Robotic AR Language

1. Overview of AR Language

Types of Commands	Symbols of Commands	Explanations of Commands
Arithmetic Operators	+	Addition
	-	Subtraction
	*	Multiplication
	/	Division
	//	Integer division, it is used for obtaining the maximal integer which is not greater than the result.
	%	Remainder
	^	Exponential operator
	-	Negative
	~	XOR
	&	AND
	|	OR
	~	NOT
	<<	Left shift operator
	>>	Right shift operator
Relational Operators	==	Equal to
	~=	Not equal to
	<=	Less than or equal to
	>=	Greater than or equal to
	<	Less than
	>	Greater than
Logical Operators	or	Logic “OR”
	not	Logic “NOT”
	and	Logic “AND”
	false	False (Notice: nil is also false)
	true	True
General Symbols	#	Solve the length of table array
	=	Assignment operator
	--	Single-line comments
	--[[	Starting line of Multi-lines comments
	--]]	Ending line of multi-lines comments
	( )	Used for function’s definition/call, and expression’s calculation
	{ }	Used for defining a table array.
	[ ]	Operator of elements in table array
	::	Define jump-position label of goto command
	;	Ending operator, this can be ignored.

	,	Used for definition, call and multi-variables assignment of function parameters, and definition of table array
	.	Used for accessing elements of table array
	..	Connection operator of characters
	…	Variable parameters of defined function
Commands of Process Control	if then else elseif end	Conditional branch commands
	while do end	Control command for cycling
	for do end	Control command for cycling
	repeat until	Control command for cycling
	goto	A jump without condition
	function end	Commands defined by user functions
General Keys	break	Jump for/while/repeat loop
	local	Define local variables
	nil	Variable is null
	return	Return a value of call function
Definitions of Robotic Axis	AX	X axis NO. under Cartesian coordinate system
	AY	Y axis NO. under Cartesian coordinate system
	AZ	Z axis NO. under Cartesian coordinate system
	AC	C axis NO. under Cartesian coordinate system
	J1	J1 joint
	J2	J2 joint
	J3	J3 joint
	J4	J4 joint
Robotic Global Variables	ON	Open state
	OFF	Close state
	p0~p999	Name of robotic default points
String manipulation	string.find	This function is used to search a pattern in a given string, then return its location
	string.match	This function is used to search a pattern in a given string, then return the matched string
	String.sub	Cut off the given string(s) from ith to jth
	string.gsub	If some characters are same with the pattern in a given string, replace them with the pattern
	string.char,string.byte	Used to convert character and the corresponding number.
	string.format	Used to format the given string, which has the same function with printf() in C language.
Global Robotic Variables	ON	Open state
	OFF	Close state
	p0~p2999	Name of robotic default points
	MovJR	The command that controls each joint to move a relative angle.
	MArchP	The command that controls robot to move with arch

		under PTP mode.
	MArc	The command that moves to absolute position from current position (Cartesian coordinate system) with arc interpolation mode.
	MCircle	A command of circle interpolation under Cartesian coordinate system
Commands of Movement Parameters	AccJ	Set a proportion of acceleration to affect MovJ / MovJR / MovP / MovPR / MArchP commands' accelerating time.
	SpdJ	Set a proportion of speed to affect MovJ / MovJR / MovP / MovPR / MArchP commands' running speed
	AccL	Set the acceleration of line movement to affect MovL/ MovLR/ MArchL/ Marc/ MCircle commands' accelerating time (unit is mm/s2)
	SpdL	Set the speed of line movement to affect MovL/ MovLR/ MArchL/ Marc/ MCircle commands' running speed (unit is mm/s)
Commands of Program Management	Delay	Delay command (unit is milliseconds)
	Exit	Exit the running program
	Pause	Pause the running program
General Commands	X	A command that builds the absolute points of specified X axis under Cartesian coordinate system
	Y	A command that builds the absolute points of specified Y axis under Cartesian coordinate system
	Z	A command that builds the absolute points of specified Z axis under Cartesian coordinate system
	C	A command that builds the absolute points of specified C axis under Cartesian coordinate system
	XYZC	A command that builds the absolute points of specified XYZC axis under Cartesian coordinate system
Commands of Input/Output	DI	Read the state of input port
	DO	Read the state of output port
	WDI	Read the state of one input port. AR program will be continued to run until this signal is effective
	WDO	Read the state of one output port. AR program will be continued to run until this signal is effective
Commands of Coordinate System	SetU	Set the current user coordinate system of robot
	WrU	Modify the data of user coordinate system
	SetT	Set the current tool coordinate system of robot
	WrT	Modify the data of tool coordinate system
	CacU	Build a new user coordinate system
	U2U	Transform **Cartesian coordinates** within user0 ~ user9
	V2Tool	Calculate a new tool

	getcart	Obtain current cartesian coordinate of robot's end
	CacU	Build a new user coordinate system
	CacT	Build a new tool coordinate system with two point
	encoderget	Obtain pulse value of corresponding encoder
Commands of Pallet	SetPlt	A command that sets the palletizing numbers
	GetPlt	A command that gets the data point of palletizing
	SET_PLT	sets palletizing parameters
	GET_PLT	Obtain position information of each point on the plate
	GetPLTPos	Obtain information: whether pallet is full、current palletizing number and current palletizing position
	ResetPLT	Reset the palletizing number
	SetArcPlt	Set Arc palletizing parameters
	GetArcPlt	Obtain position information of each point on the Arc plate
Commands of Servo Management	MotOn	Open servo enables of all the axis
	MotOff	Close servo enables of all the axis
	DragMode	Set robot to drag mode
Communication Commands	RecCom	Receive data from RS232 serial port
	SendCom	Send data to RS232 serial port
	SetCom	Set communication parameters of RS232 serial port
	ClrCom	Clear the receiving buffer of RS232 serial port
	sysnetclr	Clear the receiving buffer of network
	sysnetget	Read network data with unblock mode
	sysnetsend	Send network data
	sysnetcatch	Read network data with block mode
	CloseNet	Close connection of TCP network
	OpenNet	Build a TCP network
	ConnectNet	Connect to TCP network
	RecvNet	Receive data with TCP network
	WriteNet	Sent data with TCP network
	publicread	Read the data from GlobalData list
	publicwrite	Write data to GlobalData list
Commands of Vision	CCDrecv	Receive data which sent from a camera
	CCDtrigger	Trigger camera to take a photo
	CCDsent	Send character string to a camera
	CCDclr	Clear the network IP
	CCDoffset	Visual deviation compensation
	GetDynCCDPos	Transform the coordinate of dynamic camera to robot coordinate
Commands of Follow-camera	FollowInit	Initial parameters about follow-camera
	SetDynCatch	Open or close follow-grasping task
	GetCatchSpace	Obtain whether the workpiece has reached the grasping

		area
	SetCatch	Carry out the follow task
	GetCatchState	Obtain the catch state
	SynOver	Over the synchronization
	GetTrigger	Obtain the trigger state
	SetViewData	Send the received data to controller, then save to Cache queue
Commands of Debugging	print	Print the output of user debugging data
	Error	Terminate the running AR program and give error information
Commands of Point	Point	Call the points from point list except for CPU1
	new	Write user-defined point to DATA.PTS of current project
	teach	Write current point to DATA.PTS of current project
Commands of system	syswork	Set the working state of system
	sysstate	Obtain the state of system or current of each axis
	sysrate	Set the global speed rate
	systime	Obtain the clock time of system
Commands of Modbus	ReadRegW	Read the specified address of 16-bit word from PLC register
	ReadRegDW	Read the specified address of 32-bit word from PLC register
	WriteRegW	Write 16-bit data to specify address of PLC register
	WriteRegDW	Write 32-bit data to specify address of PLC register
Commands of File Operation	fopen	Open file
	fsize	File size
	fwrite	Write data to file
	fread	Read data from file
	fseek	Seek file to move file pointer to the appointed position
	feof	End file
	fclose	Close file
Commands of Queue Operation	qexist	Judge whether the queue exists
	qcreate	Create a new queue
	qpush	Push(write) data to the queue
	qpop	Pop(delete) the first data from the queue
	qfront	Fetch the first data from the queue
	qpopfront	Fetch the first data from the queue and then delete it
	qempty	Judge whether the queue is empty
	qsize	Calculate size of the queue
	qdestroy	Delete the queue

Note: the AR language is the size of the language, the user must be in accordance with the provisions of the operator used, and the table of all instructions in the user can only be used in

accordance with instructions, cannot be redefined.

For example：

```
Local X --Define variable X
X=10     --Assign 10 to variable X
```

X is already defined as system command, so it may cause error if it is redefined by user.

1.1 Arithmetic Operators

Symbols of Commands	Explanations of Commands
+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Integer division, it is used for obtaining the maximal integer which is not greater than the result.
%	Remainder
^	Exponential operator
-	Negative
~	XOR
&	AND
|	OR
~	NOT
<<	Left shift operator
>>	Right shift operator

Arithmetic operators are used for arithmetic function of kinds of real numbers, and bit arithmetic function of integer data.

"+": it can also realize the addition of two points' data. Example 1 is listed below:

```
local point1=p1
local point2=p2
local point3 =p3
```

- **Notice**: customized data points cannot be added directly. Data points that can be added directly must be defined in the robot system (p0~p2999 in DATA.PTS).

1.2 Relational Operators

Symbols of Commands	Explanations of Commands
==	Equal to
~=	Not equal to
<=	Less than or equal to
>=	Greater than or equal to
<	Less than

>	Greater than

Relational operators are applied in conditional judgments of process control commands.

Example 2：

```
local    a =10
local    b=20
if a == b then
…
end
if a < b then
…
end
if a < 30 then
…
end
```

== : it can also be used for comparing variables of string.

```
local    a = "ROBOT"
if a == "ROBOT" then
…
end
```

1.3 Logical Operators

Symbols of Commands	Explanations of Commands
or	Logic “OR”
not	Logic “NOT”
and	Logic “AND”
false	False (Notice：**nil** is also false)
true	True

They are also used in conditional judgments of process control commands. Example 3 is shown below:

```
if    5 or 6 then      --True
…
end
local    a =30
local    b=true
if a and b then    --True
…
end
local a=0
local b=1
if a and b then    --True
…
end
local a=0
local b=nil
if a and b then      --False (nil)
…
end
```

◆ **Notice:** In AR language, only **nil** and **false** are not true; and others are true including **0**.

1.4 General Symbols

Symbols of Commands	Explanations of Commands
#	Solve the length of table array
=	Assignment operator
--	Single-line comments
--[[	Starting line of Multi-lines comments
--]]	Ending line of multi-lines comments
( )	Used for function's definition/call, and expression's calculation
{ }	Used for defining a table array.
[ ]	Operator of elements in table array
::	Define jump-position label of "goto" command
;	Ending operator, this can be ignored.
,	Used for definition, call and multi-variables assignment of function parameters, and definition of table array
.	Used for accessing elements of table array
..	Connection operator of characters
…	Variable parameters of defined function

Example 4:

```
local a ,b, c = 1,2,3          --use " ,"   symbol to write multi-variable assignment statements
p.x                            --use "."    symbol to access element of the array.
local   a={10,20,30}           --use "{}" symbol to define an array
a [1]                          --use "[]" symbol to access element of the array
::lab::                        --use "::" symbol to define a label of jumping position for "goto"
                                 command
local str = "1234".. "abc"     --use ".." to connect two or several string
```

◆ **Notice:** the subscript of variable in table array is start with 1. Such as, a [1] is equal to 10 in above example.

1.5 General Keywords

Symbols of Commands	Explanations of Commands
break	Jump out of **for/while/repeat** loop
local	Define local variables
global.*	Variables are shared within multiple CPUs in a same project
nil	Variable is null
return	Return a value of call function

local it is used to declare local variables in AR program

Examples：

```
local   a                      --Define a local variable a, which value is nil
local   b={10,10}              -- Define an one-dimensional table b
local   c={{10,20},{30,40}}    -- Define a two-dimensional table c
local   d= "ROBOT"             -- Define a string d
```

➢ Before introducing local variables, let's first introduce how global variables are defined in the code block in lua language. Global variables do not need to be declared, the global variable is created when a variable is assigned a value. Accessing to an uninitialized global variable has no error, but result is nil

```
print(b)    -->nil
b = 10
print(b)    -->10
-- If you want to delete a global variable, you just assign it to nil; So the variable b is like
that it's never been used before. In other words, the variable exists when and only when a
variable is not equal to nil. b =nil
print(b)    -->nil
```

➢ It is different from global variable when use **local** to create a local variable, which is only valid within the declared block of code.

```
x = 10
local i = 1              -- local to the chunk
while i<=x do
      local x = i*2      -- local to the “while” body
      print(x)           --> 2, 4, 6, 8, ...
      i = i + 1
end
if i > 20 then
      local x            -- local to the “then” body
      x = 20
      print(x +2)        -->22(the local one)
else
      print(x)           --> 10 (the global one)
end
print(x)                 --> 10 (the global one)
```

global.* Depending on the actual application, a project must contain multiple CPUs. The global variables (global.*) can solve the problem of sharing the same variable among multiple CPUs.

Instructions：

- Global variables are defined as global.*, such as **global.var**, **global.a** and **global.b**;
- Global array table must be one-dimensional**. Such as:**

  global={pos={x=0,y=0},a=1,b=2} is improper;
- Global (public) variables are limited to the sharing of variables between multiple CPUs in a project. The use of global variables in a single CPU can refer to the use of local instructions in the AR programming manual.

1.6 Basic LUA APIs

API Name	Description	Example	Result
math.pi	pi	math.pi	3.1415926
math.abs	absolute	math.abs(-2012)	2012
math.ceil	ceil	math.ceil(9.1)	10
math.floor	floor	math.floor(9.9)	9
math.max	max	math.max(2,4,6,8)	8
math.min	min	math.min(2,4,6,8)	2
math.pow	pow	math.pow(2,16)	65536
math.sqrt	sqrt	math.sqrt(65536)	256

math.fmod	mode	math.fmod(65535,2)	1
math.modf	modf	math.modf(20.12)	20 0.12
math.randomseed	randomseed	math.randomseed(os.time())	
math.random	random	math.random(5,90)	5~90
math.rad	rad	math.rad(180)	3.1415926
math.deg	deg	math.deg(math.pi)	180
math.exp	exp	math.exp(4)	54.5981
math.log	log	math.log(54.5981)	4
math.log10	log10	math.log10(1000)	3
math.frexp	frexp	math.frexp(160)	0.625 8
math.ldexp	ldexp	math.ldexp(0.625,8)	160
math.sin	sin	math.sin(math.rad(30))	0.5
math.cos	cos	math.cos(math.rad(60))	0.5
math.tan	tan	math.tan(math.rad(45))	1
math.asin	asin	math.deg(math.asin(0.5))	30
math.acos	acos	math.deg(math.acos(0.5))	60
math.atan	atan	math.deg(math.atan(1))	45

1.7 Definitions of Robotic Axis

Symbols of Commands	Explanations of Commands
AX	X axis NO. under Cartesian coordinate system
AY	Y axis NO. under Cartesian coordinate system
AZ	Z axis NO. under Cartesian coordinate system
AC	C axis NO. under Cartesian coordinate system
J1	J1 joint
J2	J2 joint
J3	J3 joint
J4	J4 joint

◆ **Notice:** AX\AY\AZ\AC\J1\J2\J3\J4 are global variables which cannot be redefined by user.

1.8 Global Robotic Variables

Symbols of Commands	Explanations of Commands
ON	Open state
OFF	Close state
p0~p2999	Name of robotic default points

Global variables are already defined in the system, which have their specific meaning. So user can not redefine these variables in then system.

Point variables (p0~p2999) are belonging to table, which are defined as follows:

```
local p={x=VALUE1,y=VALUE2,z=VALUE3,c=VALUE4,h=VALUE5}
```

In program, user can access the values of point in a way as p.x\p.y\p.z\p.c\p.h

Example 6:

```
local    a = p1                    -- a is the reference of p1
a.x= 10                            --the value of p0.x is also 10
local    a = #p1                   --copy the value of p1 to a
a.x = 10                           --p1.x keeps the original value
```

1.9 String Manipulations

Symbols of Commands	Explanations of Commands
string.find	This function is used to search a pattern in a given string, then return its location
string.match	This function is used to search a pattern in a given string, then return the matched string
string.sub	Cut off the given string(s) from ith to jth
string.gsub	If some characters are same with the pattern in a given string, replace them with the pattern
string.char,string.byte	Used to convert character to its corresponding number
sring.format	Used to format the given string, which has same function with printf() in C language

1. **string.find()**

   Function: used to search a pattern in a given string; if success to find the pattern, it will return the start and end position of the pattern in the given string; if failed to find the pattern, it will return ***nil***. For example:

```
local str = “Hello World”
local i,j = string.find(str, “Hello”)    --Return the start and end position of “hello” in “str” print(i,j)    --> 1    5
```

2. **string.match()**

   Function: used to search a pattern in a given string; if success to find this pattern, it will return this pattern; if failed to find this pattern, it will return ***nil***. For example:

```
local str = “Hello12345World”
local subStr = string.match(str, “%d+”)
print(subStr)    -->1 2 3 4 5

local i,j = string.find(str, “%d+”)
subStr = string.sub(str,i,j)
print(subStr)    -->1 2 3 4 5
```

3. **string.sub(str,i,j)**

   Function: Intercept a given string from ith character to jth character and return these characters. In Lua language, the index of first character in a string is 1. You can also use the negative index, such as **-1** means last character, **-2** means penult character. For example:

```
s = “[in brackets]”
b = string.sub(s,2,-2)
print(b)      --> in brackets
```

4. **string.gsub(s, pattern, reps)**

   It has three input parameters: s(giving string), pattern(matched patterns), reps (replaceable characters).

   Function: replace all matched patterns to replaceable string and return the new string. For example:

```
s = string.gsub("Lua is cute", "cute", "great”)
print(s)                --> Lua is great
```

5. **string.char()** &&**string.byte()**

   Function: Used to convert character to its corresponding number. For example:

```
i = 97
print(string.char(i, i+1, i+2))    --> abc
print(string.byte(“abc”))          --> 97
print(string.byte(“abc”), -2)      --> 98
```

6. **string.format()**

   Function: used to format a given string. For example:

```
str = string.format(“%0.2f”, 34.2344)
print(str)      --> 34.23
str = string.format(“0x%08X”, 348)
print(str)      --> 0x0000015C
```

1.10 Commands of Process Control

Symbols of Commands	Explanations of Commands
if then else elseif end	Conditional branch commands
while do end	Control command of **while** cycling
for do end	Control command of **for** cycling
repeat until	Control command of **repeat** cycling
goto	A jump without condition
function end	Commands defined by user functions

1.10.1 if then else elseif end

Use explanation: if conditional branch

Syntax description:

Case1	Case2	Case3

if conditions then
then-part
end
	if conditions then
	then-part
	else
	else-part
	end
		if conditions then
		then-part
		elseif conditionsthen
		elseif-part
		else
		else-part
		end

"**Conditions**"are the conditions of controlled statements, if they are true, then the conditions are satisfied. "part" is the program's part to be executed.

"**Conditions**" which can be constant, variables, expressions or function calls. It is only to determine the final outcome of the conditions whether it is the false or true, then select to execute the program.

1.10.2 while do end

Use explanation: "while" loop command

Syntax description:

```
while condition do
    statements
end
```

"Conditions" which represents controlled conditions, if it is true, execute the "statements"; if it is false, do not execute the "statement".

```
a = 0
while a<10 do      --conditional judgment, if it is true, then continue to execute a = a-1
    a = a-1
end
```

1.10.3 for do do

Use explanation: "for" loop command

Syntax description:

```
for var=exp1,exp2,exp3 do
    loop-part
end
```

"for" will use exp3 as a step (step value) from the exp1 (initial value) to exp2 (end value) to execute the loop part(a loop), where exp3 can be omitted with the default step=1.

"exp" is an expression that can be a numeric constant, variable, a return value of a function call or an expression operation.

There are a few points to be paid attention to:

1. exp1, exp2 and exp3 are only calculated once before the beginning of the cycle.

```
for i=1,f(x) do          -- Call f (x) function, and the function returned value as the end
                                                    --of the cycle
    print(i)
end
for i=10,1,-1 do
    print(i)
end
```

2. The control variable is a local variable that is automatically declared, and only valid within the loop.

```
for i=1,10 do
    print(i)
end
max = i                          --error in using "I", because "i" is a local variable
```

If you need to keep the value of the control variable, you need to save it in the loop.

```
local found = nil
for i=1,a.n do
    if a[i] == value then
        found = i
        break
    end
end
print(found)
```

3. Do not change the value of the control variable in the process of the cycle; otherwise, the results are unpredictable. If you want to exit the loop, use the **break** statement.

1.10.4 repeat until

Use explanation: "repeat-until" loop command

Syntax description:

```
repeat
    statements
until conditions
```

"repeat-until" and "while" statements are roughly the same, but the executing orders of the loop part of the statements are not the same. For "repeat-until", it firstly executes the "statements", then to judge the cycling conditions; for "while", it firstly judge the cycling conditions, then to execute the "statements".

1.10.5 goto

Use explanation: goto: unconditional jump commands

Syntax description: goto lab_name

Lab_name is the user defined jump location's name of the file's row, which type is string. It will cause an error if it is undefined.

```
local a=0
::lab::                          -- Define jump location name as Lab MovL(p0)
MovL(p1)
a = a + 1
print("jump frequency:",a)
goto lab                         --Jump to the second line to re-execute the loop
```

- Notice: "goto" command cannot jump from one function to another one, and jump name of "lab_name" cannot be repeatedly used.

1.10.6 function end

Use explanation: define a function command

Syntax description: function func_name (arguments-list)

```
function  func_name(arguments-list)
    statements-list
end
```

"**func_name**" is function's name, which is defined according to the naming rules of AR language. And function's name cannot be repeated, otherwise it will go wrong.

"**arguments-list**" is the parameter list of function, in which the parameters can be data variables of any types data and they are separated with "," when there are multiple parameters. Functions can also be no parameters, the list of parameters in the definition can be empty, but the parentheses cannot be omitted. The argument list can be empty during definition, but the parentheses cannot be omitted.

"**Statements-list**" is a function body's part, which is used to realize the specific details of the function. The end of the function body can have a returned value through the keyword "return", or have no returned value.

An additive function is defined as follows:

```
1.  function add (a,b)
        return a+b
    end
    add(1,2)              --function call; the returned value is 3
```

```
2.  function addmul(a,b)
        return a+b, a*b
    end
    addmul(2,5)                          --the returned values are 7, 10
```

1.11 Commands of Movement

Symbols of Commands	explanations of Commands
MovL	The command that moves to the absolute position of Cartesian coordinate system with line mode.
MovLR	The command that moves to the relative position of Cartesian coordinate system with line mode.
MovP	The command that moves to the absolute position of Cartesian coordinate system with PTP mode.
MovPR	The command that moves to the relative position of Cartesian coordinate system with PTP mode.
MovJ	The command that controls each joint to move to target angle
MArchP	The command that controls robot to move with arch under PTP mode.
MArc	The command that moves to absolute position from current position (Cartesian coordinate system) with arc interpolation mode.
MCircle	The command that moves to the absolute position of Cartesian coordinate system with line mode.

MovL

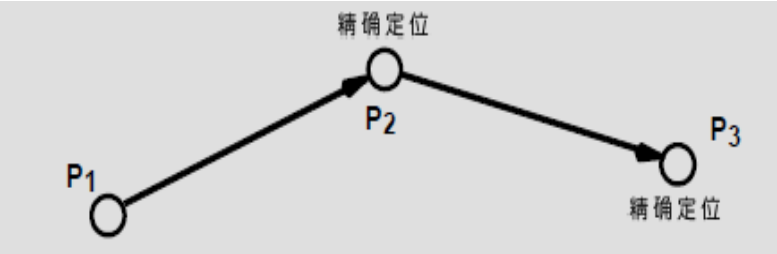
Use-explanation	Moving to the target position in the Cartesian coordinate system in a straight line.
Syntax-description	STA = MovL(A, "CP=20 Acc=20 Dec=20 Spd=100 AccC=20 SpdC=20 I=0 In=10 ON/OFF")
Parameter-description	The instruction has total of two parameters, the first parameter A is the target point, and the second parameters are optional parameters. The second parameters are default to global variables if second parameters are ignored.

Instructions for the following instructions

A	Descartes coordinates the target position. It can be the name of the point p1~p2999, also can be the point of the index 1~2999. Optional parameters, you can specify the parameters of the moving to the target location.
CP	Indicates whether the transition is smooth when moving to the target point.
Acc	Specify the acceleration, unit mm/s^2
Dec	Specify the deceleration, unit mm/s^2
Spd	Specify the speed of moving to the target position, Unit mm/s
AccC	Specify the gesture acceleration to the target position, unit is mm/s^2
SpdC	Specify the gesture speed to the target position, unit is

	mm/s
I	Current value set for third axis (Z-axis), which unit is mA. If the current in the motion exceeds the set value, the current motion command will stop and then continues other
In	Input detection signal (Optional parameter). If the input signal is detected during the movement, the current motion stops and the then continue to execute other commands
ON/OFF	ON: open OFF: close
Return values:	
STA	0 : Normal
	1: input signal is abnormal
	2: current value is abnormal

Figure



Example

1. MovL(p1) - the robot moves from the current position to the p1 target point in a straight line.
2. MovL(p10) - the robot moves from the current position to the p10 target point in a straight line.
3. MovL(p10, “Spd=1000 Acc=100”) - the robot moves from the current position to the P10 target point in a straight line, and the acceleration is 100mm/s^2, the speed is 1000mm/s
4. MovL(p20, “CP=20”) --Robot moves to the p20 position in a straight line, where the target position is 20% smooth transition

MovLR

Use-explanation	Relative movement of linear mode in Descartes coordinate system
Syntax description	STA = MovLR(A,B,“CP=20 Acc=20 Dec=20 Spd=100 AccC=20 SpdC=20 I = 0 In=10 ON/ OFF”)
Parameter declaration	This command includes three parameters, where A is Cartesian coordinate axis, B is the relative distance of movement, the third parameter are optional which are default to global values when omitted. **Instructions for the following instructions**

A	Cartesian coordinate axis, which could be one of AX,

	AY,AZ, AC
B	Relative distance of movement
CP	Indicates whether the transition is smooth when moving to the target point.
Acc	Specify the acceleration, unit mm/s^2
Dec	Specify the deceleration, unit mm/s^2
Spd	Specify the speed of moving to the target location, Unit mm/s
AccC	Specify the gesture acceleration to the target position, unit is mm/s^2
SpdC	Specify the speed of moving to the target position, Unit mm/s
I	Current value set for third axis (Z-axis), which unit is mA. If the current in the motion exceeds the set value, the current motion command will stop and then continues other
In	Specifies a trigger signal of input port to stop the movement from current position to target position
ON/OFF	ON: open OFF: close

Return values:

STA	0 : Normal
	1: input signal is abnormal
	2: current value is abnormal

Example

1. MovLR(AX,10) --x axis from the current position to the positive direction of the 10mm distance
2. MovLR(AY,10) --Y axis from the current position to the positive direction of 10mm distance
3. MovLR(AZ,-10) --Z axis from the current position to the negative direction of 10mm distance
4. MovLR(AC,10) --C axis from the current position to the positive direction of 10 degree

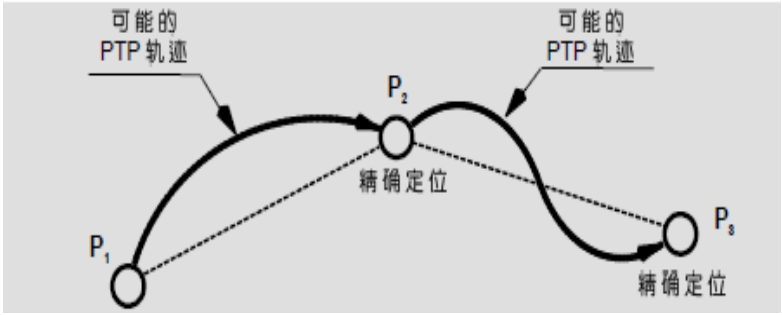
MovP

Direction	Mobile point-to-point to Cartesian coordinates of the target location
Syntax description	STA=MovP(A,"CP=20 Acc=20 Dec=20 Spd=100 I=0 In=10 ON/OFF")
Parameter declaration	The instruction is a total of two parameters, the first parameter A is the target point, and the second parameters are optional parameters, the system defaults to the global state of the system Instructions for the following instructions

A	Descartes coordinates the target position. It can be a

	point of the name p1~p2999, also can be the point of the index 1~2999
CP	Optional, indicates whether the transition is smooth when moving to the target point
Acc	Optional, specify the acceleration ratio of movement from current position to target; range: 1~100
Dec	Optional, specify the deceleration ratio of movement from current position to target; range: 1~100
Spd	Optional, specify the speed ratio of movement from current position to target; range: 1~100
I	Current value set for third axis (Z-axis), which unit is mA. If the current in the motion exceeds the set value, the current motion command will stop and then continues other
In	Specifies a trigger signal of input port to stop the movement from current position to target position
ON/OFF	ON: open OFF: close
Return value:	
STA	0 : Normal
	1: input signal is abnormal
	2: current value is abnormal

Figure



Example

1. MovP(p1) --Move to target position (p1) with in point-to-point (PTP) manner
2. MovP(10) --Move to target position (p10) in point-to-point (PTP) manner
3. MovP(10, “Acc=50 Spd=50”) --Move to target position (p10) in point-to-point (PTP) manner with 50% acceleration and speed
4. MovP(p20, “CP=20”) ----Robot moves to the p20 position in PTP mode, where the target position is 20% smooth transition

MovPR

Direction	Mobile point-to-point to Cartesian coordinates of the target location
Syntax description	STA = MovPR(A,B, “CP=20 Acc=20 Dec=20 Spd=100 I=0 In =10

	ON/OFF")	
Parameter declaration	A	Cartesian coordinate axis, which could be one of AX, AY,AZ, AC
	B	Relative distance of movement
	CP	Indicates whether the transition is smooth when moving to the target point.
	Acc	Optional, specify the acceleration ratio of movement from current position to target; range: 1~100
	Dec	Optional, specify the deceleration ratio of movement from current position to target; range: 1~100
	Spd	Optional, specify the speed ratio of movement from current position to target; range: 1~100
	I	Current value set for third axis (Z-axis), which unit is mA. If the current in the motion exceeds the set value, the current motion command will stop and then continues other
	In	Specifies a trigger signal of input port to stop the movement from current position to target position
	ON/OFF	ON: open OFF: close
	Return value:	
	STA	0 : Normal
		1: input signal is abnormal
		2: current value is abnormal
Example	1. MovPR(AX,10) --X axis from the current position to the positive direction of the 10mm distance in PTP manner 2. MovPR(AY,10) --Y axis from the current position to the positive direction of 10mm distance in PTP manner 3. MovPR(AZ,-10) --Z axis from the current position to the negative direction of 10mm distance in PTP manner 4. MovPR(AC,10) --C axis from the current position to the positive direction of 10 degree in PTP manner	

MovJ

Direction	Point-to-point mode joint mobile robot each to the specified position
Syntax description	1. MovJ (A,B, "Acc=20 Spd=100") 2. MovJ(A, "Acc=20 Spd=100")

Parameter declaration	**Usage 1：Instructions of each parameter**	
	A	Joint axes, which can be one of J1,J2,J3,J4
	B	Moving target angle for each axis
	Acc	Optional, specify the acceleration ratio of movement from current position to target; range: 1~100
	Dec	Optional, specify the deceleration ratio of movement from current position to target; range: 1~100
	Spd	Optional, specify the speed ratio of movement from current position to target; range: 1~100
	Usage 2：Instructions of each parameter	
	A	Absolute target position (p1~p2999) under Cartesian coordinate system
	Acc	Specify the acceleration ratio of movement from current position to target; range: 1~100
	Dec	Specify the deceleration ratio of movement from current position to target; range: 1~100
	Spd	Specify speed ratio of movement from current position to target; range: 1~100
Example	1. MovJ(J1,10) --First joint(J1) of the robot moves to 10 degree 2. MovJ(J3,-10) --Third joint(J3) of the robot moves to -10 degree 3. MovJ(p1,“Acc=20 Dec=20 Spd=20”) --Move to target position(p1) in joint manner	

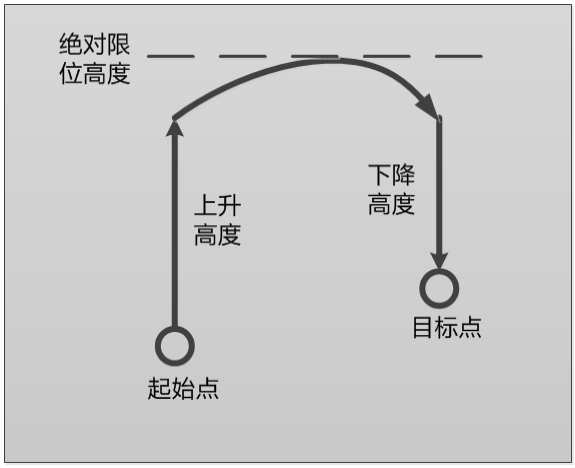
◆ **Note:** for joint movement, J3 is with millimeter (mm) unit, and the other joints are degree units.

MArchP

Direction	The robot in a point-to-point way arch movement	
Syntax description	STA = MArchP(A,B,C,D, “Acc=20 Dec=20 Spd=100 I=0 In=10 ON/OFF”)	
Parameter declaration	**Instructions for the following instructions**	
	A	Point name (p1~p2999 or 1~2999)
	B	Highest limit (absolute position) of Z axis; unit is millimeter (mm)
	C	Rising height of Z axis, unit is millimeter (mm)
	D	Falling height of Z axis, unit is millimeter (mm)
	Acc	Optional, specify acceleration ratio of the arch, range 1~100
	Dec	Optional, specify deceleration ratio of the arch, range: 1~100
	Spd	Optional, specify speed of the arch, range: 1~100

	I	Current value set for third axis (Z-axis), which unit is mA. If the current in the motion exceeds the set value, the current motion command will stop and then continues other
	In	Specifies a trigger signal of input port to stop the movement from current position to target position
	ON/OFF	ON: open OFF: close
	Return value	
	STA	0 : Normal
		1: input signal is abnormal
		2: current value is abnormal

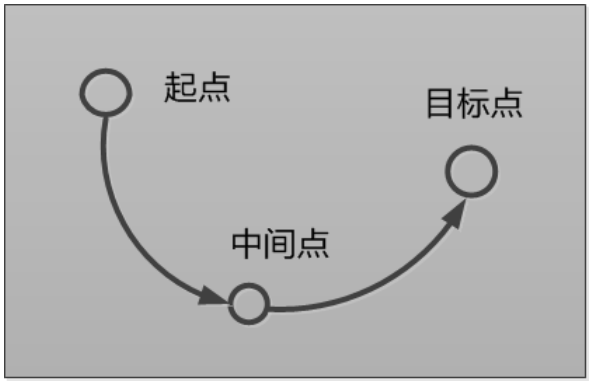
Figure



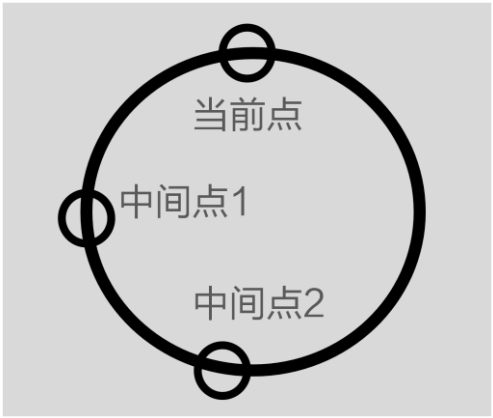
Example

1. MArchP(p1,-10,5,5)
   --p1: target position;
   --(-10): limit of Z axis, which is absolute position;
   --5: rising height of Z axis during movement, then arrive a height which cannot be over the limit;
   --5: falling height
2. MArchP(p2,10,1,1,“Acc=20 Dec=30 Spd=40”)
   --p2: target position;
   --10: limit of Z axis, which is absolute position;
   --1: rising height of Z axis during movement, then arrive a height which cannot be over the limit(10);
   --1: falling height
   --Acc=20: specify acceleration ratio (20%) during the MArchP movement
   --Dec=30: specify deceleration ratio (30%) during the MArchP movement
   --Spd= 40: specify speed ratio (40%) during the MArchP movement

MArc

Direction	Arc motion in Descartes coordinate system	
Syntax description	MArc(A,B, "CP=20 Acc=20 Dec=20 Spd=30 Angle = 120")	
Parameter declaration	Instructions for the following parameters	
	A	An intermediate point of the arc movement under Cartesian coordinate system. Name of the point can be p1~p2999 or1~2999
	B	Target position (last point) of the arc movement under Cartesian coordinate system. Name of the point can be p1~p2999 or 1~2999
	CP	Optional, specify a smooth ratio (1~100) of moving to target point.
	Acc	Optional, specify the acceleration of moving to the target position; unit is mm/s^2
	Dec	Optional, specify the deceleration of moving to the target position; unit is mm/s^2
	Spd	Optional, specify the speed of moving to the target position; unit is mm/s
	Angle	Optional, specify the arc angle, range:$1 \sim 360^0 C$
Figure		
Example	1. MArc(p1,p2) --A arc movement with p1 being intermediate point and p2 being target point 2. MArc(1,2) --A arc movement with p1 being intermediate point and p2 being target point 3. MArc(p3,p4,"CP=20 Acc=100 Dec=100 Spd=1000 Angle=120") --A arc movement with p3 being intermediate point and p4 being target point, in which smooth ratio is 20%, acceleration is 100mm/s^2, deceleration is 100mm/s^2, speed is the speed is 1000mm/s, angle is 120 degree	

MCircle

Direction for use	Circular motion of robot in Cartesian coordinate system	
Syntax description	MCircle(A,B, "CP=20 Acc=20 Dec=20 Spd=20 Angle=360")	
Parameter declaration	Instructions for the following instructions	
	A	An intermediate point of the circle movement under Cartesian coordinate system. Name of the point can be p1~p2999 or 1~2999
	B	Target position (last point) of the circle movement under Cartesian coordinate system. Name of the point can be p1~p2999 or 1~2999
	CP	Optional, specify a smooth ratio (1~100) of moving to target point
	Acc	Optional, specify the acceleration of moving to the target position; unit is mm/s^2
	Dec	Optional, specify the deceleration of moving to the target position; unit is mm/s^2
	Spd	Optional, specify the speed of moving to the target position; unit is mm/s
	Angle	Optional, specify the circle angle, range:$1 \sim 360^0 C$
Figure		

Example

1. MCircle (p1, p2) --A circle movement with p1 being intermediate point and p2 being target point
2. MCircle(1,2) --A circle movement with p1 being intermediate point and p2 being target point
3. MCircle(1,2,"CP= 20 Acc=100 Dec=150 Spd=1000 Angle=360") --A circle movement with p1 being intermediate point and p2 being target point; during the movement, smooth rate is 20%, acceleration is 100mm/s^2, deceleration is 150mm/s^2 , speed is 1000mm/s and angle is 360
4. MCircle(1,2,"CP=20") --A circle movement with p1 being intermediate point and p2 being target point; during the

movement, smooth rate is 20%

1.12 Commands of Movement Parameters

Symbols of Commands	Explanations of Commands
AccJ	Set a proportion of acceleration to affect MovJ /MovP/MovPR / MArchP commands' accelerating time.
DecJ	Set a proportion of deceleration to affect MovJ/ MovP/MovPR/ MArchP commands' decelerating time.
SpdJ	Set a proportion of speed to affect MovJ/MovP/MovPR/MArchP commands' running speed
AccL	Set the acceleration of line movement to affect MovL/MovLR/ MArchL/MArc/MCircle commands' accelerating time (unit is mm/s2)
DecL	Set the deceleration of line movement to affect MovL/MovLR/ MArchL/MArc/MCircle commands' decelerating time (unit is mm/s2)
SpdL	Set the speed of line movement to affect MovL/MovLR/MArchL/ MArc/MCircle commands' running speed (unit is mm/s)

AccJ

Use-explanation	Set the acceleration proportion of PTP movement mode
Syntax-description	AccJ(A)
Parameters-description	A is the percentage, which range is 1~100
Example	1. AccJ(50) --Set 50% of acceleration MovP(p2) --Move to p2 with 50% of acceleration

DecJ

Use-explanation	Set the deceleration proportion of PTP movement mode
Syntax-description	DecJ(A)
Parameters-description	A is the percentage, which range is 1~100
Example	1. DecJ(50) --Set 50% of deceleration MovP(p2) --Move to p2 with 50% of deceleration

SpdJ

Use-explanation	Set the speed proportion of PTP movement mode
Syntax-description	SpdJ(A)
Parameters-description	A is the percentage, which range is 1~100
Example	1. SpdJ(50) --Set 50% of speed

	MovP(p2) --Move to p2 with 50% of speed

◆ Notice: the set acceleration and speed are global variables until next update.

AccL

Use-explanation	Set the acceleration of line movement mode
Syntax-description	AccL(A)
Parameters-description	A is actual acceleration, which range is 1mm/ s^2 ~1000mm/s^2
Example	1. AccL(500) --Set the acceleration as 500 mm/ s^2 MovL(p2) --Move to p2 with 500 mm/ s^2 acceleration

DecL

Use-explanation	Set the deceleration of line movement mode
Syntax-description	DecL(A)
Parameters-description	A is actual deceleration, which range is 1 mm/ s^2~1000 mm/ s^2
Example	1. DecL(500) --Set the deceleration as 500 mm/s^2 MovL(P2) --Move to p2 with 500 mm/ s^2 deceleration

SpdL

Use-explanation	Set the speed of line movement mode
Syntax-description	SpdL(A)
Parameters-description	A is actual speed, which range is 1 mm/s ~1000mm/s
Example	1. SpdL(500) --Set the speed as 500mm/s MovL(p2) --Move to p2 with 500mm/s speed

1.13 Commands of Program Management

Symbols of Commands	Explanations of Commands
Delay	Delay command, which unit is milliseconds(ms)
Exit	Exit the running program
Pause	Pause the running program

Delay

Use-explanation	Delay command
Syntax-description	Delay(A)
Parameters-description	A is the delay time
Example	1. Delay(1000) --A delay with 1000ms 2. Delay(1) --A delay with 1ms 3. local time = 1000 Delay(time) --the parameter “time” is a local variable

Exit	
Use-explanation	Exit the running program
Syntax-description	Exit()
Parameters-description	There is no parameter in this command
Example	1. Exit() --Exit the running program MovL(1) --This command is not executed

Pause	
Use-explanation	Pause the running program
Syntax-description	Pause()
Parameters-description	There is no parameter in this command
Example	1. Pause() --pause the running program 2. MovL(1) --continue to reexecute program by pressing “Start” key

1.14 General Commands

Symbols of Commands	Explanations of Commands
X	A command that builds the absolute points of specified X axis under Cartesian coordinate system
Y	A command that builds the absolute points of specified Y axis under Cartesian coordinate system
Z	A command that builds the absolute points of specified Z axis under Cartesian coordinate system
C	A command that builds the absolute points of specified C axis under Cartesian coordinate system
XYZC	A command that builds the absolute points of specified XYZC axis under Cartesian coordinate system

X/ Y/ Z/ C	
Use-explanation	Used to build a point(under Cartesian coordinate system) of a specified axis
Syntax-description	X(A) Y(A) Z(A) C(A)
Parameters-description	A is Cartesian coordinate position of each axis, where the unit for C-axis is degree and for other three axis are millimeter.
Example	1. MovL(p10 + X(20)) --Robot moves a position which has a offset 20mm at X-axis positive direction corresponding to p10. 2. MovLR(X(20)) --Robot moves to X-axis positive direction with 20mm, which is corresponding to the current position. 3. MovLR(Z(-20)) --Robot moves to Z-axis negative direction

with 20mm, which is corresponding to the current position.

4. MovLR(C(20)) --Robot moves to C-axis positive direction with 20^o, which is corresponding to the current position.

◆ Notice: this command is used to generate a point data with no motion. Generally, it acted as a parameters assigned to motion commands.

XYZC		
Use-explanation	Used to build the point(under Cartesian coordinate system) of each specified axis	
Syntax-description	XYZC(A,B,C,D)	
Parameters-description	A	Cartesian coordinate position of X-axis; unit is millimeter
	B	Cartesian coordinate position of Y-axis; unit is millimeter
	C	Cartesian coordinate position of Z-axis; unit is millimeter
	D	Cartesian coordinate position of C-axis; unit is degree
Example	MovL(p10 + XYZC(10,20,-5,30)) --the robot moves to X-axis positive direction with 10mm, Y-axis positive direction with 20mm, Z-axis positive direction with 5mm and C-axis positive direction with 30^o, all of which are corresponding to p10.	

1.15 Commands of Input/Output

Symbols of Commands	Explanations of Commands
DI	Read the state of input port
DO	Read the state of output port
WDI	Read the state of one input port. AR program will be continued to run until this signal is effective
WDO	Read the state of one output port. AR program will be continued to run until this signal is effective

DI	
Use-explanation	Read the state of input port
Syntax-description	**Case 1:** DI(-1) or DI(-2) **Case 2:** DI(A)
Parameters-description	**Case1:** **Return value:** The return value of DI(-1) is a 32-bit binary number representing the state value of input port (0 ~ 31) from low to high. The return value of DI(-2) is also a 32-bit binary number, with the lower two representing the state value of the input port (32 ~ 33).Where 0 is closed and 1 is open.

Case 2:

Return value: ON (OFF) or Decimal value

A	Number of input port, which range is 0~33

Example

1. local input = DI(-1) --Obtain status of input ports(0~31)
   if ((input>>0)&0x0001)==1 then --Judge whether input port 0 is open, if true, then port 0 is open
   MovP(p1)
   elseif ((input>>9)&0x0001)==1 then --Judge whether input port 9 is open, if true, then port 9 is open
   MovP(p2)
   end
2. if DI(10) == ON then --If input port 10 is ON, then move to p1
   MovP(p1)
   end
3. value = DI({1,2,3}) --Obtain states of input ports(1~3); if return value is 7 (111), it means that input ports 1~3 are open; if return value is 6(110), it means that input ports 1~2 are open input port 3 is close; and so on…

- DI is only to read the state of input ports, and it will not always wait no matter the state is effective or not.
- If read several input ports at the same time, the input ports must be sorted from small to big or from big to small.

DO

Use-explanation: Read the state of output port

Syntax-description:

Case 1: DO(A)
Case 2: DO(A,B, " **Time** = C")
Case 3: DO(A,B, "**F**")
Case 4: DO(A,B, "**Time** = C **H**")
Case 5: DO(A,B, "**Time** = C **F**")
Case 6: DO(A,B, "**POS** = D")

Parameters-description: Return value ON or OFF

A	Read the number of output port, which range is 0~26
B	Write the state of output port (ON or OFF)
C	Optional parameter, which is the time to hold this state (unit is millisecond)
D	Position percentage

Example

1. DO(10,ON) --Open output port 9
2. DO(10,ON,"Time=1000") –Open output port 10 and keep 1s, and then close this port
3. DO({1,2,3,4},{ON,ON,ON,ON}) –Open outputs 1\2\3\4 to ON
   DO({5,6,7,8},{ON,OFF,ON,OFF}) –Open outputs 5 and 7 to ON,

and close outputs 6 and 8 to OFF

4. DO({9,10},{ON,ON},"Time=2000") –Open outputs 9 and 10 to ON and keep 2s, and then close them to OFF
5. DO({9,10},{0,0},"Time=2000") -- Open outputs 9 and 10 to close and keep 2s, and then open them to ON
6. DO({1,2,3},7) --Open output 1,2 and 3 to ON (because 7 is corresponding to 111 (binary))
7. DO({1,2,3},5) --Open output 1, 3 to ON and output 2 to OFF (because 5 is corresponding to 101 (binary))
8. MovP(p9)
   DO(1,ON, "F") --Move from the current position to p9 and open output port 1 after arriving at p9, then move to p10 continuously
   MovP(p10)
9. MovP(p1)
   MovP(p2)
   DO(1,ON, "POS=50") --Open the output port 1 at 50% of distance from point p1 to p2
10. MovP(p1)
    MovP(p2)
    DO(1,ON, "Time=100 H") --Open the output port 1 in 100 milliseconds before arriving at p2
11. MovP(p1)
    MovP(p2)
    DO(1,ON, "Time=50 F") --open the output port 1 at 50ms after arriving at p2

- Output ports must be set in ascending or descending order when open or close a few outputs at same time;
- Case3 ~ Case6 are belonging to cache IO function;
- Cache IO function does not apply to the combination of IO;
- Cache IO function is currently available for point-to-point movement (MovP);
- In **DO** command, **Time**、**POS**、**F**、**H** are keywords which cannot be modified;
- Current motion will not be terminated and motion is continuous once adopted cache IO function.

WDI

Use-explanation	Read the state of one input port. AR program will be continued to run until this signal is effective.	
Syntax-description	WDI(A,B) WDI(A,B, "Time=1000")	
Parameters-description	A	Input port
	B	State of input port, ON or OFF
	Time	Optional parameter, which is the waiting time(unit is millisecond)

Example	1. **WDI(10,ON)** --Wait until input port 10 is ON, then run the --following command **MovP(p1)** 2. **WDI(10,ON, "Time=2000")** --Waiting for input port 10 to ON within 2 seconds. If the input port 10 status is still OFF after 2 seconds, it will continue to perform MovP(p1) **MovP(p1)**

WDO

Use-explanation	Obtain the state of one output port. AR program will continue to run if this signal if effective	
Use-explanation	WDO(A,B) WDO(A,B, "Time=1000")	
Parameters-description	A	Output port
	B	State of output port, ON or OFF
	Time	Optional parameter, which is the waiting time(unit is millisecond)
Example	1. **WDO(10,ON)** --MovP(p2) will be executed until state of output port 10 is ON **MovP(p2)** 2. **WDO(10,ON,"Time=2000")** --MovP(p2) will be executed if state of port 10 is ON within 2 seconds; MovP(p2) will also be executed if state of port 10 is still OFF over 2 seconds **MovP(p2)**	

1.16 Commands of Coordinate System

Symbols of Commands	Explanations of Commands
SetU	Set the current user coordinate system of robot
WrU	Modify the data of user coordinate system
SetT	Set the current tool coordinate system of robot
WrT	Modify the data of tool coordinate system
CacU	Build a new user coordinate system
U2U	Transform **Cartesian coordinates** within user0 ~ user9
V2Tool	Calculate a new tool
getcart	Obtain current Cartesian coordinate of robot's end
CacT	Build a new tool coordinate system with two point
encoderget	Obtain pulse value of corresponding encoder

SetU

Use-explanation	Set current user coordinate system

Syntax-description	SetU(A)
Parameters-description	A is the number of user coordinate system, whose range is 0~6
Example	1. SetU(1) --Select user coordinate system 1 MovL(p1) --Move to p1 under user coordinate system 1

◆ Note: Once the user coordinate system is set, it is effective immediately until a new one is set.

SetT

Use-explanation	Set current tool coordinate system
Syntax-description	SetT(A)
Parameters-description	A is the number of tool coordinate system, whose range is 0~6
Example	1. SetT(1) -- select tool coordinate system 1 MovL(p1) -- move to p0 under tool coordinate system 1

WrU

Use-explanation	Modify the offset value of user coordinate system	
Syntax-description	WrU(A,B,C)	
Parameters-description	A	To be modified number of user coordinate system, which range is 1~6
	B	To be modified number of axis, which can be one of AX, AY, AZ and AC
	C	To be modified value
Example	1. WrU(1,AX,200) --Modify the X-axis offset value as 200mm under the user coordinate system 1 2. WrU(1,AC,100) --Modify the C-axis offset value as 100 degree under the user coordinate system 1	

WrT

Use-explanation	Modify the offset value of tool coordinate system	
Syntax-description	WrT (A,B,C)	
Parameters-description	A	To be modified number of tool coordinate system, which range is 1~6
	B	To be modified number of axis, which can be one of AX, AY, AZ and AC
	C	To be modified value
Example	1. WrU(1,AX,200) --Modify the X-axis offset value as 200mm under the tool coordinate system1 2. WrU(1,AC,100) --Modify the C-axis offset value as 100 degree under the user coordinate system 1	

CacU

Use-explanation	Build a new user coordinate system	
Syntax-description	CacU (pos1,pos2)	
Parameters-description	pos1	The origin of the new user coordinate system
	pos2	One point on X-axis of the new user coordinate system
Example	local pos1 = {300,100,0,0} --Define the origin of user 1 local pos2 = {300,120,0,0} --Define one point on X-axis of user 1 WrU(1,CacU(pos1,pos2)) --Name the new user coordinate as user 1	

U2U

Use-explanation	Transform **Cartesian coordinates** within user0 ~ user9	
Syntax-description	Pos = U2U (A,B,C)	
Parameters-description	A	The converted user No. (0~9)
	B	The target user No. (0~9)
	C	Cartesian coordinates under converted user No.
	Pos	Cartesian coordinates under target user No.
Example	1. local pos = U2U(0,2,p1) --Transform p1 coordinates from user 0 to user 2; return value is **pos** SetU(2) --Set user 2 as current user, and then move to **pos** point MovP(pos) 2. local pos1 = {x = 100,y = 40,z = -10,c = 30} local pos = U2U(1,3,pos1) –Transform pos1 coordinates from user 1 to user 1; return value is **pos** SetU(3) --Set user 3 as current user, and then move to **pos** point MovP(pos)	

V2Tool

Use-explanation	Calculate a new tool	
Syntax-description	V2Tool(A,B)	
Parameters-description	A	Visual Cartesian coordinate
	B	Tool No.
Example	local vis = {x=300,y=10,z=0,c=0} --Visual coordinate system V2Tool(vis,3) --Written the calculated tool to tool 3 SetT(3) --Set tool 3 as current tool	

getcart

Use-explanation	Obtain current cartesian coordinate of robot’s end	
Syntax-description	pos1,pos2 = getcart()	
Parameters-description	pos1	Current cartesian coordinate of robot’s end

	pos2	Current joint coordinate of robot's end
Example	1. local pos1,pos2= getcart () --Obtain cartresian and joint coordinate --pos1.x,pos1.y,pos1.z,pos1.c,pos1.h are respectively values of --x/y/z/c/hand --pos2.x,pos2.y,pos2.z,pos2.c,pos2.h are respectively values of --J1/J2/J3/ J4/hand 2. local pos = getcart() --Only obtain cartesian coordinate --pos.x,pos.y,pos.z,pos.c,pos.h are respectively values of x/y/z/c/hand	

CacT

Use-explanation	Build a new tool coordinate system with two point	
Syntax-description	CacT (pos1,pos2)	
Parameters-description	pos1	Cartesian position of end effector of robot's screw
	pos2	Cartesian position of end effector of gripper
Use-explanation	1. pos1= getcart( ) pos2 = {x=300,y=100,z=0,c=30} WrT(1,CacT(pos1,pos2))	

Note1: pos1 and pos2 must be belonging to same coordinate;

Note2: **CacT** generally is combined with **WrT** to use;

encoderget

Use-explanation	Obtain pulse value of corresponding encoder	
Syntax-description	Pulse = encoderget(A)	
Parameters-description	A	Encoder No. (range 1~6)
Return Value	Pulse	pulse value of corresponding encoder
Example	1. Pulse1=encoderget(1) –Obtain pulse value of M1 encoder(J1 axis) 2. Pulse2=encoderget(2) --Obtain pulse value of M2 encoder(J2 axis) 3. Pulse3=encoderget(3) --Obtain pulse value of M3 encoder(J3 axis) 4. Pulse4=encoderget(4) --Obtain pulse value of M4 encoder(J4 axis) 5. Pulse5=encoderget(5) --Obtain pulse value of M5 encoder 6. Pulse6=encoderget(6) --Obtain pulse value of M6 encoder	

1.17 Commands of Pallet

1.17.1 Programming Pallet with Three Points

Symbols of Commands	Explanations of Commands
SetPlt	A command that sets the palletizing numbers
GetPlt	A command that gets the data point of palletizing

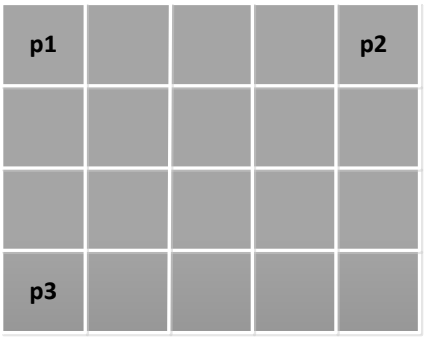
SetPlt

Use-explanation	Set parameters of pallet	
Syntax-description	Pallet on XY axis	SetPlt(A,B,C,D,E,F)
	Pallet on XYZ axis	SetPlt(A,B,C,D,E,F,G,H)
Parameters-description	**Condition 1: pallet on XY axis** SetPlt(A,B,C,D,E,F)	
	A	Pallet NO. whose Range: 1~6
	B	The palletizing origin
	C	X-axis palletizing vertex
	D	Y-axis palletizing vertex
	E	X-axis palletizing interval numbers
	F	Y-axis palletizing interval numbers
	Condition 2: pallet on XYZ axis XYZ axis' pallet SetPlt(A,B,C,D,E,F,G,H)	
	A	Pallet NO. whose Range: 1~ 6
	B	The palletizing origin
	C	X-axis palletizing vertex
	D	Y-axis palletizing vertex
	E	Z-axis palletizing vertex
	F	X-axis palletizing interval numbers
	G	Y-axis palletizing interval numbers
	H	Z-axis palletizing interval numbers
Example	1. SetPlt(1,p1,p2,p3,3,4)	--Set XY axis to pallet
	2. SetPlt(1,p1,p2,p3,p4,3,4,2)	--Set XYZ axis to pallet

GetPlt

Use-explanation	Obtain the coordinate of each point on the pallet	
Syntax-description	Pallet on XY-axis GetPlt(A,B,C)	
	Pallet on XYZ-axis GetPlt(A,B,C,D)	
Parameters-description	**Condition1:** Pallet on XY-axis GetPlt(A,B,C)	
	A	Pallet NO. whose range: 1~6
	B	X-axis palletizing start position, which is start from 1
	C	Y-axis palletizing start position, which is start from 1
	Condition2: pallet on XYZ-axis GetPlt(A,B,C,D)	
	A	Pallet NO. which Range: 1~6
	B	X-axis palletizing start position, which is start from 1
	C	Y-axis palletizing start position, which is start from 1
	D	Z-axis palletizing start position, which is start from 1

Example:




```
SetPlt(1,p1,p2,p3,5,4)    --Set XY-axis palletizing
i=1
j=1
while i <= 5 do
      j = 1
      while j <= 4 do
          pos = GetPlt(1,i, j)      --Read palletizing point data in the loop
          print(pos.x,pos.y,pos.z,pos.c) --Print the output of coordinate
          j = j + 1
          MovL(pos)                 --Moves to palletizing position with line
                                    --movement
      end
      i = i + 1
end
```

1.17.2 Programming Pallet with Four Points

Symbols of Commands	Explanations of Commands
SET_PLT	sets palletizing parameters
GET_PLT	Obtain position information of each point on the plate

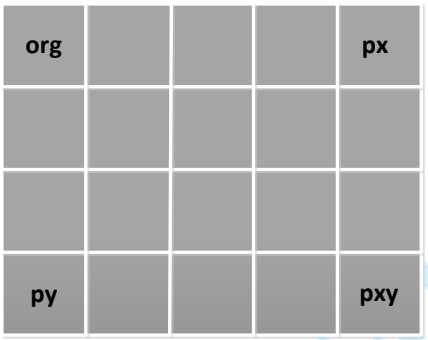
SET_PLT		
Use-explanation	sets palletizing parameters	
Syntax-description	SET_PLT(No,org,px,py,pxy,pxyz,nx,ny,nz)	
Parameters-description	No	Name of pallet, which type is number
	org	Origin position of pallet
	px	Last position of row direction
	py	Last position of column direction
	pxy	Diagonal position of first layer of pallet
	pxyz	Diagonal position of last layer of pallet
	nx	Row number of pallet
	ny	Column number of pallet
	nz	Layer number of pallet. When nz=1 is XY pallet; nz>1 is XYZ pallet

Examples	1. SET_PLT(1,p1,p2,p3,p4,p5,5,4,1)	--XY pallet
	2. SET_PLT(1,p1,p2,p3,p4,p5,5,4,3)	--XYZ pallet

GET_PLT

Use-explanation	Obtain position information of each point on the plate	
Syntax-description	pos =GET_PLT(No,num)	
Parameters-description	No	Name of pallet, which type is number
	num	Current palletizing position number

Example




```
local org = p1
local px = p2
local py = p3
local pxy = p4
local pxyz = p4
local nx,ny,nz = 5,4,1
local pos = {}
SET_PLT(1,org, px,py,pz,pxy,pxyz,nx,ny,nz)      --Set XY pallet
while true do
    for i =1, nx*ny*nz do
        pos = GET_PLT(1,i)
        print(pos.x,pos.y,pos.z,pos.c)
        MArchP(pos,0,10,10)
    end
end
```

1.17.3 Configuration Pallet

Symbols of Commands	Explanations of Commands
GetPLTPos	Obtain information: whether pallet is full、current palletizing number and current palletizing position
ResetPLT	Reset the palletizing number

GetPLTPos

Use-explanation	Obtain information: whether pallet is full、current palletizing number

	and current palletizing position
Syntax-description	flag,num,pos=ToPutPLT(“PltName”)
Parameters-description	No return

Variables

PltName	Pallet name (PLT0~PLT4)	
flag	Flag to determine whether pallet is full	
	0	Not full
	1	Full
	2	Full when last point is non-palletizing position
num	Current palletizing number	
pos	Current palletizing position	

ResetPLT

Use-explanation	Reset palletizing number
Syntax-description	ResetPLT(“PltName”,count)
Parameters-description	**Return Values**

No	

Variables

PltName	Pallet name（PLT0~PLT4）
count	Current start to palletizing number, which can be arbitrarily set (generally starting from 1)

Example:

```
local posReady = p1                --waiting position
local quliao = p2                  --Pick up position
local flag
local num
local pos ={}
MArchP(posReady,5,1,1)
while true do
    MovP(quliao)
    DO(1,ON)
    Delay(150)
    flag,num,pos =GetPLTPos(“PLT0”)
    if flag ==0 then          --Not finished
        MovP(pos)
        DO(1,OFF)
        Delay(150)
    elseif flag ==1 with      --full
        MovP(pos)             --Move to palletizing position
        DO(1,OFF)
        Delay(150)
        ResetPLT(“PLT0”,1)    --Reset palletizing number
    elseif flag==2 then           --Return value when last point
```

```
                                                    --non-palletizing postion
                    ResetPLT("PLT0",1)    -- Reset palletizing number
                end
            end
```

1.17.4 Arc Pallet

Symbols of Commands	Explanations of Commands
SetArcPlt	Set Arc palletizing parameters
GetArcPlt	Obtain position information of each point on the Arc plate

SetArcPlt

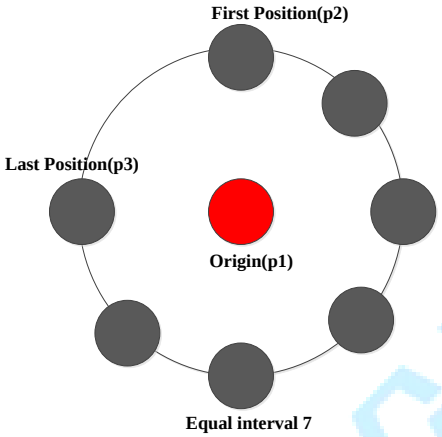
Use-explanation	Set Arc palletizing parameters	
Syntax-description	XY Arc Pallet SetArcPlt(A,B,C,D,E) XYZ Arc Pallet SetArcPlt(A,B,C,D,E,F,G)	
Parameters-description	A	Name of Arc pallet, which type is number
	B	Palletizing origin which is also the central position of Arc (circular) palletizing
	C	Start position of Arc pallet
	D	End position of Arc pallet
	E	The number of equal points for palletizing in the normal direction (the number of intervals along the circumference between the start and end of the first layer)
	XYZ Arc Pallet SetArcPlt(A,B,C,D,E,F,G)	
	A	Name of Arc pallet, which type is number
	B	Palletizing origin which is also the central position of Arc (circular) palletizing
	C	Start position of Arc pallet
	D	End position of Arc pallet
	E	Start position of Arc pallet on last layer
	F	The number of equal points for palletizing in the normal direction (the number of intervals along the circumference between the start and end of the first layer)
	G	Palletizing layer numbers
Examples	1. SetArcPlt(1,p1,p2,p3,7) --Set XY Arc pallet 2. SetArcPlt(1,p1,p2, p3, p4,7,3) --Set XYZ Arc pallet	

GetArcPlt

Use-explanation	Obtain position information of each point on the Arc plate

Syntax-description	GetArcPlt(A,B,C,D)	
Parameters-description	A	Name of Arc pallet, which type is number
	B	Which position from the first position
	C	Palletizing in which layer
	D	Pallet order: clockwise or anti-clockwise D = 1: clockwise ; D = 0: anti-clockwise

Examples




```
SetU(1)          --Arc user 1
local posready = p10          --Standby position（Teach under user 1）
local place = p11             --Placing position（Teach under user 1）
local Arc_org = {x=0,y=0,z=0,c=0} –Arc origin position（User 1 origin）
local Arc_start = p2    --Start position of first layer（Teach under user 1）
local Arc_end = p3      --End position of first layer（Teach under user 1）
local Arc_layer = p4    --Start position of last layer（Teach under user 1）
local num = 7           --interval number 7 between p2 and p3
local layer = 3          --Layer number
SetArcPlt(1,Arc_org,Arc_start,Arc_end,Arc_layer,num,layer)
MArchP(posready,0,10,10)
while true do
    i = 1
    j = 1
    for j =1,layer do
        for i =1,num do
            pos = GetArcPlt(1,i,j,1)
            MArchP(pos,0,10,10)
            Delay(5)
            MArchP(place,0,10,10)
        end
    end
end
```

- **Note1:** GetArcPlt and SetArcPlt commands are combined and used in Arc (circular) pallet.

◆ **Note2:** Must add Pallet library(pallet.lib) when use GetArcPlt and SetArcPlt commands;

Note3: Must establish a circular user coordinate system (six-point method).

1.18 Commands of Servo Management

Symbols of Commands	Explanations of Commands
MotOn	Open servo enables of all the axis
MotOff	Close servo enables of all the axis
DragMode	Set robot to drag mode

MotOn

Use-explanation	Open servo enables of all the axis
Syntax-description	MotOn( )
Parameters-description	No parameters
Example	MotOn( ) --All the axis' enables are open

◆ Notice: it will cause alarms if the servo is not enabled when the robot is on-line.

MotOff

Use-explanation	Close servo enables of all the axis
Syntax-description	MotOff( )
Parameters-description	No parameters
Example	MotOff( ) -- All the axis' enables are close

DragMode

Use-explanation	Set robot to drag mode
Syntax-description	DragMode( )
Parameters-description	No
Example	MovP(p1) MotOff( ) --Disable DragMode( ) --Drag mode

Note: Robot must be stay in disable state if want to realize drag mode in AR program.

1.19 Commands of Communication

Symbols of Commands	Explanations of Commands
RecCom	Receive data from RS232 serial port
SendCom	Send data to RS232 serial port
ClrCom	Clear the receiving buffer of RS232 serial port

sysnetclr	Clear the receiving buffer of network
sysnetget	Read network data with unblock mode
sysnetsend	Send network data
sysnetcatch	Read network data with block mode
CloseNet	Close connection of TCP network
OpenNet	Build a TCP network
ConnectNet	Connect to TCP network
RecvNet	Receive data with TCP network
WriteNet	Sent data with TCP network
publicread	Read the data from Global Data list
publicwrite	Write data to Global Data list

RecCom

Use-explanation	Receive data from RS232 serial port		
Syntax-description	Err, RecBuf=RecCom(A, "**Time** = B")		
Parameters-description	A	**1** (RS232 serial port COM1)	
	B	Optional parameter, which is timeout for receiving and whose unit is milliseconds(ms)	
	Return values is in following table:		
	RecBuf	Buffer of receiving data	
	Err	Receive error number	
		0	Receive data successfully
		Non-0	Failed to receive data
Example	1. local RecBuf --Define a receive buffer local Err --Define a receive error NO. Err,RecBuf = RecCom(1,"Time=5000") --Serial port 1 receives data with timeout 5s if Err == 0 then --Receive successfully print(RecBuf.buff) --"RecBuf" buffer receive data sent --from PC end		

Note: Data received through **RecCom** command is stored in RecBuf.buff .

SendCom

Use-explanation	Send data to RS232 serial port	
Syntax-description	SendCom(A, "B")	
Parameters-description	No value return	
	A	**1** (RS232 serial port COM1)
	B	Buffer data to be sent, which can be ASCII data and hexadecimal data. The system can automatically judge parameters' type to send.

Example	1. SendBuf={0x01,0x05,0x00,0x1A 0xff, local, 0x00} - definition of the sending and receiving buffer SendCom (1, SendBuf) - serial port 0 to send data in sixteen CRC 0x01 0xfd 0x05 0x00 0x1A 0xff 0x00 0xad 2. SendBuf local = "ROBOT" SendCom (1, SendBuf) 3. SendCom (1, “ROBOT”)

ClrCom

Use-explanation	Clear the receiving buffer of RS232 serial port	
Syntax-description	ClrCom(A)	
Parameters-description	A	**1** (RS232 serial port COM1)
Example	1. ClrCom(1) RecBuf,Err = RecCom(1, “Time=5000”) --clear the receiving buffer of RS232 port 1, then continue to receive again.	

Note : Serial port can only be 1 for RS232 serial communication.

sysnetclr

Use-explanation	Clear the receiving buffer of network	
Syntax-description	Sysnetclr(ipaton{“A”},B)	
Parameters-description	No value return **Input variables**	
	A	IP address of Network communication
	B	Port of Network communication
Example	local CameraNet={ipaton(“192.168.0.100”),8080} -- IP：192.168.0.100；Port: 8080 sysnetclr(CameraNet) --Clear the receiving buffer of network	

sysnetget

Use-explanation	Read network data with unblock mode		
Syntax-description	Err,Data=sysnetget({ipaton(“A”,B})		
Parameters-description	**Input variables**		
	A	IP address of Network communication	
	B	Port of Network communication	
	Return Value(Receiving No.)		
	Err	0	Success
		Non-zero	Fail
	Data	Buffer of receiving data	
Example	local CameraNet={ipaton(“192.168.0.100”),2000} local Err		

	local RecBuf Err,RecBuf = sysnetget(CameraNet) -

sysnetsend

Use-explanation	Send network data		
Syntax-description	sysnetsend({ipaton("A"),B},C)		
Parameters-description	No value return **Input variables**		
	A	IP address of Network communication	
	B	Port of Network communication	
	C	Send data with Network	
Example	local CameraNet={ipaton("192.168.0.100"),8080} 1、 local data1={0x23,0x11,0x33} sysnetsend(CameraNet,data1) 2、 local data2 = "trigger" sysnetsend(CameraNet,data2) 3、 local value = 123.67 sysnetsend(CameraNet,string.format("%f",value))		

sysnetcatch

Use-explanation	Read network data with blocking mode		
Syntax-description	Err,RecBuf= sysnetcatch({ipaton("A"),B},C)		
Parameters-description	**Input variables**		
	A	IP address of Network communication	
	B	Port of Network communication	
	C	Blocking time, whose unit is millisecond (ms)	
	Return value(Receiving No)		
	Err	0	Success
		Non-0	Fail
	Data	Buffer of receiving data	
Example	local CameraNet={ipaton("192.168.0.100"),2000} local Err,RecBuf = sysnetcatch(CameraNet,2000)		

CloseNet

Use-explanation	Close the TCP network connection	
Syntax-description	CloseNet ({ipaton("A"),B})	
Parameters-description	**Input variables**	
	A	For TCP Network communication，**A** is IP address when RC400 controller acts as the client.
	B	For TCP Network communication，**B** is port number

		when RC400 controller acts as the client.

OpenNet

Use-explanation	Build a TCP network		
Syntax-description	OpenNet({ipaton("A"),B})		
Parameters-description	**Input variables**		
	A	For TCP Network communication，**A** is IP address when RC400 controller acts as the client.	
	B	For TCP Network communication，**B** is port number when RC400 controller acts as the client.	
	Return values(Receiving Error No)		
	Err	0	success to build a TCP Network
		Non-0	failed to build a TCP Network

ConnectNet

Use-explanation	Connect to the TCP network		
Syntax-description	ConnectNet({ipaton("A"),B})		
Parameters-description	**Input variables**		
	A	For TCP Network communication, **A** is IP address when RC400 controller acts as the client.	
	B	For TCP Network communication, **B** is port number when RC400 controller acts as the client.	
	Return values(Receiving Error No)		
	Err	0	success to connect to a TCP Network
		Non-0	failed to build a TCP Network

RecvNet

Use-explanation	Receive data through TCP network		
Syntax-description	Err_net,RecBuf = RecvNet ({ipaton("A"),B},C)		
Parameters-description	**Input variables**		
	A	For TCP Network communication, **A** is IP address when RC400 controller acts as the client.	
	B	For TCP Network communication, **B** is port number when RC400 controller acts as the client.	
	C	Timeout time(unit is **ms**)	
	Return values(Receiving Error No)		
	Err_net	0	success to receive network data
		Non-0	failed to receive network data
	RecBuf	Receive data buffer	

WriteNet

Use-explanation	Send data through TCP Network	
Syntax-description	WriteNet ({ipaton("A"),B},C)	
Parameters-description	**Input variables**	
	A	For TCP Network communication, **A** is IP address when RC400 controller acts as the client.
	B	For TCP Network communication, **B** is port number when RC400 controller acts as the client.
	C	Data to be sent

Examples

```
local    ipPort={ipaton("192.168.0.100"),2000}    --ip,port
CloseNet(ipPort)                                  --Close TCP network
Delay(100)
print("Close TCP network!")
if OpenNet(ipPort) == 0 then                      --Open TCP network
    print("Open TCP network！Connecting...")
    repeat
        Delay(2)
    until ConnectNet(ipPort) == 0          --Connect to TCP network
    print("Has been connected to TCP network!")
end
while 1 do
    local x,y,z,c
    local err_net
    err_net,RecBuf = RecvNet(ipPort,5000)
    if err_net == 0 then                   --Receive successfully
        if RecBuf.buff == "?" then
            WriteNet(ipPort, "OK")
        end
    else
        print("接收失败:",err_net)
    end
    Delay(10)
end
```

publicread

Use-explanation	Read the data from GlobalData list	
Syntax-description	C = publicread(A, "B")	
Parameters-description	**Input variables**	
	A	Address of global data; length of address is 2
	B	The type of reading data, integer（ignore）/float/Hex
	Return value	

	C	Read the data of corresponding address in global list
Example	1. local a = publicread(0x100, “float”) --Read address 0x100 -- in floating type 2. local b = publicread(0x102) --Read address 0x102 --in integer type 3. local c=publicread(0x100,3) --Read values of addresses --0x100, 0x102 and 0x104 in integer type; return values are stored --in table **c**, in which value of 0x100 is stored in **c[1]**, value of --0x102 is stored in **c[2]**, value of 0x104 is stored in **c[3]**. 4. local d = publicread(0x100,3, “float”) --Read values of --addresses 0x100, 0x102 and 0x104 in float type; return values are --stored in table **d**, in which value of 0x100 is stored in **d[1]**, value --of 0x102 is stored in **d[2]**, value of 0x104 is stored in **d[3]**.	

publicwrite

Use-explanation	Write data to GlobalData list	
Syntax-description	◆ publicwrite(A,B,“D”) ◆ publicwrite(A,C, “D”)	
Parameters-description	No value return **Input variables**	
	A	Address of global data; length of address is 2
	B	Write data(single) to the corresponding address in **GlobalData**
	C	Write data(table) to the corresponding continuous addresses in **GlobalData**
	D	Type of writing data, integer (ignore)/float/Hex
Example	1. publicwrite(0x102,10.5,“float”) --Write float data(10.5) to --address 0x102 2. publicwrite(0x102,5) --Write Integer data(5) to address --0x102 3. publicwrite(0x100,{100,200,300}) --Write table {100,200,300} --into three consecutive addresses (0x100,0x102,0x104) with 0x100 --as the starting address 4. publicwrite(0x100,{10.1,20.1,30.1})--Write table {10.1,20.1,30.1} --into three consecutive addresses (0x100,0x102,0x104) with 0x100 --as the starting address	

1.20 Commands of Vision

Symbols of Commands	Explanations of Commands
CCDrecv	Receive data which sent from a camera
CCDtrigger	Trigger camera to take a photo
CCDsent	Send character string to a camera

CCDclr	Clear the network IP
CCDoffset	Visual deviation compensation
GetDynCCDPos	Transform the coordinate of dynamic camera to robot coordinate
CCDGet	Receive data(string) which is sent from vision(camera)

CCDrecv

Use-explanation	Receive data which sent from a camera	
Syntax-description	n,data=CCDrecv(“CamName”)	
Parameters-description	CamName	Name of camera, which set in Vision Configuration
Return values	n	Number of received data
	data	Absolute coordinate received from vision

Example

```
n,data=CCDrecv(“CAM1”)        --receive data from camera CAM1
for i=1,n do
    print(i,data[i][1],data[i][2],data[i][3])
    if data[i][1]~=0 or data[i][2]~=0 then
        pos.x=data[i][1]      --Assign data[i][1] to pos.x
        pos.y=data[i][2]      --Assign data[i][2] to pos.y
        pos.z=0
        pos.c=data[i][3]      --Assign data[i][3] to pos.c
    end
end
MovP(pos)
```

CCDtrigger

Use-explanation	Send a character string to trigger camera to take a photo	
Syntax-description	CCDtrigger(“CamName”)	
Parameters-description	CamName	Name of camera, which is set in Vision Configuration
Return	No	

CCDsent

Use-explanation	Send a character string to a camera	
Syntax-description	CCDsent(“CamName”, Buff)	
Parameters-description	CamName	Name of camera, which is set in Vision Configuration
	Buff	Character string to be sent
Return	No	

CCDclr

Use-explanation	Clear the network IP
Syntax-description	CCDclr(“CamName”)

Parameters-description	CamName	Name of camera, which is set in Vision Configuration
Return	No	

CCDoffset

Use-explanation	Visual deviation compensation	
Syntax-description	pos = CCDoffset(“CamName”,view_pos)	
Parameters-description	CamName	Name of camera, which is set in Vision Configuration
	view_pos	Receiving visual coordinates
Return	pos	Absolute coordinates after output compensation

GetDynCCDPos

Use-explanation	Transform the coordinate of dynamic camera to robot coordinate	
Syntax-description	robot_pos= GetDynCCDPos(“CamName”,view_pos)	
Parameters-description	CamName	Name of camera, which set in Vision Configuration
	view_pos	Coordinate of dynamic camera
Return	robot_pos	Absolute coordinates of the output calculation

CCDGet

Use-explanation	Receive data which is sent from vision(camera)	
Syntax-description	RecBuf=CCDGet(“CamName”)	
Parameters-description	CamName	Name of camera(string type), which is set in Vision_Configuration
Return values	RecBuf	received data(string) from vision
Example	local RecBuf = CCDGet(“CAM1”) if RecBuf.buff == “OK” then MovP(p1) elseif RecBuf.buff == “NG” then MovP(p2) end	

1.21 Commands of Follow-camera

Symbols of Commands	Explanations of Commands
FollowInit	Initial parameters about follow-camera
SetDynCatch	Open or close follow-grasping task
GetCatchSpace	Obtain whether the workpiece has reached the grasping area
SetCatch	Carry out the follow task
GetCatchState	Obtain the catch state
SynOver	Over the synchronization
GetTrigger	Obtain the trigger state

SetViewData	Send the received data to controller, then save to Catch queue

FollowInit

Use-explanation	Initial parameters about follow-camera	
Syntax-description	FollowInit(CamName)	
Parameters-description	CamName	Name of camera, which is set in Vision Configuration
Return	No	
Example	FollowInit("CAM1") --Initial parameters about --follow-camera(CAM1)	

SetDynCatch

Use-explanation	Open or close follow-grasping task	
Syntax-description	SetDynCatch(n)	
Parameters-description	n	Flag of Open or close
		0: Close follow-grasping task
		1: Open follow-grasping task
Example	SetDynCatch(0) --Close follow-grasping task SetDynCatch(1) --Open follow-grasping task	

GetCatchSpace

Use-explanation	Obtain whether the workpiece has reached the grasping area	
Syntax-description	state=GetCatchSpace()	
Parameters-description	No	
Return	state	**0:** has not reached the grasping area
		1: Has reached the grasping area

SetCatch

Use-explanation	Carry out the follow task
Syntax-description	SetCatch()
Parameters-description	No parameters
Return	No return

GetCatchState

Use-explanation	Obtain the catch state	
Syntax-description	state=GetCatchState ()	
Parameters-description	No	
Return	0	Catch over
	1	start to move to destination from current point

	2	Enter synchronization: synchronization has been successful and has the same speed and position.
	3	Exit with error: over grasping area, so it is failed to finish grasping

SynOver		
Use-explanation	Over the synchronization	
Syntax-description	SynOver()	
Parameters-description	No varibales	
Return	No return	

GetTrigger		
Use-explanation	Obtain the trigger state	
Syntax-description	num=GetTrigger ()	
Parameters-description	No variables	
Return	0	Flag of finishing the first photo to prepare to take the second photo
	1	Flag of finishing the second photo to prepare to take the fist photo again

SetViewData		
Use-explanation	Send the received data to controller, then save to Cache queue	
Syntax-description	SetViewData (pos)	
Parameters-description	pos	Data which is received from vision(camera)

Example

```
local pos={x=0,y=0,z=0,c=0,h=0}
local n,data=CCDrecv("CAM1")
if data then
    for i=1,n do
        if data[i][1]~=0 and data[i][2]~=0 then
            pos.x=data[i][1]
            pos.y=data[i][2]
            pos.c=data[i][3]
            SetViewData(pos)
        end
    end
end
```

1.22 Commands of Debugging

Symbols of Commands	Explanations of Commands
print	Print the output of user debugging data

Error	Terminate the running AR program and give error information

print

Instruction-manual	Export the output of debugging data from RS232 serial port	
Syntax-description	print(…)	
Parameters-description	Number and type of parameters can be arbitrarily	
Example	1. print(12)	--Export data 12 from serial port
	2. print (“Robot”)	--Export string (Robot) from serial port
	3. local a=10	
	print (“Robot”, a)	--Export string(Robot) and number (10)

Error

Instruction-manual	Terminate the running AR program and give error information
Syntax-description	Error(A)
Parameters-description	Error information, which type is character
Example	Error(“AR running with error”)

1.23 Commands of Point

Symbols of Commands	Explanations of Commands
Point	Call the points from point list except for CPU1
new	Write user-defined point to DATA.PTS of current project
teach	Write current point to DATA.PTS of current project

Point

Instruction-manual	Call the points from point list except for CPU1	
Syntax-description	pos1= Point (A)	
Parameters-description	A	Point data in DATA.PTS (1~2999)
Return	pos1	Assign the point data to pos1
Example	local pos={}	
	pos = Point(10)	--Call p10 of DATA.PTS and assign it to **pos** point
	print(pos.x,pos.y,pos.z,pos.c)	

- **Notice:** value of A can only be one of 1~2999, not p1~p2999.

new

Instruction-manual	Write user-defined point to DATA.PTS of current project	
Syntax-description	new(A,B,C,D,E,F)	
Parameters-description	A	Data No. (1~2999) in DATA.PTS
	B	X coordination of user-defined point

	C	Y coordination of user-defined point
	D	Z coordination of user-defined point
	E	C coordination of user-defined point
	F	Hand coordinate of user-defined point(0:left hand; 1: right hand)
Example	print(pos.x,pos.y,pos.z,pos.c) new(5,300,100,-10,30,0) --Write point (300,100,-10,30) to p5 of --DATA.PTS	

teach		
Instruction-manual	Write current point to DATA.PTS of current project	
Syntax-description	teach (A)	
Parameters-description	A	Data No. (1~2999) in DATA.PTS
Example	local pos = {x=300,y=-100,z=10,c=30,h=1} MovP(pos) teach(10) --Write pos={ x=300,y= -100,z= -10,c=30,h=1} to p10 in --DATA.PTS	

1.24 Commands of System

Symbols of Commands	Explanations of Commands
syswork	Set the working state of system
sysstate	Obtain the state of system or current of each axis
sysrate	Set the global speed rate
systime	Obtain the clock time of system

syswork		
Use-explanation	Set the working state of system	
Syntax-description	syswork(A)	
Parameters-description	**Input variables**	
	A	**1:** start AR program
		2: pause AR program
		3: stop AR program
		4: reset AR program
Examples	MovP(p1) --Move to p1 with PTP syswork(2) --Pause Delay(100) --Delay with 100ms syswork(1) --Restart	

sysstate

Use-explanation: Obtain the state of system

Syntax-description: state = sysstate(n)

n	state	
Empty variable	Obtain alarm information of the robot	**0:** No alarm
		1: servo alarm
		2: DSP alarm
		4: operation alarm
		8: system alarm
0	Obtain enable information of the robot	**0**: disable
		1: enable
1	Obtain the current value of J1 axis, unit is mA	
2	Obtain the current value of J2 axis, unit is mA	
3	Obtain the current value of J3 axis, unit is mA	
4	Obtain the current value of J4 axis, unit is mA	
5	Obtain robotic running state	0：idle
		1：pause
		2：running
6	Obtain robotic mode	0：manual mode
		1：auto mode
7	Obtain servo alarm No.	
8	Obtain DSP alarm No.	
9	Obtain operating alarm No.	
10	Obtain system alarm No.	

Examples

1.
```
local state1 = sysstate(1)    -- Obtain the current value of J1 axis
local state2 = sysstate(2)    -- Obtain the current value of J2 axis
local state3 = sysstate(3)    -- Obtain the current value of J3 axis
local state4 = sysstate(4)    -- Obtain the current value of J4 axis
```
2.
```
local state = sysstate(5)
if state == 0 then
    print("robot is idle")
elseif state==1 then
    print("robot is pause")
elseif state==2 then
    print("Robot is running")
end
```
3.
```
local state=sysstate(6)
if state == 0 then
    print("Robot stays at manual mode")
elseif state==1 then
```

```
        print("Robot stays at auto mode")
    end
4.  state7 = sysstate(7)      -- Obtain servo alarm No.
    state8 = sysstate(8)      -- Obtain DSP alarm No.
    state9 = sysstate(9)      -- Obtain operating alarm No.
    state10 = sysstate(10)    -- Obtain system alarm No.
```

sysrate		
Use-explanation	Obtain or set the global speed rate	
Syntax-description	Rate=sysrate(A)	
Parameters-description	A	Rate
	Empty variable	Obtain global speed rate
	1~100	Set global speed rate
Example	1. sysrate() --Obtain current global current speed rate 2. sysrate(50) --Set global speed rate as 50%	

systime		
Use-explanation	Obtain the clock time of system	
Syntax-description	time = systime()	
Parameters-description	Empty variable Return	
	time	System clock
Example	local time1=systime() --Obtain the current system clock MovP(p1) MovP(p2) local time2=systime()-time1 --Calculate AR running time print(time2)	

1.25 Commands of Modbus Communication (Robot as Poll)

Symbols of Commands	Explanations of Commands
ReadRegW	Read the specified address of 16-bit word from PLC register
ReadRegDW	Read the specified address of 32-bit word from PLC register
WriteRegW	Write 16-bit data to specify address of PLC register
WriteRegDW	Write 32-bit data to specify address of PLC register

ReadRegW	
Use-explanation	Read the specified address of 16-bit word from PLC register
Syntax-description	Value = ReadRegW({A,B},C,D)

Parameters-description	A	Data communication protocol: A=1 is Modbus/RTU communication A=3 is Modbus/TCP communication
	B	Station ID of slave
	C	Register address
	D	Number of variables to be read(Optional),which default is 1
	Value	Returns the value of a variable which is corresponding to the reading register address
Example	1. local plc={1,1} --PLC communication parameters(1 means UART; station number is 1) 2. ReadRegW(plc,250) --Read 16-bit data from PLC address 250 3. local a=ReadRegW(plc,250,20) --Read 20 16-bit data starting from PLC address 250 for i=1,20 do print(a[i]) end	

ReadRegDW

Use-explanation	Read the specified address of 16-bit word from PLC register	
Syntax-description	ReadRegDW ({A,B},C,D,E)	
Parameters-description	A	Data communication protocol: A=1 is Modbus/RTU communication A=3 is Modbus/TCP communication
	B	Station ID of slave
	C	Register address
	D	Number of variables to be read(Optional),which default is 1
	E	Type of variables(Optional), which can be integer or float(integer is default)
Example	1. local plc={1,1} --PLC communication parameters(1 means UART; station number is 1) print(ReadRegDW(plc,250)) --Read 32-bit integer data from PLC address 250 print(ReadRegDW(plc,250,“float”)) --Read 32-bit float data from PLC address 250 2. local a=ReadRegDW(plc,250,20) --Read 20 32-bit integer data starting from PLC address 250 3. for i=1,20 do print(a[i]) end 4. a =nil 5. local a=ReadRegDW(plc,250,20, “float”) --Read 20 32-bit float data starting from PLC address 250	

	6. for i=1,20 do print(a[i]) end 7. ReadRegDW(plc,250,0x100,10) 8. ReadRegDW(plc,250,0x100,10,“float”) 9. ReadRegDW(plc,250,{x=true,y=true,z=true,c=true,n=1},10) 10. ReadRegDW(plc,250,{x=true,y=true,z=true,c=true,n=1},10,“float”)

WriteRegW

Use-explanation	Write 16-bit data to specify address of PLC register	
Syntax-description	Value =WriteRegW ({A,B},C,D)	
Parameters-description	A	Data communication protocol: A=1 is Modbus/RTU communication A=3 is Modbus/TCP communication
	B	Station ID of slave
	C	Register address
	D	Numbers of variables to be written
Example	1. local plc={1,1} --PLC communication parameters (1 means UART; station number is 1) 2. WriteRegW(plc,250,1) --Write 16-bit data to address(250) of PLC register 3. WriteRegW(plc,250,{10,20,30}) --Continuously write 16-bit data starting from address 250 of PLC register	

WriteRegDW

Use-explanation	Write 32-bit data to specify address of PLC register	
Syntax-description	WriteRegDW ({A,B},C,D,E,F)	
Parameters-description	A	Data communication protocol: A=1 is Modbus/RTU communication A=3 is Modbus/TCP communication
	B	Station ID of slave
	C	PLC Register address
	D	Local address(Optional)
	E	Number of variables to be read
	F	Type of variables to be read(optional),which can be integer (by default) or float
Example	1. local plc={1,1} --PLC communication parameters (1 means UART; station number is 1) 2. WriteRegDW(plc,250,1) --Write 32-bit data to address 250 of PLC register 3. WriteRegDW(plc,250,{10,20,30}) --Continuously write 32-bit	

integer data starting from address 250 of PLC register

4. WriteRegDW(plc,250,1,“float”) --Write 32-bit float data to address 250 of PLC register
5. WriteRegDW(plc,250,{10,20,30},“float”) --Continuously write 32-bit float data starting from address 250 of PLC register

1.26 Commands of File Operation

Symbols of Commands	Explanations of Commands
fopen	Open file
fsize	File size
fwrite	Write data to file
fread	Read data from file
fseek	Seek file to move file pointer to the appointed position
feof	End file
fclose	Close file

fopen

Use-explanation	Open file	
Syntax-description	Address = fopen(path,mode)	
Parameters-description	path	**File path** Relative path---ignore drive; File is in the current project directory in this case; For example, current project directory is d:\projects\scara, and file locates in d:\projects\scara\test.txt; Absolute path ---full path of specified file(d:\test.txt)
	mode	**Open mode** "w" ---Only write "r" ---Only read "a" ---Additional way "+" ---read/write
Return	Address	Return address of file pointer 0: open successfully 1: open failed

fsize

Use-explanation	File size	
Use-explanation	size = fsize(file)	
Parameters-description	file	Address of file pointer

Return	size	File size

fwrite

Use-explanation	Write data to file	
Use-explanation	size = fwrite(file,data)	
Parameters-description	file	Address of file pointer
	data	Data to be written
Return	size	Size of written data >0 : success 0: fail

fread

Use-explanation	Read data from file	
Use-explanation	data = fread(file,len)	
Parameters-description	file	Address of file pointer
	len	Length of date to be read
Return	data	Data is read from file

fseek

Use-explanation	Seek file to move file pointer to the appointed position	
Use-explanation	Err = fseek(file,offset,whence)	
Parameters-description	file	Address of file pointer
	offset	displacement
	whence	“set” :locating to start “cur” :locating to current “end” :locating to end
return	Err	Error No 0: success

feof

Use-explanation	End file	
Use-explanation	flag = feof(file)	
Parameters-description	file	Address of file pointer
return	flag	Flag of ending file 0: not end -1: end of the file

fclose		
Use-explanation	Close file	
Use-explanation	Err = fclose(file)	
Parameters-description	file	Address of file pointer
return	Err	Error No 0: success
Example	function main() ----------write---------- local f=fopen("3.txt","w") if f > 0 then local len=fwrite(f,"write test!") print(len) fclose(f) end ----------Read---------- local f=fopen("3.txt","r") if f > 0 then print("size",fsize(f)) local data=fread(f,2) print(data.buff,data.len) fclose(f) end ----------read/write---------- local f=fopen("3.txt","wr") if f > 0 then local len=fwrite(f,"write test!") print(len) fseek(f,0,"set") local data=fread(f,2) print(data.buff,data.len) fclose(f) end end	

1.27 Commands of Queue Operation

Symbols of Commands	Explanations of Commands
qexist	Judge whether the queue exists
qcreate	Create a new queue
qpush	Push(write) data to the queue
qpop	Pop(delete) the first data from the queue
qfront	Fetch the first data from the queue

qpopfront	Fetch the first data from the queue and then delete it
qempty	Judge whether the queue is empty
qsize	Calculate size of the queue
qdestroy	Delete the queue

qexist

Use-explanation	Judge whether the queue exists	
Syntax-description	Flag = qexist(ID)	
Parameters-description	ID	Name of queue which type is positive integer
Return	Flag	**Flag =true** , queue exists Flag=false, queue does not exist

qcreate

Use-explanation	Create a new queue	
Syntax-description	qcreate(ID)	
Parameters-description	ID	Name of queue which type is positive integer
Return	No	

qpush

Use-explanation	Push(write) data to the queue	
Syntax-description	qpush(ID,data)	
Parameters-description	ID	Name of queue which type is positive integer
	data	Data to be written to the queue, which type can be number、string、table or combinations of them
Return	No	

qpop

Use-explanation	**Delete first data from the queue**	
Syntax-description	qpop(ID)	
Parameters-description	ID	Name of queue which type is positive integer
Return	No	

qfront

Use-explanation	**Fetch first data from the queue**	
Syntax-description	Data=qfront(ID)	
Parameters-description	ID	Name of queue which type is positive integer
Return	Data	First element(data) of the queue

qpopfront

Use-explanation	Fetch the first data from the queue and then delete it	
Syntax-description	Data=qpopfront(ID)	
Parameters-description	ID	Name of queue which type is positive integer
Return	Data	First element(data) of the queue

qempty

Use-explanation	Judge whether the queue is empty	
Syntax-description	Flag =qempty(ID)	
Parameters-description	ID	Name of queue which type is positive integer
Return	Flag	Flag = 1, queue is empty; Flag ~= 1, queue is not empty

qsize

Use-explanation	Calculate size of the queue	
Syntax-description	Len=qsize(ID)	
Parameters-description	ID	Name of queue which type is positive integer
Return	Len	Length of the queue

qdestroy

Use-explanation	**Delete the queue**	
Syntax-description	qdestroy(ID)	
Parameters-description	ID	Name of queue which type is positive integer
Return	No	

Example

```
flag1 = qexist(1)          --Judge whether queue 1 exists 否存在
if flag1 == true then      --Exist
    print("OK")
elseif flag1 ==false then    --Not exist
    qcreate(1)             --Create queue 1
end
qpush(1,123)               --Push 123 to queue 1
qpush(1, “abc”)             --Push “abc” to queue 1
qpush(1,{x=300,y=100,z=0,c=30})   --Push table (array) to queue 1
len = qsize(1)             --Calculate length of queue 1(element number)
print(len)                 --3
while qempty(1)~=1 do    --Judge whether queue 1 is empty
    data = qfront(1)       --Fetch first element of queue 1
```

```
print(data)          --打印
qpop(1)              --Delete first element of queue 1 end
qdestroy(1)          --Delete queue 1
```