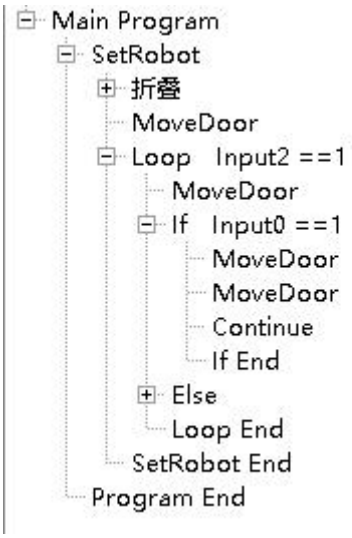
The following examples introduce the Continue command :

Example 1



As shown in the above program, when entering the “Loop true” statement, the program command 1 is executed first, and the next step is to determine whether the “if” condition is met. If it is met, the system will run the program command 2, and when it reaches the Continue command, it will jump out of the current “Loop” and re-determine whether the “Loop” condition is met. In the above figure, the “Loop” condition is true, which means it is always met and will enter the “Loop” again.

Example 2



The program is shown in the figure above. The system starts to run the MoveDoor command. If the condition “Input2==1” is met, it enters the Loop and runs MoveDoor in the Loop. Then the “If” statement is judged. If the condition “Input0==1” is met, the two MoveDoor in the If are run. When the Continue command is run, the current “Loop” is jumped out and the condition “Whether Input2 is 1” of the Loop is judged again. If Input2==1, the Loop is entered again, MoveDoor is run, and the “If” statement is judged. If Input2 is not 1, the MoveDoor command after the Loop End is directly executed without entering the loop.

Note

The Continue command can only be executed in the Loop, otherwise an error will be reported.

2.7 Wait - Wait command

Command Usage

Wait for a period of time (in seconds) or wait for a certain condition to be met.

Basic Example

The following examples introduce the Wait command :

Example 1

The function of the Wait command: After an command is executed, if you want to wait for a period of time or wait for a certain condition to be met before executing the next statement, you can use the wait command.

Example 2

As shown in the program, when the program starts, it runs the first MoveDoor command in the main program, then enters the “Loop true” statement and runs the second MoveDoor command. The program then reaches the Wait command, pausing for a period of time (the stop time is a specific number set by yourself), or the time of the wait command can be controlled through association with a Double variable. To associate with a Double variable, click the “connect” button in the Wait command interface and select the appropriate variable from the list.

Example 3

As shown in the figure above, if the program runs to the Wait command shown in the figure above, the program will stop until the condition “Input0==true” is met, and then the program after Wait will continue to run.

2.8 Call Subprogram - Call subprogram

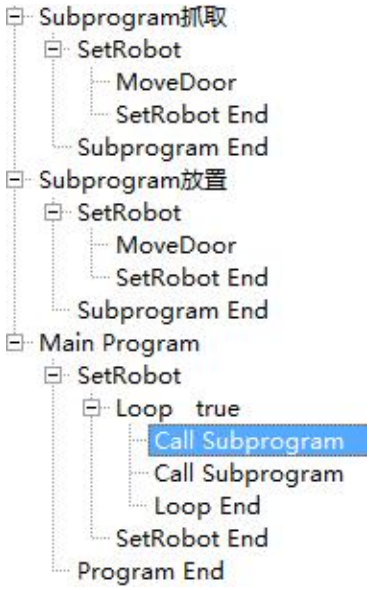
Command Usage

Call subprograms. Please note that subprograms can be called by the main program and other subprograms, but they cannot call each other. For example, there are three subprograms A, B, and C. It is possible for the main program to call A, A to call B, and B to call C. After that, C cannot call A and B again, B cannot call A again, and the subprogram cannot call the main program again.

Basic Example

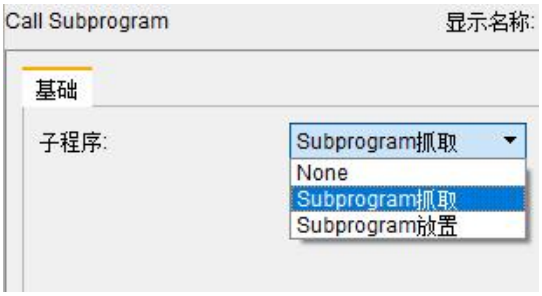
The following examples introduce the Call Subprogram command :

Example 1



In the program, the function of the Call Subprogram command is to call the subprogram. Under some conditions, the subprogram is needed to judge or control the input and output. Each Call Subprogram command can only call one subprogram. After the subprogram is executed, the next command of the Call Subprogram command in the main program will continue to be executed.

Example 2



The Call Subprogram command can select any subprogram that has been created in the program, but only one subprogram can be selected.

Note: The creation of the Call Subprogram command is not limited to being in a subprogram or main program, but its conditions must be met. (The conditions are indicated in the command usage)

2.9 Goto - Program jump

Command Usage

The program jump statement allows the program to jump to a specified label and start executing the program from the label. It is used in conjunction with the Label command.

Basic Example

The example is introduced together with the Label command.

2.10 Label - Label

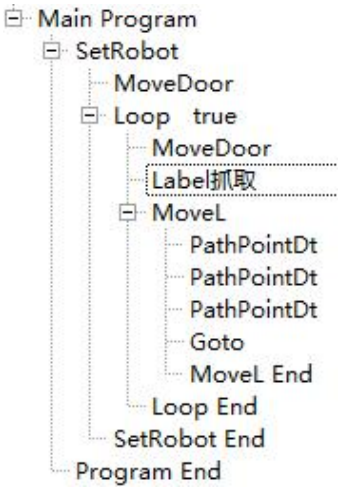
Command Usage

Label, used in conjunction with the Goto command.

Basic Example

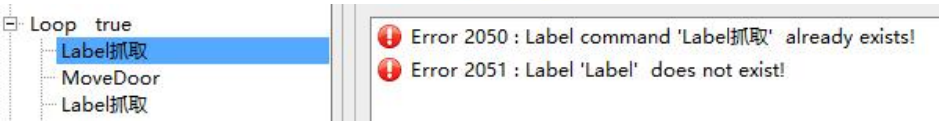
The following example introduces the Goto command and Label command:

Example 1



In the Goto command interface, select the Label we need to jump to. As in the above program, it will cycle between “Label grab” and Goto, and will not start executing from the MoveDoor command.

Note:



The same Label name cannot appear in the same main program. At the same time, Goto in the main program can only call Label in the main program, and Label in the subprogram cannot be called by Goto in the main program.

It is not recommended to mix Loop command and goto+label. In particular, you cannot use goto+label to jump into or out of the Loop.

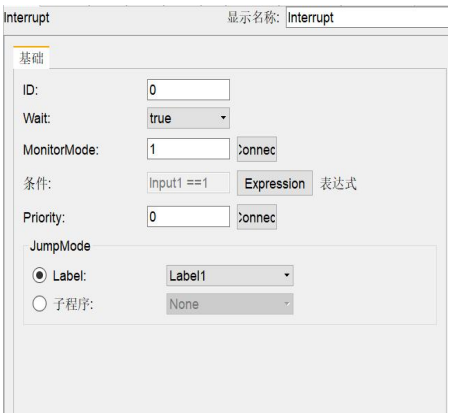
2.11 Interrupt - Interrupt

Command Usage

Program interrupt command, when the set conditions are met, the jump logic in JumpMode is executed. If multiple interrupts meet the conditions at the same time, the command with the highest priority will be executed. If the priorities are the same, they will be executed in the order in which the interrupts are created, and the one created first will be executed first.

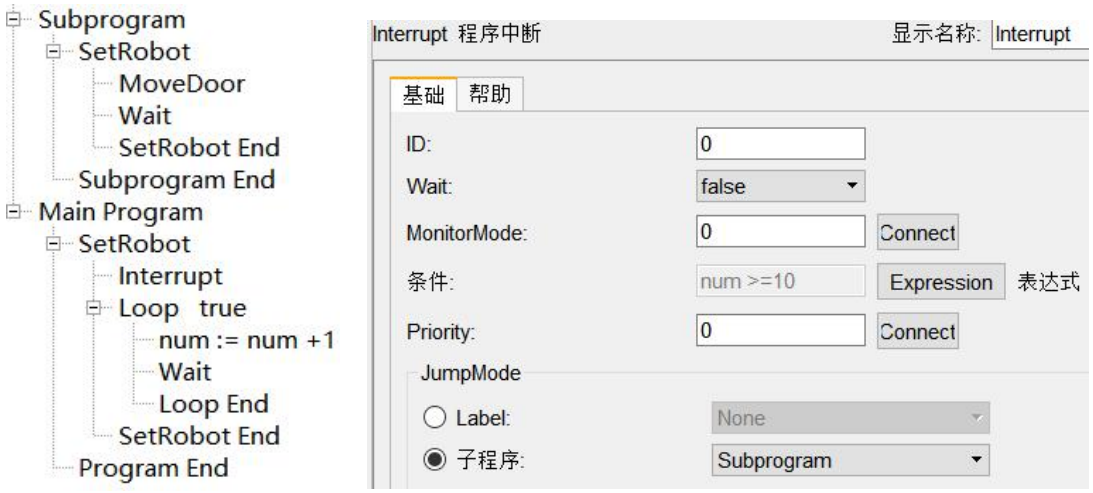
Basic Example

Example 1



Basic column	Functions and parameters
ID	The ID number of the interrupt command
Wait	Whether to wait for the current motion to complete
MonitorMode	This function is reserved and is not enabled yet.
Condition	The content of the conditional expression is met. When Input==1, the Interrupt command is executed.
Priority	Set the program execution priority (0~5). The larger the value, the higher the program execution priority.
JumpMode	Label: jump to the corresponding Label. Subprogram: jump to the corresponding subprogram.

Example 2



As shown in the program above, an interrupt is set. When the condition “num >= 10” is met, it will jump to the Subprogram. The Loop loops once and num will be increased by 1. When num is greater than or equal to 10, it will jump to the subprogram to execute MoveDoor.

2.12 Pause - Pause/Continue program execution

Command Usage

Pause/Continue program execution

Basic Example

The following example introduces the Pause command :

Example 1

Its basic tab functions:

Basic tab	Functions and parameters
Mode	When Mode=0, the program execution will be paused when the program reaches pause.
	When Mode=1, the program will resume execution when it reaches pause.
	When Mode=2, the program will jump to the next command when it reaches pause.

Note: Pause is only valid for the Mainprogram and the subprograms it calls. It is invalid for the Thread and the subprograms it calls. Pause (Mode=0) can be written in Mainprogram or Thread, while Pause (Mode=1) needs to be written in Thread. If it is written in Mainprogram, once the program is paused, it cannot be resumed.

2.13 StopCmdRun - Stop mode setting

Command Usage

Stop mode setting

Basic Example

The following example introduces the StopCmdRun command :

Its basic tab functions:

Basic tab	Basic tab
Mode	Stop mode. When Mode=0, after pressing the End button or the End All button, if there is a stop subprogram Stop_Subprogram(), the motion will be stopped after the current motion command is executed or after the next two motion commands, and the stop subprogram is executed. If there is no stop subprogram, it will stop immediately after the current motion command is executed.
	When Mode=1, it is used in conjunction with the IsStopTriggered() function. When the Stop button or StopAll button is pressed, the return value of the IsStopTriggered() function is true. It can be used to respond

	to the stop signal at a specific location and execute the corresponding subprogram.
	When Mode=2, after pressing the End button or the End All button, if there is a stop subprogram, it is the same as Mode=0. If there is no stop subprogram, the motion stops immediately at the current position and the program ends.

Chapter 3 Motion

3.1 MoveL - Linear motion command

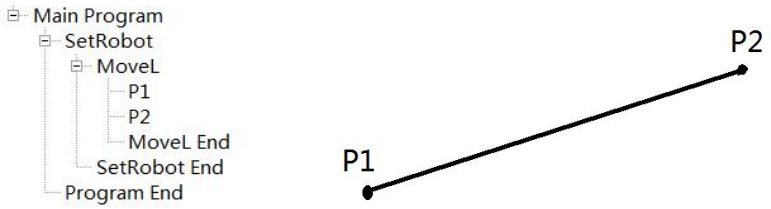
Command Usage

Linear motion. If the robot is required to move in a straight line, add this command to move the tool center point along a straight line to a given destination. Used with PathPoint and PathPointDt.

Basic Example

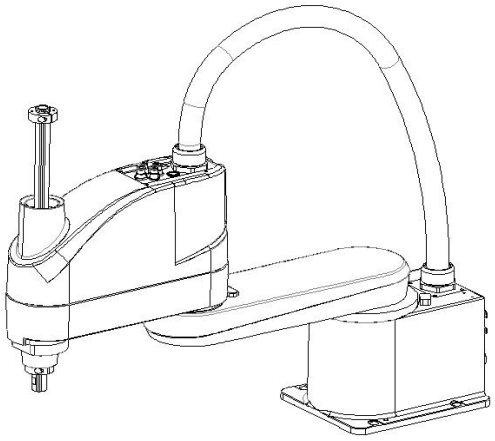
The following example introduces the MoveL command :

Example 1



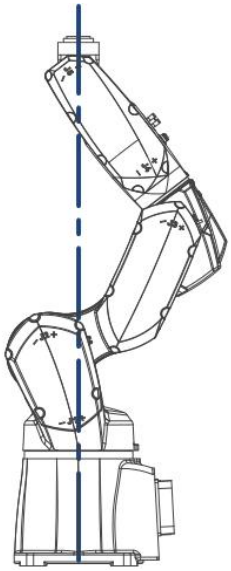
As shown in the above program, after running the main program, it enters the MoveL command, moves in a straight line from the current point to point P1, and then moves in a straight line from point P1 to point P2.

When using MoveL, singularity points should be avoided. If a singularity point occurs during the MoveL process, the robot will report an error and the program cannot continue to run. The figure shows the status of the robot singularity point.



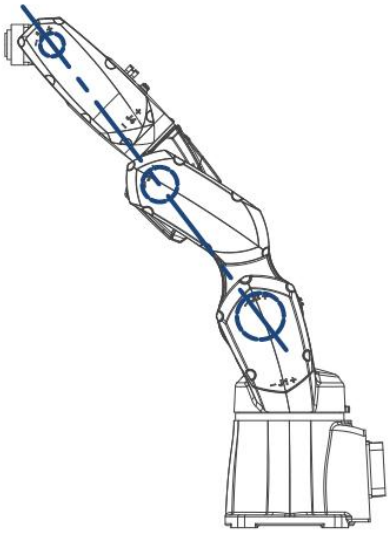
SCARA singularity point position J2=0°

SCARA robot has only one singularity point position, when J2=0°, the 1st and 2nd arms are in a straight line.



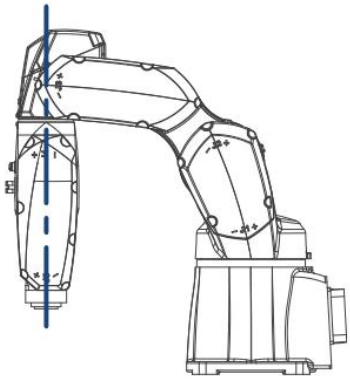
Top singularity

The intersection of the 4th, 5th and 6th axes is located directly above J1.



Extended singularity

The 2nd, 3rd and 5th axis joint centers are collinear.



Wrist singularity

The 4th and 6th axes are collinear.

Three singularity points of 6-axis robots

Bat series

Tracking unit

Speed — Maximum speed (0-1)

Acceleration — Maximum acceleration (0-1)

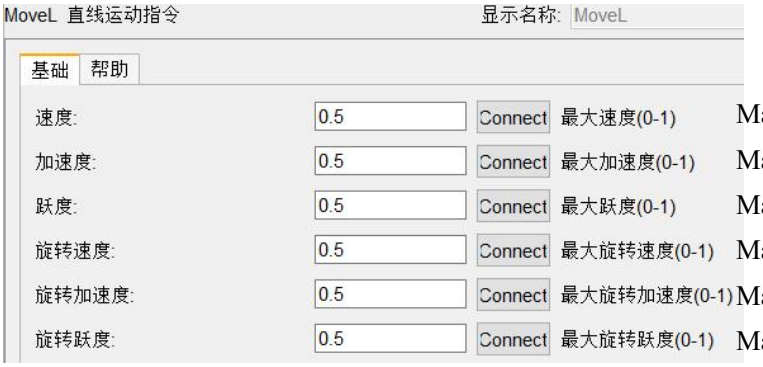
Jerk — Maximum jerk (0-1)

Rotation speed — Maximum rotation speed (0-1)

Rotation acceleration — Maximum rotation acceleration (0-1)

Rotation jerk — Maximum rotation jerk(0-1)

Python series



Speed — Maximum speed (0-1)

Acceleration — Maximum acceleration (0-1)

Jerk — Maximum jerk (0-1)

Rotation speed — Maximum rotation speed (0-1)

Rotation acceleration — Maximum rotation acceleration (0-1)

Rotation jerk — Maximum rotation jerk(0-1)

Mantis series

Speed — Maximum speed (0-1)

Acceleration — Maximum acceleration (0-1)

Jerk — Maximum jerk (0-1)

Rotation speed — Maximum rotation speed (0-1)

Rotation acceleration — Maximum rotation acceleration (0-1)

Rotation jerk — Maximum rotation jerk(0-1)

Collision monitoring settings

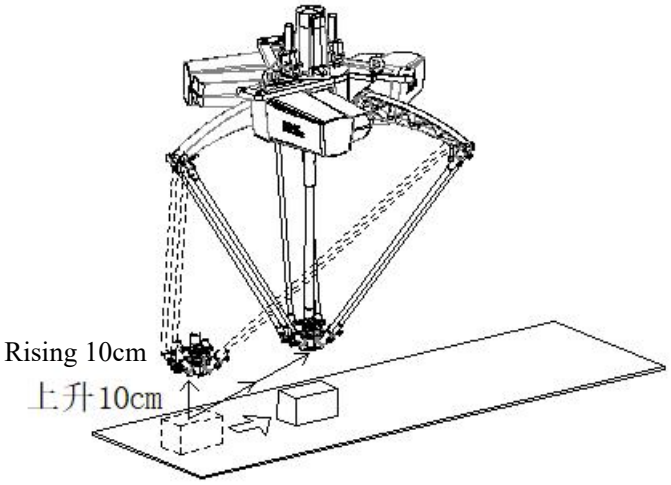
Use custom parameters

Monitoring validity

Monitoring level — Sensitivity increases as the value decreases.

MoveL parameters are as above, and their specific meanings are as follows (parentheses indicate models that include this parameter, and no parentheses after the parameter indicate that all models include this parameter):

Basic column	Functions and parameters
Tracking unit (Bat)	True to enable, false to disable. After Pick/PlaceMobileBox turns on the tracking unit, if the following MoveL/MoveC command also turns on the tracking unit, the MoveL/MoveC motion will also be superimposed with the conveyor belt tracking motion. Until a motion command that does not turn on the tracking unit is

	encountered, the following motion will no longer be superimposed with the tracking motion regardless of whether the tracking is turned on or not. Superimposed tracking conditions: 1. Pick/PlaceMobileBox turns on the tracking unit. 2. MoveL/MoveC behind Pick/PlaceMobileBox also turns on the tracking unit. 3. Pick/PlaceMobileBox tracking action is executed. 4. The tracking is not interrupted (if there is an command that does not turn on the tracking unit in the middle of multiple commands that turn on the tracking unit, the tracking is interrupted here, and the tracking unit cannot be tracked if it is turned on again later). As shown in the figure, the tracking unit is turned on and the robot follows the conveyor belt. 
Speed	Maximum speed percentage. For example, if the robot's running trajectory is 2 meters per second and the speed is set to 0.5, the robot's maximum speed during the execution of the trajectory is 2*0.5=1 meter per second.
Acceleration	The ratio of the change in the speed of an object to the time taken is called acceleration. The higher the percentage, the faster the speed increases within the range.
Jerk	Maximum jerk percentage
Rotation speed	The maximum speed percentage of the rotational motion part. The maximum speed at which the robot executes commands during the rotational motion.
Rotation acceleration	The acceleration percentage of the rotational motion part. The larger the value, the faster the rotation acceleration.
Rotation jerk	The rotation jerk is the derivative of the rotation acceleration. The derivative describes a degree of change, which refers to the change of the maximum falling deceleration.

Associated variables	There is a Connect button behind each parameter. Pressing it will automatically display all variables that match the type in the Variable tab. Double-click the variable ID to associate it with the corresponding variable. After association, the Connect button will change to a Disconnect button. Press Disconnect to cancel the association.
Using custom parameters (Mantis)	Whether to use custom parameters. When a robot is in motion, its arms and other parts may interfere with surrounding objects and collide with them. When a collision occurs during motion, it is necessary to detect the collision immediately and perform an emergency stop on the robot. This is called collision detection.
Monitoring validity (Mantis)	Whether collision can be detected. 0 means invalid, 1 means valid.
Monitoring level (Mantis)	Monitoring sensitivity, which increases as the value decreases.

3.2 MoveJ - Joint motion command

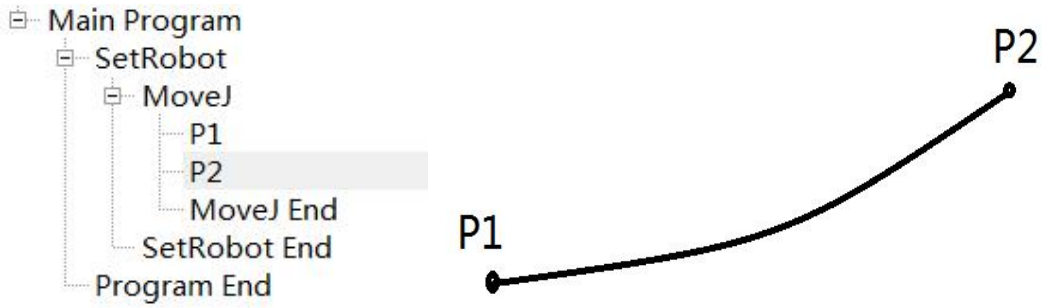
Command Usage

Joint motion command, MoveJ is used to quickly move the tool center point of the robot from one point to another. All axes reach the target position at the same time. Used with PathPoint and PathPointDt.

Basic Example

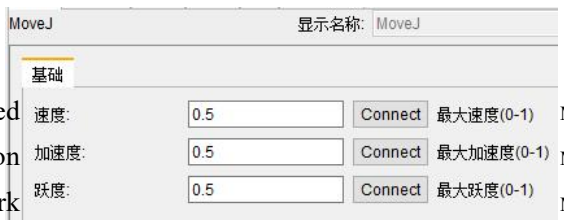
The following example introduces the MoveJ command :

Example 1



For example, in the above program, after the system runs the main program, it enters the MoveJ command, moving from the current point to point P1, and from point P1 to point P2. MoveJ is a point-to-point motion, and there is no requirement for the trajectory during the motion process.

Bat series, Python series



Speed — Maximum speed (0-1)

Acceleration — Maximum acceleration (0-1)

Jerk — Maximum jerk (0-1)

Mantis series

Speed — Maximum speed (0-1)

Acceleration — Maximum acceleration (0-1)

Jerk — Maximum jerk (0-1)

Collision monitoring settings

Use custom parameters

Monitoring validity

Monitoring level — Sensitivity increases as the value decreases.

MoveJ parameters are as above, and their specific meanings are as follows (parentheses indicate models that include this parameter, and no parentheses after the parameter indicate that all models include this parameter):

Basic column	Functions and parameters
Speed	Maximum speed percentage. For example, if the robot's running trajectory is 2 meters per second and the speed is set to 0.5, the robot's maximum speed during the execution of the trajectory is 2*0.5=1 meter per second.
Acceleration	The ratio of the change in the speed of an object to the time taken is called acceleration. The higher the percentage, the faster the speed increases within the range.
Jerk	Maximum jerk percentage
Associated variables	There is a Connect button behind each parameter. Pressing it will automatically display all variables that match the type in the Variable tab. Double-click the variable ID to associate it with the corresponding variable. After association, the Connect button will change to a Disconnect button. Press Disconnect to cancel the association.
Use custom parameters (Mantis)	Whether to use custom parameters. When a robot is in motion, its arms and other parts may interfere with surrounding objects and collide with them. When a collision occurs during motion, it is necessary to detect the collision immediately and perform an emergency stop on the robot. This is called collision detection.
Monitoring validity (Mantis)	Whether collision can be detected. 0 means invalid, 1 means valid.
Monitoring level (Mantis)	Monitoring sensitivity, which increases as the value decreases.

3.3 MoveC - Circular motion command

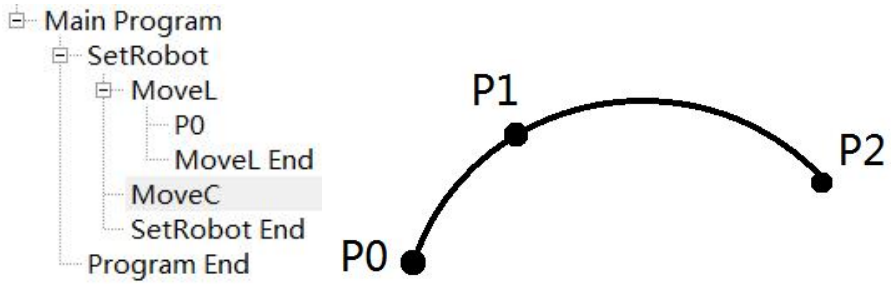
Command Usage

Circular motion command is used to move the tool center point (TCP) along a circle to a given destination. During the motion, the orientation of the cycle usually remains relatively unchanged. The previous command is used as the starting point, and the two points set in MoveC make the robot move to form a circular arc.

Basic Example

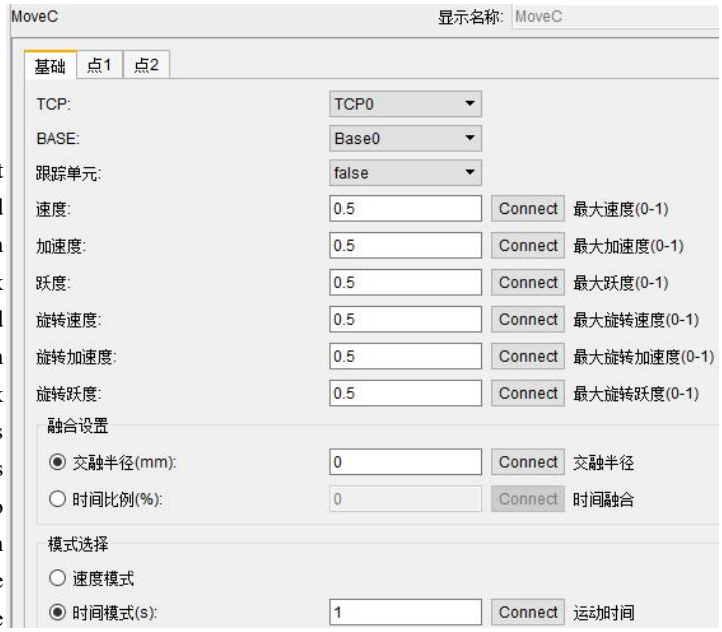
The following examples introduce the MoveC command :

Example 1



MoveC starts from P0, passes through the middle point P1, and finally moves to the end point P2.

Bat series



Tracking unit

Speed — Maximum speed (0-1)

Acceleration — Maximum acceleration (0-1)

Jerk — Maximum jerk (0-1)

Rotation speed — Maximum rotation speed (0-1)

Rotation acceleration — Maximum rotation acceleration (0-1)

Rotation jerk — Maximum rotation jerk(0-1)

Blending settings

Blending radius — Blending radius

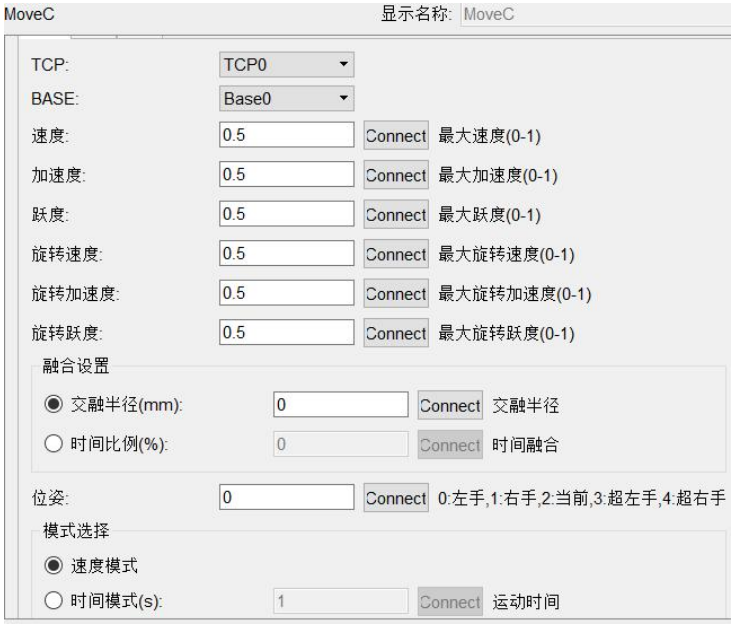
Time ratio — Time blending

Mode selection

Speed mode

Time mode — Motion time

Python series



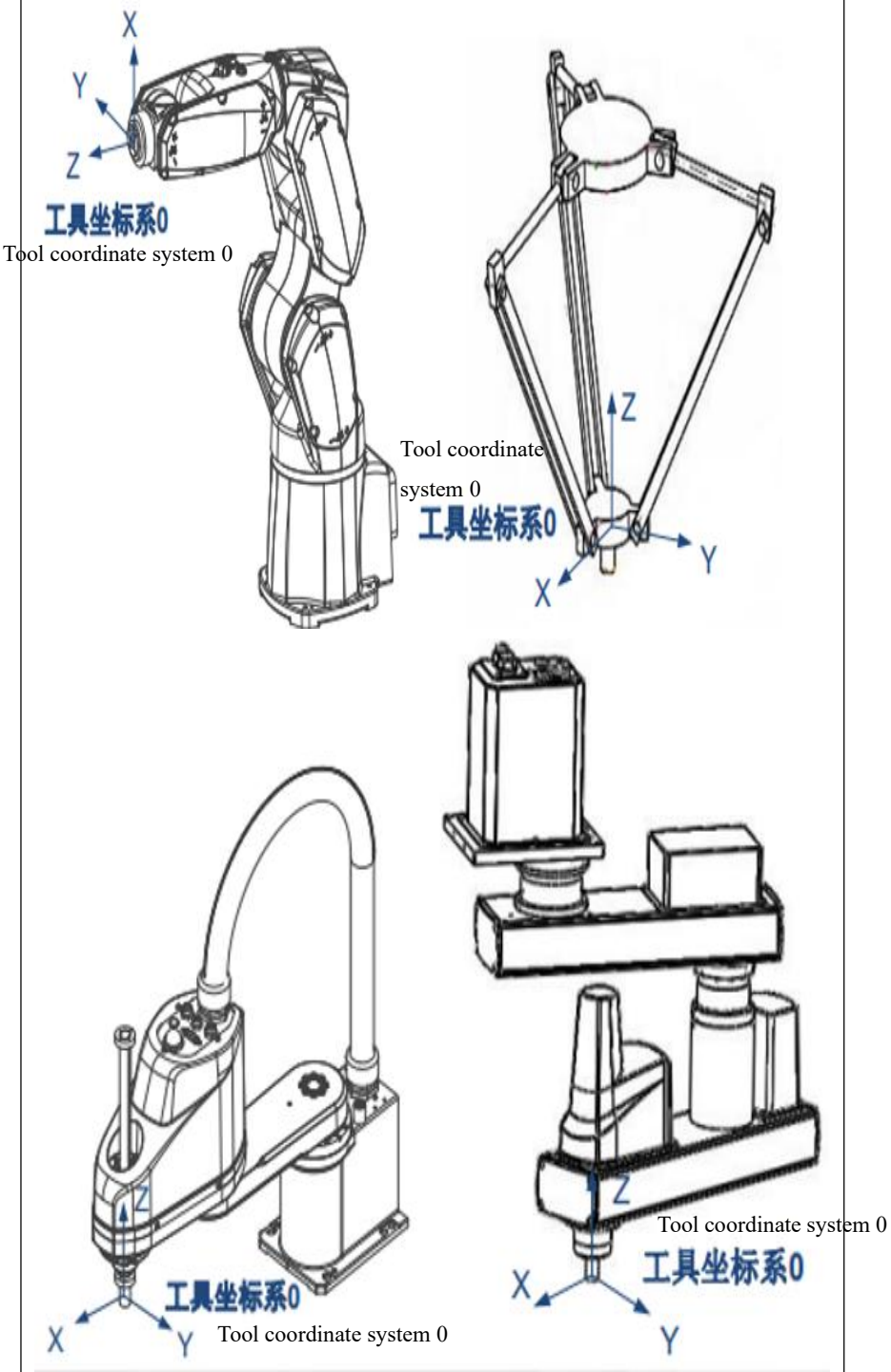
Speed — Maximum speed (0-1)
Acceleration — Maximum acceleration (0-1)
Jerk — Maximum jerk (0-1)
Rotation speed — Maximum rotation speed (0-1)
Rotation acceleration — Maximum rotation acceleration (0-1)
Rotation jerk — Maximum rotation jerk(0-1)
Blending settings
Blending radius — Blending radius
Time ratio — Time blending
Pose — 0: left hand, 1: right hand, 2: current, 3: beyond left hand, 4: beyond right hand.
Mode selection
Speed mode
Time mode — Motion time

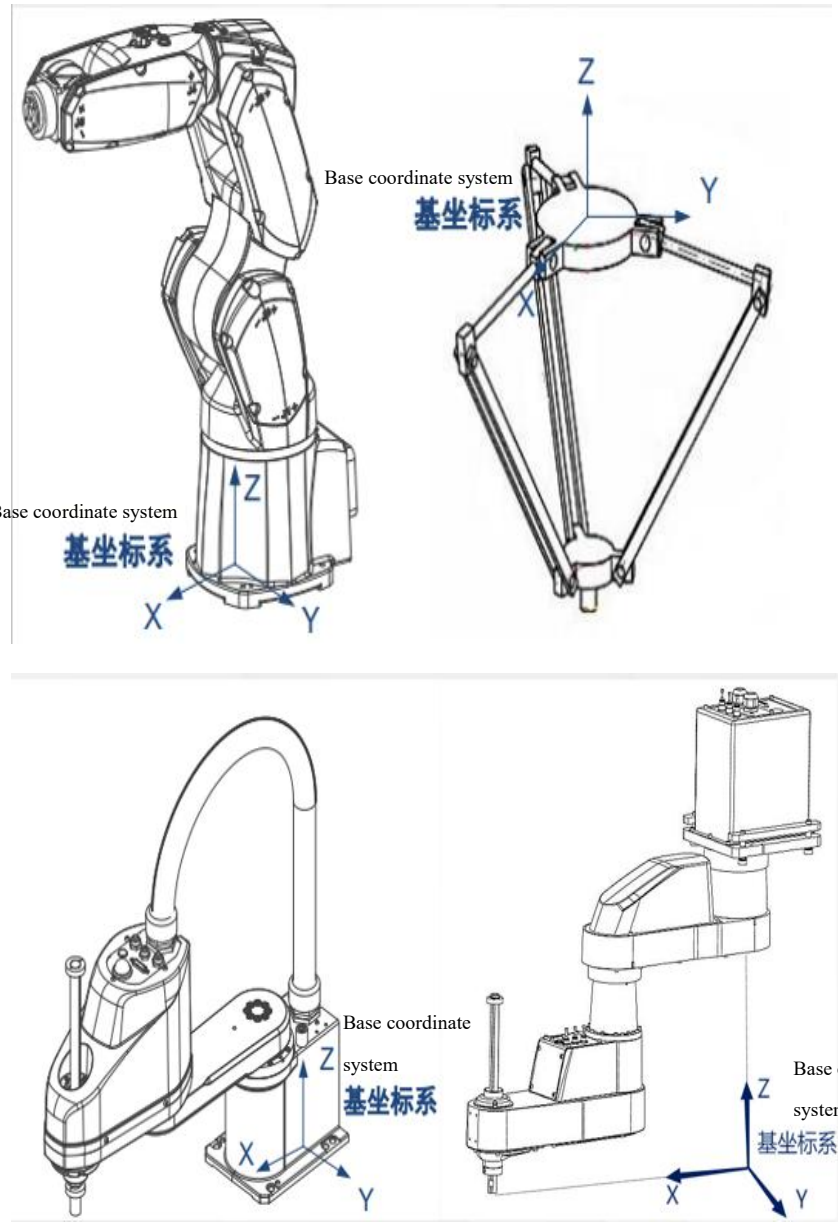
Mantis series

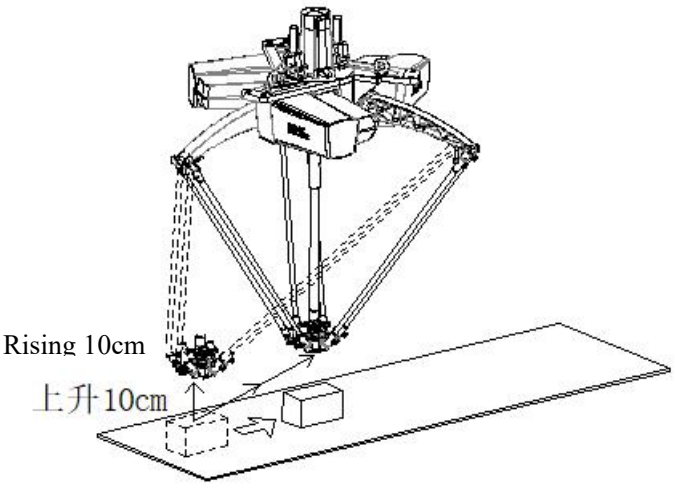
Speed — Maximum speed (0-1)
Acceleration — Maximum acceleration (0-1)
Jerk — Maximum jerk (0-1)
Rotation speed — Maximum rotation speed (0-1)
Rotation acceleration — Maximum rotation acceleration (0-1)
Rotation jerk — Maximum rotation jerk(0-1)
Blending settings
Blending radius — Blending radius
Time ratio — Time blending
Mode selection
Speed mode
Time mode — Motion time

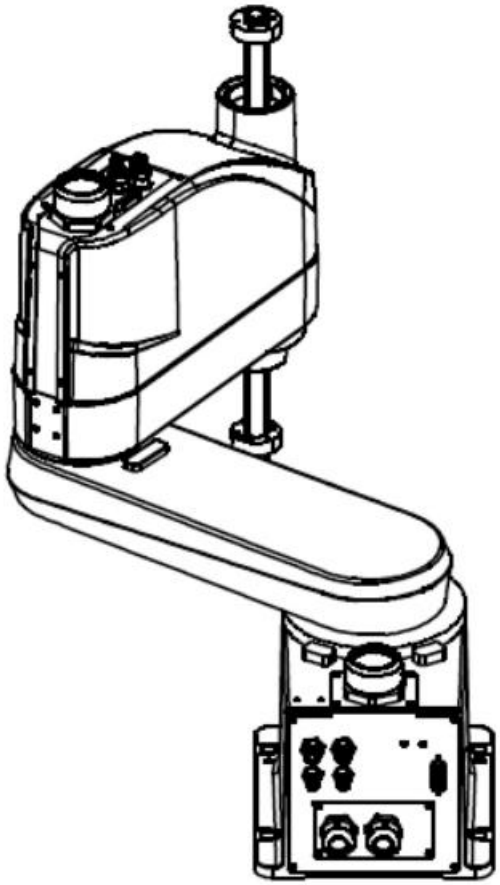
The parameters of the MoveC command are as above, and their specific meanings are as follows:

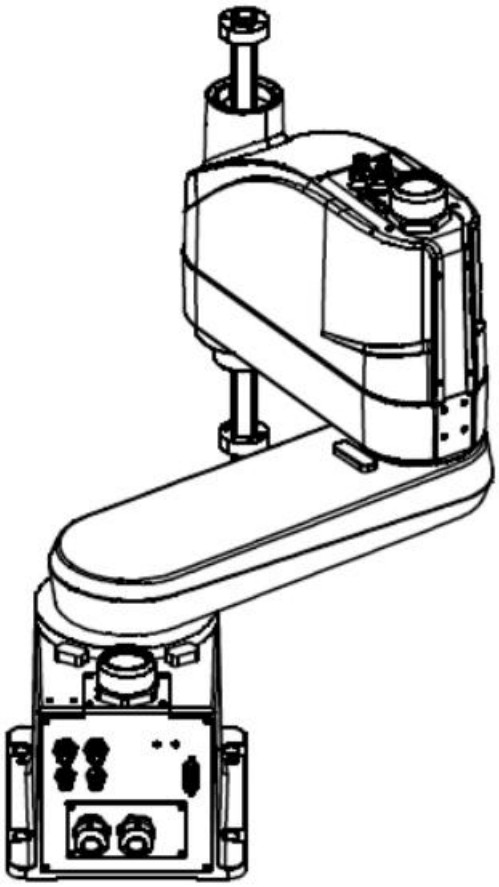
Basic column	Functions and parameters
TCP	Select the TCP coordinate system to be used. The tool coordinate system is attached to the tool. TCP (Tool Center Point, the origin of the tool coordinates) is used as the reference point to measure the robot's arrival position. Generally, the tool end point is taken as TCP, and the direction can be freely defined. In the YiKingSoft control system, up to 23 tool coordinate systems can be defined. Tool0 means no tool is used (tool0 cannot be changed by default). At this time, the tool coordinate system is located at the end of the robot body.

	
Base	Select the BASE coordinate system to be used. The base coordinate system is also called the robot coordinate system and is generally located at the root of the robot.

	
Tracking unit (Bat)	True to enable, false to disable. After Pick/PlaceMobileBox turns on the tracking unit, if the following MoveL/MoveC command also turns on the tracking unit, the MoveL/MoveC motion will also be superimposed with the conveyor belt tracking motion. Until a motion command that does not turn on the tracking unit is encountered, the following motion will no longer be superimposed with the tracking motion regardless of whether the tracking is turned on or not. Superimposed tracking conditions: 1. Pick/PlaceMobileBox turns on the tracking unit. 2. MoveL/MoveC behind Pick/PlaceMobileBox also turns on the tracking unit. 3. Pick/PlaceMobileBox tracking action is executed. 4. The tracking is not interrupted (if there is an command that does not turn on the tracking unit in the middle of multiple commands that turn on

	the tracking unit, the tracking is interrupted here, and the tracking unit cannot be tracked if it is turned on again later). As shown in the figure, the tracking unit is turned on and the robot follows the conveyor belt. 
Speed	Maximum speed percentage. For example, if the robot's running trajectory is 2 meters per second and the speed is set to 0.5, the robot's maximum speed during the execution of the trajectory is 2*0.5=1 meter per second.
Acceleration	The ratio of the change in the speed of an object to the time taken is called acceleration. The higher the percentage, the faster the speed increases within the range.
Jerk	Maximum jerk percentage
Rotation speed	The maximum speed percentage of the rotational motion part. The maximum speed at which the robot executes commands during the rotational motion.
Rotation acceleration	The acceleration percentage of the rotational motion part. The larger the value, the faster the rotation acceleration.
Rotation jerk	The rotation jerk is the derivative of the rotation acceleration. The derivative describes a degree of change, which refers to the change of the maximum deceleration of the descent.
Blending radius	The motion synthesis radius, in mm. (Details are introduced in PathPoint)
Time ratio %	Motion synthesis percentage (according to the command motion time)
Pose (Python)	Python model-specific parameters that determine the robot's pose. 0: left hand,

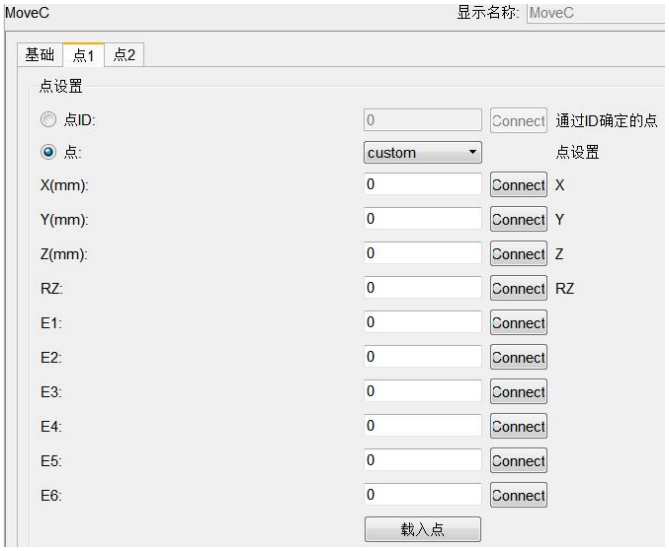
	 1: right hand,

	 2: current 3: beyond left hand (PythonF series) The robot runs in a left-handed pose. When the second axis is less than -180°, it is called beyond left hand. 4: beyond right hand (PythonF series) The robot runs in a right-handed pose. When the second axis is more than 180°, it is called beyond right hand.
Speed mode	The actual motion speed and acceleration of the robot are the product of these parameters and the speed bar in the figure above. Example: speed 0.5, speed bar 50%, the actual speed is 0.5*0.5=0.25.

Time mode	This action takes time as priority. If the action can be completed within the time specified by the time mode within the maximum speed and acceleration range of the robot, it will be completed according to the time. If the action cannot be completed within the time specified by the time mode, the action will be completed at the maximum speed and acceleration. The maximum speed and acceleration of the robot are still determined by the product of these parameters and the speed bar in the figure above. For example: Speed = Acceleration = Jerk = Rotation Speed = Rotation Acceleration = Rot Jerk = 0.5, Speed Bar 50%, Actual Maximum Speed = Acceleration = Jerk = Rotation Speed = Rotation Acceleration = Rot Jerk = 0.5*0.5 = 0.25, Time Mode = 3, assuming that the robot can complete this action at 20% of the speed acceleration, and does not exceed 25% of the maximum capacity, then the robot will complete this action at 20% of the speed acceleration in 3 seconds. When Time mode = 0.5, the robot cannot complete the action in 0.5 seconds even at 25% of the actual maximum capacity, so the robot will complete the action at 25% of the capacity.

Example 3

Bat series, Python series

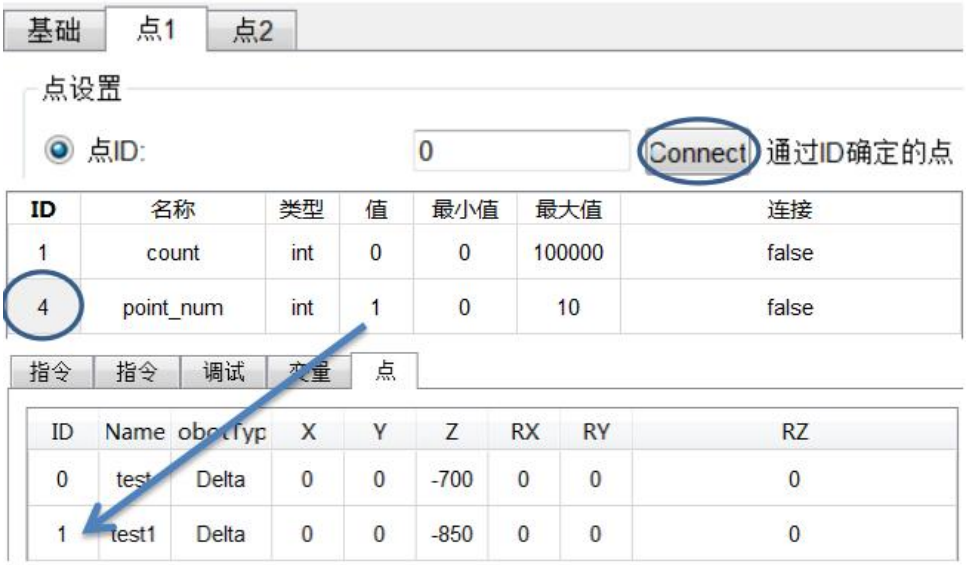


Mantis series

The functions of point 1 and point 2 in the MoveC command are as follows:

Point ID: Associate a path point in the Point tab through an int type variable in the "Variable" tab.

For example: Press the Connect button, all the int type variables in the variable tab will appear, double-click ID4, the current variable Point_num=1 will automatically associate with the path point of ID=1 in the Point tab, that is, assign the coordinates of the path point of ID=1 to point1 of MoveC.



Point: X(mm), Ymm), Z(mm), RZ are the X, Y, Z coordinate values of the path point and the rotation angle of the Z axis (4th axis). E1-E6 are the angle values of the external axes. You can enter the coordinate values manually, or you can jog the robot to the desired position and press "Load Point" to automatically pick up the current coordinate values of the robot. TargetSovleN (mantis) six-axis solving parameters, TargetSixAxisCfg (mantis) six-axis circle parameters.

Jog: Select manual mode with the key switch on the teach pendant - select the jog interface - press the enable key to the middle position (hear the sound of the motor powering on) - press the direction arrow (the robot moves, the corresponding coordinate value changes) to move the robot to the desired position - release the enable key - return to the programming interface and press "Load Point" - X (mm), Y mm), Z (mm), RZ are automatically filled in as the current coordinate values.

3.4 PathPoint - Path point command

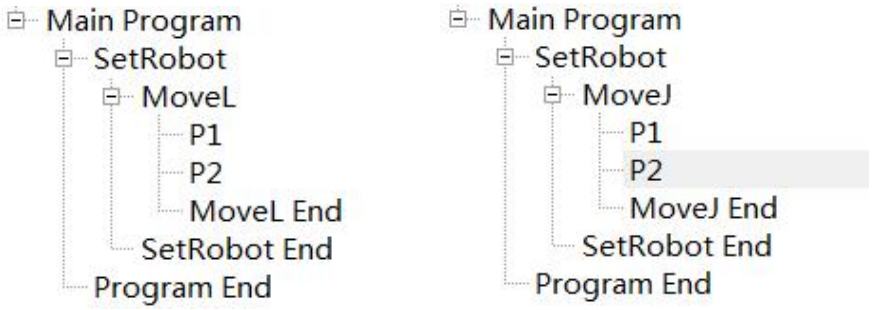
Command Usage

Path point coordinates. Specify a coordinate point in space. Users can name path points, such as P1.

Basic Example

The following examples introduce the PathPoint command :

Example 1



The application cases of the PathPoint command have been mentioned before. PathPoint is required in both the MoveJ command and the MoveL command. For example, the points P1 and P2 in the above two programs are both PathPoint commands.

Example 2

Bat series, Python series

Speed and acceleration

Using custom speed and acceleration

Speed — Maximum speed (0-1)

Acceleration — Maximum acceleration (0-1)

Jerk — Maximum jerk (0-1)

Rotation speed — Maximum rotation speed (0-1)

Rotation acceleration — Maximum rotation acceleration (0-1)

Rotation jerk — Maximum rotation jerk(0-1)

Blending settings

Blending radius — Blending radius

Time ratio — Time blending

Mode selection

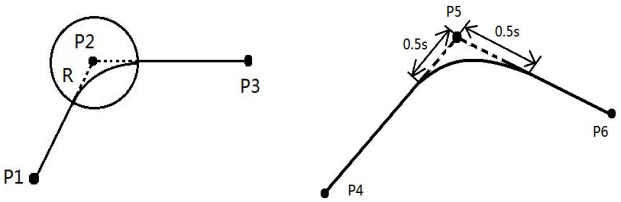
Speed mode

Time mode — Motion time

Mantis series

The parameters of the Pathpoint command are as above, and their specific meanings are as follows:

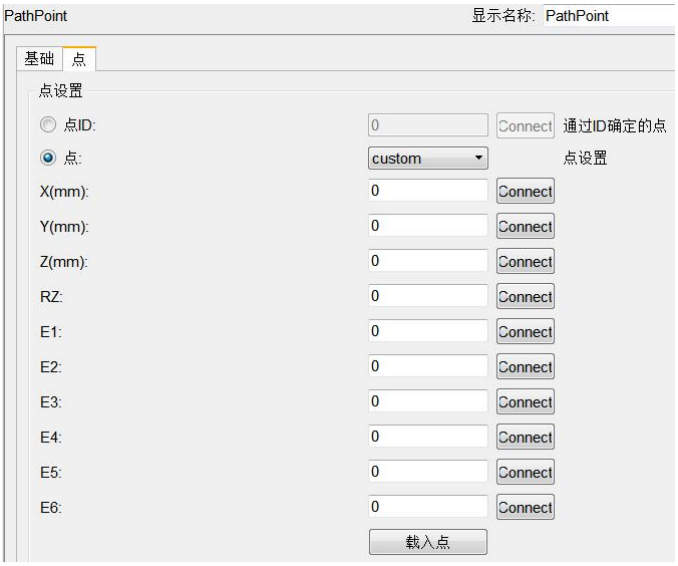
Basic tab	Functions and parameters
Name	Point name
TCP	Select the TCP coordinate system to be used. The tool coordinate system is attached to the tool. TCP (Tool Center Point, the origin of the tool coordinates) is used as the reference point to measure the robot's arrival position. Generally, the tool end point is taken as TCP, and the direction can be freely defined. In the Robotphoenix's YiKingSoft control system, up to 23 tool coordinate systems can be defined. Tool0 means no tool is used (tool0 cannot be changed by default). At this time, the tool coordinate system is located at the end of the robot body.
Base coordinate	Select the BASE coordinate system to be used. The base coordinate system is also called the robot coordinate system and is generally located at the root of the robot.
Speed and acceleration settings	When a check √ is placed in front of "Use custom speed and acceleration", the maximum speed, maximum acceleration, maximum jerk, maximum angular velocity, maximum angular acceleration, and maximum rotation jerk can be modified. These parameters determine the maximum motion capability of the robot from the current point to this point. If there is no check mark √ in front of "Use custom speed and acceleration", these parameters cannot be modified. The maximum motion capacity of the robot from the current point to this point depends on the maximum speed, maximum acceleration, maximum jerk,

	maximum rotation speed, maximum rotation acceleration, and maximum rotation jerk set in the previous MoveJ/MoveL.
Blending settings — Blending radius	The blending radius, in mm, the blending radius R of point P2 cannot be greater than half of the shortest distance between the two adjacent path points. For example: the distance between P1 and P2 is 10mm, and the distance between P2 and P3 is 20mm, then R cannot be greater than 5mm. When the robot Tcp enters the blending radius of point P2, it is considered that the robot has reached P2 and starts to move to the next point P3. During this process, the robot will not decelerate to 0. If the blending radius is set to 0, the robot will reach point P2 exactly, and the speed at point P2 will be reduced to 0. Setting the blending radius can make the robot's motions smoother.
Blending settings — Time ratio %	Compare the time taken for the two motions, take the shorter time and multiply it by the time ratio to get the synthesized time. As follows, assuming that the motion starts from P4 and ends from P5 to P6, the motion time from P4 to P5 is 2s, the motion time from P5 to P6 is 1s, and the time ratio of P5 point = 0.5. Compare the time taken by P4, P5 and P5, P6, and use the shorter time 1s*0.5=0.5s to get the time of the synthetic motion. 
Mode selection - Speed mode	The actual motion speed and acceleration of the robot are the product of these parameters and the speed bar in the figure above. Example: speed 0.5, speed bar 50%, the actual speed is 0.5*0.5=0.25.
Mode selection - Time mode	This action takes time as priority. If the action can be completed within the time specified by the time mode within the maximum speed and acceleration range of the robot motion, it will be completed according to the time. If the action cannot be completed within the time specified by the time mode, the action will be completed at the maximum speed and acceleration. The maximum speed and acceleration of the robot are still determined by the product of these parameters and the speed bar in the figure above. For example: Speed = Acceleration = Jerk = Rotation Speed = Rotation Acceleration = Rot Jerk = 0.5, Speed Bar 50%, Actual Maximum Speed = Acceleration = Jerk = Rotation Speed = Rotation Acceleration = Rot Jerk = 0.5*0.5 = 0.25, Time Mode = 3, assuming that the robot can complete this action at 20% of the speed acceleration, and does not exceed 25% of the maximum capacity, then the robot will complete this action at 20% of the speed acceleration in 3 seconds. When Time mode = 0.5, the robot cannot complete the action in 0.5 seconds even at 25% of the actual maximum capacity, so the robot will complete the action at 25% of the capacity.
Use custom parameters (Mantis)	Whether to use custom parameters. When a robot is in motion, its arms and other parts may interfere with

	surrounding objects and collide with them. When a collision occurs during motion, it is necessary to detect the collision immediately and perform an emergency stop on the robot. This is called collision detection.
Monitoring validity (Mantis)	Whether collision can be detected. 0 means invalid, 1 means valid.
Monitoring level (Mantis)	Monitoring sensitivity, which increases as the value decreases.

Example 3

Bat series



Python series

Mantis series

The settings of the point of the PathPoint command are the same as those of the MoveC command.
Point ID: Associate a path point in the Point tab through an int type variable in the "Variable" tab.
"Point" tab: X(mm), Ymm), Z(mm), RZ are the X, Y, Z coordinate values of the path point and the rotation angle of the Z axis (4th axis). E1-E6 are the angle values of the external axes. TargetSovleN (mantis) six-axis solving parameters, TargetSixAxisCfg (mantis) six-axis circle parameters. You can enter the coordinate values manually, or you can jog the robot to the desired position and press "Load Point" to automatically pick up the current coordinate values of the robot.

3.5 PathPointDt - Relative path point command

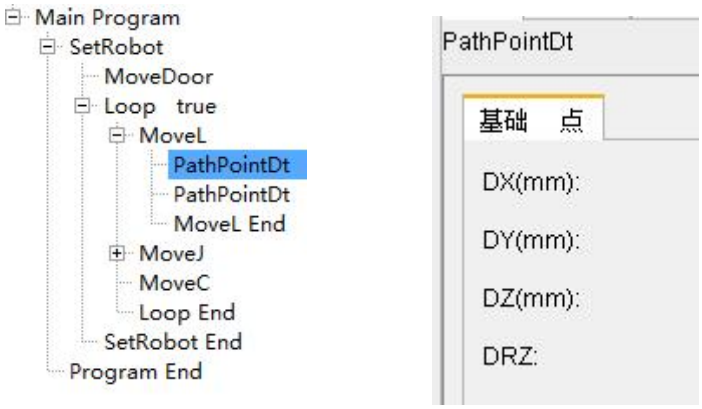
Command Usage

Relative path point. (Specify a vector distance, and the robot will leave the current point according to this vector.)

Basic Example

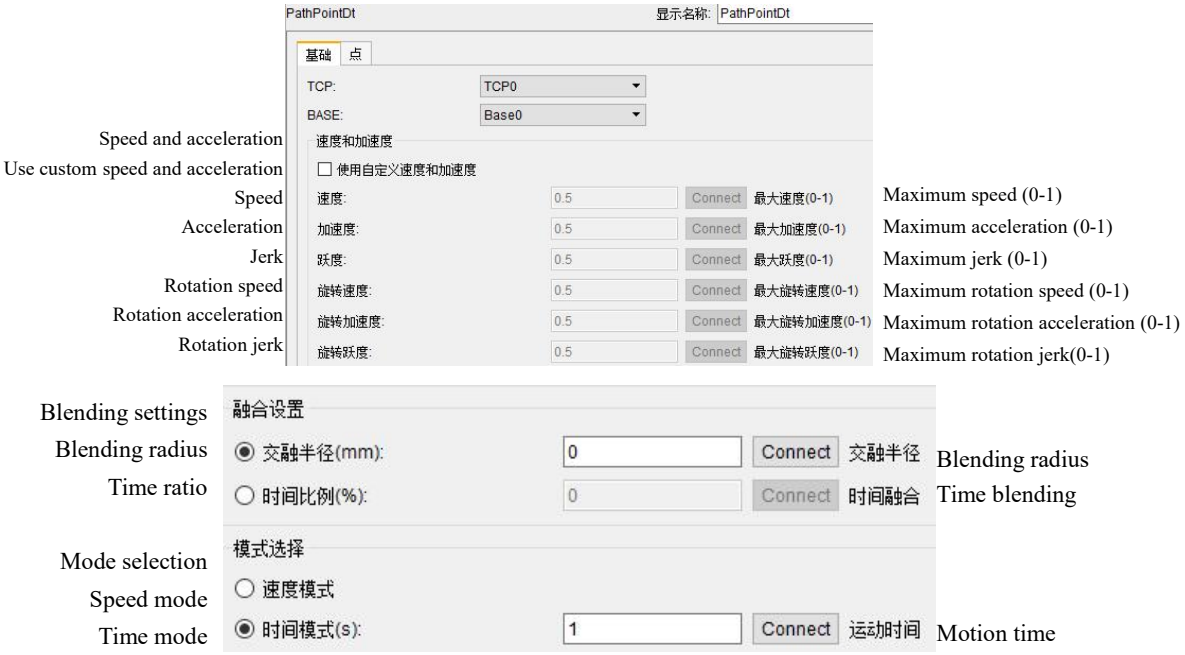
The following examples introduce the PathPointDt command :

Example 1



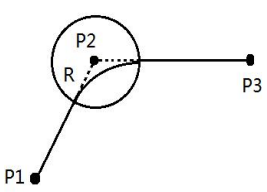
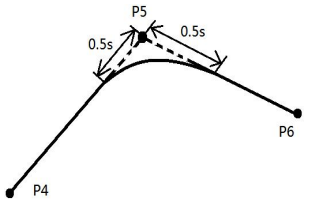
In the above program, the robot takes the current position of the robot as the starting point, and follows the PathPointDt command to travel a relative distance. For example, in the PathPointDt command, 30 is written to DX, 20 to DY, 10 to DZ, and 90 to DRZ, and the current position of the robot is (200, 100, -100, 0). After running PathPointDt, the position of the robot is (230, 120, -90, 90).

Example 2



Basic tab: Same usage as the Basic tab of PathPoint.

Basic tab	Functions and parameters
Name	Point name
TCP	Select the TCP coordinate system to be used. The tool coordinate system is attached to the tool. TCP (Tool Center Point, the origin of the tool coordinates) is used as the reference point to measure the robot's arrival position. Generally, the tool end point is taken as TCP, and the direction can be freely defined. In the Robotphoenix's YiKingSoft control system, up to 23 tool coordinate systems can be defined. Tool0 means no tool is used (tool0 cannot be changed by default). At this time, the tool coordinate system is located at the end of the

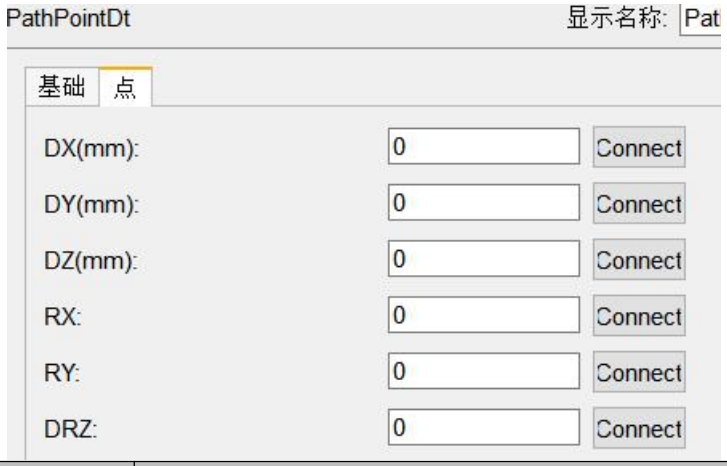
	robot body.
Base coordinate	Select the BASE coordinate system to be used. The base coordinate system is also called the robot coordinate system and is generally located at the root of the robot.
Speed and acceleration settings	When a check √ is placed in front of "Use custom speed and acceleration", the maximum speed, maximum acceleration, maximum jerk, maximum angular velocity, maximum angular acceleration, and maximum rotation jerk can be modified. These parameters determine the maximum motion capability of the robot from the current point to this point. If there is no check mark √ in front of "Use custom speed and acceleration", these parameters cannot be modified. The maximum motion capacity of the robot from the current point to this point depends on the maximum speed, maximum acceleration, maximum jerk, maximum rotation speed, maximum rotation acceleration, and maximum rotation jerk set in the previous MoveJ/MoveL.
Blending settings — Blending radius	The blending radius, in mm, the blending radius R of point P2 cannot be greater than half of the shortest distance between the two adjacent path points. For example: the distance between P1 and P2 is 10mm, and the distance between P2 and P3 is 20mm, then R cannot be greater than 5mm. When the robot Tcp enters the blending radius of point P2, it is considered that the robot has reached P2 and starts to move to the next point P3. During this process, the robot will not decelerate to 0. If the blending radius is set to 0, the robot will reach point P2 exactly, and the speed at point P2 will be reduced to 0. Setting the blending radius can make the robot's motions smoother.
Blending settings — Time ratio %	Compare the time taken for the two motions, take the shorter time and multiply it by the time ratio to get the synthesized time. As follows, assuming that the motion starts from P4 and ends from P5 to P6, the motion time from P4 to P5 is 2s, the motion time from P5 to P6 is 1s, and the time ratio of P5 point = 0.5. Compare the time taken by P4, P5 and P5, P6, and use the shorter time 1s*0.5=0.5s to get the time of the synthetic motion.  
Mode selection - Speed	The actual motion speed and acceleration of the robot are

mode	the product of these parameters and the speed bar in the figure above. Example: speed 0.5, speed bar 50%, the actual speed is 0.5*0.5=0.25.
Mode selection - Time mode	This action takes time as priority. If the action can be completed within the time specified by the time mode within the maximum speed and acceleration range of the robot motion, it will be completed according to the time. If the action cannot be completed within the time specified by the time mode, the action will be completed at the maximum speed and acceleration. The maximum speed and acceleration of the robot are still determined by the product of these parameters and the speed bar in the figure above. For example: Speed = Acceleration = Jerk = Rotation Speed = Rotation Acceleration = Rot Jerk = 0.5, Speed Bar 50%, Actual Maximum Speed = Acceleration = Jerk = Rotation Speed = Rotation Acceleration = Rot Jerk = 0.5*0.5 = 0.25, Time Mode = 3, assuming that the robot can complete this action at 20% of the speed acceleration, and does not exceed 25% of the maximum capacity, then the robot will complete this action at 20% of the speed acceleration in 3 seconds. When Time mode = 0.5, the robot cannot complete the action in 0.5 seconds even at 25% of the actual maximum capacity, so the robot will complete the action at 25% of the capacity.

Example 3

Bat series, Python series

Mantis series



Point tab	Functions and parameters
DX	The distance the target point moves on the X axis relative to the current point.
DY	The distance the target point moves on the Y axis relative to the current point.
DZ	The distance the target point moves on the Z axis relative to the current point.
RX（Mantis）	The rotation angle of the target point relative to the current point on the X axis.
RY（Mantis）	The rotation angle of the target point relative to the current point on the Y axis.
DRZ	The rotation angle of the target point relative to the current point on the Z axis.

3.6 MoveAbsJ - Absolute position motion command

Command Usage

Absolute position motion command. Specify a coordinate point in space and the rise and fall distance.

Basic Example

The following examples introduce the MoveAbsJ command :

Example 1

Bat series, Python series

Base

Speed and acceleration

Use custom speed and acceleration

Speed

Acceleration

Jerk

Rotation speed

Rotation acceleration

Rotation jerk

Maximum speed (0-1)

Maximum acceleration (0-1)

Maximum jerk (0-1)

Maximum rotation speed (0-1)

Maximum rotation acceleration (0-1)

Maximum rotation jerk(0-1)

Blending settings

Blending radius

Time ratio

Blending radius

Time blending

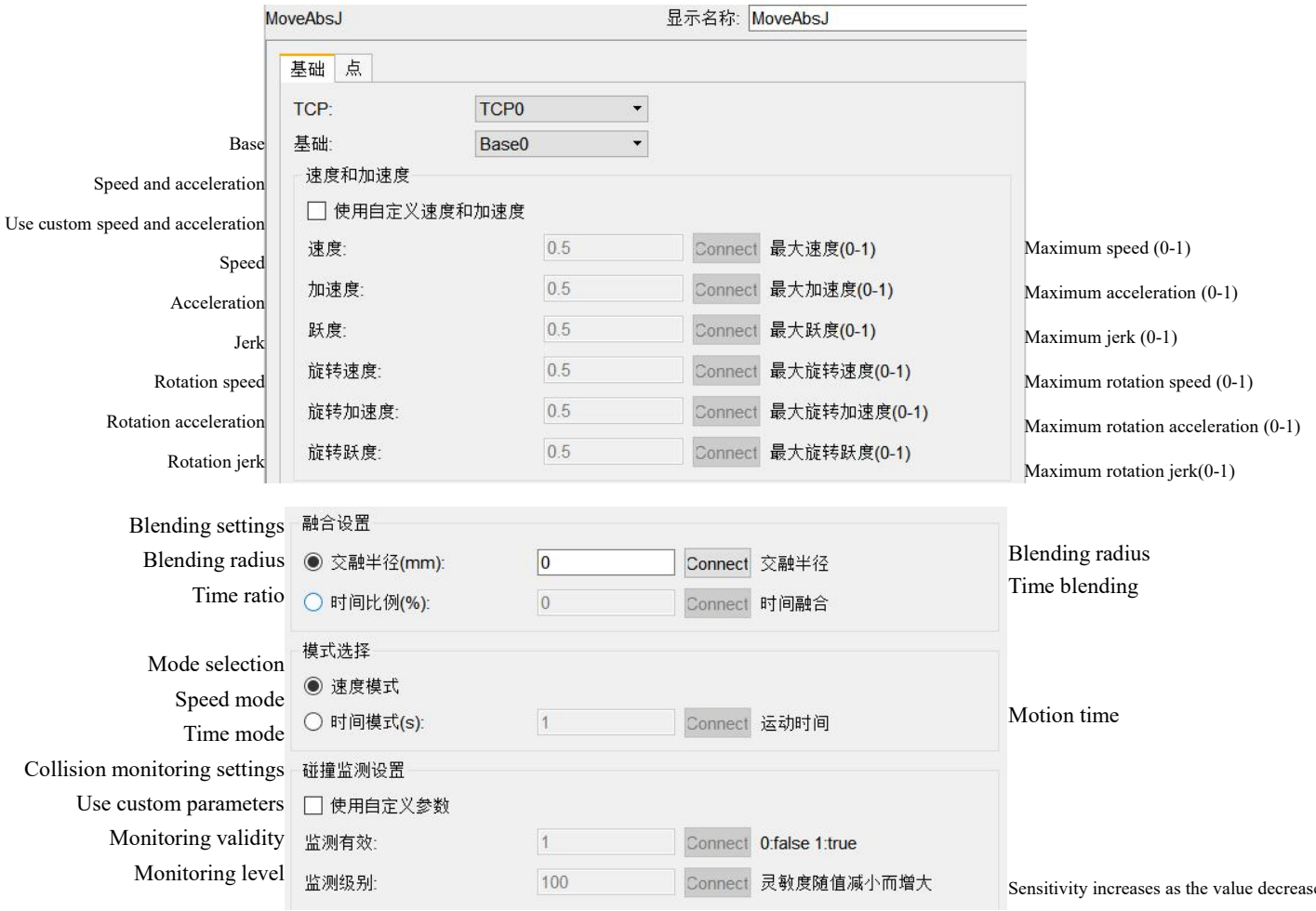
Mode selection

Speed mode

Time mode

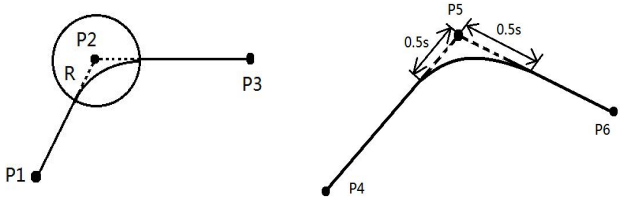
Motion time

Mantis series



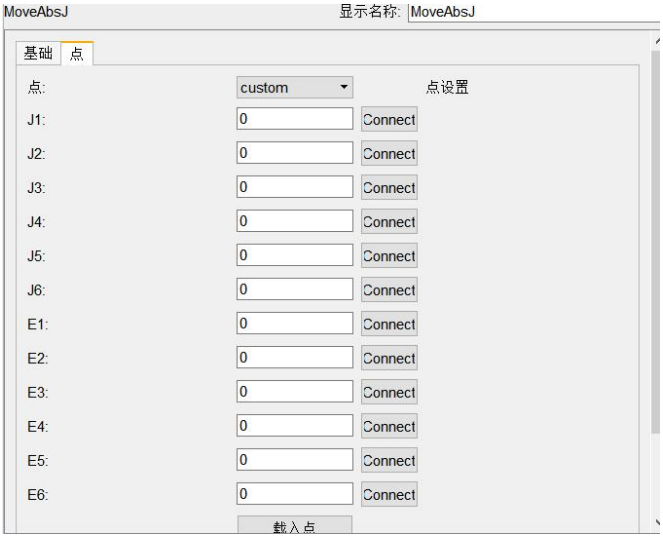
The parameters of the Pathpoint command are as above, and their specific meanings are as follows:

Basic tab	Functions and parameters
Name	Point name
TCP	Select the TCP coordinate system to be used. The tool coordinate system is attached to the tool. TCP (Tool Center Point, the origin of the tool coordinates) is used as the reference point to measure the robot's arrival position. Generally, the tool end point is taken as TCP, and the direction can be freely defined. In the Robotphoenix's YiKingSoft control system, up to 23 tool coordinate systems can be defined. Tool0 means no tool is used (tool0 cannot be changed by default). At this time, the tool coordinate system is located at the end of the robot body.
Base coordinate	Select the BASE coordinate system to be used. The base coordinate system is also called the robot coordinate system and is generally located at the root of the robot.
Speed and acceleration settings	When a check √ is placed in front of "Use custom speed and acceleration", the maximum speed, maximum acceleration, maximum jerk, maximum angular velocity, maximum angular acceleration, and maximum rotation jerk can be modified. These parameters determine the maximum motion capability of the

	robot from the current point to this point. If there is no check mark √ in front of "Use custom speed and acceleration", these parameters cannot be modified. The maximum motion capacity of the robot from the current point to this point depends on the maximum speed, maximum acceleration, maximum jerk, maximum rotation speed, maximum rotation acceleration, and maximum rotation jerk set in the previous MoveJ/MoveL.
Blending settings — Blending radius	The blending radius, in mm, the blending radius R of point P2 cannot be greater than half of the shortest distance between the two adjacent path points. For example: the distance between P1 and P2 is 10mm, and the distance between P2 and P3 is 20mm, then R cannot be greater than 5mm. When the robot Tcp enters the blending radius of point P2, it is considered that the robot has reached P2 and starts to move to the next point P3. During this process, the robot will not decelerate to 0. If the blending radius is set to 0, the robot will reach point P2 exactly, and the speed at point P2 will be reduced to 0. Setting the blending radius can make the robot's motions smoother.
Blending settings — Time ratio %	Compare the time taken for the two motions, take the shorter time and multiply it by the time ratio to get the synthesized time. As follows, assuming that the motion starts from P4 and ends from P5 to P6, the motion time from P4 to P5 is 2s, the motion time from P5 to P6 is 1s, and the time ratio of P5 point = 0.5. Compare the time taken by P4, P5 and P5, P6, and use the shorter time 1s*0.5=0.5s to get the time of the synthetic motion. 
Mode selection - Speed mode	The actual motion speed and acceleration of the robot are the product of these parameters and the speed bar in the figure above. Example: speed 0.5, speed bar 50%, the actual speed is 0.5*0.5=0.25.
Mode selection - Time mode	This action takes time as priority. If the action can be completed within the time specified by the time mode within the maximum speed and acceleration range of the robot motion, it will be completed according to the time. If the action cannot be completed within the time specified by the time mode, the action will be completed at the maximum speed and acceleration. The maximum speed and acceleration of the robot are still determined by the product of these parameters and the speed bar in the figure above. For example: Speed = Acceleration = Jerk = Rotation Speed = Rotation Acceleration = Rot Jerk = 0.5, Speed Bar 50%, Actual Maximum Speed = Acceleration = Jerk = Rotation Speed = Rotation Acceleration = Rot Jerk = 0.5*0.5 = 0.25, Time Mode = 3, assuming that the robot can complete this action at 20% of the speed acceleration, and does not exceed 25% of the maximum capacity, then the robot will complete this action at 20% of the speed acceleration in 3 seconds. When Time mode = 0.5, the robot cannot complete the action in 0.5 seconds even at 25% of the actual maximum

	capacity, so the robot will complete the action at 25% of the capacity.
Use custom parameters (Mantis)	Whether to use custom parameters. When a robot is in motion, its arms and other parts may interfere with surrounding objects and collide with them. When a collision occurs during motion, it is necessary to detect the collision immediately and perform an emergency stop on the robot. This is called collision detection.
Monitoring validity (Mantis)	Whether collision can be detected. 0 means invalid, 1 means valid.
Monitoring level (Mantis)	Monitoring sensitivity, which increases as the value decreases.

Example 2



"Point" tab: J1-J6 are the rotation angles of the robot's six axes. E1-E6 are the angle values of the external axes. TargetSovleN (mantis) is the six-axis solving parameter, TargetSixAxisCfg (mantis) is the six-axis circle parameter. You can enter the coordinate value manually, or you can jog the robot to the desired position and press "Load Point" to automatically pick up the robot's current coordinate value.

3.7 MoveDoor - Door frame motion command

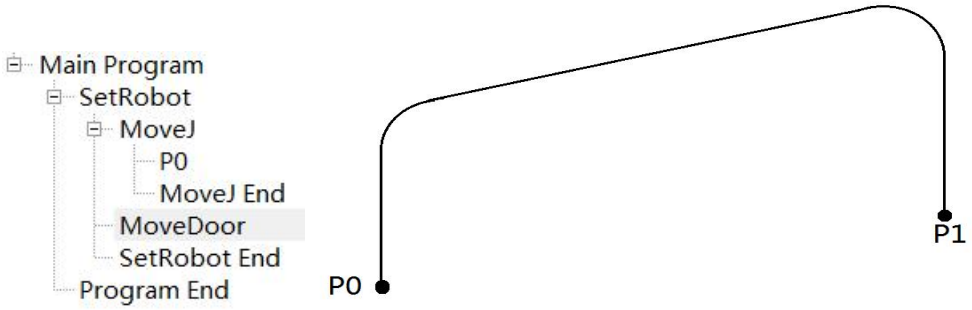
Command Usage

Door frame motion. Specify a coordinate point in space and the rise and fall distance. Mantis models do not support MoveDoor command.

Basic Example

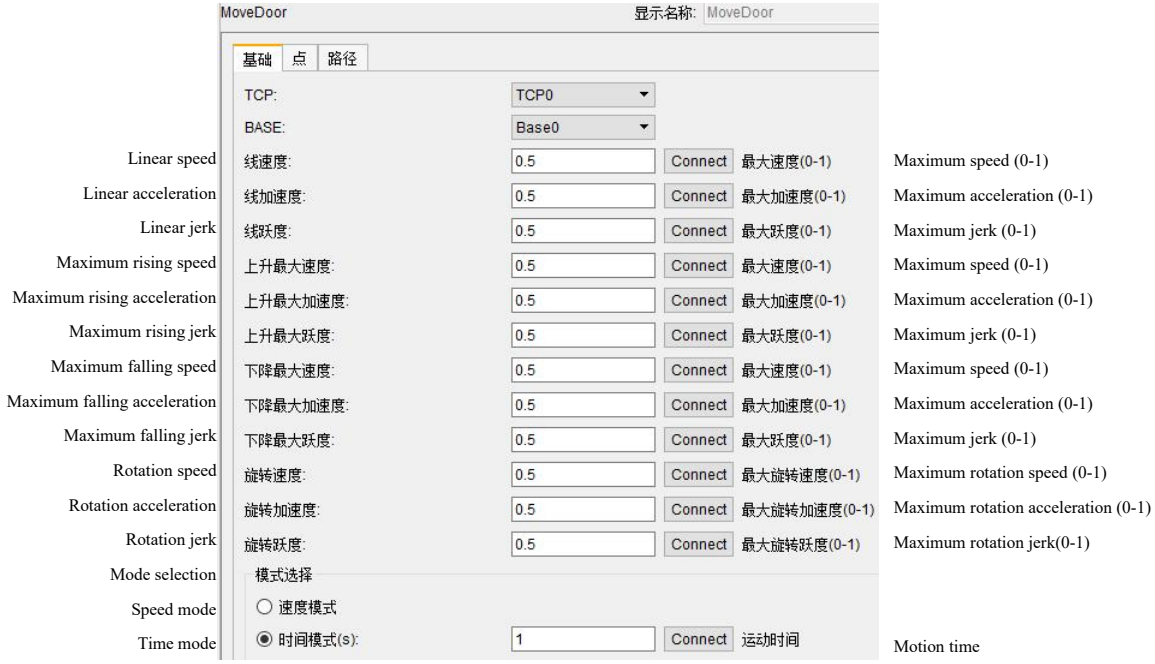
The following examples introduce the MoveDoor command :

Example 1



As shown in the above program, the robot runs the program, first runs P0 in the command “MoveJ”, and then runs the command “MoveDoor” after the operation is completed. The motion from point P0 to P1 follows a door frame motion.

Example 2

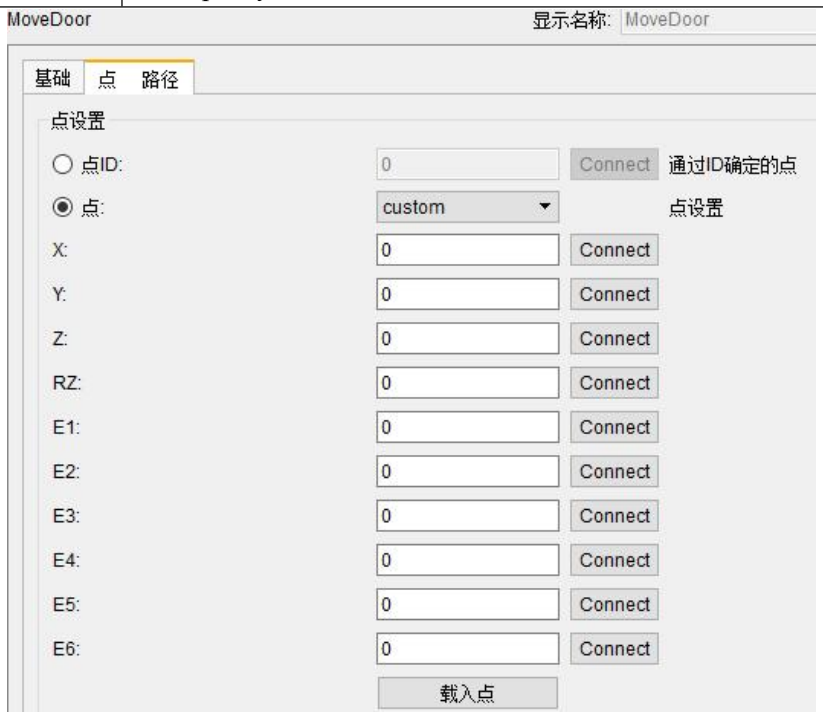


The functions of its basic tab are:

Basic tab	Functions and parameters
TCP	Select the TCP coordinate system to be used. The tool coordinate system is attached to the tool. TCP (Tool Center Point, the origin of the tool coordinates) is used as the reference point to measure the robot’s arrival position. Generally, the tool end point is taken as TCP, and the direction can be freely defined. In the Robotphoenix’s YiKingSoft control system, up to 23 tool coordinate systems can be defined. Tool0 means no tool is used (tool0 cannot be changed by default). At this time, the tool coordinate system is located at the end of the robot body.
BASE	Select the BASE coordinate system to be used. The base coordinate system is also called the robot coordinate system and is generally located at the root of the robot.
Linear speed	The maximum speed percentage of the linear motion part. For example, if the robot’s running trajectory is 2 meters per second and

	the linear speed is set to 0.5, the maximum speed of the robot during the execution of the trajectory is 2*0.5=1 meter per second
Linear acceleration	The percentage of acceleration in the linear motion part. Linear acceleration: It is the ratio of the change in velocity to the time it takes to occur. It is a physical quantity that describes how fast an object's velocity changes. The ratio of the change in an object's velocity to the time it takes is called acceleration. The higher the percentage, the faster the acceleration in the range. For example; the robot's current running speed is 1 meter per second, set to 2 meters per second, and the linear acceleration is set to 0.5 and 1 respectively. When the linear speed is 1, the robot's acceleration is faster.
Linear jerk	The percentage of speed in the linear motion part. Linear jerk is the derivative of linear acceleration, which describes a degree of change, which refers to the change of linear acceleration.
Maximum rising speed	The maximum speed percentage of the rising motion part. The maximum speed at which the robot executes the command during the rising process.
Maximum rising acceleration	Maximum rising jerk is the derivative of maximum rising acceleration, which describes a degree of change and refers to the change of maximum rising acceleration.
Maximum rising jerk	The percentage of acceleration in the rising motion part. The larger the value, the faster the acceleration.
Maximum falling speed	The maximum speed percentage of the falling motion part. The maximum speed at which the robot executes the command during the falling process.
Maximum falling acceleration	The percentage of acceleration in the falling motion part. The larger the value, the faster the deceleration.
Maximum falling jerk	The maximum falling jerk is the derivative of the maximum falling deceleration, which describes a degree of change and refers to the change of the maximum falling deceleration.
Rotation speed	The maximum speed percentage of the rotational motion part. The maximum speed at which the robot executes the command during the rotational motion.
Rotation acceleration	The percentage of acceleration in the rotational motion part. The larger the value, the faster the rotation acceleration.
Rotation jerk	The rotation jerk is the derivative of the rotation acceleration, which describes a degree of change and refers to the change of the maximum falling deceleration.
Mode selection - Speed mode	The actual motion speed and acceleration of the robot are the product of these parameters and the speed bar in the figure above. Example: speed 0.5, speed bar 50%, the actual speed is 0.5*0.5=0.25.
Mode selection — Time mode	This action takes time as priority. If the action can be completed within the time specified by the time mode within the maximum speed and acceleration range of the robot motion, it will be completed

	according to the time. If the action cannot be completed within the time specified by the time mode, the action will be completed at the maximum speed and acceleration. The maximum speed and acceleration of the robot are still determined by the product of these parameters and the speed bar in the figure above. For example: Speed = Acceleration = Jerk = Rotation Speed = Rotation Acceleration = Rot Jerk = 0.5, Speed Bar 50%, Actual Maximum Speed = Acceleration = Jerk = Rotation Speed = Rotation Acceleration = Rot Jerk = 0.5*0.5 = 0.25, Time Mode = 3, assuming that the robot can complete this action at 20% of the speed acceleration, and does not exceed 25% of the maximum capacity, then the robot will complete this action at 20% of the speed acceleration in 3 seconds. When Time mode = 0.5, the robot cannot complete the action in 0.5 seconds even at 25% of the actual maximum capacity, so the robot will complete the action at 25% of the capacity.



The function of its point tab:

Point ID: Associate a path point in the Point tab through an int type variable in the “Variable” tab.

“Point” tab: X(mm), Ymm), Z(mm), RZ are the X, Y, Z coordinate values of the path point and the rotation angle of the Z axis (4th axis). E1-E6 are the angle values of the external axes. You can enter the coordinate values manually, or you can jog the robot to the desired position and press “Load Point” to automatically pick up the current coordinate values of the robot.

Bat series, Mantis series

Python series

The functions of its path tab:

Path tab	Functions and parameters
Middle segment mode (Python)	Mid-section running mode of the MoveDoor command
Rising distance	Rising distance of the door frame motion
Falling distance	Falling distance of the door frame motion
Rising limit	The highest point limit of the rise
Falling limit	The highest limit of the fall
Rising linear ratio	The ratio of linear motion in the rising motion
Falling linear ratio	The ratio of linear motion in the falling motion
Rotation ratio	Rotation percentage

Chapter 4 Input/Output

4.1 SetOutput - Set the output level signal

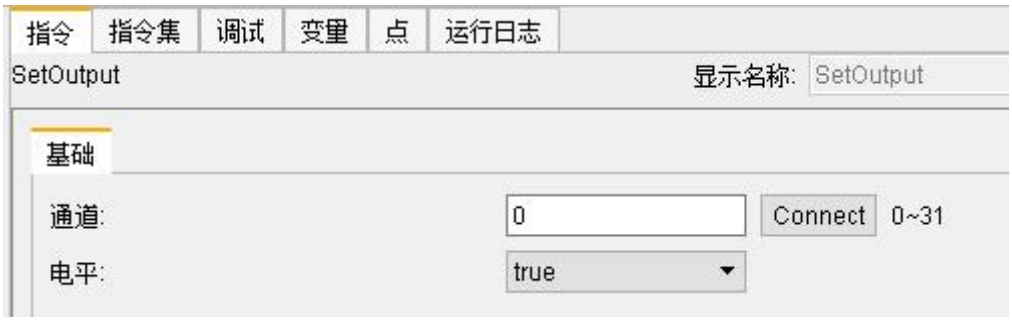
Command Usage

Make a certain output port output signal.

Basic Example

The following example introduces the SetOutput command :

Example 1



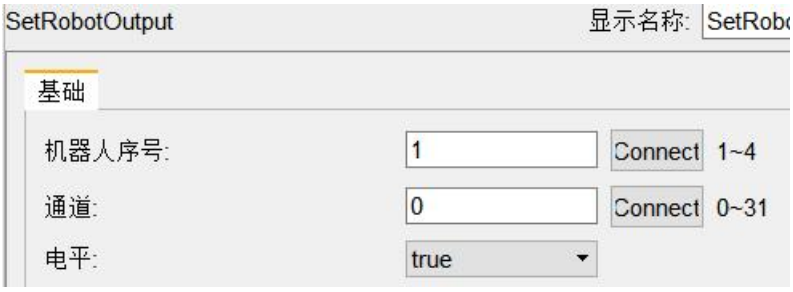
For the SetOutput command, select the IO channel. In the above example, 0 is the current channel. When the level is in the “true” state, it outputs a signal by default. When the level is in the “false” state, it is turned off by default.

4.2 SetRobotOutput - Specify the robot to output level signal

Command Usage

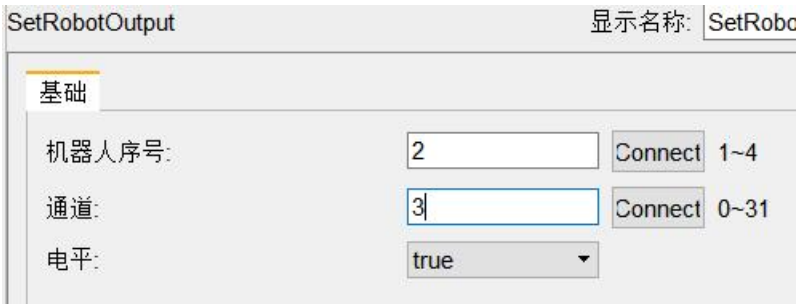
Compared with the SetOutput introduced earlier, it only has an additional RobotNO parameter, which is used to control and monitor the IO of other robots. Commands without the RobotNO parameter can only control and monitor the local IO. Controlling and monitoring the IO of other robots is very convenient in dual or multi-machines. One master can be equipped with up to three slaves, so the input value of the RobotNO parameter can only be an integer from 1 to 4, 1 represents the master and 2, 3, 4 represent the 1st, 2nd, and 3rd slaves.

Example 1



In the above example, Robot number 1 represents the master, and 0 is the current channel. When the level is in the “true” state, it outputs a signal by default. When the level is in the “false” state, it is turned off by default.

Example 2



In the above example, Robot number 2 represents the first slave, and 3 is the current channel. When the level is in the “true” state, it outputs a signal by default. When the level is in the “false” state, it is turned off by default.

4.3 SetOutputPulse - Output pulse signal

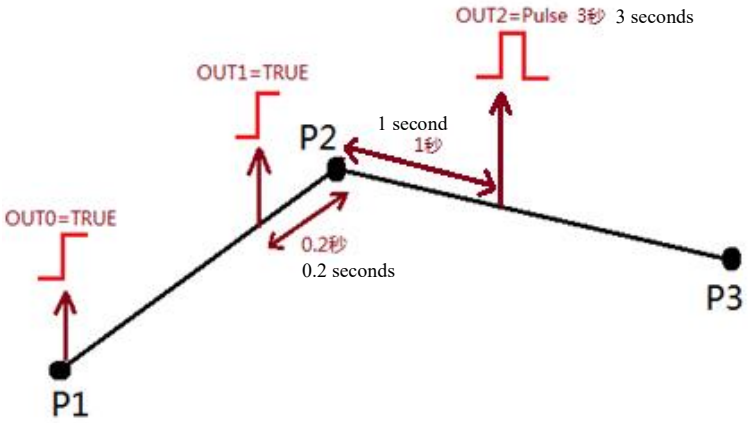
Command Usage

Make a certain output port output a pulse signal of specified duration.

Basic Example

The following example introduces the SetOutputPulse command :

Example 1



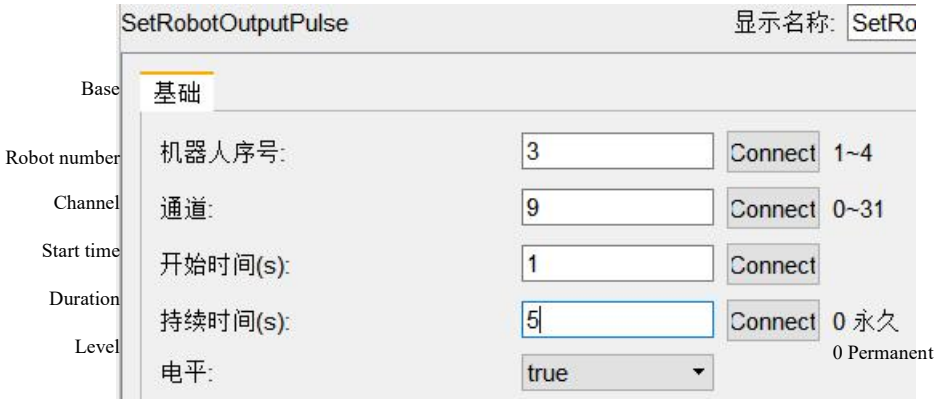
Taking the program in the above figure as an example, a signal output diagram is given. Based on the

original two commands, the “SetOutputPulse” command increases the duration, making the level in the “true” state, starting at 1 second and lasting for 3 seconds.
Note: For the same channel, if you call this command repeatedly, you need to ensure that the calling time interval is greater than the duration time, otherwise an IO error will be reported.

4.4 SetRobotOutputPulse - Specify the robot to output pulse signal

Command Usage
Compared with the SetOutputPulse introduced earlier, it only has an additional RobotNO parameter, which is used to control and monitor the IO of other robots. Commands without the RobotNO parameter can only control and monitor the local IO. Controlling and monitoring the IO of other robots is very convenient in dual or multi-machines. One master can be equipped with up to three slaves, so the input value of the RobotNO parameter can only be an integer from 1 to 4, 1 represents the master and 2, 3, 4 represent the 1st, 2nd, and 3rd slaves.

Example 1



Make the level of output channel 9 of robot slave 2 in the “true” state, starting at 1 second and lasting for 5 seconds.
Note: For the same channel, if you call this command repeatedly, you need to ensure that the calling time interval is greater than the duration time, otherwise an IO error will be reported.

4.5 SetOutputPWM - Output continuous pulse signal

Command Usage
Make a certain output port output pulse signal continuously (not supported yet)

4.6 SetRobotOutputPWM - Specify the robot to output continuous pulse signal

Command Usage
Compared with the SetOutputPWM introduced earlier, it only has an additional RobotNO parameter, which is used to control and monitor the IO of other robots. Commands without the RobotNO parameter can only control and monitor the local IO. Controlling and monitoring the IO of other robots is very convenient in dual or multi-machines. One master can be equipped with up to three slaves, so the input value of the RobotNO parameter can only be an integer from 1 to 4, 1 represents the master and 2, 3, 4 represent the 1st, 2nd, and 3rd slaves.

4.7 SetOutputBeforEnd - Output level signal in advance

Command Usage

Output a level signal a certain period of time before reaching the next motion point. This command cannot be used in a thread. The motion command bound to this command must have a clear motion command, such as MoveDoor, etc. It is not allowed to bind commands such as Pick/Place that may not generate motion when executed.

Basic Example

The following example introduces the SetOutputBeforEnd command :

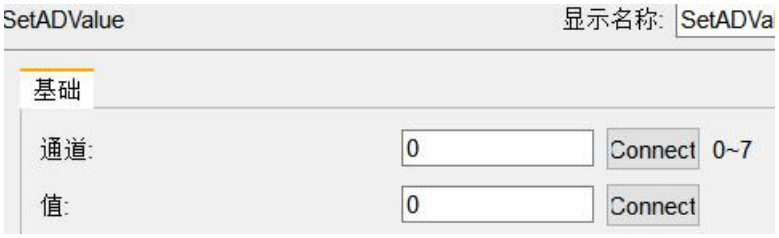
Example 1

In time mode, the corresponding channel sets the level to true 0.2S before the bound motion reaches the target point. If percentage mode is selected, the corresponding channel sets the level to true at 20% of the path before the bound motion reaches the target point.

4.8 SetADValue - Analog signal output

Command Usage

Analog signal output



Chapter 5 Settings

5.1 BaseOffset - Set the motion offset

Command Usage

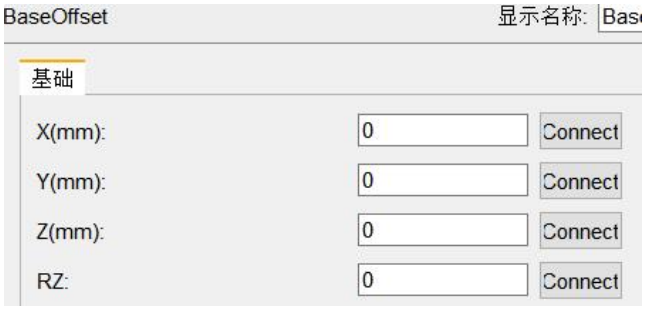
Set the motion offset and offset based on the basic coordinate system.

Basic introduction

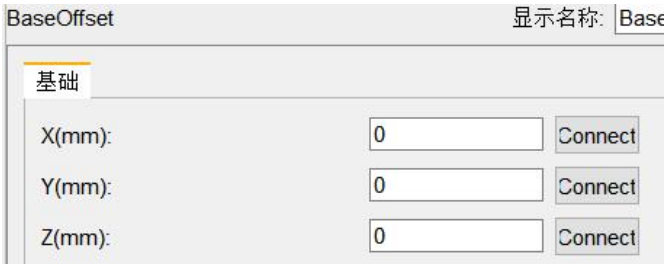
The following examples introduce the BaseOffset command :

Example 1

Bat series, Python series



Mantis series



Basic tab	Functions and parameters
X	The offset distance along the X axis, in millimeters
Y	The offset distance along the Y axis, in millimeters
Z	The offset distance along the Z axis, in millimeters
RZ（Bat、Python）	The angle of the offset around the Z axis

Note: The set Baseoffset is valid for each subsequent motion statement until the next Baseoffset statement is encountered.

Example 2

The X, Y, Z, and RZ coordinates of the base coordinate system are offset by 1mm, 2mm, 3mm, and 4 degrees respectively.

5.2 ToolOffset - Set the motion offset

Command Usage

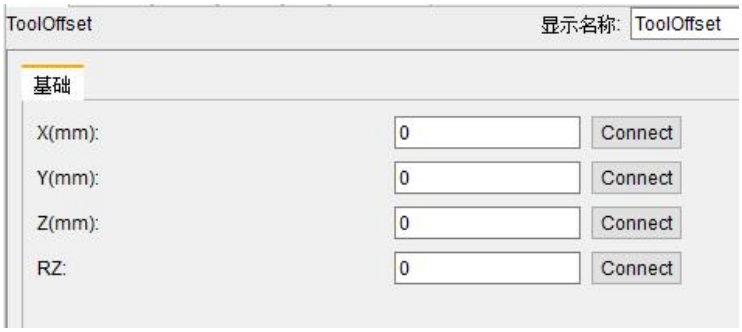
Set the motion offset and offset based on the tool coordinate system.

Basic introduction

The following examples introduce the ToolOffset command :

Example 1

Bat series, Python series



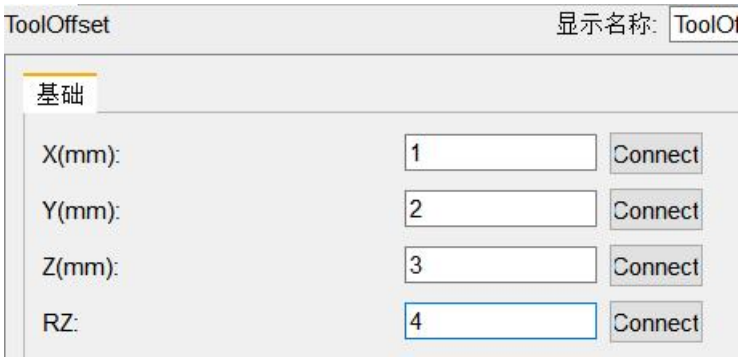
Mantis series

Its ToolOffset basic tab: offset based on tool coordinates.

Basic tab	Functions and parameters
X	The offset distance along the X axis
Y	The offset distance along the Y axis
Z	The offset distance along the Z axis
RX	The angle of the offset around the X axis
RY	The angle of the offset around the Y axis
RZ	The angle of the offset around the Z axis

Note: The set Tooloffset is valid for each subsequent motion statement until the next Tooloffset statement is encountered

Example 2



The X, Y, Z, and RZ coordinates of the tool coordinate system are offset by 1 mm, 2 mm, 3 mm, and 4 degrees respectively.

5.3 SetCuboidSpace - Set the monitoring area

Command Usage

Set the monitoring area and use it with the function IsCurPickObjInCuboidSpace(CuboidSpaceID)

or IsCurPosInCuboidSpace(CuboidSpaceID，TCPID).

Basic Example

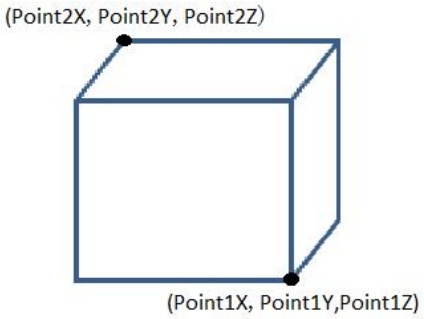
The following examples introduce the SetCuboidSpace command :

Example 1

Its basic tab functions:

Basic tab	Functions and parameters
CuboidSpaceID	Area ID number. Multiple areas can be set
Point1X	The starting value of the X coordinate in the cuboid area
Point1Y	The starting value of the Y coordinate in the cuboid area
Point1Z	The starting value of the Z coordinate in the cuboid area
Point2X	The end value of the X coordinate in the cuboid area
Point2Y	The end value of the Y coordinate in the cuboid area
Point2Z	The end value of the Z coordinate in the cuboid area

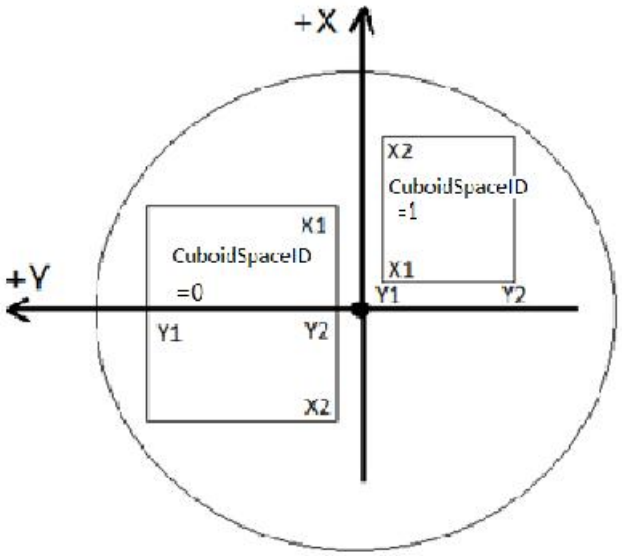
Example 2



If the object is within the CuboidSpaceD=0 area, the return value of IsCurPickObjInCuboidSpace(0) is true.
If the object is within the CuboidSpaceID=1 area, the return value of IsCurPickObjInCuboidSpace(1) is true.

If the TCP1 is within the CuboidSpaceD=0 area, the return value of IsCurPosInCuboidSpace(0，1) is true.

If the TCP1 is within the CuboidSpaceID=1 area, the return value of IsCurPosInCuboidSpace(1，1) is true.



5.4 SetStopMode - Stop mode setting

Command Usage

Stop mode setting

Basic Example

The following example introduces the SetStopMode command :

Example 1

Its basic tab functions:

Basic tab	Functions and parameters
Mode	Stop mode. When Mode=0, after pressing the End button or the End All button, if there is a stop subprogram Stop_Subprogram(), the motion will be stopped after the current motion command is executed or after the next two motion commands, and the stop subprogram is executed. If there is no stop subprogram, it will stop immediately after the current motion command is executed.
	Mode=1, it is used in conjunction with the IsStopTriggered() function. When the End button or EndAll button is pressed, the return value of the IsStopTriggered() function is true. It can be used to respond to the stop signal at a specific location and execute the corresponding subprogram.
	When Mode=2, after pressing the End button or the End All button, if there is a stop subprogram, it is the same as Mode=0. If there is no stop subprogram, the motion stops immediately at the current position and the program ends.

5.5 SetFixedSpeed - Speed bar settings

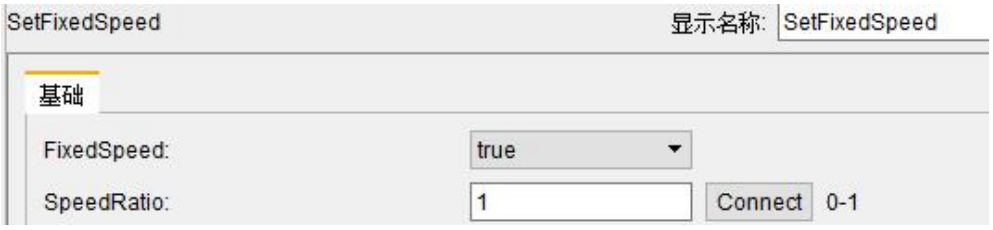
Command Usage

Set the speed percentage in automatic running mode.

Basic Example

The following example introduces the SetFixedSpeed command :

Example 1



When the parameter FixedSpeed is true, the speed of all motion statements following SetFixedSpeed is not controlled by the speed bar, and all run at the percentage specified in the parameter SpeedRatio until a SetFixedSpeed statement with a false parameter is encountered.

When the parameter FixedSpeed is false, the speed of all motion statements following SetFixedSpeed is controlled by the speed bar until a SetFixedSpeed statement with a true parameter is encountered.

Its basic tab parameters:

Basic tab	Functions and parameters

FixedSpeed	true—Subsequent motion statements are not controlled by the speed bar and only run at the speed specified by SpeedRatio. False—Subsequent motion statements are controlled by the speed bar.
SpeedRatio	Specify the speed at which action commands will be executed when FixedSpeed is set to true.

5.6 VarArrayInit - Multiple variable initialization

Command Usage

Initialize multiple variables at one time, up to 10 variables at a time.

Basic introduction

The following example introduces the VarArrayInit command :

Example 1

The variables associated with output 1 to output 6 are initialized to 0.

Basic tab	Functions and parameters
Initial value	The target value for variable initialization
Number of initialized variables	Output 1 - Output 10 is the number of initialization assignments to be made
Output 1 - Output 10	The variable value to be initialized must be associated with 10 variables, otherwise the addition will fail. If it needs to be initialized, associate the variable normally, and if it does not need to be initialized, associate a useless temporary variable.

5.7 AddLog - Add log output

Command Usage

The command can output log information to the running log. By viewing the log, you can determine the program running process, which is used for program debugging or alarm error location, etc.

Basic Example

The following examples introduce the AddLog command :

Example 1

After executing the above commands, you will see the “Data abnormality” information in the running log, which can be used to locate the command position that caused this exception.

Basic tab	Functions and parameters
Log information	Printed information content, can be associated with String variables
Repeat	Whether to deduplicate the printed content to facilitate log viewing.

Example 2

After executing the above command, the AddLog command will be executed 5 times. Because the repeat parameter is false, the “Data abnormality” information will only be seen once in the running log.

ID	Robot	Function code	Information
ID	**机器人**	**功能码**	**信息**
15	1	7039	数据错误 Data error
14	0	1002	----Start ./cmds/1.xml----

Example 3

After executing the above command, the AddLog command will be executed 5 times. Because the repeat parameter is true, the “Data abnormality” information will be seen once in the running log.

ID	Robot	Function code	Information	Time
ID	**机器人**	**功能码**	**信息**	**时间**
23	1	7039	数据错误4 Data error 4	2023-11-03 14:20:54:062
22	1	7039	数据错误3 Data error 3	2023-11-03 14:20:54:060
21	1	7039	数据错误2 Data error 2	2023-11-03 14:20:54:057
20	1	7039	数据错误1 Data error 1	2023-11-03 14:20:54:055
19	1	7039	数据错误 Data error	2023-11-03 14:20:54:052
18	0	1002	----Start ./cmds/1.xml----	2023-11-03 14:20:54:042

5.8 ConfJ - Whether to use solving parameters

Command Usage

Dedicated commands for Mantis series robots, whether to use six-axis solving parameters.

Basic Introduction

The following example introduces the ConfJ command :

Example 1

If ConfJ is true, the TargetSovleN parameter in the command after ConfJ is valid. If it is false, the TargetSovleN parameter in the command after ConfJ is treated as 0.

5.9 SetLoad - Set load

Command Usage

Set the current motion load. The subsequent motion commands will automatically adjust the speed according to the current load. You need to configure the load in Function Operation->LOAD configuration. When editing commands, the payload created by the current robot is automatically loaded.

Basic Example

The following example introduces the SteLoad command :

Example 1

Its basic tab functions:

Basic tab	Functions and parameters
LOAD	Load name

5.10 SetMoveDoorSolveMode - Set the door frame motion solve mode

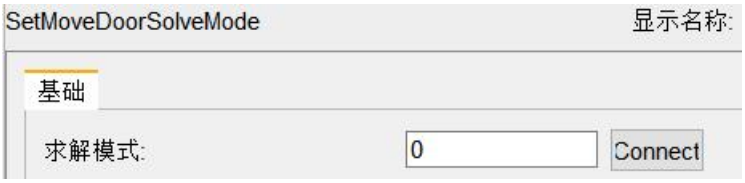
Command Usage

Set the door frame motion solve mode for Python series models. The subsequent MoveDoor motion commands will be run the solution according to the set mode.

Basic Example

The following example introduces the SetMoveDoorSolveMode command :

Example 1



Its basic tab functions:

Basic tab	Functions and parameters
Solving mode	0: Do not allow left and right hand switching 1: Allow left and right hand switching

5.11 GetDoubleRunTime - Get running time

Command Usage

Get the time from the current moment to the robot starting to run the program, in seconds (s).

Basic Example

The following example introduces the GetDoubleRunTime command :

Example 1

Get the current moment value and output it to the double type variable time1. If the value is 10.2, the robot command program has been running for 10.2S at the current moment.

Version Change History

Version Number	Modifications	Modified by	Approver	Modification time
V1.0	Reorganized the overall command manual layout, format, and content, added introductions, command examples, page numbers, and directory command explanations	Fu Guocheng		2022\11\10
V1.1	Revised based on version 2.8.9(RU2)	Chen Biao		2023\11\29
V1.2	Review and improvement	Cui Rongxiang		2023\12\04