Overview of
AR Language

Date: Jun, 2017

Vision: V2.0 (English)

SHINI

Contents

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
General Keys	break	ump 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
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
Movement 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.

	linerun	Achieve the manual control in Cartesian coordinate system under the condtion of automation.
	stoprun	Cancel the manual control in Cartesian coordinate system under the condtion of automation.
Commands of Movement Parameters	AccJ	Set a proportion of acceleration to affect MovJ /MovJR / MovP / MovPR / MArchP commands'accelerating time.
	DecJ	Set a proportion of deceleration to affect MovJ / MovJR / MovP / MovPR / MArchP commands' decelerating 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)
	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)
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
Input/Output 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.
Commands of Coordinate System	SetU	Set the current user coordinate system of robot
	SetT	Set the current tool coordinate system of robot
	WrU	Modify the data of user coordinate system
	WrT	Modify the data of tool coordinate system
	CacU	Build a new user coordinate system

	V2Tool	Calculate a new tool
	getcart	Obtain current cartesian coordinate of robot's end
	CacU	Build a new user coordinate system
Commands of Pallet	SetPlt	A command that sets the palletizing numbers
	GetPlt	A command that gets the data point of palletizing
	ToPutPLT	Put piecework to the pallet
	FromPutPLT	Leave the pallet
	ResetPLT	reset the palletizing number
Commands of Servo Management	MotOn	Open servo enables of all the axis
	MotOff	Close servo enables of all the axis
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
Vision Commands	VisToRob	Transform the pixel coordinate to the appointed user coordinate
	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
Follow-camera 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 Cache queue
Debugging Commands	print	Print the output of user debugging data
	Error	Terminate the running AR program and give error information
Point Commands	Point	Call the points from point list except for CPU1
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 PLCregister
	ReadRegDW	Read the specified address of 32-bit word from PLCregister
	WriteRegW	Write 16-bit data to specify address of PLC register
	WriteRegDW	Write 32-bit data to specify address of PLC register

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:

```
local point1={X=10,Y=20,Z=30,C=10,H=0 }
local point2={X=10,Y=20,Z=30,C=10,H=1 }
local point3 =point1+point2
```

After running this program, the result of point3 is: X = 20, Y = 40, Z = 60 and H = 0. H is handle coordinate system, which is subjected to the handle coordinate system of point1.

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 ：

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

== : it is can also 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:

```
if 5 or 6 then
…
end
local a =30
local b=true
if a and b then
…
end
local a=0
local b=1
if a and b then     ---line 11
…
end
local a=0
local b=nil
if a and b then
…
end
```

Notice: In AR language, only nil and false are not true, and the others are true including 0. The demonstration for line 11 above are valid.

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 :

```
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
```

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 Keyword

Symbols of Commands	Explanations of Commands
break	ump for/while/repeat loop
local	Define local variables
nil	Variable is null
return	Return a value of call function

local: it is used to declare user variables.

Example :

```
local a                    --declare a local variable a, which value is nil
local b={10,20}            --declare a table array
local b={{10,20},{30,40}}  --declare a table array with two dimensions
local a= "ROBOT"           --declare a string variable a
```

1.6 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

NO. of axis are also the global variables, which are acted as some parameters of movement commands. And they are not redefined by user.

1.7 Global Robotic Variables

Symbols of Commands	Explanations of Commands
ON	Open state
OFF	Close state
p0~p999	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 the system.

Point variables (p0~p999) are belonging to table, which are defined as follows when starts robot:

```
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

Example :

```
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
```

1.8 Commands of Process Control

Symbols of Commands	Explanations of Commands
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

1.8.1 if then else elseif end

Use explanation: a command of “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 conditions then
		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**”: they 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 to select to execute the program.

1.8.2 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”.

Syntax description:

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

1.8.3 for do end

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 var 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.

1.8.4 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”.

1.8.5 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.

Syntax description:

```
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 loo
```

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

1.8.6 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 is to realize the specific details of thefunction. 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:

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

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

1.9 Movement Commands

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.
linerun	Achieve the manual control in Cartesian coordinate system under the condtion of automation.
stoprun	Cancel the manual control in Cartesian coordinate system under the condtion of automation.

1.9.1 MovL

Use-explanation	Moving to the target position in the Cartesian coordinate system in a straight line.
Syntax-description	MovL (A,"CP=20 Acc=20 Dec=20 Spd=100 AccC=20 SpdC=20 In=10 PULSE+/ PULSE-")
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.

Some instructions as follows :

A	Descartes coordinates the target position. It can be the name of the point p0~p1000, also can be the point of the index 0~1000. Optional parameters, you can specify theparameters 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 location, Unit mm/s
AccC	Specify the acceleration of equipment's condition , unit mm/s^2
SpdC	Specify the speed of equipment's condition during moving to the target location, Unit mm/s
In	Specifies a trigger signal of input port to stop the movement from current position to target position.
PULSE+ /PULSE	PULSE+: rising edge is effective; PULSE-: falling edge is effective

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") -- the robot moves to the p20 position in a

straight line, where the target position is p20 smooth transition.

5. MovL (p20, " In=10 PULSE+") - the robot moves in a straight line to the p20 target position, and if the input signal 10 is detected as rising edge during movement, then current movement will be stopped.

1.9.2 MovLR

Use-explanation	Relative movement of linear mode in Descartes coordinate system
Syntax description	MovLR(A,B,“CP=20 Acc=20 Dec=20 Spd=100 AccC=20 SpdC=20 In=10 PULSE+/PULSE-")
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 acceleration of equipment’s condition , unit mm/s^2
SpdC	Specify the speed of equipment’s condition during moving to the target location, Unit mm/s
In	Specifies a trigger signal of input port to stop the movement from current position to target position.
PULSE+ /PULSE	PULSE+: rising edge is effective; PULSE-: falling edge is effective

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.

1.9.3 MovP

Direction	Mobile point-to-point to Cartesian coordinates of the target location
Syntax description	MovP (A, “CP=20 Acc=20 Dec=20 Spd=20”)
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, thesystem 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 p0~p1000, also can be the point of the index 0~1000
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

Example:

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

1.9.4 MovPR

Direction	Mobile point-to-point to Cartesian coordinates of the target location
Syntax description	MovPR(A,B, “CP = 20 Acc=20 Dec=20 Spd=20”)
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

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.

1.9.5 MovJ

Direction	Point-to-point mode joint mobile robot each to the specified position	
Syntax description	1. MovJ (A,B, “Acc=20 Dec=20 Spd=100”) 2. MovJ(A, “Acc=20 Dec=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~p999) under Cartesian coordinate system
	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
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), and the other joints are angle units.

1.9.6 MArchP

Direction	The robot in a point-to-point way arch movement	
Syntax description	MArchP(A,B,C,D, “Acc=20 Dec=20 Spd=20”)	
Parameter declaration	Instructions for the following instructions	
	A	Point name (p1~p999 or 1~999)
	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

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, this arriving height 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.

1.9.7 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~p999 or 1~999
	B	Target position (last point) of the arc movement under Cartesian coordinate system. Name of the point can be p1~p999 or 1~999
	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 ~ 360℃

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.

1.9.8 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~p999 or 1~999
	B	Target position (last point) of the circle movement under Cartesian coordinate system. Name of the point can be p1~p999 or 1~999
	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 ~ 360℃
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%.	

1.9.9 linerun

Use-explanation	Achieve the manual control in Cartesian coordinate system under the condtion of automation.
Syntax-description	line(A,B,C)

Parameters-description	Parameter declaration:	
	A	Cartesian coordinate axis, which could be one of AX,AY,AZ, AC
	B	moving direction; 1 represent positive direction，-1 represent negative direction
	C	moving distance
Example	1. linerun(AX,1,10) --X axis +direction moving 10mm / manual manipulation 2. linerun(AX,-1,10) --X axis -direction moving 10 mm/manual manipulation 3. linerun(AY,1,10) --Y axis +direction moving 10mm / manual manipulation 4. linerun(AY,-1,10) --Yaxis -direction moving 10mm / manual manipulation 5. linerun(AZ,1,10) --Z axis +direction moving 10mm / manual manipulation 6. linerun(AZ,-1,10) --Z axis -direction moving 10mm / manual manipulation 7. linerun(AC,1,10) --C axis -direction moving 10degrees/ manual manipulation 8. linerun(AC,-1,10) --C axis -direction moving 10degrees / manual manipulation	

*: linerun - Achieve the series of movement in Cartesian coordinate

1.9.10 stoprun

Use-explanation	Cancel the manual control in Cartesian coordinate system under the condtion of automation.
Syntax-description	stoprun()
Parameters-description	There is no parameter in this command

Example	function Bit(value,bit)
	if (value & (0x0001<<bit)) == 0 then
	return nil
	else
	return 1
	end
	end
	function ExtManu()
	local value
	value = publicread(0x100)
	if DI(0)==ON then --X+
	print("X+\n")
	linerun(AX,1,50)
	elseif DI(1)==ON then --X-
	print("X-\n")
	linerun(AX,-1,50)

```
    elseif DI(2)==ON then --Y+
        print("Y+\n")
        linerun(AY,1,50)
    elseif DI(3)==ON then --Y-
        print("Y-\n")
        linerun(AY,-1,50)
    elseif DI(4)==ON then --Z+
        print("Z+\n")
        linerun(AZ,1,50)
    elseif DI(5)==ON then --Z-
        print("Z-\n")
        linerun(AZ,-1,50)
    elseif DI(6)==ON then --C+
        print("C+\n")
        linerun(AC,1,50)
    elseif DI(7)==ON then --C
        print("C-\n")
        linerun(AC,-1,0)
    else
        stoprun()
    end
end
function main()
    while 1 do
        ExtManu()
        Delay(10)
    end
end
```

*:stoprun - Stop the series of movement in Cartesian coordinate，so commands “linerun” and “stoprun” both be used. Without “stoprun”， command “linerun” will continuously run until the robot arm reaching the limit position.

1.10 Commands of Movement Parameters

Symbols of Commands	Explanations of Commands
AccJ	Set a proportion of acceleration to affect MovJ / MovJR / MovP / MovPR / MArchP commands’ accelerating time.
DecJ	Set a proportion of deceleration to affect MovJ / MovJR / 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)

1.10.1 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

1.10.2 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

1.10.3 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.

1.10.4 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

1.10.5 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. AccL(500) --Set the deceleration as 500 mm/s^2 MovL(p2) --Move to p2 with 500 mm/ s^2 deceleration

1.10.6 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

1.11 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

1.11.1 Delay

Use-explanation	Delay command
Syntax-description	Delay(A)
Parameters-description	A is the delay time, which range is 1ms~100000ms
Example	1. Delay(1000) --A delay with 1000ms 2. Delay(1) --A delay with 1ms 3. local time = 1000 4. Delay(time) --the parameter “time” is a local variable

1.11.2 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 2. MovL(1) --This command is not executed

1.11.3 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

1.12 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

1.12.1 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℃ , which is corresponding to the current position.

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

1.12.2 XYZC

Use-explanation	Used to build the point(under Cartesian coordinate system) of each specified axis.	
Syntax-description	XYZC(A,B,C,D)	
Parameter declaration	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℃ , all of which are corresponding to p10.	

1.13 Input/Output Commands

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.

1.13.1 DI

Use-explanation	Read the state of input port
Syntax-description	**Type1**: DI() **Type2**: DI(A)
Parameters-description	**Type1**: **Return value:** A 32-bit binary number, from low to high respectively representing the status value of the input port 0~33 **Type2**: **Return value:** ON or OFF
A	Number of input port, which range is 0~33
Example	1. local input = DI() --Obtain status of all input ports if ((input>>0)&0x0001)==1 then --Judge whether input port 0 is open, if true, thenport 0 is open MovP(p1) elseif ((input>>9)&0x0001)==1 then -- Judge whether input port 9 is open, if true, thenport 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

Notice: DI is only to read the state of input ports, but it will not always wait no matter the state is effective or not.

1.13.2 DO

Use-explanation	Read the state of output port
Syntax-description	DO(A) DO(A,B, " Time = 1000")
Parameters-description	Return value ON or OFF
A	Read the number of output port, which range is 0~23
B	Write the state of output port (ON or OFF)
Time	Optional parameter, which is the time to hold this state, (unit is millisecond).

Example	1. if DO(10) == ON then --If output port 10 is ON, then move to p1 MovP(p1) end 2. DO(10,ON) --Open the output port 10 to ON 3. DO(10,ON,"Time=1000") --Open the output port 10 to ON holding 1000ms;Over 1000ms, the output port 10 will turn to OFF.

1.13.3 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	**Return** ON or OFF

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)

1.13.4 WDO

Use-explanation	Read the state of one output port. AR program will be continued to run until this signal is effective.
Syntax-description	WDO(A,B) WDO(A,B, "Time=1000")
Parameters-description	**Return** value ON or OFF

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) --Wait until output port 10 is ON, then run the --following command MovP(p2) 2. WDO(10,ON, "Time=2000") --Waiting for output 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)

1.14 Commands of Coordinate System

Symbols of Commands	Explanations of Commands
SetU	Set the current user coordinate system of robot
SetT	Set the current tool coordinate system of robot
WrU	Modify the data of user coordinate system
WrT	Modify the data of tool coordinate system
CacU	Build a new user coordinate system
V2Tool	Calculate a new tool
getcart	Obtain current cartesian coordinate of robot's end

1.14.1 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

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

1.14.2 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

1.14.3 WrU

Use-explanation	Modify the offset value of user coordinate system	
Syntax-description	WrU(A,B,C)	
Parameter declaration	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	

1.14.4 WrT

Use-explanation	Modify the offset value of tool coordinate system	
Syntax-description	WrT (A,B,C)	
Parameter declaration	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, A Y, 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 3. WrT(1,{x= 10,y=20,z=0,c=30}) –1Modify the offset value of tool coordinate system when the end of robot arm at X、Y、Z、C axis and when the moving distance and angle are 10mm,20mm,0mm,30 degrees. 4. Offset = { x=10,y=10,z=0,c=-30} WrT(2,Offset)--the parameter “Offset” is a local variable, 2Modify the offset value of tool coordinate system when the end of robot arm at X、Y、Z、C axis and when the moving distance and angle are 10mm,10mm,0mm,-30 degrees.	

1.14.5 CacU

Use-explanation	Build a new user coordinate system	
Syntax-description	CacU (pos1,pos2)	
Parameter declaration	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 localpos2 = {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	

1.14.6 V2Tool

Use-explanation	Calculate a new tool	
Syntax-description	V2Tool(A,B)	
Parameter declaration	A	Visual Cartesian coordinate
	B	Number of tool
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	

1.14.7 getcart

Use-explanation	Obtain current cartesian coordinate of robot's end	
Syntax-description	pos1,pos2 = getcart()	
Parameter declaration	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	

1.15 Commands of Pallet

Symbols of Commands	Explanations of Commands
SetPlt	A command that sets the palletizing numbers
GetPlt	A command that gets the data point of palletizing
ToPutPLT	Put piecework to the pallet
FromPutPLT	Leave the pallet
ResetPLT	Reset the palletizing number

1.15.1 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	-axis palletizing interval numbers
Example	1. SetPlt(1,p1,p2,p3,5,11) --set XY axis to pallet 2. SetPlt(1,p1,p2,p3,p4,11,5,3) --set XYZ axis to pallet	

Notice: palletizing commands is only executed one time at the beginning of program, which automatically determines to palletize on XY-axis or XYZ-axis according to the numbers of parameters.

1.15.2 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,p3,5,11) --Set XY-axis palletizing
i=1
j=1
while i <= 5 do
      j=1
      while j <= 11 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
```

1.15.3 ToPutPLT

Use-explanation	Put piecework to the pallet

Syntax-description	ToPutPLT(PltName)
Parameters-description	**Inputs**
	PltName　Palletizing name（ PLT0~PLT4）

1.15.4 FromPutPLT

Use-explanation	Leave the pallet
Syntax-description	FromPutPLT(PltName)
Parameters-description	**Return values**
	O　Pallet is not full; continue to palletize
	1　Pallet is full, please change a new pallet
	Inputs
	PltName　Palletizing name（ PLT0~PLT4）

1.15.5 ResetPLT

Use-explanation	Reset the palletizing number
Syntax-description	ResetPLT(PltName,count)
Parameters-description	**No return** **Inputs**
	PltName　Palletizing name（ PLT0~PLT4）
	count　which means that the beginning position to be palletized

Example	local posReady = p1　　--Safety point or waiting point
	local quliao = p2　　--Point of picking up piecework
	MArchP(posReady,5,1,1)　--Move to safety point with MArchP
	while 1 do
	　MArchP(posReady,5,1,1) --Move to safety point with MArchP
	　MArchP(quliao,5,1,1) --Move to pickingup point with MArchP
	　ToPutPLT(“PLT1”)　　-Put piecework to pallet PLT1
	　full=FromPutPLT(“PLT1”) --Leave pallet PLT1
	　　--“full” is used to determine whether the PLT1 has being
	　　--palletized fully; if full, then “full” == 1
	　if full==1 then
	　ResetPLT(“PLT1”,1) --If full is 1, then restart palletizing from 1
	end
	　end

1.16 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

1.16.1 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.

1.16.2 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

1.17 Communication Commands

Symbols of Commands	Explanations of 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

1.17.1 RecCom

Use-explanation	Receive data from RS232 serial port	
Syntax-description	Err, RecBuf=RecCom(A, "Time=5000")	
Parameter declaration	A	1 (RS232 serial port COM1)
	Time	Optional parameter, which is timeout for receiving and whose unit is milliseconds
	Return values is in following table:	
	RecBuf	Receive data to buffer
	Err	Receive error number

		0 Receive successfully
		1 Receive timeout
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	

1.17.2 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”)	

1.17.3 ClrCom

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

1.17.4 sysnetclr

Use-explanation	Clear the receiving buffer of network	
Syntax-description	Sysnetclr(ipaton{“A”},B)	
Parameters n	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

1.17.5 sysnetget

Use-explanation	Read network data with unblock mode	
Syntax-description	Err,Data=sysnetget({ipaton(“A”,B})	
Parameter declaration	Input variables	
	A	IP address of Network communication
	B	Port of Network communication
	Return Value(Receiving No)	
	Err	0 Success
		1 Fail
	Data	Buffer of receiving data
Example	local CameraNet={ipaton(“192.168.0.100”),8080} local Err local Data Err,Data = sysnetget(CameraNet)	

1.17.6 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} local data={0x23,0x11,0x33} sysnetsend(CameraNet,data) local data = “trigger” sysnetsend(CameraNet,data)	

1.17.7 sysnetcatch

Use-explanation	Read network data with block mode	
Syntax-description	Err,Data= sysnetcatch({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

		1:Fail
	Data	Buffer of receiving data
Example	local CameraNet={ipaton("192.168.0.100"),8080} local Err,Data = sysnetcatch(CameraNet)	

1.17.8 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.

1.17.9 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
		1: failed to build a TCP Network

1.17.10 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
		1: failed to build a TCP Network

1.17.11 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
		1: failed to receive network data
	RecBuf	Receive data buffer

1.17.12 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(“Receive failure:”,err_net)
    end
    Delay(10)
```

end

1.17.13 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， int（ ignore） /float/Hex

Return value

C	Read the data of corresponding address in global list

Example	local a = publicread(0x100, “float”) local b = publicread(0x102)

1.17.14 publicwrite

Use-explanation	Write data to GlobalData list
Syntax-description	publicwrite(A,B,“C”)
Parameters-description	**No value return** Input variables

A	Address of global data; length of address is 2
B	Write data to the corresponding address in global list
C	The type of writing data, int(ignore)/float/Hex

Example	publicwrite(0x102,10.5,“float”) publicwrite(0x102,5)

1.18 Vision Commands

Symbols of Commands	Explanations of Commands
VisToRob	Transform the pixel coordinate to the appointed user coordinate
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

1.18.1 VisToRob

Use-explanation	Transform the pixel coordinate to the appointed user coordinate
Syntax-description	D = VisToRob (A,B,C)
Parameter declaration	

A	IP address of Network communication
B	Port of Network communication

	C	Flag of camera's pixel, which range is 0~9
		0: 0.3 mega pixels（640*480）
		1: 0.5 mega pixels（800*600）
		2: 0.8 mega pixels（1024*768）
		3: 1 mega pixels（1140*900）
		4: 1.3 mega pixels（1280*960）
		5: 2 mega pixels（1600*1200）
		6: 3 mega pixels（2048*1536）
		7: 5 mega pixels（2576*1932）
		8: 5 mega pixels（2592*1944）
		9: 5 mega pixels（2560*1920）
Return value	D	Coordinates under world coordinate system

1.18.2 CCDrecv

Use-explanation	Receive data which sent from a camera	
Syntax-description	n,data=CCDrecv(CamName)	
Parameter declaration	CamName	Name of camera, which set in Vision Configuration
	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)
```

1.18.3 CCDtrigger

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

1.18.4 CCDsent

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

1.18.5 CCDclr

Use-explanation	Clear the network IP	
Syntax-description	CCDclr(CamName)	
Parameter declaration	CamName	Name of camera, which is set in Vision Configuration
Return	No	

1.18.6 CCDoffset

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

1.18.7 GetDynCCDPos

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

1.19 Follow-camera Commands

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 Cache queue

1.19.1 FollowInit

Use-explanation:	Initial parameters about follow-camera	
Syntax-description:	FollowInit(CamName)	
Parameter declaration	CamName	Name of camera, which is set in Vision Configuration
Return	No	
Example	FollowInit(“CAM1”) --Initial parameters about --follow-camera(CAM1)	

1.19.2 SetDynCatch

Use-explanation	Open or close follow-grasping task	
Syntax-description	SetDynCatch(n)	
Parameter declaration	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	

1.19.3 GetCatchSpace

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

1.19.4 SetCatch

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

1.19.5 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 andposition.
3 Exit with error: over grasping area, so it is failed to finish grasping

1.19.6 SynOver

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

1.19.7 GetTrigger

Use-explanation	Obtain the trigger state
Syntax-description	num=GetTrigger ()
Parameters-description	No
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

1.19.8 SetViewData

Use-explanation	Send the received data to controller, then save to Cache queue
Syntax-description	SetViewData (pos)
Parameters-description	Pos: data from vision

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

1.20 Debugging Commands

Symbols of Commands	Explanations of Commands
print	Print the output of user debugging data
Error	Terminate the running AR program and give error information

1.20.1 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 4. print (“Robot”, a) --Export string(Robot) and number (10)

1.20.2 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”)

1.21 Commands of Point

Symbols of Commands	Explanations of Commands
Point	Call the points from point list except for CPU1

1.21.1 Point

Instruction-manual	Call the points from point list except for CPU1	
Syntax-description	pos1= Point (A)	
Parameter declaration	A	Point data in DATA.PTS (1~999)
Return	pos1	Assign the point data to pos1

Note that: value of A can only be one of 1~999, not p1~p999.

1.22 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

1.22.1 syswork

Use-explanation	Set the working state of system	
Syntax-description	syswork(A)	
Parameters-description	**Input variables**	
Parameter declaration	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	

1.22.2 sysstate

Use-explanation	Obtain the state of system	
Syntax-description	state = sysstate(n)	
Parameters-description	**No input; return value(state)**	
	0	Normal running state
	!0	Abnormal state
	With Input variable n(1~4), which represents each axis of robot Return value(state)	
	Returns	current value of the designated axis(unit is mA)
Examples	1. local state state = sysstate() --Obtain system's state if state ~= 0 then --if not 0 syswork(4) --Reset end 2. local state1 = sysstate(1) --Obtain the current of first axis local state2= sysstate(2) --Obtain the current of second axis local state3 = sysstate(3) --Obtain the current of third axis local state4 = sysatate(4) -- Obtain the current of fourth axis	

1.22.3 sysrate

Use-explanation	Set the global speed rate	
Syntax-description	sysrate(A)	
Parameters-description	**Input variables**	
	A	Speed rate, which range from 1 to 100
Example	sysrate(50) --Set global speed rate as 50%	

1.22.4 systime

Use-explanation	Obtain the clock time of system	
Syntax-description	time = systime()	
Parameters-description	No input variables **Return(time)**	
	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)	

1.23 Commands of Modbus

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

1.23.1 ReadRegW

Use-explanation	Read the specified address of 16-bit word from PLC register	
Syntax-description	Value = ReadRegW({A,B},C,D)	
Parameter declaration	A	Data communication protocol; A means universal asynchronous Receiver/Transmitter(UART)
	B	Station number
	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	

1.23.2 ReadRegDW

Use-explanation	Read the specified address of 16-bit word from PLC register

Syntax-description	ReadRegDW ({A,B},C,D,E)	
Parameter declaration	A	Data communication protocol; A means universal asynchronous Receiver/Transmitter(UART)
	B	Station number
	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")	

1.23.3 WriteRegW

Use-explanation	Write 16-bit data to specify address of PLC register	
Syntax-description	Value =WriteRegW ({A,B},C,D)	
Parameter declaration	A	Data communication protocol; A means universal asynchronous Receiver/Transmitter(UART)
	B	Station number
	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

1.23.4 WriteRegDW

Use-explanation	Write 32-bit data to specify address of PLC register	
Syntax-description	WriteRegDW ({A,B},C,D,E,F)	
Parameter declaration	A	Data communication protocol; A means universal asynchronous Receiver/Transmitter(UART)
	B	Station number
	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 (1means 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