Robot Instruction Manual

Version： v0.0.1

Date： 2026.01.22

Contents

Command Overview

General Statements

Call	Call a function
Do..Loop	Loop statement that repeatedly executes between Do..Loop based on a condition
Else/ElseIf	Used with conditional statements; executes code between
Exit	Forcefully exit a loop or function
For..To..Next	Repeatedly executes the code between For...Next a specified number oftimes
Function..Fend	Function definition
Global	Declare a global variable
GoSub..Return	Jump to a subroutine
GoTo	Jump to a labeled point
If..Then..Else..EndIf	Ifthe statement is true, execute the code between Then... EndIf
Select..Case..Default..Send	Execute corresponding statements based on the Case results

Data Type

Boolean	Boolean type, True/False
Byte	Byte type, -128 to 127
Double	Double precis ion floating-point number
Int32	32-bit integer, -2147483648 to 2147483647
Integer	16-bit integer, -32768 to 32767
Long	Long integer, -2147483648 to 2147483647
Real	Sing le precis ion floating-point number
Short	Short integer, -32768 to 32767
String	String, up to 255 characters
UByte	Unsigned byte, 0 to 255
UInt32	Unsigned 32-bit integer, 0 to 4294967295
UShort	Unsigned short integer, 0 to 65535
Preserve	Define a global variable to retain memory during power loss; retains ,var iab le memory when stopping the program or rebooting the controller.

Motion commands

!...!	Parallel processing of input and output ofI/O during motion
Motor	Overall up and down enablement
ServoOff	Power-off Specified Axis
ServoOn	Power-on Specified Axis
Power	High/Low Power Switching
SpeedFactor	Global Speed Configuration (Affects Speed/SpeedS Parameters
Speed	Set/Display PTP Speed (Go/Jump)
Accel	PTP Motion Acceleration/Deceleration Configuration
Go	PTP to Target from Current Position
BGo	PTP Motion with Offset in Selected User Frame
TGo	Execute offset PTP motion in the selected tool coordinate system
SpeedS	Function for setting/displaying CP motion speeds for Move, Arc, Arc3, etc.
AccelS	Set/Display Acceleration/Deceleration for Linear Interpolation Motion
Move	Perform linear interpolation motion between the current position and a specified target position
BMove	Execute offset linear interpolation motion in the selected user coordinate system
TMove	Perform offset linear interpolation motion on the selected tool coordinate system.
Jump	PTP Gate-type Motion
Arc	Circular Interpolation Motion in the XY Plane
Arc3	3D Circular Interpolation Motion
CP	For configuring/displaying motion smoothing
CPZone	For configuring/displaying continuous motion smoothing parameters
R180	Restrict U-axis angular range to ±180°
Here	Used for teaching/returning the current coordinates of the robot
Home	Return to Home Position
Local	For configuring/displaying user coordinate system
TLSet	For configuring/displaying tool coordinate system
Tool	For selecting/displaying a specified tool coordinate system by ID
TCalib	For defining tool coordinate system (TCS) using three-point method
Till	For conditional motion termination during execution
Agl	For retrieving the angle/position of a specified joint

PAgl	For retrieving joint angles of a specified target position
AglToPls	For converting robot joint angles to servo pulses
Arch	Used to set/display the Arch parameters for Jump, Jump3, and Jump3CP commands
Find	Used to set/display the conditions for saving coordinates in action commands
FindPos	Used to return the saved coordinates via the Find function during the execution of motion commands.
InPos	Return whether the robot has reached its designated position
JA	Return the robot's coordinates based on specified joint angles
Joint	Used to display the robot's current coordinates in the joint coordinate system
LimZ	Used to set/display the initial value of the 3rd joint height (Z-coordinate value) for the Jump command
Pass	Used for performing PTP (Point-to-Point) motion that passes near a specified point without stopping
Pls	Used to return the pulse values of each joint at the current position
PPls	Used to specify the joint pulse values for a given coordinate point.
JS	To return whether a Sense input is valid
PosFound	To return the execution status obtained from the Find command
Pulse	Use PTP to move to the specified joint pulse positions, or return points based on pulse values
RobLoad	Used for selecting/displaying the load with the specified number
Sense	Used to set/show the conditions for Jump, Jump3, and Jump3CP
	To stop at a position above the target coordinates
TillOn	To return the status of Till
WaitPos	Await the cessation of robot motion
XY	Return the point position based on the specified numerical value
CX	For setting/getting the X - coordinate of a point.
CY	For setting/getting the Y - coordinate of a point.
CZ	For setting/getting the Z - coordinate of a point.
CU	For setting/getting the U - coordinate of a point.
CV	For setting/getting the V - coordinate of a point.
CW	For setting/getting the W - coordinate of a point.
CR	For setting/getting the R - coordinate of a point.

CS	For setting/getting the S - coordinate of a point.
CT	For setting/getting the T - coordinate of a point.
TargetOK	For detecting if the specified point's coordinates are reachable.
ColSup	For setting/getting the collision detection coefficient
ZeroPls	For setting/getting the robot's zero position
Hand	For setting/getting the current hand system.
Elbow	For setting/getting the elbow posture
Wrist	For setting/getting the wrist posture.

Independent axis command

IndPower	Used to set the enable status of the independent axis
IndHalt	Used to control the independent axis to stop moving
IndMovDone	Used to control the independent axis to stop moving
IndMovAbs	Used to control the independent axis to move to the specified absolute position
IndMoveRel	Used to control the independent axis to move to a relative motion
IndMovVel	Used to control the independent axis to maintain constant speed movement
IndReadPos	Used to read the current position of the independent axis
IndReadVel	Used to read the current movement speed of the independent axis
IndReadStatus	Used to read the current status of the independent axis
IndReset	for clearing independent axis errors

Point、I/O、Encoder sremark instructions

PDef	For checking if a point position is defined.
PDel	For deleting point positions.
PLabel	For setting labels for point - position data.
PLabel$	For getting the labels of point positions.
PNumber	For getting the point numbers
PList	For getting point position data.
PLocal	To return user coordinates of a point
PDescription	For setting a point's description.
PDescription$	For returning a point's description.
GetPointFiles	For getting a point - related file list.

DIDescription	Description of setting digital input data
DIDescription$	Description of returning digital input data
DODescription	Description of setting digital output data
DODescription$	Description of returning digital output data
AIDescription	Description of setting analog input data
AIDescription$	Description of returning analog input data
AODescription	Description of setting analog output data
AODescription$	Description of returning analog output data
EncDescription	Description of setting encoder data
EncDescription$	Description of returning encoder data

I/O commands

Wait	Delay order, or wait for a condition to hold within a set time.
Mask	Using bits, it conducts a bit - AND on the value for Wait input conditions.
On	Set DO to ON, or turn DO ON for a certain time then OFF.
Off	Set DO to OFF, or activate DO for a defined period and then leave it ON.
Oport	Get the state of the specified DO.
Sw	Get the status of the given DI.
SetSw	Set the given DI to ON or OFF.
In	Return 8-bit DI status according to 8421 code, range 0 - 255
SetIn	Set the 8-bit DI state using 8421 binary weights, with values from 0 to 255
Out	Set/Return the state of 8-bit DO according to 8421 code, ranging from 0 to 255
InW	Return the 16-bit DI status based on '8421', ranging from 0 to 65535.
SetInW	Set the state of 16-bit DI according to 8421, ranging from 0 to 65535.
OutW	Set/return the state of 16-bit DO according to 8421, ranging from 0 to 65535
InBCD	Return the state of 8-bit DI according to 8421, ranging from 0 to 99.
OpBCD	Set the state of 8-bit DO according to 8421, ranging from 0 to 99.
InReal	Return the DI value as a 32-bit floating-point number.
OutReal	Set/return the DO value as a 32-bit floating-point number.
AIO_InW	Return the value of the analog input.
AIO_OutW	Set/return the value of the analog output.
MemOn	Set bit 1 of the user register to 1.

MemOff	Set bit 1 of the user register to 0.
MemSw	Return the value of bit 1 of the user register.
MemIn	Return the value of 8 bits of the user register.
MemOut	Set the value of 8 bits of the user register.
MemInW	Return the value of 16 bits of the user register.
MemOutW	Set the value of 16 bits of the user register.
MemIn32	Return the value of 32 bits of the user register.
MemOut32	Set the value of 32 bits of the user register.
MemInReal	Return the 32-bit value of the user register as a 32-bit floating-point number.
MemOutReal	Set the 32-bit value of the user register as a 32-bit floating-point number.
SysMemOn	Set bit 1 of the system register to 1.
SysMemOff	Set bit 1 of the system register to 0.
SysMemSw	Return the value of bit 1 of the system register.
SysMemIn	Return the 8-bit value of the system register.
SysMemOut	Set the 8-bit value of the system register.
SysMemInW	Return the 16-bit value of the system register.
SysMemOutW	Set the 16-bit value of the system register.
SysMemIn32	Return the 32-bit value of the system register.
SysMemOut32	Set the 32-bit value of the system register.
SysMemInReal	Return the 32-bit value of the system register as a 32-bit floating-point number.
SysMemOutReal	Set the 32-bit value of the system register using a 32-bit floating-point number.

Communication commands

SetNet	For setting TCP/IP port parameters
OpenNet	For opening a TCP/IP port
WaitNet	For waiting for a TCP/IP port to establish a connection
ChkNet	For returning the number of bytes in the receive buffer of the network port
ClrNet	For clearing the receive buffer of the network port
CloseNet	For closing the TCP/IP port opened by OpenNet
SetCom	For setting serial communication parameters
OpenCom	For opening the serial port and configuring communication parameters

ChkCom	For the number of bytes in the receive buffer of the serial port
ClrCom	For clearing the receive buffer of the serial port
CloseCom	For closing the serial port opened by OpenCom
Read	For reading a specified number of bytes from the communication port
ReadBin	For reading binary data from the communication port
Write	Write the string to the communication port
WriteBin	For writing binary data to the communication port
Print #	For outputting data to the specified communication port
Input #	For reading strings or data from the communication port
Line Input #	For reading one line of data from the communication port

System commands

Print	Print a string or data
Xqt	Start multi-threading
Pause	Pause all threads
Halt	Pause the specified thread
Quit	Stop all or specified threads
Resume	Resume all or specified threads
Reset	Reset the controller to its initial state
MyTask	Return the current thread number
TaskDone	Check if a thread has completed execution
TaskState	Return the status of the specified thread
TaskWait	For waiting for the specified thread to end
SyncLock	Used for mutual exclusive locking to synchronize multiple threads
SyncUnLock	Used to unlock the signal locked by SyncLock
Signal	Used to send a signal to the task that is executing the WaitSig command
WaitSig	Used to wait for synchronization signals sent by the Signal command from other tasks
TW	Used to return the status of the Wait, WaitNet, and WaitSig commands
ElapsedTime	Return the elapsed time since the start of the calculation cycle time timer, in seconds.
ResetElapsedTime	Reset the ElapsedTime timer

Tmr	Return the elapsed time since the timer started, in seconds.
TmReset	Reset the Tmr timer
Eval	Used to execute statements in the command window.
Time	Used to print/return the controller's system time.
Time$	Used to return the controller's system time in the specified format hh:mm:ss.
Date	Used to print/return the controller's system date.
Date$	Used to return the controller's system date in the specified format yyyy:MM:dd.

Palletizing commands

Pallet	Define an array, get array points, print the array
PalletClr	Clear the array

Conveyor Tracking Commands

SyStart	Start Conveyor Tracking Function
SyEnd	End Conveyor Tracking Function
SyReset	Clear the current conveyor tracking queue data
SyGetPoint	Obtain the position data available for conveyor tracking
SyGetUserData	Get the additional information of the current tracking target
SyGetPose	Get the position of the current tracking target on the conveyor belt
SyGetNum	Get the number of targets in the conveyor tracking queue
SyMove	Conveyor tracking linear interpolation movement
SyArc	Conveyor tracking circular interpolation movement
SyTrig	Record the encoder pulse value of the specified conveyor belt
SyPoint	Generate tracking points based on current conveyor belt tracking
SyAddVisTarget	Register the vision coordinates into the conveyor tracking queue
SyRejectDis	Set/Get the filtering distance for the conveyor tracking function
SySave	Save the modifications to the conveyor tracking function in the program

Mathematical operation commands

+	Add
-	Subtract
*	Multiply
/	Divide
**	Exponentiation
Mod	Integer modulo
And	And
Or	Or
Xor	Xor
Not	Not
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
=	Equal to / Assign
DegToRad	Degrees to radians
RadToDeg	Radians to degrees
Sin	Sine function
Cos	Cosine function
Tan	Tangent function
Asin	Arcsine function
Acos	Arccosine function
Atan	Arctangent function
Atan2	Arctangent function 2
HexToFloat	Hexadecimal float to single-precision float
FloatToHex	Single-precision float to hexadecimal float
Abs	Absolute value
Sqr	Square root
Sgn	Return the sign of a value
LShift	Left shift

RShift	Right shift
BClr	Set the specified bit of a value to 0 and return the value
BSet	Set the specified bit of a value to 1 and return the value
BTst	Return the value of the specified bit of a number
Ubound	Get the array length
Dist	Returns the straight-line distance between two points.

String operation commands

ParseStr	String splitting
+	String concatenation
<>	Not equal to
=	Equal to, assignment
Val	String to numeric value
Str$	Numeric value to string
Len	Get string length
Asc	Character to ASCII code
Chr$	ASCII code to character
Left$	Extract specified characters from the left side of the string
Mid$	Extract characters or substrings within a specified range from the string
Right$	Extract specified characters from the right side of the string
LSet$	Get a string of specified length starting from the left side of the string
RSet$	Get a string of specified length starting from the right side of the string
Space$	Return a space string of specified length
LCase$	Return a string in lowercase letters
UCase$	Return a string in uppercase letters
LTrim$	Remove spaces from the left side of the string and return it
RTrim$	Remove spaces from the right side of the string and return it
Trim$	Remove spaces from both sides of the string and return it
InStr	Retrieve the position of a specified character from the string and return it
Tab$	Return a string of specified number of tabs
Hex$	Convert a hexadecimal value to a string and return it

Point operation commands

+	Relative coordinate incremental movement
-	Relative coordinate negative incremental movement
/	Modify hand system, modify user coordinate system
:	Absolute coordinate movement
=	Assignment
@	Convert to specified user coordinate system
X	X-direction component operation
Y	Y-direction component operation
Z	Z-direction component operation
U	U-direction component operation
V	V-direction component operation
W	W-direction component operation
RX	User coordinate system rotation component operation around X-axis
RY	User coordinate system rotation component operation around Y-axis
RZ	User coordinate system rotation component operation around Z-axis
TLX	Tool coordinate system rotation component operation around X-axis
TLY	Tool coordinate system rotation component operation around Y-axis
TLZ	Tool coordinate system rotation component operation around Z-axis
TLU	Tool coordinate system rotation component operation around U-axis
TLV	Tool coordinate system rotation component operation around V-axis
TLW	Tool coordinate system rotation component operation around W-axis
SavePoints	Save point file
LoadPoints	Load point file
ClearPoints	Clear points

Declaration of text symbols:

1. {Parameter}: The curly braces indicate that the parameter within them can be omitted and is not a mandatory syntax requirement.
2. [Parameter]: The square brackets indicate that one of the parameters within them must be chosen and filled in, which is a mandatory syntax requirement.

Common statements

Call

[Function]

Invokes a function.

[Syntax]

Call FunctionName

[Parameter Description]

FunctionName: The name of the Function to be called.

[Usage Example]

```
Function main
    Call Task2
    Task2
    Task2 ()
Fend
```

Do..Loop

[Function]

Execute code repeatedly between Do..Loop based on whether the condition is met.

[Syntax]

Do {While / Until conditional expression}

...

Loop

Or

Do

...

Loop {While/Until conditional expression}

[Parameter Description]

While/Until	Keywords that can be appended after Do or Loop to use the conditional expression.
Conditional expression:	Determine whether to exit the Do..Loop based whether the conditional expression is met.

[Usage Example]

```
Do                    'Loop continuously
    Go P1
    Wait 1
    Go P2
    Wait 2
Loop

    Do Until Sw (1) <> 1  'Enter the loop when DI1 is not equal to 1
    Go P1
    Wait 1
    Go P2
    Wait 2
Loop

Do                   'Loop continuously
    Go P1
    Wait 1
    Go P2
    Wait 2
Loop Until ChkNet (201) < 0  'Exit the loop when TCP/IP port 201 communication is abnormal.
```

Else/ElseIf

[Function]

Used in conjunction with judgment statements; when the If condition is not met, execute the code within Else/ElseIf.

[Syntax]

```
If  Conditional expression Then
    ...
ElseIf  Conditional expression  Then
    ...
Else
    ...
EndIf
```

[Parameter Description]

Conditional expression Execute the code within Else/ElseIf based on whether the conditional expression is satisfied.

[Usage Example]

```
If Sw(1) = 1 Then
    Print "DI1=On"

ElseIf Sw(1) = 0  Then
    Print "DI1=Off"

Else
    Print "DI1 abnormal"
End If
```

Exit

[Function]

Used to forcibly terminate a loop or function.

[Syntax]

Exit [Do/For/Function]

[Parameter Description]

Exit Do Exit the Do..Loop. If there are multiple nested Do..Loops, Exit Do will only exit one level of the loop.

Exit For Exit the For loop. If there are multiple nested For loops, Exit For will only exit one level of the loop.

Exit Function Exit the Function. If there are multiple nested Functions, Exit Function will only exit to the immediately preceding Function.

[Usage Example]

```
Do                          'Loop continuously
    Go P1
    Go P2
    If Sw(1) = 1 Then 'Enter the conditional check when DI1 equals 1
        Exit Do         'Exit Do loop
    End If
Loop

Integer var
For var = 1 To 5      'Execute a loop 5 times.
    Go P1
    Go P2
    If Sw(1) = 1 Then 'Enter the conditional check when DI1 equals 1
        Exit For
    End If
Next

Function main
    Exit Function
Fend
```

For..To..Next

[Function]

Execute the code between For..Next a certain number of times repeatedly.

[Syntax]

```
For Variable = Start Value To End Value {Step Increment Value}

    ...

Next
```

[Parameter Description]

Variable: The variable used for iteration in the For loop. This variable needs to be defined in advance.

Start Value: The initial value of the variable when the For loop starts.

End Value: The terminal value of the variable when the For loop ends.

Increment Value: The value by which the variable is incremented after each iteration. Defaults to 1 if not specified.

[Usage Example]

```
Integer var
For var = 1 To 5 Step 1   'Execute a loop 5 times.
    Go P1
    Go P2
    Print var
Next
```

Function..Fend

[Function]

Global function definition.

[Syntax]

Function Function name{(argument As data type,argument As data type, ...)}{As data type}

...

Fend

[Parameter Description]

Function name A Function name that must be defined.

Argument list Formal parameters that require passing numerical values or numerical operations within the Function. When there are multiple variables, separate them with commas.

ByRef When invoking a function with an array as a formal parameter, this instruction must be used in conjunction to modify the argument's index value.

ByVal When invoking a function with an array as a formal parameter, this instruction must be used in conjunction. The index value of the argument cannot be modified.

Argument As data type Required parameter. Please declare the type of the argument.

Function name As data type Generate the function name as a variable and store the return value. Please declare the variable type to which the function name belongs.

[Usage Example]

```
Function aa
    Print "123"
    Print "456"
    Print "789"
Fend

Call aa

Function Trig_CCD(IO_Idx As Integer, Integer_Time As Double)
    Do
        On IO_Idx
        Wait Integer_Time
        Off IO_Idx
    Loop
Fend
```

```
Function Add(Num1 As Integer, Num2 As Integer) As Integer
    Integer result
    result = Num1 + Num2
    Add = result
Fend


Function main
    Integer AAA(2)
    AAA(0) = 10
    AAA(1) = 20
    test(ByRef AAA())
    Print AAA(0),AAA(1)
Fend

Function Return_Array(ByRef BBB() As Integer)
    BBB(0) = BBB(0)*2
Fend
```

Global

[Function]

Global variable definition

[Syntax]

Global data type Variac

[Parameter Description]

Variable name A variable name that must be defined.

Precautions:

1. Global variables can only be defined outside functions, not inside functions.
2. The total number of global variables must not exceed 100,000. The number of String-type variables must not exceed 10,000.

[Usage Example]

```
Global String   PointList$(300,24)
Global Boolean  P_Mode
Global String   CCD_Dev_X$
Global Double   SafeHight
Global Integer  ABCD
Global Integer  AABB(10)
```

GoSub..Return

[Function]

GoSub to the label to transfer program control to the subroutine. The subroutine returns control to the original GoSub position via Return.

[Syntax]

GoSub [Label]

...

[Label]:

...

Return

[Parameter Description]

Label	A label name that must be defined.

[Usage Example]

```
Function main
    GoSub check in              ' Jump to the label
        On 1
        On 2
    Exit Function

Check in:                       'Label
     If  In (0) = 1 And  In (1) = 1 Then
        On  1
    Else
        Off 1
    End If
    Return                      'Return to the GoSub statement
Fend
```

GoTo

[Function]

Jump to the label to continue executing the program.

[Syntax]

GoTo [Label]

...

[Label]:

...

[Parameter Description]

Label	A label name that must be defined.

Precautions:

1. The difference from GoSub is that GoTo does not have a Return operation.

[Usage Example]

```
Function main
    If Sw(1) = Off Then
        GoTo mainAbort
    End If
    Print    "DI1=On"

    Exit Function

    mainAbort:
    Print    "DI1=Off"
Fend
```

If..Then..Else..EndIf

[Function]

Conditional statement that executes corresponding program code based on the result of a conditional expression.

[Syntax]

```
If Conditional expression Then
    ...
ElseIf Conditional expression Then
    ...
Else
    ...
EndIf
```

[Parameter Description]

Conditional expression	Execute the corresponding program code depending on whether the conditional expression evaluates to true or false.

[Usage Example]

```
Function main
    String AA$
    Integer num

    Input #201, AA$

    Num = val (AA$)              'Convert the received content to a numeric value and assign it to Num.
    If num > 1 Then
        Print    "num > 1"
    ElseIf num < 1 Then
        Print    "num < 1"
    Else
        Print    "num = 1"

    End If
Fend
```

Select..Case..Default..Send

[Function]

Selection statement that determines which section of the program to execute based on the value.

[Syntax]

Select Variable

Case Value 1

...

{Case Value 2

...}

Default

...

Send

[Parameter Description]

Variable	Variable name used for conditional judgment.
Value 1、Value 2	Check if the variable equals the specified value, and if true, execute the corresponding program block.
Default	When all Cases are not satisfied, execute the program block under Default.

[Usage Example]

```
Function main
    Integer I
    For I = 0 To 10
        Select I
            Case 0
                Print "I=1"
                Off 1; On 2; Jump P1
            Case 3
                Print "I=3"
                On 1; Off 2
                Jump P2;Move P3; On 3
            Case 7
                Print "I=3"
                On 4
            Default
                On 7
        Send
    Next
Fend
```

Detailed explanation of data types

Boolean

[Function]

Boolean Type, 2 Bytes.

[Syntax]

Boolean Variable{(Number of Array Elements)}

Global Boolean Variable{(Number of Array Elements)}

[Parameter Description]

Variable	Required Variable Name.
Range of values	True/False.

Precautions:

1. Maximum number of local variables: 2000
2. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    Boolean AA
    IfAA = True Then
        Print       "AA = True"
    ElseIf AA = False Then
        Print       "AA= Flase "
    EndIf
Fend
```

Byte

[Function]

Byte Type, 2 Bytes.

[Syntax]

Byte Variable{(Number of Array Elements)}

Global Byte Variable{(Number of Array Elements)}

[Parameter Description]

Variable	Required Variable Name.
Range of values	-128~127

Precautions:

3. Maximum number of local variables: 2000
4. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    Byte A1(10)
    Byte B1(10, 10)
    Byte CC(5,5,5)
    Byte Test_ok

    Test_ok = 15

    Test_ok =   (test_ok And 8)

    If    test_ok <> 8 Then
        Print "The high-order bit is ON."
    Else
        Print "The high-order bit is OFF."
    EndIf
Fend
```

Double

[Function]

Double-precision floating-point number, 8 bytes.

[Syntax]

Double Variable{(Number of Array Elements)}

Global Double Variable{(Number of Array Elements)}

[Parameter Description]

Variable	Required Variable Name.
Range of values	Precision up to 14 decimal places.

Precautions:

5. Maximum number of local variables: 2000
6. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    Double var1
    Double A1 (10)
    Double B1 (5,5)
    Double CC (5,5,5)
    Double arrayvar (10)
     Integer  i

    For   i = 1  To 5
        Print arrayvar (i)
    Next
Fend
```

Int32

[Function]

32 bit integer, 4 bytes.

[Syntax]

Int32 Variable{(Number of Array Elements)}

Global Int32 Variable{(Number of Array Elements)}

[Parameter Description]

Variable	Required Variable Name.
Range of values	-2147483648~2147483647

Precautions:

7. Maximum number of local variables: 2000
8. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    Int32 A1 (10)
    Int32 B1 (5,5)
    Int32 CC (5,5,5)
    Int32 var1,arrayvar (10)
Fend
```

Integer

[Function]

Integer type, 2 bytes.

[Syntax]

Integer Variable{(Number of Array Elements)}

Global Integer Variable{(Number of Array Elements)}

[Parameter Description]

Variable	Required Variable Name.
Range of values	-32768~32767

Precautions:

9. Maximum number of local variables: 2000
10. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    Integer A1 (10)
    Integer B1 (5, 5)
    Integer CC (5, 5, 5)
    Integer var1, arrayvar (10)
Fend
```

Long

[Function]

Long Integer type, 4 bytes.

[Syntax]

Long Variable{(Number of Array Elements)}

Global Long Variable{(Number of Array Elements)}

[Parameter Description]

Variable	Required Variable Name.
Range of values	-2147483648~2147483647

Precautions:

11. Maximum number of local variables: 2000

12. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    Long A1 (10)
    Long B1 (5, 5)
    Long CC (5, 5, 5)
    Long var1, arrayvar (10)
Fend
```

Real

[Function]

Single-precision floating-point number, 4 bytes.

[Syntax]

Real Variable{(Number of Array Elements)}

Global Real Variable{(Number of Array Elements)}

[Parameter Description]

Variable	Required Variable Name.
Range of values	Precision up to 6 decimal places.

Precautions:

13. Maximum number of local variables: 2000

14. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    Real A1 (10)
    Real B1 (5, 5)
    Real CC (5, 5, 5)
    Real var1, arrayvar (10)
Fend
```

Short

[Function]

Short Integer type, 2 bytes.

[Syntax]

Short Variable{(Number of Array Elements)}

Global Short Variable{(Number of Array Elements)}

[Parameter Description]

Variable	Required Variable Name.
Range of values	-32768~32767

Precautions:

15. Maximum number of local variables: 2000

16. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    Short A1 (10)
    Short B1 (5, 5)
    Short CC (5, 5, 5)
    Short var1, arrayvar (10)
Fend
```

String

[Function]

String type.

[Syntax]

String Variable$ {(Number of Array Elements)}

Global String Variable$ {(Number of Array Elements)}

[Parameter Description]

Variable$	Required Variable Name.
Range of values	A maximum of255 characters.

Precautions:

17. Maximum number of local variables: 2000

18. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    String A1$(10)
    String B1$(5,5)
    String CC$(1,2,3)
    String var1$,arrayvar$(10)
Fend
```

UByte

[Function]

Unsigned byte type, 2 bytes.

[Syntax]

UByte Variable{(Number of Array Elements)}

Global UByte Variable{(Number of Array Elements)}

[Parameter Description]

Variable	Required Variable Name.
Range of values	0~255

Precautions:

19. Maximum number of local variables: 2000

20. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    UByte A1 (10)
    UByte B1 (5,5)
    UByte CC (1,2,3)
    UByte var1,arrayvar (10)
Fend
```

UInt32

[Function]

Unsigned 32-bit integer, 4 bytes.

[Syntax]

UInt32 Variable{(Number of Array Elements)}

Global UInt32 Variable{(Number of Array Elements)}

[Parameter Description]

Variable	Required Variable Name.
Range of values	0~4294967295

Precautions:

21. Maximum number of local variables: 2000

22. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    UInt32 A1 (10)
    UInt32 B1 (5,5)
    UInt32 CC (1,2,3)
    UInt32 var1,arrayvar (10)
Fend
```

UShort

[Function]

Unsigned short integer, 2 bytes.

[Syntax]

UShort Variable{(Number of Array Elements)}

Global UShort Variable{(Number of Array Elements)}

[Parameter Description]

Variable	Required Variable Name.
Range of values	0~65535

Precautions:

23. Maximum number of local variables: 2000

24. Maximum number of global variables: 100000

[Usage Example]

```
Function main
    UShort A1 (10)                  'one-dimensional array of type UShort
    UShort B1 (5,5)                 'two-dimensional array of type UShort
    UShort CC (1,2,3)               'three-dimensional array of type UShort
    UShort var1,arrayvar (10)
Fend
```

Preserve

[Function]

Power-down retention for global variables.

[Syntax]

Global Preserve Integer Variable

[Parameter Description]

Variable	Required Variable Name.
Range of values	-2147483648~2147483647

Precautions:

25. Maximum number of local variables: 2000

26. Maximum number of global variables: 100000

[Usage Example]

```
Global Preserve Integer Hi
Global Preserve Integer OS
```

Detailed description of motion commands

!...!

[Function]

Process I/O inputs and outputs in parallel during motion.

[Syntax]

Motion Command Syntax !Parallel Statements!

[Parameter Description]

Motion Command Syntax Motion instructions such as Go, Move, Arc, Jump, etc.

Parallel Statements as follows.

1. D: Percentage processing statement.

2. I/O instructions such as On/Off, MemOff, Out, OutW, MemOut, Signal, Wait, Print, Print#, etc. For details, please refer to the detailed description of I/O instructions.

[Usage Example]

```
Function main
    Go P1 !D50; On 1!
     'During the PTP movement to point P1, set DO1 to ON when reaching 50% of the path.

    Jump P2 !D30; On 1; D70; Off 1!
     'During the gantry-type movement to point P2, set DO1 to ON when reaching 30% of the journey, and set DO1 to OFF when reaching 70%

    Move P3 !D10; On 5; Wait 0.5; Off 5!
     'During the linear movement to point P3, set DO5 to ON when reaching 10% of the journey, and then set DO5 to OFF after 0.5 seconds.
Fend
```

Motor

[Function]

Robot overall enable/disable, or return robot enable status.

[Syntax]

Motor {On/Off}

[Parameter Description]

On/Off On means overall enabling; Off means overall disabling. If omitted, it returns the robot's enable status.

[Usage Example]

```
Function main
    If Motor = Off Then
        Motor On
    EndIf
Fend
```

[Return Description]

Return Data 1 0 / 1, where 0 represents Off and 1 represents On.

ServoOn 、ServoOff

[Function]

Single-axis enable/disable, or return single-axis enable/disable status.

[Syntax]

Enable and disable:

ServoOn Joint number 1 {, joint number 2…}

ServoOff Joint number 1 {, joint number 2…}

Obtain the enable status:

ServoOn(Joint number)

ServoOff(Joint number)

[Parameter Description]

Joint number Specify the joints that need to be enabled or disabled using 1, 2, 3, 4, 5, 6.

Or specify the joints whose enable status needs to be returned.

[Usage Example]

```
Function main
    If ServoOff =  True Then
        ServoOn 1
    End If
Fend
```

[Return Description]

Return Data 1 TRUE / FALSE

Power

[Function]

Switch between high and low power modes, or return the current power mode.

[Syntax]

Power {High/Low}

[Parameter Description]

High/Low	High refers to high power mode, Low refers to low power mode. If omitted, the current power mode will be returned.

[Usage Example]

```
Function main
    If Power =  Low Then
        Power High
    End If
Fend
```

[Return Description]

Return Data 1	0 / 1, where 0 represents the low power mode and 1 represents the high power mode.

SpeedFactor

[Function]

Set the global motion speed, or return the current global speed.

[Syntax]

SpeedFactor {Percentage}

[Parameter Description]

Percentage Set the current global speed as a percentage between 1% and 100%.

If omitted, the current global speed will be returned.

[Usage Example]

```
Function main
    Motor On
    Power High

    Speed 100          'PTP Speed
    Accel 100,100
    SpeedS 2000        'Line Speed
    AccelS 100,100

    SpeedFactor 80     'Global Speed

    Print SpeedFactor
Fend
```

[Return Description]

Return Data 1 Percentage value: 1 - 100%, default data type is floating-point.

Speed

[Function]

Set the running speed of PTP motion, or return the current running speed of PTP motion.

[Syntax]

Speed {Percentage {, rotation speed percentage, approach speed percentage}}

[Parameter Description]

Percentage Set the current PTP motion speed within the range of 1% to 100%.

rotation speed percentage Set the transfer speed for the Jump command within the range of 1% to 100%, which can be omitted.

approach speed percentage Set the transfer speed for the Jump command within the range of 1% to 100%, which can be omitted.

Precautions:

If the PTP motion speed is not defined using this instruction in the program, the default speed of 20% will be used.

[Usage Example]

```
Function main
    Motor On
    Power High

    Speed 100          'PTP Speed
    Accel 100,100
    SpeedS 2000        'Line Speed
    AccelS 100,100

    SpeedFactor 80     'Global Speed

    Print SpeedFactor
Fend
```

[Return Description]

Return Data 1 Percentage value: 1 - 100%, default data type is floating-point.

Accel

[Function]

Set the running acceleration and deceleration of PTP motion, or return the current running acceleration and deceleration of PTP motion.

[Syntax]

Accel {Acceleration percentage, deceleration percentage, {rotational speed acceleration percentage, rotational speed deceleration percentage, approach acceleration percentage, approach deceleration percentage}}.

[Parameter Description]

Acceleration percentage	Set the current PTP motion deceleration as a percentage within the range of 1% to 100%.
deceleration percentage	Set the current PTP motion deceleration within the range of 1% to 100%.
rotational speed acceleration percentage	Set the transfer acceleration for the Jump command within the range of 1% to 100%, which can be omitted.
rotational speed deceleration percentage	Set the transfer deceleration for the Jump command within the range of 1% to 100%, which can be omitted.
approach acceleration percentage	Set the approach acceleration for the Jump command within the range of 1% to 100%, which can be omitted.
approach deceleration percentage	Set the approach deceleration for the Jump command within the range of 1% to 100%, which can be omitted.

[Usage Example]

```
Function main
    Accel 100,100,50,50,50,50          'Set the PTP motion plus or minus speed
    Print Accel(1)                     'Set the acceleration of the PTP motion
    Print Accel(2)                     'Set the PTP motion deceleration
    Print Accel(3)                     'Set the acceleration of the JUMP motion transfer action
    Print Accel(4)                     'Set the JUMP motion transfer action to decelerate
    Print Accel(5)                     'Set the JUMP motion close to the action acceleration
    Print Accel(6)                     'Set the JUMP motion close to the action deceleration
Fend
```

[Return Description]

Return Data 1	Percentage value: 1 - 100%, default data type is floating-point.

Go

[Function]

Used to execute PTP motion from the current position to the specified position.

[Syntax]

Go Target coordinate {CP}{Till/Find}{!...!}

[Parameter Description]

Target coordinate Specify the target position using point data.

CP Set motion smoothness, which can be omitted.

Till/Find Till expression = On/Off. Find expression = On/Off. Optional. Refer to the detailed description of Till/Find for specifics.

!...! Process I/O and other input/output operations in parallel during motion. Optional. Refer to the detailed description of parallel processing for specifics.

[Usage Example]

```
Function main
    Go P1
    Go P2 CP

    Go P1+X (20)
     ' Move to point P1 with a relative offset of 20 mm in the X direction.

    Go P2:X (200)
    'Move to point P2 with the X coordinate changed to 200 mm.

    Go P3 Till Sw (1) = On And Sw (5) = Off
     ' Stop the motion when DI1 is On and DI5 is Off during the movement to point P3.

    Go P4 Till Sw (1) = On !D50; On 5!
     ' Move to point P4; stop the motion when DI1 is On, and set DO5 to On when 50% of the path is reached.

    Go P5 /R            'Move to point P5 in the right-hand coordinate system.

    Go XY (X coordinate , Y coordinate , Z coordinate ,U coordinate)        'Move to the specified coordinate.
Fend
```

BGo

[Function]

Used to execute offset PTP motion in the selected user coordinate system.

[Syntax]

BGo Target coordinate {CP}{Till/Find}{!...!}

[Parameter Description]

Target coordinate	Specify the target position using point data.
CP	Set motion smoothness, which can be omitted.
Till/Find	Till expression = On/Off. Find expression = On/Off. Optional. Refer to the detailed description of Till/Find for specifics.
!...!	Process I/O and other input/output operations in parallel during motion. Optional. Refer to the detailed description of parallel processing for specifics.

[Usage Example]

```
Function main
    BGo XY (100, 0, 0, 0)              'Move 100mm in the X direction on the selected user coordinate system.

    P1 = XY (300, 300, -20, 0)
    P2 = XY (300, 300, -20, 0) /L
    Local 1, XY (0, 0, 0, 45)

    Go P1
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /R /0
    BGo XY (0, 50, 0, 0)
    Print Here
    'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /R /0
    Go P2
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0
    BGo XY (0, 50, 0, 0)
    Print Here
    'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0
    BGo XY (0, 50, 0, 0) /L
    Print Here
    'X:264.645 Y:385.355 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0
Fend
```

TGo

[Function]

Used to execute offset PTP motion in the current tool coordinate system.

[Syntax]

TGo Target coordinate {CP}{Till/Find}{!...!}

[Parameter Description]

Target coordinate	Specify the target position using point data.
CP	Set motion smoothness, which can be omitted.
Till/Find	Till expression = On/Off. Find expression = On/Off. Optional. Refer to the detailed description of Till/Find for specifics.
!...!	Process I/O and other input/output operations in parallel during motion. Optional. Refer to the detailed description of parallel processing for specifics.

[Usage Example]

```
Function main
    TGo XY (100, 0, 0, 0)                'Move 100mm in the X direction on the selected tool coordinate system.

    Tool 0
    P1 = XY (300, 300, -20, 0)
    P2 = XY (300, 300, -20, 0) /L

    Go P1
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /R /0

    TGo XY (0, 0, -30, 0)
    Print Here
    'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000 /R /0

    Go P2
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0

    TGo XY (0, 0, 30, 0)
    Print Here
    'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000 /L /0
Fend
```

Speeds

[Function]

Set the running speed of the linear interpolation motion, or return the current running speed of the linear interpolation motion.

[Syntax]

SpeedS {Speed{, Rotational Speed, Approach Speed}}

[Parameter Description]

Speed	Set the current linear interpolation motion speed within the range of 0.1~2000 mm/s.
Rotational Speed	Set the transfer speed of the Jump3 command within the range of 0.1~2000 mm/s.
Approach Speed	Set the transfer speed for the Jump3 command within the range of 0.1~2000 mm/s.

[Usage Example]

```
Function main
    Motor On
    Power High

    Speed 100          'PTP Speed
    Accel 100,100
    SpeedS 2000        'Line Speed
    AccelS 100,100

    SpeedFactor 80     'Global Speed

    Print SpeedFactor
Fend
```

[Return Description]

Return Data 1	Percentage value: 1 - 100%, default data type is floating-point.

AccelS

[Function]

Set the acceleration and deceleration of the linear interpolation motion, or return the current acceleration and deceleration values.

[Syntax]

AccelS {Acceleration percentage, deceleration percentage, {rotational speed acceleration percentage, rotational speed deceleration percentage, approach acceleration percentage, approach deceleration percentage}}.

[Parameter Description]

Acceleration percentage	Set the current acceleration of linear interpolation motion as a percentage in the range of 1% to 100%.
deceleration percentage	Set the current deceleration of linear interpolation motion as a percentage in the range of 1% to 100%.
rotational speed acceleration percentage	Set the transfer acceleration for the Jump3 command within the range of 1% to 100%, which can be omitted.
rotational speed deceleration percentage	Set the transfer deceleration for the Jump3 command within the range of 1% to 100%, which can be omitted.
approach acceleration percentage	Set the approach acceleration for the Jump3 command within the range of 1% to 100%, which can be omitted.
approach deceleration percentage	Set the approach deceleration for the Jump3 command within the range of 1% to 100%, which can be omitted.

[Usage Example]

```
Function main
    Motor On
    Power High

    Speed 100          'PTP Speed
    Accel 100,100
    SpeedS 2000        'Line Speed
    AccelS 100,100

    SpeedFactor 80     'Global Speed

    Print SpeedFactor
Fend
```

Move

[Function]

Used for linear interpolation motion from the current position to a specified position.

[Syntax]

Move Target coordinate {CP}{Till/Find}{!...!}

[Parameter Description]

Target coordinate	Specify the target position using point data.
CP	Set motion smoothness, which can be omitted.
Till/Find	Till expression = On/Off. Find expression = On/Off. Optional. Refer to the detailed description of Till/Find for specifics.
!...!	Process I/O and other input/output operations in parallel during motion. Optional. Refer to the detailed description of parallel processing for specifics.

[Usage Example]

```
Function main
    Move P1          'Move linearly to point P1.
    Move P2 CP       'Move linearly to point P2 with a smooth transition at P2.

    Move P3 Till Sw(1) = On And Sw(5) = Off
    'Stop the motion when DI1 is On and DI5 is Off during the movement to point P3.

    Move P4 Till Sw(1) = On !D50; On 5!
    'During the movement to point P4, stop the motion when DI1 is On, and set DO5 to On when 50% of the path is reached.
Fend
```

BMove

[Function]

Used to perform offset linear interpolation motion in the selected user coordinate system.

[Syntax]

BMove Target coordinate {CP}{Till/Find}{!...!}

[Parameter Description]

Target coordinate Specify the target position using point data.

CP Set motion smoothness, which can be omitted.

Till/Find Till expression = On/Off. Find expression = On/Off. Optional. Refer to the detailed description of Till/Find for specifics.

!...! Process I/O and other input/output operations in parallel during motion. Optional. Refer to the detailed description of parallel processing for specifics.

[Usage Example]

```
Function main
BMove XY (100 , 0 , 0 , 0)          'Used to perform offset linear interpolation motion in the selected user coordinate system.
    P1 = XY (300 , 300 , -20 , 0)
    P2 = XY (300 , 300 , -20 , 0) /L
    Local 1 , XY (0 , 0 , 0 , 45)

    Go P1
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000

    BMove XY (0 , 50 , 0 , 0)
    Print Here
    'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000

    Go P2
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000

    BMove XY (0 , 50 , 0 , 0)
    Print Here
    'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000

Fend
```

TMove

[Function]

Used to perform offset linear motion in the current tool coordinate system.

[Syntax]

TMove Target coordinate {CP}{Till/Find}{!...!}

[Parameter Description]

Target coordinate Specify the target position using point data.

CP Set motion smoothness, which can be omitted.

Till/Find Till expression = On/Off. Find expression = On/Off. Optional. Refer to the detailed description of Till/Find for specifics.

!...! Process I/O and other input/output operations in parallel during motion. Optional. Refer to the detailed description of parallel processing for specifics.

[Usage Example]

```
Function main
    TMove XY(100, 0, 0, 0) 'Move 100mm in the X direction on the selected tool coordinate system.
    Tool 0                 'Select Tool 0.
    P1 = XY(300, 300, -20, 0)
    P2 = XY(300, 300, -20, 0) /L

    Go P1
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000

    TMove XY(0, 0, -30, 0)
    Print Here
    'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000

    Go P2
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000

    TMove XY(0, 0, -30, 0)
    Print Here
    'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000
Fend
```

Jump

[Function]

Used for gantry-type PTP motion from the current position to a specified position: a gantry movement that first rises, then translates, then descends.

[Syntax]

Jump Target coordinate {C Arch Number}{LimZ Coordinate Value}{CP}{Till/Find/Sense}{!...!}

[Parameter Description]

Target coordinate	Specify the target position using point data.
C Arch Number	Select different Arches using C0~C7. For details, refer to the detailed description of Arch.
LimZ Coordinate Value	Set the maximum movable value (limit value) of the 3rd joint during the Jump motion. For details, refer to the detailed description of LimZ; can be omitted
CP	Set motion smoothness, which can be omitted.
Till/Find/Sense	Till expression = On/Off; Find expression = On/Off; Sense expression = On/Off; can be omitted. For details, refer to the detailed description of Till/Find/Sense.
!...!	Process I/O and other input/output operations in parallel during motion. Optional. Refer to the detailed description of parallel processing for specifics.

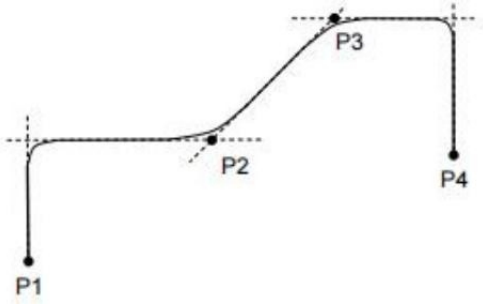
[Usage Example]

```
Function main
    Go P1
    Jump P2:Z(-50) C0 LimZ -50 CP
    Go P3:Z(0) CP
    Jump P4 C0 LimZ 0

    Jump P1
    Jump P2 CP
    Jump P3 Till Sw(1) = On And Sw(5) = Off
    'Stop the motion when DI1 is ON and DI5 is OFF during the movement to point P3

    Jump P4 Till Sw(1) = On !D50; On 5!
    'During the movement to point P4, stop the motion when DI1 is On, and set DO5 to On when 50% of the path is reached.

    Jump P5 /R C1 LimZ-10 Sense Sw(2) = On
    'Select Arch1 as the approach parameter; during the right-hand coordinate system movement to point P5, the Z-axis can only move a maximum of -10mm.
    'Stop above the target point when DI2 is On, and set DO5 to On when 50% of the path is reached.
Fend
```



Arc、Arc3

[Function]

Arc is used to move the robotic arm from the current position to a specified position with circular interpolation motion on the XY plane.

Arc3 is used to move the robotic arm from the current position to a specified position with circular interpolation motion in three-dimensional space.

[Syntax]

Arc Pass-through Coordinate , Target Coordinate {CP}{Till/Find}{!...!}

Arc3 Pass-through Coordinate , Target Coordinate {CP}{Till/Find}{!...!}

[Parameter Description]

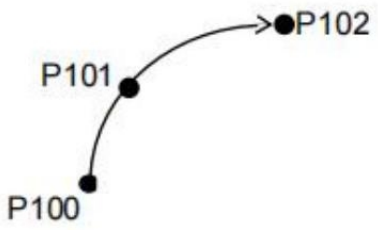
Pass-through Coordinate	Specify the position that the designated arc will pass through using point data.
Target coordinate	Specify the target position using point data.
CP	Set motion smoothness, which can be omitted.
Till/Find	Till expression = On/Off. Find expression = On/Off. Optional. Refer to the detailed description of Till/Find for specifics.
!...!	Process I/O and other input/output operations in parallel during motion. Optional. Refer to the detailed description of parallel processing for specifics.

[Usage Example]

```
Function main

    Go P100
    Arc P101 , P102

Fend
```



CP

[Function]

Enable motion smoothness.

[Syntax]

CP {On/Off}

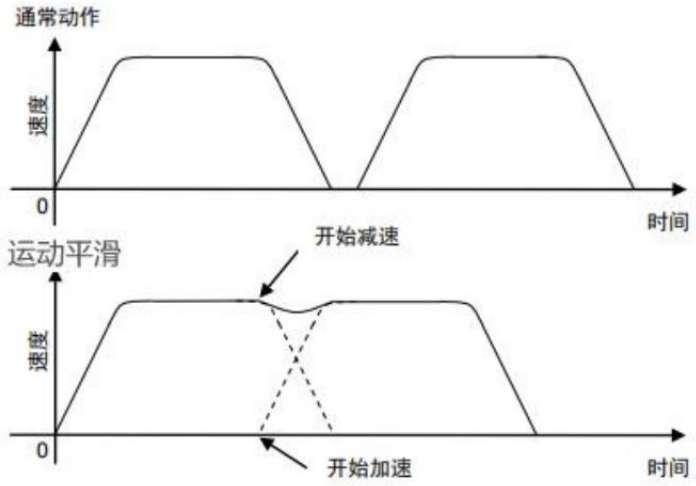
Motion command Target coordinate CP

[Parameter Description]

Motion command On enables motion smoothing, and Off disables it.

Target coordinate Motion instructions such as Go, Move, Arc, Jump, etc.

[Usage Example]




```
Function main
    Go P1 CP
    Move P2 CP

    CP On              'After "CP On", the subsequent motion commands will have smooth transition by default.
    Move P3
    Move P4
    CP Off

    If CP = Off Then
        CP On
    End If
Fend
```

CPZone

[Function]

Set motion smoothness.

[Syntax]

CPZone {On/Off} { PTP motion smoothing ratio, linear interpolation smoothing ratio }

[Parameter Description]

On/Off	On enables the motion smoothing ratio, and Off disables the smoothing ratio. When Off, the system default ratio is adopted.
PTP motion smoothing ratio	Set the fusion degree of PTP motion smoothing in the range of 1% to 200%.
linear interpolation smoothing ratio	Set the fusion degree of linear interpolation motion smoothing in the range of 1~200 mm.

[Usage Example]

```
Function main

    CPZone On , 50 , 50

Fend
```

R180

[Function]

Used to constrain the positive and negative angles ofthe U-axis to ±180 degrees.

[Syntax]

R180 {On/Off}

Go **Target coordinate** R180

Print R180

[Parameter Description]

On/Off	On enables the constraint, and Off disables the constraint.

[Usage Example]

```
Function main

    If R180 = Off Then
        R180 On
    End If

    Go P1 R180
    Go P2 R180
    Go P3

    R180 Off

    Print R180

Fend
```

Here

[Function]

Used to teach the robot's coordinates to a point location or obtain the robot's current coordinates.

[Syntax]

Here {Point number / Point name}

[Parameter Description]

Point number / Point name	Teach the current coordinates to the point location.

[Usage Example]

```
Function main

    Here P1             'Teach the current coordinates to point P1.
    Go Here + X(10)     'Offset 10mm in the X direction from the current position.

Fend
```

Home

[Function]

Move the robot to the user-defined Home position.

[Syntax]

Home

[Parameter Description]

[Usage Example]

```
Function main

    Home        'Go Home

Fend
```

Local

[Function]

Used to define and display the user coordinate system.

[Syntax]

One-point teaching: Local user coordinate number, point position

Two-point teaching: Local user coordinate number, point 1, point 2, {X/Y}

Three-point teaching: Local user coordinate number, origin, X-axis point, Y-axis point, {X/Y}

[Parameter Description]

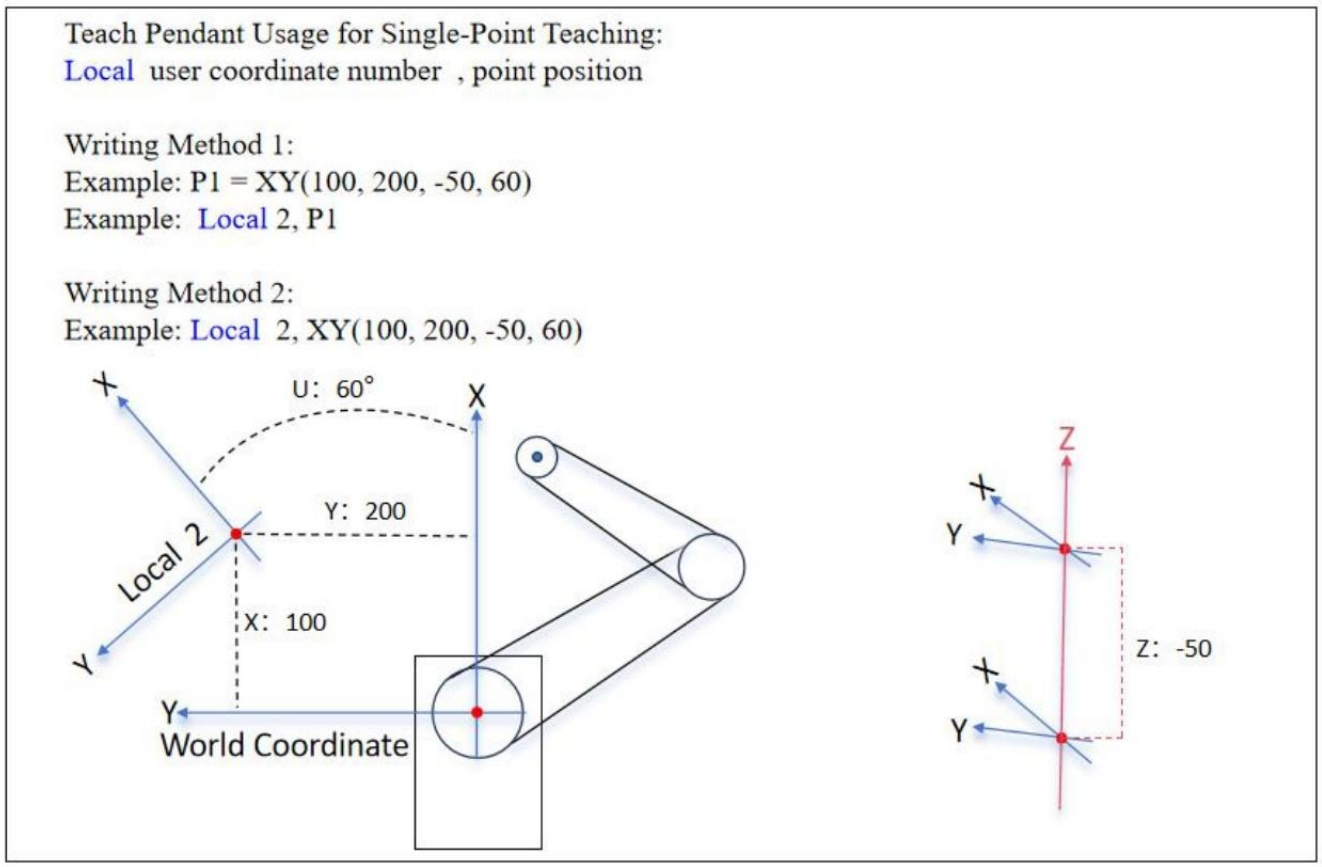
User coordinate number	Specify the user coordinate system to be taught using a number from 1 to 64.
Origin, Point 1...	Specify the point data of the user coordinate system using point variables.
L/R	Align the origin of the user coordinate system to the left (No. 1) or right (No. 2); this parameter can be omitted.
X/Y	Define the straight line connecting the origin and the specified X-axis or Y-axis point as the X-axis or Y-axis of the local coordinate system; this parameter can be omitted, and the default is X.

[Usage Example]

Teach Pendant Usage for Single-Point Teaching:
Local user coordinate number , point position

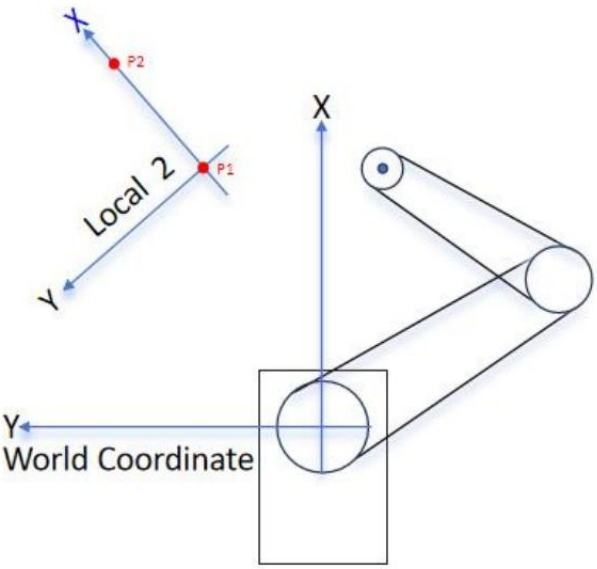
Writing Method 1:
Example: P1 = XY(100, 200, -50, 60)
Example: Local 2, P1

Writing Method 2:
Example: Local 2, XY(100, 200, -50, 60)

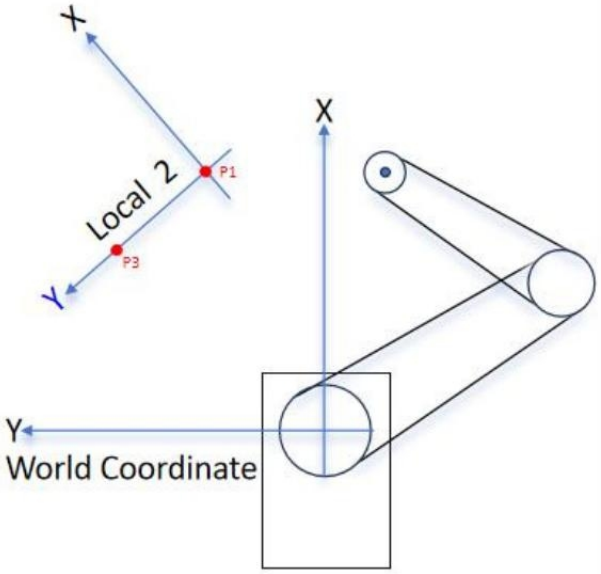


Teach Pendant Usage for Two-Point Teaching:
Local user coordinate number, point 1, point 2, {X/Y}

Writing Method 1:
Example: Local 2, P1, P2, X

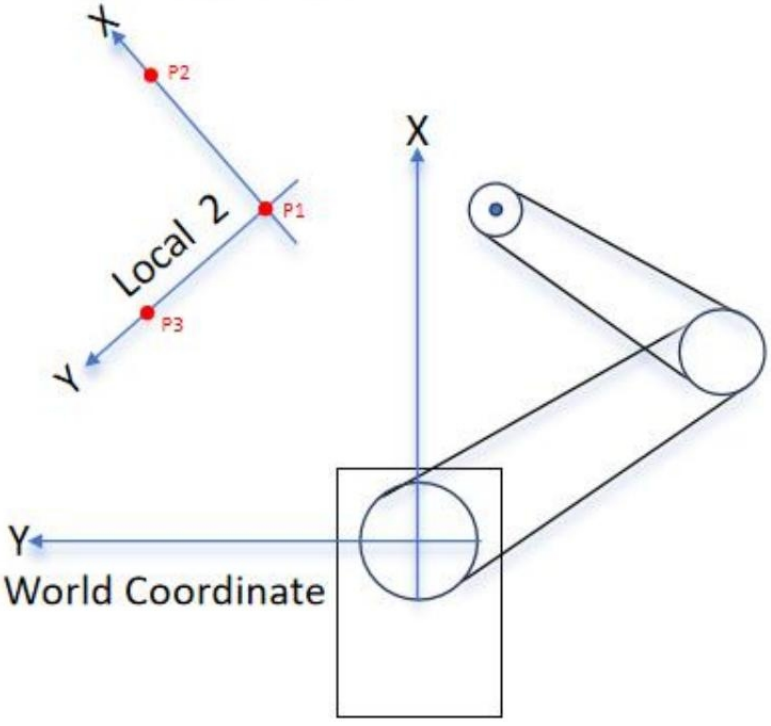


Writing Method 1:
Example: Local 2, P1, P3, Y



Teach Pendant Usage for Three-Point Teaching:
Local user coordinate number, origin, X-axis point, Y-axis point

Example: Local 2, P1, P2, P3



TLSet

[Function]

Used for defining and displaying the tool coordinate system.

[Syntax]

TLSet {Tool coordinate number, { Tool setting data }}

[Parameter Description]

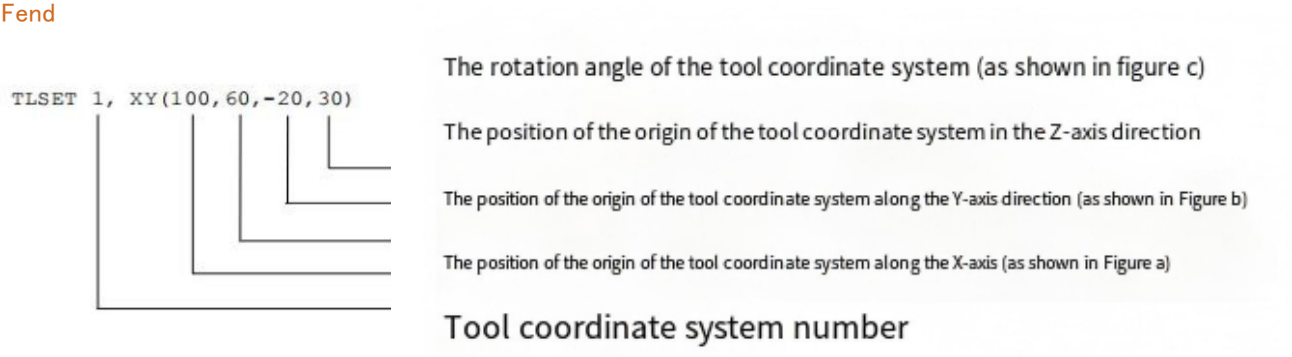
Tool coordinate number Tool coordinate systems to be taught are specified using numbers from 1 to 64.

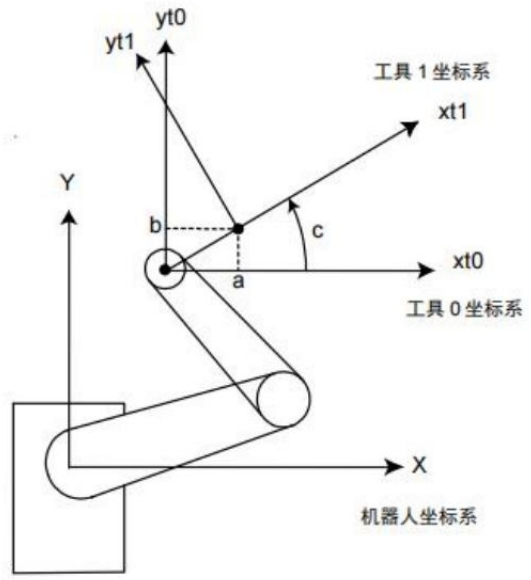
Tool setting data Set the origin and orientation of the tool coordinate system using point numbers, point names, and point variables.

Precautions: If all parameters are omitted, all TLSet settings will be displayed.

[Usage Example]

```
Function main
    TLSet 1,XY(50 , 100 , -20 , 30)
    TLSet 2 , P10 +X(20)
Fend
```





Tool

[Function]

Used to select a tool coordinate system or return the currently selected tool coordinate system.

[Syntax]

Tool { Tool coordinate number }

[Parameter Description]

Tool coordinate number Specify the selected tool coordinate system using numbers from 1 to 64.

[Usage Example]

```
Function main

    Tool 1                                  'Select tool coordinate system 1.

    If Tool = 0 Then
        Tool 2
    End If

Fend
```

TCalib

[Function]

Define the tool coordinate system using the three-point method

[Syntax]

TCalib Tool coordinate number, Tool auxiliary point 1, Tool auxiliary point 2, Tool auxiliary point 3

[Parameter Description]

Tool coordinate number	Specify the selected tool coordinate system using numbers from 1 to 64.
Tool auxiliary point 1	Align the center of the fixture end mounted on the robot's lead screw with the teaching marker.
Tool auxiliary point 2	On the basis of Tool Auxiliary Point 1, rotate the U-axis clockwise or counterclockwise by 60° to 120°, and use the jog function to align the center of the fixture end with the teaching marker again.
Tool auxiliary point 3	On the basis of Tool Auxiliary Point 2, rotate the U-axis again in the same clockwise or counterclockwise direction by 60° to 120°, and use the jog function to align the center of the fixture end with the teaching marker once more.

[Usage Example]

```
Function main

    TCalib 1 , P0 , P1 , P2                    'Teach tool coordinate 1 using P0, P1, and P2 as auxiliary points.

Fend
```

Till

[Function]

Used to set the conditions for stopping movements during motion commands such as Jump, Go, and Move.

[Syntax]

Till { Conditional expression }

[Parameter Description]

Conditional expression	Conditions for stopping movement during motion. Various I/O commands, operators, etc., can be used.

[Usage Example]

```
Function main
    Till Sw(1) = Off        'Set the Till condition (DI1 is OFF).
        Go P1 Till          'Stop when the condition in the previous line is met.

    Till Sw(1) = On And Sw(1) = On   'Set a new Till condition.
        Move P2 Till                 ' Stop when the condition from the previous line is satisfied.

        Move P5 Till Sw(10) = On     ' Stop when the condition in this line is met.
Fend
```

Agl

[Function]

Function used to return the angle or position of a specified joint.

[Syntax]

Agl(Joint number)

[Parameter Description]

Joint number	Specify the joint number as an integer. The range is 1-4 for SCARA robots, 1-6 for 6-axis robots. If there are extension axes, the numbering increases accordingly.

[Usage Example]

```
Function main
    Print PAgl(1), Agl(2)
Fend
```

PAgl

[Function]

Function to return the joint angles of a specified point.

[Syntax]

PAgl(Point number, Joint number)

[Parameter Description]

Point number Use the sequential number in the point file as the index.

Joint number Specify the joint number as an integer. For SCARA robots, the range is 1-4; for 6-axis robots, it is 1-6. If there are extended axes, the numbering increases accordingly based on the extended axes.

[Usage Example]

```
Function main
    Print PAgl(P1,1)
    Print PAgl(P1,2)
    Print PAgl(P1,3)
    Print PAgl(P1,4)
Fend
```

AglToPls

[Function]

Function to convert each robot joint angle into pulse values.

[Syntax]

AglToPls(Joint Position 1, Joint Position 2, Joint Position 3, Joint Position 4, {Joint Position 5, Joint Position 6}….)

[Parameter Description]

Joint position	Specify the joint angles of each joint in sequence. The number of parameters can be increased as appropriate according to the machine model and extended axes.
Instruction Coordination	When used as a Go parameter, it executes absolute joint motion; when used as a Print parameter, it outputs pulse values; in all other usages, it returns spatial points.

[Usage Example]

```
Function main

    P1 = AglToPls (0 , 0 , 0 , 90 , 0 , 0)
    Go P1

    Move AglToPls (0 , 0 , 0 , 90 , 0 , 0)

    Go AglToPls (0 , 0 , 0 , 90 , 0 , 0)

Fend
```

Arch

[Function]

Used to set/display the Arch parameters for the Jump, Jump3, and Jump3CP commands.

[Syntax]

Arch {Arch number{, transfer distance, approach distance}}
Arch(Arch number, parameter number)

[Parameter Description]

Arch Number Specifies the Arch number as an integer from 0 to 6. If not specified, the system uses default values (see table below for details).

Transfer Distance Vertical distance measured from the starting point, in millimeters (mm).

Approach Distance Vertical distance measured from the target point, in millimeters (mm).

Parameter Number 1: Transfer Distance.2: Approach Distance.

Arch 表格默认值

Arch 编号	转移距离	接近距离
0	30	30
1	40	40
2	50	50
3	60	60
4	70	70
5	80	80
6	90	90

[Usage Example]

```
Function main

    Arch 0 , 20 , 20                    'Set the two distance parameters for Arch 0.

    Jump P1 C0                          'Enable Arch0 in Jump command.

    If Arch(0,1) = 30 And Arch(0,2) = 30 Then
        'Check if the two distance parameters of Arch0 meet the criteria.
        Arch 0 , 25 , 25
    End If

Fend
```

Find

[Function]

Used to set the saving of coordinates when the conditional expression is satisfied during the movement.

[Syntax]

Find conditional expression

[Parameter Description]

conditional expression The condition for saving coordinates during movement. Various I/O commands, operators, etc., can be used.

[Usage Example]

```
Function main

    Go P10 Find Sw(5) = On          'Save the coordinates when DI5 is set to On during movement.

    If PosFound Then
        Go FindPos                  'Retrieve the coordinates saved by Find using FindPos.
    Else
        Print "Retrieve the coordinates saved by Find using FindPos."
    End If

Fend
```

FindPos

[Function]

Used to return the coordinates saved by Find.

[Syntax]

```
FindPos
Go FindPos
```

[Parameter Description]

[Usage Example]

```
Function main

    Go P10 Find Sw(5) = On          'Save the coordinates when DI5 is set to On during movement.

    If PosFound Then
        Go FindPos                  'Retrieve the coordinates saved by Find using FindPos.
    Else
        Print "Retrieve the coordinates saved by Find using FindPos."
    EndIf

Fend
```

InPos

[Function]

Return the robot's positioning status.

[Syntax]

InPos

[Parameter Description]

Precautions:

1.Returns True when positioning is complete.

2.Returns False while the robot is in motion.

[Usage Example]

```
Function main

    P0 = XY (0 , -100 , 0 , 0)
    P1 = XY (0 , 100 , 0 , 0)
    Xqt MonitorPosition
    Do
        Jump P0
        Wait 0.5
        Jump P1
        Wait 0.5
    Loop
Fend

Function MonitorPosition
    Boolean oldInPos , Pos

    Do
        Pos = InPos
        If Pos <> oldInPos Then
            Print "InPos=" , Pos
        End If

        oldInPos = Pos

    Loop
Fend
```

JA

[Function]

Used to return the robot coordinates based on joint angles.

[Syntax]

JA(Joint angle 1, joint angle 2, joint angle 3, joint angle 4, {joint angle 5, joint angle 6}….)

[Parameter Description]

Joint angle Specify the joint angles of each joint in sequence. Unit: Degrees (deg) for rotational joints, Millimeters (mm) for linear joints. The number of parameters can be adjusted according to the robot model and additional axes.

[Usage Example]

```
Function main
    P10 = JA (60 , 30 , -50 , 45)
    Go JA (135 , 90 , -50 ,90)
    P3 = JA (0 , 0 , 0 , 0 , 0 , 0)
Fend
```

Joint

[Function]

Used to display the current robot position in the joint coordinate system.

[Syntax]

Joint

[Parameter Description]

[Usage Example]

```
Function main
    Joint
    'Print information: Joint: 1:-11.880deg 2:30.432deg 3:-21.902mm 4:-18.552deg
Fend
```

LimZ

[Function]

Used to set/display the initial value of the 3rd joint height (Z coordinate) during the Jump movement.

[Syntax]

LimZ {Z coordinate value}

[Parameter Description]

Z coordinate value	Specify the coordinate value of the 3rd joint within the motion range.

[Usage Example]

```
Function main
    LimZ -10                'Set the default value of Limz
    Jump P1                 ''When executing JUMP, keep the Z-axis height at -10 during horizontal movement.
    Jump P2 LimZ -20        ' When executing JUMP, keep the Z-axis height at -20 during horizontal movement.
    Jump P3                 ' When executing JUMP, keep the Z-axis height at -10 during horizontal movement.
    If LimZ > -10 Then      'When the set value is greater than -10
        LimZ -30            'Set the default value of Limz to -30.
    End If
Fend
```

Pass

[Function]

Used for PTP movement that passes near the specified point (without stopping).

[Syntax]

Pass Target coordinate 1{, target coordinate 2, target coordinate 3….}

[Parameter Description]

Target coordinate	Set the position to move to using point numbers or point names. Additional point parameters can be added as needed.

[Usage Example]

```
Function main
    Pass P1 , P2 , P3       ''Continuous smooth transition movement from P1 → P2 → P3
Fend
```

Pls

[Function]

Used to return the pulse value of the specified joint at the current position.

[Syntax]

Pls(Joint number)

[Parameter Description]

Joint number Specify the joint for which the pulse value needs to be obtained using an integer from 1 to 6 (the number ofjoints).

[Usage Example]

```
Function main
    Real Axle_1 , Axle_2 , Axle_3 , Axle_4
    Axle_1 = pls(1)
    Axle_2 = pls(2)
    Axle_3 = pls(3)
    Axle_4 = pls(4)
    Print "Current position, axis 1 pulse:", Axle_1
    Print "Current position, axis 2 pulse:", Axle_2
    Print "Current position, axis 3 pulse:", Axle_3
    Print "Current position, axis 4 pulse:", Axle_4
Fend
```

PPls

[Function]

Used to specify the joint pulse values of a coordinate point.

[Syntax]

PPls(Point number,Joint number)

[Parameter Description]

Point number	Use the sequence number in the point file as the index.
Joint number	Specify the joint for which the pulse value needs to be obtained using an integer from 1 to 6 (the number ofjoints).

[Usage Example]

```
Function main
    Print "P1 point, axis 1 pulse position:", PPls(P1, 1)
    Print "P1 point, axis 1 pulse position:", PPls(P1, 2)
    Print "P1 point, axis 1 pulse position:", PPls(P1, 3)
    Print "P1 point, axis 1 pulse position:", PPls(P1, 4)
Fend
```

JS

[Function]

Returns whether the Sense condition is met during Jump, Jump3, or Jump3CP movements.

[Syntax]

JS

[Parameter Description]

Precautions:

1. Returns True when the Sense condition is met and the movement stops above the target point.
2. Returns False when the Jump action is completed normally and the target point is reached.

[Usage Example]

```
Function main
    Print searchSensor
Fend

Function searchSensor As Boolean
    Sense Sw(5) = On
    Jump P0
    Jump P1 Sense

    If JS = True Then
        Print "Sense condition is met"
        searchSensor = True
    End If
Fend
```

PosFound

[Function]

Used to return the execution status of the Find command.

[Syntax]

PosFound

[Parameter Description]

Precautions:

1. Returns True if Find is satisfied during the movement; otherwise, returns False.

[Usage Example]

```
Function main

    Go P10 Find Sw(5) = On 'Save the coordinates when DI5 is set to On during movement.

    If PosFound Then
        Go FindPos              'Obtain the coordinates saved by Find via FindPos.
    Else
        Print "DI5 is not set to ON during movement."
    End If
Fend
```

Pulse

[Function]

Move the robotic arm to the position specified by each joint's pulse value via PTP, or return the point corresponding to the pulse value.

[Syntax]

Pulse{Joint 1 pulse value, Joint 2 pulse value, Joint 3 pulse value, Joint 4 pulse value{, Joint 5 pulse value, ...}}

Pulse{Joint 1 pulse value, Joint 2 pulse value, Joint 3 pulse value, Joint 4 pulse value{, Joint 5 pulse value, ...}}

[Parameter Description]

Joint pulse value Fill in the pulse values of each joint in sequence, and the numberof formal parameters can be increased as appropriate according to the actual number of axes of the robot.

[Usage Example]

```
Function main
    Pulse 123456 , 222333 , 333444 , 111111
    'Move to the specified pulse position

    P1 = Pulse (123456 , 222333 , 333444 , 111111)
     ' Save the point with the specified pulses to P1.

    Pulse 'Print the current pulse values of all axes.
Fend
```

RobLoad

[Function]

Used to select specified load parameters or return the specified load parameters.

[Syntax]

RobLoad {Load number}

[Parameter Description]

Load number	Specify the load to be selected with an integer from 1 to 64.

[Usage Example]

```
Function main
    RobLoad 3
    Go P1

    If RobLoad = 0 Then
        RobLoad 1
    End If
Fend
```

Sense

[Function]

Used to set/display the conditions for Jump, Jump3, and Jump3CP to stop above the target point.

[Syntax]

Sense {Conditional expression}

[Parameter Description]

Conditional expression The condition to stop above the target point during movement. Various I/O commands, operators, etc., can be used.

[Usage Example]

```
Function main
    Sense Sw(5) = Off
    Jump P1 C2 Sense
    'Stop above the target point when DI5 is Off.

    If JS = True Then
        Print "The robot has stopped above the target point."
    End If
    On 1; Wait 0.2; Off 1
Fend
```

TillOn

[Function]

Used to return the status of Till.

[Syntax]

TillOn

[Parameter Description]

Precautions:

1. If the Till condition is met during movement, return True; otherwise, return False.

[Usage Example]

```
Function main
    Go P1 Till Sw(1) = On

    If TillOn Then
        Print "Movement has stopped because DI1 is set to On."
    End If
Fend
```

WaitPos

[Function]

Used to wait for the robot to decelerate and stop after the motion smoothing command before transferring the program to the subsequent program.

[Syntax]

WaitPos

[Parameter Description]

Precautions:

1. Under normal circumstances, when motion smoothing is active (CP On or when the motion command includes CP), the action command will transfer control to the subsequent program in advance while starting to decelerate. If you do not want to transfer in advance, you can insert WaitPos afterward, and the transfer will only occur after the deceleration action is completed.

[Usage Example]

```
Function main
    Off 1
    CP On
    Move P1
    Move P2
    WaitPos          "Wait for the robot to decelerate
    On 1
    CP Off
Fend
```

XY

[Function]

Return the point position based on the specified data.

[Syntax]

XY(X coordinate value, Y coordinate value, Z coordinate value, U coordinate value{, V coordinate value, W coordinate value})

[Parameter Description]

coordinate value	Fill in the coordinates of each axis in sequence. The coordinate values of the V-axis and W-axis can be added as appropriate according to the actual robot model. The unit is mm.

[Usage Example]

```
Function main
    P10 = XY (60 , 30 , -50 , 45) + P20
    'Generate point P10 by adding the specified offset coordinates to the data of index point P20

    P10 = XY (60 , 30 , -50 , 45) /L
    'Generate the P10 point based on the specified coordinates and hand system.

    Go XY (10 , 20 , 30 , 40)
    'Arc move to the specified coordinate position.

Fend
```

CX、CY、CZ、CU、CV、CW、CR、CS、CT

[Function]

Used to set (modify) the coordinate value of a point.

[Syntax]

CX(Point Number) {= coordinate value}

CY(Point Number) {= coordinate value}

CZ(Point Number) {= coordinate value}

....

[Parameter Description]

Point Number point number、point name.

coordinate value The set coordinate value, in millimeters (mm).

[Usage Example]

```
Function main
    Double New_Y

    CX (P1) = 123                   'Set the X coordinate of P1 to 123 mm.
    CU (P1) = 321                   'Set the U coordinate of P1 to 321 degrees (deg).

    If CY (P1) > 200 Then
        CY (P1) = 200               'If the Y coordinate of P1 is greater than 200, force it to be equal to 200.
    End If

    P1 = Here                       'Acquire the current position and assign it to point P1.
    Print CX (P1)                   'Print the X coordinate of the current position.
    New_Y = CY (P1)                 'Assign the current Y coordinate to the variable New_Y.
Fend
```

TargetOK

[Function]

Used to detect whether the coordinates of a specified point are reachable.

[Syntax]

TargetOK(Point Number)

....

[Parameter Description]

Point Number point number、point name.

[Usage Example]

```
Function main
    If TargetOK(P1) = False Then
        Print "This point position is unreachable."
    ElseIf TargetOK(P1) = True Then
        Go P1
    End If
Fend
```

ColSup

[Function]

Used to set (modify) the collision detection coefficient.

[Syntax]

ColSup On/Off [,level]

ColSup

....

[Parameter Description]

On/Off Whether the collision detection coefficient needs to be set.

level The value of the collision detection coefficient to be set, with a range of 1 - 300.

[Usage Example]

```
Function main
    ColSup On, 50          'Set the collision detection coefficient.
    ColSup Off             'Turn off the collision detection setting.
    Print ColSup           "Print the current collision detection coefficient
Fend
```

ZeroPls

[Function]

Used to set (modify) the robot's zero position.

[Syntax]

ZeroPls Pulse 1, Pulse 2, Pulse 3 [, Pulse 4, Pulse 5, Pulse 6]

ZeroPls

....

[Parameter Description]

Pulse 1, Pulse 2, Pulse 3 [, Pulse 4, Pulse 5, Pulse 6] The pulse values for setting the zero position required by each member axis.

[Usage Example]

```
Function main

    ZeroPls 0 , 0  , 0  , 0
    'Set the zero position ofthe X, Y, Z, and U axes to the position where the pulse value is 0.

    Print ZeroPls
    'Print the current zero position.

Fend
```

Hand

[Function]

Used to set (modify) the hand posture.

[Syntax]

Hand [Point Number , hand posture]
Hand

....

[Parameter Description]

Point Number point number、point name.

hand posture Modify the point's hand configuration , Lefty: Left-hand configuration

Righty: Right-hand configuration

[Usage Example]

```
Function main

    Print Hand (P0)             'Print the hand configuration of point P0.

    Hand P0, Righty             'Set point P0 to the right-hand configuration.

    Print Hand                  'Print the current hand configuration.

Fend
```

Elbow

[Function]

Used to set (modify) the elbow posture.

[Syntax]

Elbow [Point Number , elbow posture]
Elbow
....

[Parameter Description]

Point Number point number、point name.

elbow posture The elbow posture of the point: Above indicates the elbow is up, and Below indicates the elbow is down.

[Usage Example]

```
Function main
    Print Elbow (P0)            'Print the elbow posture of point P0.
    Elbow P0 , Above            ' Set the posture of point P0 with the elbow up.
    Print Elbow                 ' Print the elbow posture.
Fend
```

Wrist

[Function]

Used to set (modify) the wrist posture.

[Syntax]

Wrist [Point Number , wrist posture]
Wrist
....

[Parameter Description]

Point Number point number、point name.

wrist posture] The wrist posture of the point: Flip indicates the wrist is flipped, and NoFlip indicates the wrist is not flipped.

[Usage Example]

```
Function main
    Print Wrist(P0)                    'Print the wrist posture of point P0.
    Wrist P0 , NoFlip                  'Set the posture of point P0 with the elbow in the upward position.
    Print Wrist                        'Print the wrist posture.
Fend
```

Detailed Description of Independent Axis Commands

IndPower

[Function]

for setting the enable status of the independent axis.

[Syntax]

IndPower Independent Axis ID , On / Off

[Parameter Description]

Independent Axis ID	Specify the independent axis number (Range: 1 - 16) .
On / Off	Specify the independent axis enable status (On: Upper Enable / Off: Lower Enable).

[Usage Example]

```
Function main

    IndPower 1 , On          'Upper Enable for Independent Axis 1
   Wait 1
    IndPower 1 , Off         ' Lower Enable for Independent Axis 1

Fend
```

IndHalt

[Function]

for controlling the independent axis to stop moving.

[Syntax]

IndHalt Independent Axis ID , Accel/Decel

[Parameter Description]

Independent Axis ID	Specify the independent axis number (Range: 1 - 16) .
Accel/Decel	Specify the acceleration and deceleration of the independent axis; enter 0 to stop.

[Usage Example]

```
Function main

    IndPower 1 , On                 'Upper Enable for Independent Axis 1
    IndMovAbs 1 , 1000, 30, 100     ' Specify Independent Axis 1 to move to 1000 (mm/deg/Pulse)
   Wait 1
    IndHalt 1 , 0                   'Independent Axis 1 stops moving.

Fend
```

IndMovDone

[Function]

for waiting for the independent axis to finish moving.

[Syntax]

IndMovDone Independent Axis ID

[Parameter Description]

Independent Axis ID	Specify the independent axis number (Range: 1 - 16) .

Precautions:

1.Commands related to the independent axis will not wait for the command execution to complete; that is, after the command takes effect, it proceeds to the next step.

[Usage Example]

```
Function main

    IndPower 1 , On                 'Upper Enable for Independent Axis 1
    IndMovAbs 1 , 1000, 30, 100     ' Specify Independent Axis 1 to move to 1000 (mm/deg/Pulse)
    IndMovDone 1                    'Wait for Independent Axis 1 to finish moving.

Fend
```

IndMovAbs

[Function]

for controlling the independent axis to move to the specified absolute position.

[Syntax]

IndMovAbs Independent Axis ID , absolute position , Speed , Accel/Decel [, speed lock]

[Parameter Description]

Independent Axis ID Specify the independent axis number (Range: 1 - 16) .

absolute position Specify the absolute position of the independent axis, unit (Unit).

Speed Specify the maximum movement speed of the independent axis (Unit / S).

Accel/Decel Specify the acceleration and deceleration time of the independent axis (1- 1000 ms).

speed lock Specify that the movement speed of the independent axis is not affected by the global speed percentage.

Precautions:

1.Commands related to the independent axis will not wait for the command execution to complete; that is, after the command takes effect, it proceeds to the next step.

2.Calculation method of Unit: 1 Unit = N pulses = 1 mm/degree.

[Usage Example]

```
Function main

    IndPower 1 , On                 'Upper Enable for Independent Axis 1
    IndMovAbs 1 , 1000, 30, 100     ' Specify Independent Axis 1 to move to 1000 (mm/deg/Pulse)
    IndMovDone 1                    'Wait for Independent Axis 1 to finish moving.

Fend
```

IndMovRel

[Function]

for controlling the independent axis to move to the relative position.

[Syntax]

IndMovRel Independent Axis ID , relative position , Speed , Accel/Decel [, speed lock]

[Parameter Description]

Independent Axis ID	Specify the independent axis number (Range: 1 - 16) .
relative position	Specify the relative position of the independent axis, unit (Unit).
Speed	Specify the maximum movement speed of the independent axis (Unit / S).
Accel/Decel	Specify the acceleration and deceleration time of the independent axis (1- 1000ms).
speed lock	Specify that the movement speed of the independent axis is not affected by the global speed percentage.

Precautions:

1.Commands related to the independent axis will not wait for the command execution to complete; that is, after the command takes effect, it proceeds to the next step.

2.Calculation method of Unit: 1 Unit = N pulses = 1 mm/degree.

[Usage Example]

```
Function main

    IndPower 1 , On                 'Upper Enable for Independent Axis 1
    IndMovRel 1 , 1000, 30, 100     'Specify Independent Axis 1 to offset 1000 (mm/deg/Pulse) relative to its
                                    current position
    IndMovDone 1                    'Wait for Independent Axis 1 to finish moving.

Fend
```

IndMovVel

[Function]

for controlling the independent axis to move at a constant speed.

[Syntax]

IndMovRel Independent Axis ID , Speed , Accel/Decel [, speed lock]

[Parameter Description]

Independent Axis ID	Specify the independent axis number (Range: 1 - 16) .
Speed	Specify the maximum movement speed of the independent axis (Unit / S).
Accel/Decel	Specify the acceleration and deceleration time of the independent axis (1- 1000 ms).
speed lock	Specify that the movement speed of the independent axis is not affected by the global speed percentage.

Precautions:

1.Commands related to the independent axis will not wait for the command execution to complete; that is, after the command takes effect, it proceeds to the next step.

[Usage Example]

```
Function main

    IndPower 1 , On                 'Upper Enable for Independent Axis 1
    IndMovVel 1 , 30, 100           'Specify the constant speed of Independent Axis 1.
    IndMovDone 1                    'Wait for Independent Axis 1 to finish moving.

Fend
```

IndReadPos

[Function]

for reading the current position of the independent axis.

[Syntax]

IndReadPos（Independent Axis ID）

[Parameter Description]

Independent Axis ID	Specify the independent axis number (Range: 1 - 16) .

[Usage Example]

```
Function main

    Double IndPos
    IndPos = IndReadPos(1)

Fend
```

[Return Description]

Return Data 1	Current position of the independent axis, data type: floating-point number.

IndReadVel

[Function]

for reading the current movement speed of the independent axis.

[Syntax]

IndPeadVel（Independent Axis ID）

[Parameter Description]

Independent Axis ID	Specify the independent axis number (Range: 1 - 16) .

[Usage Example]

```
Function main

    Double IndVel

    IndVel = IndReadVel(1)

Fend
```

[Return Description]

Return Data 1	Current speed of the independent axis, data type: floating-point number.

IndReadStatus

[Function]

for reading the current state of the independent axis.

[Syntax]

IndReadStatus（Independent Axis ID）

[Parameter Description]

Independent Axis ID	Specify the independent axis number (Range: 1 - 16) .

[Usage Example]

```
Function main

    Integer  IndStatus
    IndStatus =  IndReadStatus (1)

Fend
```

[Return Description]

Return Data 1	Current state of the independent axis, data type: integer.

[状态说明]

Return Data	State
0	Not Enabled
1	Error Stop
2	Stopping
3	Ready
4	Discrete Motion(IndMovAbs 、 IndMovRel)
5	Continuous Motion(IndMovVel)
6	Reserved
7	GO Home

IndReset

[Function]

for clearing the error of the independent axis.

[Syntax]

IndPower　Independent Axis ID

[Parameter Description]

Independent Axis ID	Specify the independent axis number (Range: 1 - 16) .

[Usage Example]

```
Function main

    IndPower 1 , On                 'Upper Enable for Independent Axis 1
    IndMovAbs 1 , 1000, 30, 100     ' Specify Independent Axis 1 to move to 1000 (mm/deg/Pulse)
    IndMovDone 1                    'Wait for Independent Axis 1 to finish moving.

    IndReset 1                      'Reset Independent Axis 1

Fend
```

Detailed description of remark instructions such as point location, I/O, encoder, etc.

PDef

[Function]

Used to return whether a point has been defined.

[Syntax]

PDef（Point Number）

PDef（P （Point Number））

PDef（Label name）

[Parameter Description]

Point Number Index by the sequence number in the point file.

P Point Number Index by the sequence number in the point file.

Label name After assigning a tag variable, index by the sequence number of the variable value in the point file.

[Usage Example]

```
Function main

    Print "0:" , PDef (0)                    'Print whether point P0 is defined.
    Print "P1:" , PDef (P1)                   'Print whether point P1 is defined.

    Print "P(1):" , PDef (P (1))              'Print whether point P1 is defined.

    Integer Point_Num
    Point_Num =  0
    Print "Point_Num:" , PDef (Point_Num)  'Print whether the corresponding point is defined.

Fend
```

[Return Description]

Return Data 1: TRUE / FALSE. TRUE indicates that the point is defined, and FALSE indicates that the point is not defined.

PDel

[Function]

Used to delete points.

[Syntax]

PDel　Point Number

PDel　Start point number, end point number

[Parameter Description]

Point Number	Index by the sequence number in the point file.
Start point number	Delete points starting from the specified number.
end point number	Delete points up to the specified number.

[Usage Example]

```
Function main

    PDel 1                    'Delete the position information of point P1.

    PDel 1 , 5                'Delete the position information of points P1 to P5.

    SavePoints "robot1"       'Save to the point table.

Fend
```

PLabel

[Function]

Used to set the label for the specified point data.

[Syntax]

PLabel　Point number, new label

[Parameter Description]

Point Number　Index by the sequence number in the point file.

new label　Assign a new label to the specified point number (Chinese is not supported temporarily).

[Usage Example]

```
Function main

    PLabel 1 , "New_Point"        'Assign a new label to point P1.

    Go P(New_Point)

    SavePoints "robot1"           'Save to the point table.

Fend
```

PLabel$

[Function]

Used to return the label of the specified point data.

[Syntax]

PLabel$ (Point number)

PLabel$ (P Point number)

PLabel$ (P(Point number,))

[Parameter Description]

Point Number Index by the sequence number in the point file.

P Point Number Index by the sequence number in the point file.

P(Point Number) Index by the sequence number in the point file.

Precautions:

1.If the instruction is used to index an empty point or a point without a label, no output will be generated.

[Usage Example]

```
Function main

    Print PLabel$ (1)                    'Print the label of point 1.
    Print PLabel$ (P1)                   'Print the label of point 1.
    Print PLabel$ (P (1))                'Print the label of point 1.
    If PLabel$ (2) = "" Then             'Determine whether the label of point 2 is empty.
        Print "This point has no label."
    End If

Fend
```

PNumber

[Function]

Used to return the point number corresponding to the specified label name.

[Syntax]

PNumber (Label name)

PNumber (“Label name”)

[Parameter Description]

Label name	One of the items in the point data is the label.

[Usage Example]

```
Function main

    Print PNumber(EA)              'Print the point number(s) with the label "EA".
    Print PNumber("AA")            'Print the point number(s) with the label "AA".

Fend
```

PList

[Function]

Used to display the specified point data.

[Syntax]

PList

PList　Point Number

PList　Start point number　,　end point number

[Parameter Description]

Point Number	Index by the sequence number in the point file.
Start point number	Delete points starting from the specified number.
end point number	Delete points up to the specified number.

[Usage Example]

```
Function main

    PList                   'Print all point data.

    PList 1                 'Print the data ofpoint 1.

    PList 2 , 10            'Print the data from point 2 to point 10.

Fend
```

PLocal

[Function]

Used to specify/acquire the user - coordinate information of the point.

[Syntax]

PLocal (Point Number)
PLocal (Point Number) = Value

[Parameter Description]

Point Number Index by the sequence number in the point file.

Value The specified user coordinate system number value

[Usage Example]

```
Function main

    Print PLocal (1)          'Print the user coordinates to which point 1 belongs.

    PLocal (1) = 1            'Set the user coordinates to which point 1 belongs.

Fend
```

PDescription

[Function]

Used to set the description of the specified point.

[Syntax]

PDescription　Point Number ， New description

[Parameter Description]

Point Number	Index by the sequence number in the point file.
New description	The descriptive data displayed in the point file.

[Usage Example]

```
Function main

    PDescription 1 , "New description"          'Change the description of point P1 to "New description".

    Print PDescription$ (1)                     'Print the description of point P1.

Fend
```

PDescription$

[Function]

Used to return the description of the specified point.

[Syntax]

PDescription$ （Point Number）

PDescription$ （P（Point Number））

PDescription$ （Label name）

[Parameter Description]

Point Number Index by the sequence number in the point file.

P(Point Number) Index by the sequence number in the point file.

Label name After assigning a label variable, index based on the sequence number of the variable value in the point file.

[Usage Example]

```
Function main

    PDescription 1 , "New description"          'Change the description of point P1 to "New description".

    Print PDescription$ (1)                     ' Print the description of point P1.

    Print PDescription$ (P1)                    ' Print the description of point P1.

    Print PDescription$ (P (1))                 ' Print the description of point P1.

Fend
```

GetPointFiles

[Function]

Used to obtain the list of point files.

[Syntax]

GetPointFiles String array $()

[Parameter Description]

String array	Store the names of the point tables in the variable array.

[Usage Example]

```
Function main
    Integer i
    String names$(0)

    GetPointFiles names$()      'Store the names of the point tables in the variable array.
    For i = 0 To UBound(names$())
        Print names$(i)
    Next
Fend
```

DIDescription

[Function]

Description of setting digital input data.

[Syntax]

DIDescription Number , description

[Parameter Description]

Number Specify the DI number whose description needs to be modified with a value ranging from 0 to 1023.

Description Provide the description to be written in string format.

[Usage Example]

```
Function main

    DIDescript ion 0 , "New description"        'Modify the description of DI-0 to "New description".

Fend
```

DIDescription$

[Function]

Description of returning digital input data

[Syntax]

DIDescription$ (Number)

[Parameter Description]

Number Specify the DI number whose description needs to be returned with a value ranging from 0 to 1023.

[Usage Example]

```
Function main

    String AA$
    AA$ = DIDescription$ (1)
    Print AA$                    'Print the description of DI-1

    Print DIDescription$ (1)     'Print the description of DI-1

Fend
```

DODescription

[Function]

Description of setting digital output data.

[Syntax]

DODescription Number , description

[Parameter Description]

Number Specify the DO number whose description needs to be modified with a value ranging from 0 to 1023.

Description Provide the description to be written in string format.

[Usage Example]

```
Function main

    DODescription 0 , "New description"          'Modify the description of DO-0 to "New description".

Fend
```

DODescription$

[Function]

Description of returning digital output data

[Syntax]

DODescription$ (Number)

[Parameter Description]

Number Specify the DO number whose description needs to be returned with a value ranging from 0 to 1023.

[Usage Example]

```
Function main

    String AA$
    AA$ = DODescription$ (1)
    Print AA$                    'Print the description of DO-1

    Print DODescription$ (1)     'Print the description of DO-1

Fend
```

AIDescription

[Function]

Description of setting analog input data.

[Syntax]

AIDescription Number , description

[Parameter Description]

Number Specify the AI number whose description needs to be modified with a value ranging from 0 to 255.

Description Provide the description to be written in string format.

[Usage Example]

```
Function main

    AIDescription 0 , "New description"          'Modify the description of AI-0 to "New description".

Fend
```

AIDescription$

[Function]

Description of returning analog input data

[Syntax]

AIDescription$ (Number)

[Parameter Description]

Number Specify the AI number whose description needs to be returned with a value ranging from 0 to 255.

[Usage Example]

```
Function main

    String AA$
    AA$ = AIDescription$ (1)
    Print AA$                    'Print the description of AI-1

    Print AIDescription$ (1)     'Print the description of AI-1

Fend
```

AODescription

[Function]

Description of setting analog output data.

[Syntax]

AODescription Number , description

[Parameter Description]

Number Specify the AO number whose description needs to be modified with a value ranging from 0 to 255.

Description Provide the description to be written in string format.

[Usage Example]

```
Function main

    AODescription 0 , "New description"          'Modify the description of AO-0 to "New description".

Fend
```

AODescription$

[Function]

Description of returning analog output data

[Syntax]

AODescription$(Number)

[Parameter Description]

Number Specify the AO number whose description needs to be returned with a value ranging from 0 to 255.

[Usage Example]

```
Function main

    String AA$
    AA$ = AODescription$ (1)
    Print AA$                    'Print the description of AO-1

    Print AODescription$ (1)     'Print the description of AO-1

Fend
```

EncDescription

[Function]

Description of setting encoder data.

[Syntax]

EncDescription Number , description

[Parameter Description]

Number Specify the Encoder number whose description needs to be modified with a value ranging from 0 to 7.

Description Provide the description to be written in string format.

[Usage Example]

```
Function main

    EncDescription 0 , "New description"        'Modify the description of Enc-0 to "New description".

Fend
```

EncDescription$

[Function]

Description of returning encoder data

[Syntax]

EncDescription$ (Number)

[Parameter Description]

Number Specify the Encoder number whose description needs to be returned with a value ranging from 0 to 7.

[Usage Example]

```
Function main

    String AA$
    AA$ = EncDescription$ (1)
    Print AA$                   'Print the description of Enc-1

    Print EncDescription$ (1)   'Print the description of Enc-1

Fend
```

Detailed Description ofI/O Instructions

Wait

[Function]

Delay instruction, or wait for a condition to be satisfied within a specified time.

[Syntax]

Wait Time

Wait Conditional expression

Wait Conditional expression, Time

[Parameter Description]

Time	Specify the delay time with a number ranging from 0 to 2147483. Unit: second. Minimum delay: 0.01 seconds.
Conditional expression	Terminate the Wait when the specified condition is met.

[Usage Example]

```
Function main

    ' Wait for input 0 to turn to the On state.
    Wait Sw(0) = On

    'Continue execution after waiting for 60.5 seconds.
    Wait 60.5

    'Wait for input 0 to turn OFF and input 1 to turn ON.
    Wait Sw(0) = Off And Sw(1) = On

    'Wait for register bit 0 to turn ON or register bit 1 to turn ON
    Wait MemSw(0) = On Or MemSw(1) = On

    'Wait for 1 second, then set output 1 to ON.
    Wait 1; On 1

Fend
```

Mask

[Function]

Used to perform a bit wise AND operation on the value representing the Wait input condition, on a bit-by-bit basis.

[Syntax]

Wait Conditional expression Mask Maximum value of the bit = Condition return value

[Parameter Description]

Time Specify the delay time with a number ranging from 0 to 2147483.
Unit: second. Minimum delay: 0.01 seconds.

Conditional expression Specified DIO bit or group, register address or bit.

Mask Used to perform a bitwise AND operation on the value representing the Wait input condition.

Maximum value of the bit Maximum value of the indexed bit.

Condition return value The target value that satisfies the current conditional expression.

[Usage Example]

```
Function main
    Do
        Wait InW(0) Mask 1 = 0
        'The exit condition is met when the return value of the 16-bit DI group 0 (first 1 bit) is 0.

        Wait InW(0) Mask 3 = 2
        'The exit condition is met when the return value of the 16-bit DI group 0 (first 2 bits) is 2.

        Wait InW(0) Mask 7 = 4
        'The exit condition is satisfied when the return value of the 16-bit DI group 0 (first 3 bits) is 4.

        Wait InW(0) Mask 15 = 8
        'The exit condition is satisfied when the return value of the 16-bit DI group 0 (first 4 bits) is 8.

        Wait InW(0) Mask 31 = 16
        'The exit condition is satisfied when the return value of the 16-bit DI group 0 (first 5 bits) is 16.

        Wait InW(0) Mask 63 = 32
        The exit condition is satisfied when the return value of the 16-bit DI group 0 (first 6 bits) is 32.
    Loop
Fend
```

On

[Function]

Control the DO to turn ON, or turn it OFF after it has been ON for a certain period of time.

[Syntax]

On DO number / DO label {, time}

[Parameter Description]

DO number / DO label	Specify the DO number as an integer, or use a DO label for specification.
Time	Specify the duration for which the DO is turned ON in seconds. It will be automatically turned OFF once this duration is reached.

[Usage Example]

```
Function main

    On 1                'DO1 to be ON.
    On 3                'DO3 to be ON.

    On 5 , 1            'DO5 is turned ON, and then turned OFF after 1 second.

Fend
```

Off

[Function]

Control the DO to turn OFF, or turn it ON after it has been OFF for a certain period of time.

[Syntax]

Off DO number / DO label {, time}

[Parameter Description]

DO number / DO label Specify the DO number as an integer, or use a DO label for specification.

Time Specify the duration for which the DO is turned OFF in seconds. It will be automatically turned ON once this duration is reached.

Synchronous processing After specifying the time of Off, the method of synchronous processing can be set to specify the execution sequence of subsequent instructions, which can be omitted

0 Set DO to Off while executing the following command, and set the maximum time to 10 seconds.

1 Default Settings: Set DO to Off, wait a specified amount of time and then set to On before allowing subsequent instructions to execute.

[Usage Example]

```
Function main

    Off 1               'DO1 to be Off.
    Off 3               'DO3 to be Off.

    Off 5 , 1           'DO5 is turned OFF, and then turned ON after 1 second.

Fend
```

Oport

[Function]

Used to return the status of the specified DO.

[Syntax]

Oport(DO number)

[Parameter Description]

DO number	Specify the DO number as an integer, or use a DO label for specification.

Precautions:

1.Returns 1 when the DO is ON.

2.Returns 0 when the DO is OFF.

[Usage Example]

```
Function main

    If Oport (1) = 0 Then
            On 1
    End If

    Wait Oport (3)
Fend
```

Sw

[Function]

Used to return the status of the specified DI.

[Syntax]

Sw(DI Number)

[Parameter Description]

DI Number	Specify the DI whose status needs to be obtained as an integer.

Precautions:

1.Returns 1 when the DI is ON.

2.Returns 0 when the DI is OFF.

[Usage Example]

```
Function main

    If Sw(1) = 0 Then
        SetSw 1 , On
    End If

    Wait Sw(1)
Fend
```

SetSw

[Function]

Used to turn the specified DI ON or OFF.

[Syntax]

SetSw DI Number, status

[Parameter Description]

DI Number	Specify the DI whose status needs to be obtained as an integer.
status	0 means OFF, and 1 means ON. Entering OFF or ON will also take effect.

[Usage Example]

```
Function main

    SetSw 1 , On
    SetSw 4 , Off


Fend
```

In

[Function]

Return the status of the 8-bit DI as a value ranging from 0 to 255 based on the 8421 code.

[Syntax]

In(DI group number)

[Parameter Description]

DI group number Specify the group of DIs whose status needs to be read using an integer.

Precautions:

1. When the DI group number is set to 0, it returns the status of DI0 to DI7. When set to 1, it returns the status of DI8 to DI15, and so on.
2. The schematic diagram of the 8421 principle is as follows:

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[Usage Example]

```
Function main

    Print In(0)

    Integer A1
    A1 =  In(1)

    Select A1
    Case 0
        Go P1

    Case 8
        Go P2

    Default
        Go P3

    Send
Fend
```

SetIn

[Function]

Set the status of the 8-bit DI as a value ranging from 0 to 255 based on the 8421 code.

[Syntax]

SetIn DI group number, status value

[Parameter Description]

DI group number	Specify the group of DIs whose status needs to be read using an integer.
status value	Set the status of this group of DIs according to the 8421 code.

Precautions:

1. When the DI group number is set to 0, it returns the status of DI0 to DI7. When set to 1, it returns the status of DI8 to DI15, and so on.

2. The schematic diagram of the 8421 principle is as follows:

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[Usage Example]

```
Function main
    Set In 0 ,85                'Turn DI0, DI2, DI4, and DI6 ON.
Fend
```

Out

[Function]

Set/return the status of the 8-bit DO using the 8421 code, with values ranging from 0 to 255.

[Syntax]

Out DO group number, status value

Out(DO group number)

[Parameter Description]

DO group number	Specify the group of DO whose status needs to be read using an integer.
status value	Set the status of this group of DO according to the 8421 code.

Precautions:

1. DO group number 0 refers to DO0~DO7; number 1 refers to DO8~DO15, and so on.

2. The schematic diagram of the 8421 principle is as follows:

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[Usage Example]

```
Function main
    Out 0 ,85                    'Turn DI0, DI2, DI4, and DI6 ON.

    If Out (0) = 0 Then          'Read the status of DO0~DO7.
        Out 0 ,85
    End If
Fend
```

InW

[Function]

Return the status of the 16-bit DI as 0~65535 based on 8421.

[Syntax]

InW(DI group number)

[Parameter Description]

DI group number Specify the group of DIs whose status needs to be read using an integer.

Precautions:

1.DI group number 0 corresponds to DI0~DI15; number 1 corresponds to DI16~DI31, and so on.

2.Schematic diagram of the 8421 principle is as follows:

返回值	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	On	On	On	On
255	Off	Off	Off	Off	Off	Off	Off	Off	On	On	On	On	On	On	On	On

[Usage Example]

```
Function main
    Print  InW (0)

     Integer A1
    A1 =   InW (1)

    Se lect A1
    Case 0
        Go P1

    Case 8
        Go P2

    Defau lt
        Go P3
    Send
Fend
```

SetInW

[Function]

Set the 16-bit DI status via 8421, 0~65535.

[Syntax]

SetInW DI group number, status value

[Parameter Description]

DI group number Specify the group of DIs whose status needs to be read using an integer.

status value Set the status of this group of DIs according to the 8421 code.

Precautions:

1.DI group 0 refers to DI0~DI15; group 1 refers to DI16~DI31, and so on.

2.Schematic diagram of the 8421 principle is as follows:

返回值	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	On	On	On	On
255	Off	Off	Off	Off	Off	Off	Off	Off	On	On	On	On	On	On	On	On

[Usage Example]

```
Function main
    SetInW 0 ,85                ' Turn DI0, DI2, DI4, and DI6 ON.
Fend
```

OutW

[Function]

Set/return 16-bit DO status via 8421, 0~65535.

[Syntax]

OutW DO group number, status value
OutW(DO group number)

[Parameter Description]

DO group number	Specify the group of DO whose status needs to be read using an integer.
status value	Set the status of this group of DO according to the 8421 code.

Precautions:

1.DO group 0 refers to DO0~DO15; group 1 refers to DO16~DO31, and so on.

2.Schematic diagram of the 8421 principle is as follows:

返回值	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	Off	On	On	On	On
255	Off	Off	Off	Off	Off	Off	Off	Off	On	On	On	On	On	On	On	On

[Usage Example]

```
Function main
    OutW 0 ,85                    'Turn DI0, DI2, DI4, and DI6 ON.

    If OutW (0) = 0 Then          'Read the status of DO0~DO15.
        OutW 0 ,85
    End If
Fend
```

InBCD

[Function]

Return the status of 8-bit DI according to 8421, ranging from 0 to 99.

[Syntax]

InBCD(DI group number)

[Parameter Description]

DI group number Specify the group of DIs whose status needs to be read using an integer.

Precautions:

1. When the DI group number is set to 0, the status of DI0~DI7 is returned. When it is set to 1, the status of DI8~DI15 is returned, and so on.
2. The difference between this function and In lies in the value range. The value range of In is 0~255.
3. The schematic diagram of the 8421 principle is as follows:

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[Usage Example]

```
Function main
    Print  InBCD (0)

     Integer A1
    A1 =   InBCD (1)

    Se lect A1
    Case 0
        Go P1

    Case 8
        Go P2

    Defau lt
        Go P3

    Send
Fend
```

OpBCD

[Function]

Set the status of 8-bit DO according to 8421, ranging from 0 to 99.

[Syntax]

OpBCD DO group number, status value

[Parameter Description]

DO group number Specify the group of DOs whose status needs to be read using an integer.

status value Set the status of this group of DOs according to the 8421 code.

Precautions:

1. When the DO group number is set to 0, it refers to DO0~DO7. When set to 1, it refers to DO8~DO15, and so on.

2. The difference between this function and Out is in the value range. The value range of Out is 0~255.

3. The schematic diagram of the 8421 principle is as follows:

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[Usage Example]

```
Function main
    OpBCD 0 ,85                    ' Turn on DO0, DO2, DO4, and DO6.
Fend
```

InReal

[Function]

Return the DI status as a 32-bit floating-point number.

[Syntax]

InReal(DI group number)

[Parameter Description]

DI group number	Specify the group of DIs whose status needs to be read using an integer.

Precautions:

1. When the DI group number is set to 0, it refers to DI0~DI31. When set to 1, it refers to DI32~DI63, and so on.

[Usage Example]

```
Function main
    Real realVal

    realVal =  InReal (32)
Fend
```

OutReal

[Function]

Set/return the DO status as a 32-bit floating-point number.

[Syntax]

OutReal DO group number, status value

OutReal(DO group number)

[Parameter Description]

DO group number	Specify the group of DOs whose status needs to be read using an integer.
status value	Set the status of this group of DOs according to the 8421 code.

Precautions:

1. When the DO group number is set to 0, it refers to DO0~DO31. When set to 1, it refers to DO32~DO63, and so on.

[Usage Example]

```
Function main
    OutReal 32 , 2.543
Fend
```

AIO_InW

[Function]

Read the value of analog input (AI).

[Syntax]

AIO_InW(AI Number)

[Parameter Description]

AI Number Specify the analog input (AI) whose status needs to be read with an integer.

[Usage Example]

```
Function main
    If AIO_InW(1) > 123456 Then
        Print "The analog quantity is greater than 123456."
    End If
Fend
```

AIO_OutW

[Function]

Set/read the value of analog output (AO).

[Syntax]

AIO_OutW AO Number, analog value

AIO_OutW(AO Number)

[Parameter Description]

AO Number Specify the analog output (AO) whose status needs to be set or read with an integer.

analog value Set the value for this AO.

[Usage Example]

```
Function main
    If AIO_OutW (1) > 123456 Then
        AIO_OutW 1 , 0          'If the analog quantity of AO1 is greater than 123456, clear it.
    End If
Fend
```

Detailed explanation of register read/write instructions

MemOn

[Function]

Specify a bit in the user register to be 1.

[Syntax]

MemOn bit position

[Parameter Description]

bit position Specify which bit to set to 1.

Precautions:

1. A user register is 16 bits, so bits 0~15 belong to the 0th register. Bits 16~31 belong to the 1st register, and soon.

[Usage Example]

```
Function main
    MemOn 3          'Set the 3rd bit of the first register to 1.
    MemOn 16         'Set the 0th bit of the second register to 1.
Fend
```

MemOff

[Function]

Specify a bit in the user register to be 0.

[Syntax]

MemOff bit position

[Parameter Description]

bit position Specify which bit to set to 0.

Precautions:

1. A user register is 16 bits, so bits 0~15 correspond to the 0th register, bits 16~31 correspond to the 1st register, and so on.

[Usage Example]

```
Function main
    MemOff 3            'Set the 3rd bit of the first register to 0.
    MemOff 16           'Set the 0th bit of the second register to 1.
Fend
```

MemSw

[Function]

Get the value of a bit in the user register.

[Syntax]

MemSw(bit position)

[Parameter Description]

bit position Specify which bit.

Precautions:

1. A user register is 16 bits, so bits 0~15 belong to the 0th register, bits 16~31 belong to the 1st register, and so on.

[Usage Example]

```
Function main

    If MemSw (16) = 1 Then              'Get the value of the 0th bit in the second register.
        Go P1
    Else
        Go P2
    End If

Fend
```

MemIn

[Function]

Get the value of 8 bits in the user register.

[Syntax]

MemIn(bit numbering, {Signed/Unsigned})

[Parameter Description]

bit numbering Specify which group of bits.

Signed/Unsigned Read a signed value or an unsigned value. The default is unsigned.

Precautions:

1. A user register is 16 bits, so bits 0~7 and 8~15 are respectively the lower 8 bits and higher 8 bits ofthe 0th register.

Bits 16~23 and 24~31 are respectively the lower 8 bits and higher 8 bits ofthe 1st register, and so on.

Bit numbering	Corresponding address
0	user register0 (Bit0 - Bit7)
1	user register0 (Bit8 - Bit15)
2	user register1 (Bit0 - Bit7)
3	user register1 (Bit8 - Bit15)
4	user register2 (Bit0 - Bit7)
5	user register2 (Bit8 - Bit15)

[Usage Example]

```
Function main

    Print MemIn (2)
    'Get the lower 8 bits ofthe first register.

    Print MemIn (1 , Signal)
    'Obtain the higher 8 bits ofthe 0th register as a signed value.

Fend
```

MemOut

[Function]

Set the value of 8 bits in the user register.

[Syntax]

MemOut bit numbering, value

[Parameter Description]

bit numbering Specify which group of bits.

value The value set for the register.

Precautions:

1.A user register is 16 bits, so bits 0-7 and bits 8-15 correspond to the lower 8 bits and higher 8 bits ofthe 0th register respectively; bits 16-23 and bits 24-31 correspond to the lower 8 bits and higher 8 bits ofthe 1st register respectively, and so on.

Bit numbering	Corresponding address
0	user register0 (Bit0 - Bit7)
1	user register0 (Bit8 - Bit15)
2	user register1 (Bit0 - Bit7)
3	user register1 (Bit8 - Bit15)
4	user register2 (Bit0 - Bit7)
5	user register2 (Bit8 - Bit15)

[Usage Example]

```
Function main
    MemOut 1 , 123
    'Set the higher 8 bits ofthe 0th register to 123.
Fend
```

MemInW

[Function]

Get the 16 - bit value in the user register.

[Syntax]

MemInW(register number, {Signed/Unsigned})

[Parameter Description]

register number Specify which register (by number).

Precautions:

1.A user register is 16 bits, so 0 is the 0th register, 1 is the 1st register, and so on.

[Usage Example]

```
Function main

    Print MemInW(1)
    'Get the value of the first register.

    Print MemInW(2 , Signal)
    'Get the value of the 2nd register as a signed value.
Fend
```

MemOutW

[Function]

Set the 16-bit value in the user register.

[Syntax]

MemOutW register number, value

[Parameter Description]

register number	Specify which register (by number).
value	The value set for the register.

Precautions:

1. A user register is 16 bits, so 0 refers to the 0th register, 1 refers to the 1st register, and so on.

[Usage Example]

```
Function main
    MemOutW 5 , 1234
    'Set the value of the 5th register to 1234.
Fend
```

MemIn32

[Function]

Get the 32-bit value in the user register.

[Syntax]

MemIn32(register group, {Signed/Unsigned})

[Parameter Description]

register number Specify which register (by number).

Signed/Unsigned Read a signed value or an unsigned value. The default is unsigned.

Precautions:

1. A user register is 16 bits, so 0 represents the 32 - bit value formed by combining the 0th register and the 1st register; 1 represents the 32 - bit value formed by combining the 2nd register and the 3rd register, and so on.

Bit numbering	Corresponding address
0	user register0 (Bit0 - Bit1)
1	user register0 (Bit2 - Bit3)
2	user register1 (Bit4 - Bit5)
3	user register1 (Bit6 - Bit7)
4	user register2 (Bit8 - Bit9)
5	user register2 (Bit10 - Bit11)

[Usage Example]

```
Function main

    Print MemIn32(0)
    'Get the 32-bit value formed by combining the 0th register and the 1st register.

    Print MemIn32(1 , Signal)
    'Get the 32-bit value formed by combining the 2nd register and the 3rd register as a signed value.
Fend
```

MemOut32

[Function]

Set the 32-bit value in the user register.

[Syntax]

MemOut32 register group, value

[Parameter Description]

register group Specify which group of registers.

value The value set for the register.

Precautions:

1. A user register is 16 bits, so 0 is the 32 - bit value formed by combining the 0th register and the 1st register; 1 is the 32 - bit value formed by combining the 2nd register and the 3rd register, and so on.

Bit numbering	Corresponding address
0	user register0 (Bit0 - Bit1)
1	user register0 (Bit2 - Bit3)
2	user register1 (Bit4 - Bit5)
3	user register1 (Bit6 - Bit7)
4	user register2 (Bit8 - Bit9)
5	user register2 (Bit10 - Bit11)

[Usage Example]

```
Function main
    MemOut32 1 , 1234
    'Set the 32-bit value formed by combining the 2nd register and the 3rd register to 1234.
Fend
```

MemInReal

[Function]

Get the 32-bit value in the user register as a 32-bit floating-point number.

[Syntax]

MemInReal(register group)

[Parameter Description]

register group Specify which group of registers.

Bit numbering	Corresponding address
0	user register0 (Bit0 - Bit1)
1	user register0 (Bit2 - Bit3)
2	user register1 (Bit4 - Bit5)
3	user register1 (Bit6 - Bit7)
4	user register2 (Bit8 - Bit9)
5	user register2 (Bit10 - Bit11)

Precautions:

1. A user register is 16 bits, so 0 corresponds to the 32-bit value formed by combining the 0th register and the 1st register;

1 corresponds to the 32-bit value formed by combining the 2nd register and the 3rd register, and so on.

[Usage Example]

```
Function main

    Print MemInReal (0)
    'Get the 32-bit value formed by combining the 0th register and the 1st register as a 32-bit floating-point number.
Fend
```

MemOutReal

[Function]

Set the 32-bit value in the user register.

[Syntax]

MemOutReal register group, value

[Parameter Description]

register group Specify which group of registers.

value The value set for the register.

Bit numbering	Corresponding address
0	user register0 (Bit0 - Bit1)
1	user register0 (Bit2 - Bit3)
2	user register1 (Bit4 - Bit5)
3	user register1 (Bit6 - Bit7)
4	user register2 (Bit8 - Bit9)
5	user register2 (Bit10 - Bit11)

Precautions:

1. A user register is 16 bits, so 0 is the 32-bit value formed by combining the 0th register and the 1st register; 1 is the 32-bit value formed by combining the 2nd register and the 3rd register; and so on.

[Usage Example]

```
Function main
    MemOutReal 1 , 123.321
    'Set the 32-bit value formed by combining the 2nd register and the 3rd register to 123.321.
Fend
```

SysMemOn

[Function]

Specify a bit in the system register to be set to 1.

[Syntax]

SysMemOn bit position

[Parameter Description]

bit position Specify which bit to set to 1.

Precautions:

1. A system register is 16 bits, so bits 0~15 belong to the 0th register; bits 16~31 belong to the 1st register, and soon.

[Usage Example]

```
Function main
    SysMemOn 0          'Set the 1st bit of the 0th system register to On.
    SysMemOn 15         'Set the 16th bit of the 0th system register to On.
    SysMemOn 16         'Set the 1st bit of the 1st system register to On.
Fend
```

SysMemOff

[Function]

Specify a bit in the system register to be set to 0.

[Syntax]

SysMemOff bit position

[Parameter Description]

bit position Specify which bit to set to 0.

Precautions:

1. A system register is 16 bits, so bits 0~15 correspond to the 0th register; bits 16~31 correspond to the 1st register, and so on.

[Usage Example]

```
Function main
    SysMemOff 0              'Set the 1st bit of the 0th system register to Off.
    SysMemOff 15             'Set the 16th bit of the 0th system register to Off.
    SysMemOff 16             'Set the 1st bit of the 1st system register to Off.
Fend
```

SysMemSw

[Function]

Get the value of a bit in the system register.

[Syntax]

SysMemSw bit position

[Parameter Description]

bit position Specify which bit.

Precautions:

1.A system register is 16 bits, so bits 0~15 belong to the 0th register; bits 16~31 belong to the 1st register, and so on.

[Usage Example]

```
Function main

    If SysMemSw (0) = On Then                    'It holds when the 1st bit of the 0th system register is on.
        Go P1

    Else If SysMemSw (15) = On Then              ' It holds when the 16th bit of the 0th system register is on.
        Go P2
    End If

Fend
```

SysMemIn

[Function]

Get the 8-bit value in the system register.

[Syntax]

SysMemIn(bit numbering, {Signed/Unsigned})

[Parameter Description]

bit numbering Specify which group of bits.

Signed/Unsigned Read a signed value or an unsigned value. The default is unsigned.

Bit numbering	Corresponding address
0	user register0 (Bit0 – Bit7)
1	user register0 (Bit8 – Bit15)
2	user register1 (Bit0 – Bit7)
3	user register1 (Bit8 – Bit15)
4	user register2 (Bit0 – Bit7)
5	user register2 (Bit8 – Bit15)

Precautions:

1.A system register is 16 bits, so 0~1 correspond to the low 8 bits and high 8 bits ofthe 0th register respectively; 2~3 correspond to the low 8 bits and high 8 bits ofthe 1st register respectively, and so on.

[Usage Example]

```
Function main

    Print SysMemIn (0)
    'Print the low 8 bits ofthe 0th register.

    Print SysMemIn (1 , Unsigned)
    'Print the high 8 bits ofthe 0th register as an unsigned value.

Fend
```

SysMemOut

[Function]

Set the value of 8 bits in the system register.

[Syntax]

SysMemOut bit numbering, value

[Parameter Description]

bit numbering Specify which group of bits.

value The value set for the register.

Precautions:

1. A system register is 16 bits, so 0~1 are the low 8 bits and high 8 bits ofthe 0th register respectively; 2~3 are the low 8 bits and high 8 bits ofthe 1st register respectively, and so on.

Bit numbering	Corresponding address
0	user register0 (Bit0 - Bit7)
1	user register0 (Bit8 - Bit15)
2	user register1 (Bit0 - Bit7)
3	user register1 (Bit8 - Bit15)
4	user register2 (Bit0 - Bit7)
5	user register2 (Bit8 - Bit15)

[Usage Example]

```
Function main
    SysMemOut 1 , 123
    'Set the high 8 bits ofthe 0th register to 123.
Fend
```

SysMemInW

[Function]

Get the 16-bit value in the system register.

[Syntax]

SysMemInW (register number, {Signed/Unsigned})

[Parameter Description]

register number	Specify which register (by number).
Signed/Unsigned	Read a signed value or an unsigned value. The default is unsigned.

Precautions:

1.A system register is 16 bits, so 0 corresponds to the 0th register; 1 corresponds to the 1st register, and so on.

[Usage Example]

```
Function main

    Print SysMemInW (0,Unsigned)
    'Print the No. 0 system register in unsigned integer format.

    Print SysMemInW (1 , Signal)
    'Print the No. 1 system register as a signed value.
Fend
```

SysMemOutW

[Function]

Set the 16-bit value in the system register.

[Syntax]

SysMemOutW register number, value

[Parameter Description]

register number	Specify which register (by number).
value	The value set for the register.

Precautions:

1. A system register is 16 bits, so 0 is the 0th register; 1 is the 1st register, and so on.

[Usage Example]

```
Function main
    SysMemOutW 0 , 123
    'Write the value 123 to the address of the 0th register.
Fend
```

SysMemIn32

[Function]

Get the 32-bit value in the user register.

[Syntax]

SysMemIn32(register group, {Signed/Unsigned})

[Parameter Description]

register number Specify which register (by number).

Signed/Unsigned Read a signed value or an unsigned value. The default is unsigned.

Precautions:

1. A system register is 16 bits, so 0 is the 32-bit value formed by combining the 0th register and the 1st register; 1 is the 32-bit value formed by combining the 2nd register and the 3rd register, and so on.

Bit numbering	Corresponding address
0	user register0 (Bit0 - Bit1)
1	user register0 (Bit2 - Bit3)
2	user register1 (Bit4 - Bit5)
3	user register1 (Bit6 - Bit7)
4	user register2 (Bit8 - Bit9)
5	user register2 (Bit10 - Bit11)

[Usage Example]

```
Function main

    Print SysMemIn32 (0)
    'Get the 32-bit value formed by combining the 0th register and the 1st register.

    Print SysMemIn32 (1 , Unsigned)
    'Obtain the 32-bit value formed by combining the 2nd register and the 3rd register as a signed value.
Fend
```

SysMemOut32

[Function]

Set the 32-bit value in the system register.

[Syntax]

SysMemOut32 register group, value

[Parameter Description]

register group Specify which group of registers.

value The value set for the register.

Precautions:

1. A system register is 16 bits, so 0 is the 32-bit value formed by combining the 0th register and the 1st register; 1 is the 32-bit value formed by combining the 2nd register and the 3rd register, and so on.

Bit numbering	Corresponding address
0	user register0 (Bit0 - Bit1)
1	user register0 (Bit2 - Bit3)
2	user register1 (Bit4 - Bit5)
3	user register1 (Bit6 - Bit7)
4	user register2 (Bit8 - Bit9)
5	user register2 (Bit10 - Bit11)

[Usage Example]

```
Function main
    SysMemOut32 1 , 1234
    'Set the 32-bit value formed by combining the 2nd system register and the 3rd system register to 1234.
Fend
```

SysMemInReal

[Function]

Obtain the 32-bit value in the system register as a 32-bit floating-point number.

[Syntax]

SysMemInReal(register group)

[Parameter Description]

register group Specify which group of registers.

Precautions:

1. A system register is 16 bits, so 0 is the 32-bit value formed by combining the 0th register and the 1st register; 1 is the 32-bit value formed by combining the 2nd register and the 3rd register, and so on.

Bit numbering	Corresponding address
0	user register0 (Bit0 - Bit1)
1	user register0 (Bit2 - Bit3)
2	user register1 (Bit4 - Bit5)
3	user register1 (Bit6 - Bit7)
4	user register2 (Bit8 - Bit9)
5	user register2 (Bit10 - Bit11)

[Usage Example]

```
Function main

    Print SysMemInReal (0)
    'Obtain the 32-bit value formed by combining the 0th register and the 1st register as a 32-bit floating-point number.

Fend
```

SysMemOutReal

[Function]

Set the 32-bit value in the system register as a 32-bit floating-point number.

[Syntax]

SysMemOutReal register group, value

[Parameter Description]

register group Specify which group of registers.

value The value set for the register.

Precautions:

1. A system register is 16 bits, so 0 represents the 32-bit value formed by combining the 0th register and the 1st register; 1 represents the 32-bit value formed by combining the 2nd register and the 3rd register, and so on.

Bit numbering	Corresponding address
0	user register0 (Bit0 - Bit1)
1	user register0 (Bit2 - Bit3)
2	user register1 (Bit4 - Bit5)
3	user register1 (Bit6 - Bit7)
4	user register2 (Bit8 - Bit9)
5	user register2 (Bit10 - Bit11)

[Usage Example]

```
Function main
    SysMemOutReal 1 , 123.321
    'Set the 32-bit value formed by combining the 2nd register and the 3rd register to 123.321.
Fend
```

Detailed description of communication commands

SetNet

[Function]

Used to set the communication parameters of TCP/IP.

[Syntax]

SetNet #Communication port number, IP address, port number{, terminator, flow control, timeout, communication protocol}

[Parameter Description]

#Communication port number	Specify the port number for which communication parameters need to be modified. Range: #201~216.
IP address	IP address used for TCP/IP communication. For example: 192.168.1.1
port number	Port number used for TCP/IP communication. For example: 8090, range: 0~65535.
terminator	The trailing character of communication data. It can be set to CR, LF, or CRLF (carriage return, line feed, carriage return plus line feed) and can be omitted.
flow control	Refers to software flow control. It is set to NONE and can be omitted.
Timeout	The maximum time for sending and receiving is set in seconds. 0 means never time out, and it can be omitted.
communication protocol	Currently, only the TCP protocol is supported, and it can be omitted.

[Usage Example]

```
Function main
   ReConnect:
   CloseNet #201
   SetNet #201 , "192.168.1.220" , 9000 ,CRLF
   OpenNet #201 As Client
   WaitNet #201, 5
    If ChkNet (201) = -1 Then
        Print "Communication error: The port is open, but no communication has been established."
        Wait 1
        GoTo ReConnect
   ElseIf ChkNet (201) = -2 Then
        Print "Communication error: The port is being used by another task."
        Wait 1
        GoTo ReConnect
```

```
    ElseIf ChkNet (201) = -3 Then
        Print "Communication error: The port is not open."
        Wait 1
        GoTo ReConnect
    End If
Fend
```

OpenNet

[Function]

Used to start TCP/IP communication or obtain the thread number of OpenNet.

[Syntax]

OpenNet #Communication port number As [Client/Server]
OpenNet(Communication port number)

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified. Range: #201~216.

Precautions:

1. When using OpenNet as a client, the SetNet directive parameter (IP address) is filled in with the server IP address.

2. When using OpenNet as a server, SetNet directive parameter (IP address) fills in the controller IP address.

[Usage Example]

```
Function main
   ReConnect:
   CloseNet #201
   SetNet #201 , "192.168.1.220" , 9000 ,CRLF
   OpenNet #201 As Client
   WaitNet #201, 5
    If ChkNet (201) = -1 Then
        Print "Communication error: The port is open, but no communication has been established."
        Wait 1
        GoTo ReConnect
   ElseIf ChkNet (201) = -2 Then
        Print "Communication error: The port is being used by another task."
        Wait 1
        GoTo ReConnect
   ElseIf ChkNet (201) = -3 Then
        Print "Communication error: The port is not open."
        Wait 1
        GoTo ReConnect
   End If
Fend
```

WaitNet

[Function]

Used to wait for the establishment of a TCP/IP communication port connection.

[Syntax]

WaitNet #Communication port number{, Time-out period}

[Parameter Description]

#Communication port number	Specify the port number for which communication parameters need to be modified. Range: #201~216.
Time-out period	The waiting time is set in seconds. If omitted, the wait is infinite.

Precautions:

1. When the timeout is not omitted, the program waits at this statement line for the specified timeout or for the connection bar to be established

[Usage Example]

```
Function main
    ReConnect:
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    If ChkNet(201) = -1 Then
        Print "Communication error: The port is open, but no communication has been established."
        Wait 1
        GoTo ReConnect
    ElseIf ChkNet(201) = -2 Then
        Print "Communication error: The port is being used by another task."
        Wait 1
        GoTo ReConnect
    ElseIf ChkNet(201) = -3 Then
        Print "Communication error: The port is not open."
        Wait 1
        GoTo ReConnect
    End If
Fend
```

ChkNet

[Function]

Used to return the number of bytes in the receive buffer of the communication port.

[Syntax]

ChkNet(Communication port number)

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified. Range: #201~216.

Precautions:

1. Returns the number of bytes as an integer. If no data exists, returns the communication port status as a negative value.

   - -1 indicates the port is open but there is no communication connection.
   - -2 indicates the communication port is being used by another thread.
   - -3 indicates the communication port is not open.

[Usage Example]

```
Function main
   ReConnect:
   CloseNet #201
   SetNet #201 , "192.168.1.220" , 9000 ,CRLF
   OpenNet #201 As Client
   WaitNet #201, 5
    If ChkNet(201) = -1 Then
        Print "Communication error: The port is open, but no communication has been established."
        Wait 1
        GoTo ReConnect
   ElseIf ChkNet(201) = -2 Then
        Print "Communication error: The port is being used by another task."
        Wait 1
        GoTo ReConnect
   ElseIf ChkNet(201) = -3 Then
        Print "Communication error: The port is not open."
        Wait 1
        GoTo ReConnect
   End If
Fend
```

ClrNet

[Function]

Used to clear the communication buffer.

[Syntax]

ClrNet #Communication port number

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified. Range: #201~216.

[Usage Example]

```
Function main
   ReConnect:
   CloseNet #201
   SetNet #201 , "192.168.1.220" , 9000 ,CRLF
   OpenNet #201 As Client
   WaitNet #201, 5
    If ChkNet (201) = -1 Then
        Print "Communication error: The port is open, but no communication has been established."
        Wait 1
        GoTo ReConnect
   ElseIf ChkNet (201) = -2 Then
        Print "Communication error: The port is being used by another task."
        Wait 1
        GoTo ReConnect
   ElseIf ChkNet (201) = -3 Then
        Print "Communication error: The port is not open."
        Wait 1
        GoTo ReConnect
   End If

   Print #201 , "EXW,1"
    Input #201 , CCD_X$,CCD_Y$,CCD_U$
   ClrNet #201

Fend
```

CloseNet

[Function]

Used to close the TCP/IP communication port opened by OpenNet.

[Syntax]

CloseNet [#Communication port number/All]

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified. Range: #201~216.

Precautions:
1. When the trigger stops, the communication is shut down.

[Usage Example]

```
Function main
   ReConnect:
   CloseNet #201
   SetNet #201 , "192.168.1.220" , 9000 ,CRLF
   OpenNet #201 As Client
   WaitNet #201, 5
    If ChkNet (201) = -1 Then
        Print "Communication error: The port is open, but no communication has been established."
        Wait 1
        GoTo ReConnect
   ElseIf ChkNet (201) = -2 Then
        Print "Communication error: The port is being used by another task."
        Wait 1
        GoTo ReConnect
   ElseIf ChkNet (201) = -3 Then
        Print "Communication error: The port is not open."
        Wait 1
        GoTo ReConnect
   End If
Fend
```

SetCom

[Function]

Used to set the RS232/485 serial communication parameters.

[Syntax]

SetCom #Communication port number{, baud rate, data bit length, stop bit length, parity, terminator,H/W flow control, S/W flow control, timeout period}

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified. Range: #1~16.

baud rate Can specify 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 230400, 256000. The default value is 19200 if omitted.

data bit length The data bit length of each character is specified by 7 or 8, and it can be omitted.

stop bit length The stop bit length of each character is specified by 1 or 2, and it can be omitted.

terminator The trailing character of communication data. It can be set to CR, LF, or CRLF (carriage return, line feed, carriage return and line feed), and can be omitted.

H/W flow control When hardware control is valid, specify RTS; when invalid, specify NONE. It can be omitted.

S/W flow control When software control is valid, specify XON; when invalid, specify NONE. It can be omitted.

timeout period The maximum time for sending and receiving is set in seconds. 0 means never time out, and it can be omitted.

[Usage Example]

```
Function main
    Integer AA
    ReConnect:
    SetCom #2,38400,8,2,N,CRLF,NONE,NONE,0
    OpenCom #2
    Do
        Print #2 ,"OK"
         Input #2,AA
        Print AA
    Loop

Fend
```

OpenCom

[Function]

Used to start RS232/485 serial communication or obtain the thread number ofthe RS232/485 serial port.

[Syntax]

OpenCom #Communication port number

OpenCom(Communication port number)

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified. Range: #1~16.

[Usage Example]

```
Function main
    Integer AA
    ReConnect:
    SetCom #2,38400,8,2,N,CRLF,NONE,NONE,0
    OpenCom #2
    Do
        Print #2 ,"OK"
        Input #2,AA
        Print AA
    Loop

Fend
```

ChkCom

[Function]

Used to return the number of bytes in the receive buffer of the serial communication port.

[Syntax]

ChkCom(Communication port number)

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified. Range: #1~16.

Precautions:

1. Returns the number of bytes as an integer. If no data exists, returns the communication port status as a negative value.

- -1 indicates the port is open but there is no communication connection.
- -2 indicates the communication port is being used by another thread.
- -3 indicates the communication port is not open.

[Usage Example]

```
Function main
    Integer numChars
    numChars = ChkCom(1)
Fend
```

ClrCom

[Function]

Used to clear the communication buffer.

[Syntax]

ClrCom #Communication port number

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified. Range: #1~16.

[Usage Example]

```
Function main
    ClrCom #1
Fend
```

CloseCom

[Function]

Used to close the serial communication port opened by OpenCom.

[Syntax]

CloseCom [#Communication port number/All]

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified. Range: #1~16.

[Usage Example]

```
Function main
    CloseCom #1
Fend
```

Read

[Function]

Used to read a specified number of characters from the communication port, excluding the terminator.

[Syntax]

Read #Communication port number, String variable$, Number of characters

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified. The range for TCP/IP is #201~216, and the range for serial ports is #1~16.

String variable$ A string variable used to receive characters from the port.

Number of characters The number of characters to read.

[Usage Example]

```
Function main
    String strBin$
    Read #201,strBin$,4          'Read four characters from port #201
Fend
```

ReadBin

[Function]

Used to read binary data from a TCP/IP communication port.

[Syntax]

ReadBin #Communication port number , Variable name

ReadBin #Communication port number , Array variable name() , number of bytes

[Parameter Description]

#Communication port number	Specify the port number for which communication parameters need to be modified. The range for TCP/IP is #201~216, and the range for serial ports is #1~16.
Variable name	A variable used to receive binary data from the port.
Array variable name()	An array variable used to receive binary data from the port.
number of bytes	The number of bytes to read.

[Usage Example]

```
Function main
    Integer num
    ReadBin #201,num
    Integer arr(5)
    ReadBin #201,arr(),5
Fend
```

Write

[Function]

Write the string to the communication port, excluding the terminator.

[Syntax]

Write　#Communication port number ， String

[Parameter Description]

#Communication port number　Specify the port number for which communication parameters need to be modified.

The range for TCP/IP is #201~216, and the range for serial ports is #1~16.

String　The string to be written to the communication port.

[Usage Example]

```
Function main
    Wr ite #201 ,  "abcd"
Fend
```

WriteBin

[Function]

Write binary data to the communication port, excluding the terminator.

[Syntax]

WriteBin #Communication port number , Variable name

WriteBin #Communication port number , Array variable name(), number of bytes

[Parameter Description]

#Communication port number	Specify the port number for which communication parameters need to be modified. The range for TCP/IP is #201~216, and the range for serial ports is #1~16.
Variable name	A variable used to receive binary data from the port.
Array variable name()	An array variable used to receive binary data from the port.
Number of bytes	The number of bytes to read.

[Usage Example]

```
Function main
    WriteBin #201 , 123456
    Integer arr(5)
    arr(0) = 22
    arr(1) = 43
    arr(2) = 67
    arr(3) = 45
    arr(4) = 33
    WriteBin #201 , arr() , 5
Fend
```

Print

[Function]

Output data or string to the specified communication port, including the terminator.

[Syntax]

Print #Communication port number, Data 1{, Data 2…}

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified. The range for TCP/IP is #201~216, and the range for serial ports is #1~16.

Data 1{, Data 2...} You can use variables, values, strings, etc. to output the content you want to output. You can appropriately increase the amount of data according to your needs.

[Usage Example]

```
Function main
    String CCD_X$,CCD_Y$,CCD_U$
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    Print #201 , "EXW,1"
     Input #201 , CCD_X$,CCD_Y$,CCD_U$
    Print CCD_X$,CCD_Y$,CCD_U$
Fend
```

Input

[Function]

Used to read data or strings from the communication port, including the terminator.

[Syntax]

Input #Communication port number, Variable 1{, Variable 2…}

[Parameter Description]

#Communication port number	Specify the port number for which communication parameters need to be modified. The range for TCP/IP is #201~216, and the range for serial ports is #1~16.
Variable 1{, Variable 2...}	The variable name used to receive data or strings. Variables can be added as appropriate according to the data length.

Precautions:

1. Input # By default, strings are split by "," (comma) or " " (whitespace). For other delimiters, please use the token string instruction.

[Usage Example]

```
Function main
    String CCD_X$,CCD_Y$,CCD_U$
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    Print #201 , "EXW,1"
     Input #201 , CCD_X$,CCD_Y$,CCD_U$
    Print CCD_X$,CCD_Y$,CCD_U$
Fend
```

Line Input

[Function]

Used to read a line of data or string from the communication port, including the terminator.

[Syntax]

Line Input #Communication port number, Variable

[Parameter Description]

#Communication port number Specify the port number for which communication parameters need to be modified.

The range for TCP/IP is #201~216, and the range for serial ports is #1~16.

Variable The variable name used to receive data or strings.

[Usage Example]

```
Function main
    String CCD_TxT$ ,strArr$(0)
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5

    Print #201 , "EXW,1"
    Line Input #201 , CCD_TxT$
    ParseStr CCD_TxT$ , strArr$() , "$"        'Split the content of the string using $ as the delimiter and store it
                                                in an array.

    For i = 0 To UBound(strArr$())
        Print "Array content = " , strArr$(i)
    Next
Fend
```

Detailed description of system commands

Print

[Function]

Print command: print strings, values, variables.

[Syntax]

Print Content 1{, Content 2.....}

[Parameter Description]

Content 1{, Content 2.....}	The string, value, or variable to be printed. You can add variables as appropriate according to your needs.

[Usage Example]

```
Function main
   ReConnect:
   CloseNet #201
   SetNet #201 , "192.168.1.220" , 9000 ,CRLF
   OpenNet #201 As Client
   WaitNet #201, 5
    If ChkNet (201) = -1 Then
        Print "Communication error: The port is open, but no communication has been established."
        Wait 1
        GoTo ReConnect
   ElseIf ChkNet (201) = -2 Then
        Print "Communication error: The port is being used by another task."
        Wait 1
        GoTo ReConnect
   ElseIf ChkNet (201) = -3 Then
        Print "Communication error: The port is not open."
        Wait 1
        GoTo ReConnect
   End If

   Print ChkNet (201)

Fend
```

Xqt

[Function]

Start multi-threading.

[Syntax]

Xqt {Task number, } Function name(Parameter 1....) {, Normal/NoPause/NoEmgAbort}

[Parameter Description]

Task number	Specify the multi-thread number as an integer, which can be omitted.
Function name	The name of the function to be executed in multi-threading. If the function has formal parameters, you need to add () and fill in the formal parameters; if there are no formal parameters, you can directly fill in the function name.
Normal	Ordinary multi-threading. If all three are omitted, the default is Normal.
NoPause	A multi-thread that will not pause when a Pause occurs, there is an external pause signal, or the safety door is open.
NoEmgAbort	A multi-thread that can continue processing even when an emergency stop occurs or an error happens.

Precautions:

1. For foreground multi-threads, the task numbers range from 1 to 16. For background multi-threads, the task numbers range from 66 to 80.

[Usage Example]

```
Function main
    Xqt A1
Fend

Function A1
    Print "Runing_____"
Fend
```

Pause

[Function]

Used to pause all pausable multi-threads.

[Syntax]

Pause

[Parameter Description]

Precautions:

1. For foreground multi-threads, the task numbers range from 1 to 16. For background multi-threads, the task numbers range from 66 to 80.

[Usage Example]

```
Function main
    Xqt A1
    Wait 1
    Pauser                  'Pause the program
Fend

Function A1
    Print "Runing_____"
Fend
```

Halt

[Function]

Used to pause the specified running multi-thread.

[Syntax]

Halt [Function name/Task number]

[Parameter Description]

Function name	The name of the multi-thread function that is currently executing.
Task number	The task number of the multi-thread that is currently executing.

Precautions:

1. For foreground multi-threads, the task numbers range from 1 to 16. For background multi-threads, the task numbers range from 66 to 80.

[Usage Example]

```
Function main
    Xqt A1
    Wait 1
    Halt A1            'Pause the A1 thread
Fend

Function A1
    Do
        Print "Runing_____"
    Loop
Fend
```

Quit

[Function]

Stop all or specified threads.

[Syntax]

Quit [Function name/Task number/All]

[Parameter Description]

Function name	The name of the multi-thread function that is currently executing.
Task number	The task number of the multi-thread that is currently executing.
All	Stop all threads.

Precautions:

1. For foreground multi-threads, the task numbers range from 1 to 16. For background multi-threads, the task numbers range from 66 to 80.

[Usage Example]

```
Function main
    Xqt A1
    Wait 1
    Quit A1          'Stop the A1 thread.
Fend

Function A1
    Do
        Print "Runing_____"
    Loop
Fend
```

StartMain

[Function]

Used to enable the main task.

[Syntax]

StartMain [Main task name]

[Parameter Description]

Main task name　　main - main63。

[Usage Example]

```
Function bgmain                 'Background task
    If MemInW(0) = 1 Then
        StartMain main
        Wait MemInW(0) <> 1     'Wait for the signal to disappear to prevent repeated triggering.
    End If
Fend
```

Resume

[Function]

Used to resume the multi-threads that were paused by the Halt command.

[Syntax]

Resume [Function name/Task number/All]

[Parameter Description]

Function name	The name of the multi-thread function that is currently executing.
Task number	The task number of the multi-thread that is currently executing.
All	Resume all threads.

Precautions:

1. For foreground multi-threads, the task numbers range from 1 to 16. For background multi-threads, the task numbers range from 66 to 80.

[Usage Example]

```
Function main
    Xqt A1
    Wait 1
    Halt A1          'Pause the A1 thread.
    Wait Sw(1) = 1
    Resume A1        'Resume the operation of the A1 thread.
Fend

Function A1
    Do
        Print "Running_____"
    Loop
Fend
```

Reset

[Function]

Used to reset the alarm status of the controller.

[Syntax]

Reset [Error]

[Parameter Description]

Error	It means that all errors will be cleared by the instruction.

Precautions:

1. When the instruction is called, if "Error" is filled in, all alarms will be cleared; if "Error" is not filled in, only the emergency stop status will be cleared.

[Usage Example]

```
Function main
    Reset              'Clear the emergency stop status.
Fend

Function A1
    Reset Error        'Clear the emergency stop status.
Fend
```

MyTask

[Function]

Used to return the task number of the current thread.

[Syntax]

MyTask

[Parameter Description]

Precautions:

1. For foreground multi-threads, the task numbers range from 1 to 16. For background multi-threads, the task numbers range from 66 to 80.

[Usage Example]

```
Function main
    Xqt 2 , A1
    Xqt 3 , A1
    Xqt 4 , A1
Fend

Function A1
    On MyTask          ' Set the IO port with the same number as the current task number to On.
    Wait 1
    Off MyTask         'Set the IO port with the same number as the current task number to Off.
Fend
```

TaskDone

[Function]

Confirm whether the thread has ended.

[Syntax]

TaskDone(Function name/Task number)

[Parameter Description]

Function name	The name of the multi-thread function that is currently executing.
Task number	The task number of the multi-thread that is currently executing.

Precautions:

1. For foreground multi-threads, the task numbers range from 1 to 16. For background multi-threads, the task numbers range from 66 to 80.
2. If the thread has ended, return True; otherwise, return False.

[Usage Example]

```
Function main
    Xqt A1
    Do
        Print "A1"
    Loop Until TaskDone(A1)
Fend

Function A1
    Go P1
Fend
```

TaskState

[Function]

Return the current status of the thread.

[Syntax]

TaskState(Function name/Task number)

[Parameter Description]

Function name	The name of the multi-thread function that is currently executing.
Task number	The task number of the multi-thread that is currently executing.

Precautions:

1. For foreground multi-threads, the task numbers range from 1 to 16. For background multi-threads, the task numbers range from 66 to 80.

2. Feedback value:

0 indicates "Not executed".

4 indicates "Standby".

5 indicates "Starting up".

6 indicates "Running".

7 indicates "Error state".

8 indicates "Paused state".

9 indicates "Resuming state".

10 indicates "Stopping state".

[Usage Example]

```
Function main
    If TaskState (A1) = 0 Then
        Xqt 2 , A1
    End If
Fend

Function A1
    Go P1
Fend
```

TaskWait

[Function]

Used to wait for the specified thread to end.

[Syntax]

TaskWait(Function name/Task number)

[Parameter Description]

Function name	The name of the multi-thread function that is currently executing.
Task number	The task number of the multi-thread that is currently executing.

Precautions:

1. For foreground multi-threads, the task numbers range from 1 to 16. For background multi-threads, the task numbers range from 66 to 80.

[Usage Example]

```
Function main
    Xqt 2 , A1
    TaskWait A1
Fend

Function A1
    Go P1
Fend
```

SyncLock

[Function]

Used to synchronize multiple threads by using mutual exclusive locking.

[Syntax]

SyncLock Signal number {, timeout period}

[Parameter Description]

Signal number Specify the signal number to be received with an expression or a numerical value, ranging from 0 to 63.

timeout period Specify the waiting time before locking with an expression or a numerical value, which can be omitted.

[Usage Example]

```
Function Main
    Xqt Func1
    Xqt Func2
Fend
Function Func1
    Long count
    Do
    Wait 0.5
        count = count+1
        LogMsg ("Msg from Func1,"+ Str$ (count))
    Loop
Fend

Function Func2
    Long count
    Do
    Wait 0.5
        count = count+1
        LogMsg ("Msg from FuncmsgSCloseCom2,"+ Str$ (count))
    Loop
Fend

Function LogMsg (msg$ As String)
    SyncLock 1
    Print msg$
    Wait 1
    SyncUn lock 1
Fend
```

SyncUnLock

[Function]

Used to release the signal number locked by SyncLock.

[Syntax]

SyncUnLock Signal number

[Parameter Description]

Signal number Specify the signal number to be received with an expression or a numerical value, ranging from 0 to 63.

[Usage Example]

```
Function Main
    Xqt Func1
    Xqt Func2
Fend
Function Func1
    Long count
    Do
    Wait 0.5
        count = count+1
        LogMsg ("Msg from Func1,"+ Str$ (count))
    Loop
Fend

Function Func2
    Long count
    Do
    Wait 0.5
        count = count+1
        LogMsg ("Msg from FuncmsgSCloseCom2,"+ Str$ (count))
    Loop
Fend

Function LogMsg (msg$ As String)
    SyncLock 1
    Print msg$
    Wait 1
    SyncUn lock 1

Fend
```

Signal

[Function]

Used to send a signal to the task that is executing the WaitSig command.

[Syntax]

Signal　Signal number

[Parameter Description]

Signal number　Specify the signal number to be received with an expression or a numerical value, ranging from 0 to 63.

[Usage Example]

```
Function main
    Xqt 2 , A1
    Signal 1
Fend

Function A1
    WaitSig 1
Fend
```

WaitSig

[Function]

Used to wait for synchronization signals sent by the Signal command of other tasks.

[Syntax]

WaitSig Signal number{, timeout period}

[Parameter Description]

Signal number Specify the signal number to be received with an expression or a numerical value, ranging from 0 to 63.

timeout period Specify the maximum waiting time with an expression or a numerical value, which can be omitted.

[Usage Example]

```
Function main
    Xqt 2 , A1
    Signal 1
Fend

Function A1
    WaitSig 1
Fend
```

TW

[Function]

Used to return the status of the Wait, WaitNet, and WaitSig commands.

[Syntax]

TW

[Parameter Description]

Precautions:

1.Within the specified time, the conditions for Wait, WaitNet, and WaitSig being met return False; if timed out, they return True.

[Usage Example]

```
Function main
    Wait Sw(0) = On , 5
     If TW = True Then
          ' Wait for 5 seconds before the input DI0 turns to ON state.
         Print "DI0 is not set to ON."
    End If
Fend
```

ElapsedTime

[Function]

Returns the elapsed time in seconds since the calculation cycle timer started, excluding program pause time.

[Syntax]

ElapsedTime

[Parameter Description]

Precautions:

1. The measurement range of the timer is 0 to 1.7E+31. The minimum unit is 0.001 seconds.

[Usage Example]

```
Function Main
    Integer i
    ResetElapsedTime                    'Reset the timer used for calculating the cycle time.
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "Number of times: ", i, ElapsedTime / 10  'Calculate and print the cycle time.
    If i < 10 Then
        Return
    End If
Fend
```

ResetElapsedTime

[Function]

Used to reset the timer for calculating cycle time used by ElapsedTime.

[Syntax]

ResetElapsedTime

[Parameter Description]

[Usage Example]

```
Function Main
    Integer i
    ResetElapsedTime                    'Reset the timer used for calculating the cycle time.
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "Number of times: ", i, ElapsedTime / 10        'Calculate and print the cycle time.
    If i < 10 Then
        Return
    End If
Fend
```

Tmr

[Function]

Returns the elapsed time in seconds since the timer started, including program pause time.

[Syntax]

Tmr(Timer number)

[Parameter Description]

Timer number	Specify the time of which timer to return using an expression or a numerical value, with the range being 0 to 63.

[Usage Example]

```
Function Main
    Integer i
    TmReset 0                          ' Reset the timer used for calculating the cycle time.
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "Number of times: ", i, Tmr (0) / 10    'Calculate and print the cycle time.
    If i < 10 Then
        Return
    End If
Fend
```

TmReset

[Function]

Used to reset the Tmr timer.

[Syntax]

TmReset Timer number

[Parameter Description]

Timer number Specify the time of which timer to return using an expression or a numerical value, with the range being 0 to 63.

[Usage Example]

```
Function Main
    Integer i
    TmReset 0                          ' Reset the timer used for calculating the cycle time.
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "Number of times: ", i, Tmr (0) / 10    'Calculate and print the cycle time.
    If i < 10 Then
        Return
    End If
Fend
```

EVal

[Function]

Used to execute statements in the command window.

[Syntax]

Eval Command statement

[Parameter Description]

Command statement	Specify the command to be executed as a string.
Return Value	Returns the error code returned by executing the command. When the command ends normally, it returns 0.

[Usage Example]

```
Function Main

Integer errCMD
String strBin$

    Do
        strBin$ = "Motor On"            'String assignment
        EVal(strBin$ )                  'Executing strings

        Print EVal("Motor On")          'Execute servo power on and print EVal status code

        errCMD = EVal("Motor On")       'Perform servo power-on and assign status code to
ErrCMD variable
    Loop

Fend
```

Time

[Function]

Used to print/return the controller's system time.

[Syntax]

Time

Print Time(Number)

[Parameter Description]

Number Specify to obtain the hour/minute/second of the controller's system time with a value ranging from 0 to 2.

[Usage Example]

```
Function Main
    Time      'Print the controller's system time.

    Integer New_Time
    New_Time = Time (0)      'Return the hour of the controller's system time.
    Print New_Time

    New_Time = Time (1)      'Return the minute of the controller's system time.
    Print New_Time

    New_Time = Time (2)      'Return the second of the controller's system time.
    Print New_Time
Fend
```

Time$

[Function]

Used to return the controller's system time in the specified format hh:mm:ss.

[Syntax]

Print Time$

[Parameter Description]

Time$	Return the controller's system time in hour/minute/second format.

[Usage Example]

```
Function Main

    Print Time$        'Print the controller's system time.

Fend
```

Date

[Function]

Used to print/return the controller's system date.

[Syntax]

Date

Print Date(Number)

[Parameter Description]

Number Specify to obtain the year/month/day of the controller's system date with a value ranging from 0 to 2.

[Usage Example]

```
Function Main
    Date  'Print the controller's system date.

    Integer New_Date
    New_Date = Date (0)  'Return the year of the controller's system date.
    Print New_Date

    New_Date = Date (1)  ' Return the month of the controller's system date.
    Print New_Date

    New_Date = Date (2)  'Return the controller's system date.
    Print New_Date
Fend
```

Date$

[Function]

Used to return the controller's system date in the specified format yyyy:MM:dd.

[Syntax]

Print Date$

[Parameter Description]

Date$	Return the controller's system date in year/month/day format.

[Usage Example]

```
Function Main
    Print Date$  'Print the controller's system date.
Fend
```

Detailed Description of Palletizing Statements

Pallet

[Function]

Used to define/display pallets.

[Syntax]

Define pallets：

Pallet {OutSide,} Pallet Number, Point 1 , Point 2 , Point 3{, Point 4,} Quantity 1, Quantity 2

Print a single pallet:

Pallet Pallet Number

Print all pallets:

Pallet

Get pallet points:

Pallet(Pallet Number, Pallet position number)

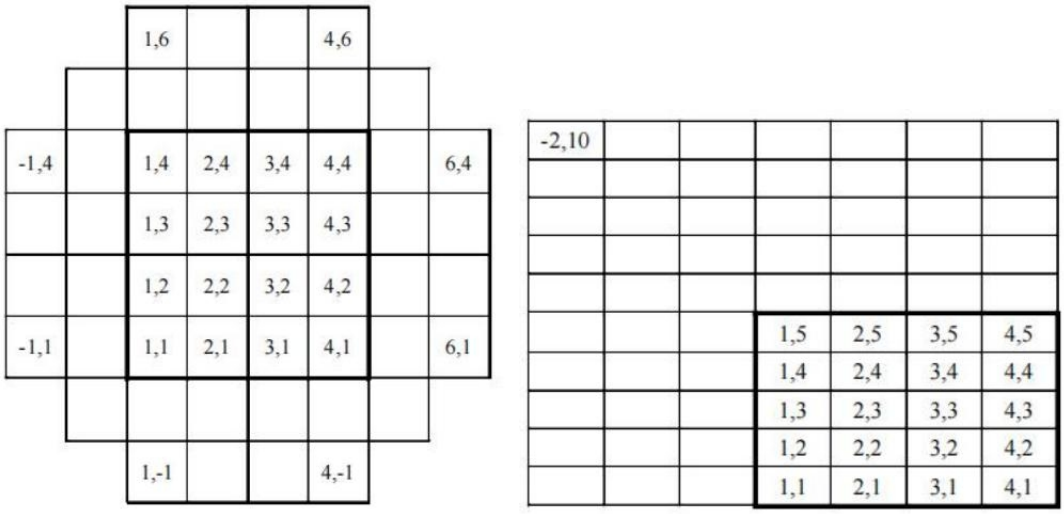
Pallet(Pallet Number, Row coordinate, Column coordinate)

[Parameter Description]

OutSide	Generate additional pallets outside the specified row and column range; this can be omitted.
Pallet Number	Specify the pallet number using an expression or a numerical value, with a range of 0 to 15.
Point 1~3	Standard 3-point method for defining a pallet。
Point 4	The 4-point method can be used to define a pallet, further improving the accuracy of pallet positioning.
Quantity1	Specify the number of points between Point 1 and Point 2 using an integer, with a range of 1 to 32767.
Quantity2	Specify the number of points between Point 1 and Point 3 using an integer, with a range of 1 to 32767.
Row coordinate	Specify the row number of the pallet with an integer.
Column coordinate	Specify the column number of the pallet with an integer.

Precautions:

The effect of "OutSide" is as shown in the figure below: additional points outside the pallet are expanded on the basis of the original pallet.



[Usage Example]

```
Function main
    Integer var

    'Define a 5*3 pallet
    Pallet 1 , P1 , P2 , P3 , 5 , 3

    'Print pallet No. 2
    Pallet 2

    'Print all pallets
    Pallet

    'Loop through pallet points with a for loop.
    For var = 1 To 15
        Go Pallet(1 , var)
    Next

    Go Pallet(1,2,3)  ' Go to the position of the 2nd row and 3rd column of pallet No. 1.

Fend
```

PalletClr

[Function]

Used to clear the set pallet(s).

[Syntax]

PalletClr Pallet Number

[Parameter Description]

Pallet Number Specify the pallet number using an expression or a numerical value, with a range of 0 to 15.

[Usage Example]

```
Function main
    Integer var

    'Define a 5*3 pallet
    Pallet 1 , P1 , P2 , P3 , 5 , 3

    'Clear pallet No. 1
    PalletClr 1

Fend
```

Detailed explanation of the conveyor tracking statement

SyStart

[Function]

Start the conveyor tracking function.

[Syntax]

SyStart　Conveyor number

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.

Precautions:

1. This instruction is used when the robot needs to start tracking materials on the conveyor belt.

[Usage Example]

```
Recrt：
    SyStart Cvt_One  ' Enable belt tracking

If SyGetPose (Cvt_One) > 700 Then
    Print "The materials are insufficient for the processing distance."
    SyEnd Cvt_One 'Disable belt tracking
    GoTo Recry
End If

SyStart Cvt_One        'Enable belt tracking to proceed with the production process.
.
.
.
.
SyEnd Cvt_One          'End belt tracking
```

SyEnd

[Function]

End the conveyor tracking function.

[Syntax]

SyEnd Conveyor number

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.

Precautions:

1. This instruction is used when the robot has finished tracking the conveyor belt and no longer needs to track it.

[Usage Example]

```
Recrt :
    SyStart Cvt_One  ' Enable belt tracking

If SyGetPose (Cvt_One) > 700 Then
    Print "The materials are insufficient for the processing distance."
    SyEnd Cvt_One 'Disable belt tracking
    GoTo Recry
End If

SyStart Cvt_One        'Enable belt tracking to proceed with the production process.
.
.
.
.
SyEnd Cvt_One          'End belt tracking
```

SyReset

[Function]

Clear the current conveyor tracking target queue and reset the conveyor tracking.

[Syntax]

SyReset Conveyor number

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.

[Usage Example]

```
SyReset Cvt_One       'Initialize belt tracking
Recrt :
    SyStart Cvt_One  ' Enable belt tracking

If SyGetPose (Cvt_One) > 700 Then
    Print "The materials are insufficient for the processing distance."
    SyEnd Cvt_One 'Disable belt tracking
    GoTo Recry
End If

SyStart Cvt_One       'Enable belt tracking to proceed with the production process.
.
.
.
.
SyEnd Cvt_One         'End belt tracking
```

SyGetPoint

[Function]

Obtain the point information in the target queue of the specified conveyor belt.

[Syntax]

SyGetPoint (Conveyor number)

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.

[Usage Example]

```
SyMove SyGetPoint (Cvt_One) :X (X1)  :Y (Y1)  :Z (Z1)  :U (RZ1) CP

SyMove SyGetPoint (Cvt_One) :X (X1)  :Y (Y1)  :Z (Z1)  :U (RZ1)

On 0
Wait 1

Off 0
Wait 1
```

SyGetUserData

[Function]

Obtain the additional information of the points in the target queue of the specified conveyor belt.

[Syntax]

SyGetUserData(Conveyor number)

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.

Precautions:

1.This information is added by the SyAddVisTarget instruction.

[Usage Example]

```
Print SyGetUserData (1)
```

SyGetPose

[Function]

Obtain the current position of the specified conveyor target on the conveyor belt, in millimeters.

[Syntax]

SyGetPose(Conveyor number)

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.

[Usage Example]

```
SyReset Cvt_One        'Initialize belt tracking
Recrt :
    SyStart Cvt_One  ' Enable belt tracking

If SyGetPose (Cvt_One) > 700 Then
    Print "The materials are insufficient for the processing distance."
    SyEnd Cvt_One 'Disable belt tracking
    GoTo Recry
End If

SyStart Cvt_One        'Enable belt tracking to proceed with the production process.
.
.
.
.
SyEnd Cvt_One          'End belt tracking
```

SyGetNum

[Function]

Get the number of targets in the queue of the specified conveyor belt.

[Syntax]

SyGetNum（Conveyor number）

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.

[Usage Example]

```
If SyGetNum(1) > 1 Then
    Print "Targets exist in the tracking queue."
End If
```

SyGetQuePose

[Function]

Obtain the position of the point in the queue of the specified conveyor belt on the conveyor belt.

[Syntax]

SyGetQuePose(Conveyor number , queue number)

[Parameter Description]

Conveyor number Specify the conveyor number with an expression or value, in the range of 1 to 4.

queue number Specify the number in the conveyor belt tracking queue with an integer ranging from 1 to 64.

[Usage Example]

```
If SyGetQuePose (1, 1) > 600 Then
    Print "Beyond the tracking range"
    Print SyGetQuePose (1, 1)     'Print the target distance position.
End If
```

[Return Description]

Return data 1: distance value, unit in millimeters

SyMove

[Function]

Conveyor tracking linear interpolation movement.

[Syntax]

SyMove Target coordinate {CP}{Till/Find}{!...!}

[Parameter Description]

Target coordinate	Specify the target position with the point data obtained by the conveyor tracking instruction.
CP	Set motion smoothness, which can be omitted.
Till/Find	Till expression = On/Off. Find expression = On/Off. Optional. Refer to the detailed description of Till/Find for specifics.
!...!	Process I/O and other input/output operations in parallel during motion. Optional. Refer to the detailed description of parallel processing for specifics.

[Usage Example]

```
SyMove SyGetPoint(Cvt_One) :X(X1) :Y(Y1) :Z(Z1) :U(RZ1) CP

SyMove SyGetPoint(Cvt_One) :X(X1) :Y(Y1) :Z(Z1) :U(RZ1)

On 0
Wait 1

Off 0
Wait 1
```

SyArc

[Function]

Conveyor tracking circular interpolation movement.

[Syntax]

SyArc Passing coordinate , Target coordinate {CP}{Till/Find}{!...!}

[Parameter Description]

Passing coordinate Specify the positions that the arc will pass through using the point data obtained by the conveyor tracking instruction.

Target coordinate Specify the target position with the point data acquired by the conveyor tracking instruction.

CP Set motion smoothness, which can be omitted.

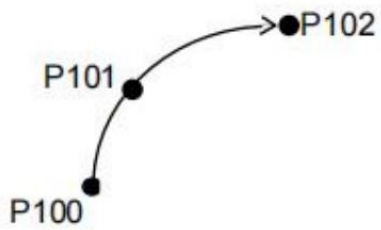
Till/Find Till expression = On/Off. Find expression = On/Off. Optional. Refer to the detailed description of Till/Find for specifics.

!...! Process I/O and other input/output operations in parallel during motion. Optional. Refer to the detailed description of parallel processing for specifics.

[Usage Example]

```
On 0
Wait 1
SyArc SyGetPoint(Cvt_One):X(X1) :Y(Y1) :Z(Z1) :U(RZ1)

On 0
Wait 1
```



SyTrig

[Function]

Record the current encoder pulse value of the conveyor belt.

[Syntax]

SyTrig (Conveyor number)

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.

Precautions:

1. Subsequent SyAddVisTarget instructions will use the encoder pulse value recorded by this instruction.

[Usage Example]

```
SyTrig 1
```

SyPoint

[Function]

Convert ordinary coordinates or points to conveyor tracking coordinates or points.

[Syntax]

SyPoint(Conveyor number, Coordinate X, Coordinate Y, Coordinate U)
SyPoint(Conveyor number, Point)

[Parameter Description]

Conveyor number Specify the conveyor number with an expression or value, in the range of 1 to 4.

Coordinate Specified in millimeters, it refers to the coordinates that need to be converted to conveyor tracking.

Point Specified by expressions, point names, or point numbers, it refers to the coordinates that need to be converted to conveyor tracking.

[Usage Example]

```
SyAddVisTarget 1 , SyPoint (1 , 10 , 20 , 30)
SyMove SyPoint (1 , 5 , 5 , 5)
```

SyAddVisTarget

[Function]

Register the visual photographing results into the conveyor belt tracking queue.

[Syntax]

SyAddVisTarget Conveyor number, Point{, Additional information}

[Parameter Description]

Conveyor number Specify the conveyor number with an expression or value, in the range of 1 to 4.

Point Specify the points to be registered by means of expressions, point names, or point numbers.

Additional information Specify additional information with integers.

[Usage Example]

```
Do
    Input #TCP_Id , temp1$(0) , temp1$(1) , temp1$(2) , temp1$(3)
    Print temp1$(0) ,",", temp1$(1) ,",", temp1$(2) ,",", temp1$(3)
    CCD_X$ = temp1$(1)
    CCD_Y$ = temp1$(2)
    CCD_U$ = temp1$(3)
    If Val(temp1$(0)) = 1 Then
        Print "Registered points"
        SyAddVisTarget Cvt_One , SyPoint(Cvt_One , Val(CCD_X$) , Val(CCD_Y$) , Val(CCD_U$)
    End If
Loop Until ChkNet(TCP_Id) < 0
```

SyTimeOffset

[Function]

Set / Get the tracking time offset.

[Syntax]

SyTimeOffset Conveyor number , Offset Time
SyTimeOffset (Conveyor number)

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.
Offset Time	Set the tracking offset time in milliseconds.

Precautions:

Do not use this function during tracking.

[Usage Example]

```
SyReset Cvt_One        'Initialize belt tracking
Recrt :
    SyTimeOffset Cvt_One , 20
    SyStart Cvt_One  ' Enable belt tracking

If SyGetPose (Cvt_One) > 700 Then
    Print "The materials are insufficient for the processing distance."
    SyEnd Cvt_One 'Disable belt tracking
    GoTo Recry
End If

SyStart Cvt_One        'Enable belt tracking to proceed with the production process.
.
.
.
.
SyEnd Cvt_One          'End belt tracking
```

SyRejectDis

[Function]

Set/obtain the duplicate filtering distance for the conveyor tracking function.

[Syntax]

SyRejectDis Conveyor number , Distance
SyRejectDis (Conveyor number)

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.
Distance	Distinguish material spacing in millimeters.

Precautions:

1. Do not use this function during tracking.

[Usage Example]

```
SyReset Cvt_One       'Initialize belt tracking
Recrt :
    SyStart Cvt_One  ' Enable belt tracking

If SyGetPose (Cvt_One) > 700 Then
    Print "The materials are insufficient for the processing distance."
    SyEnd Cvt_One 'Disable belt tracking
    GoTo Recry
End If

SyStart Cvt_One       'Enable belt tracking to proceed with the production process.
.
.
.
.
SyEnd Cvt_One         'End belt tracking
```

SyStarDis

[Function]

Set / Get the duplicate filtering distance for the conveyor belt tracking function.

[Syntax]

SyStarDis Conveyor number , Start Distance
SyStarDis (Conveyor number)

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.
Start Distance	Set the distance from the camera to the start tracking point in millimeters.

Precautions:

Do not use this function during tracking.

[Usage Example]

```
Recrt :
    SyStarDis Cvt_One , 200
    SyStart Cvt_One  ' Enable belt tracking

If SyGetPose (Cvt_One) > 700 Then
    Print "The materials are insufficient for the processing distance."
    SyEnd Cvt_One 'Disable belt tracking
    GoTo Recry
End If

SyStart Cvt_One        'Enable belt tracking to proceed with the production process.
.
.
.
.
SyEnd Cvt_One          'End belt tracking
```

SyEndDis

[Function]

Set / Get the tracking end distance.

[Syntax]

SyStarDis Conveyor number , End Distance
SyStarDis (Conveyor number)

[Parameter Description]

Conveyor number	Specify the conveyor number with an expression or value, in the range of 1 to 4.
End Distance	Set the distance to the tracking end point in millimeters.

Precautions:

Do not use this function during tracking.

[Usage Example]

```
Recrt :
    SyEndDis Cvt_One , 700
    SyStart Cvt_One  ' Enable belt tracking

If SyGetPose (Cvt_One) > 700 Then
    Print "The materials are insufficient for the processing distance."
    SyEnd Cvt_One 'Disable belt tracking
    GoTo Recry
End If

SyStart Cvt_One       'Enable belt tracking to proceed with the production process.
.
.
.
.
SyEnd Cvt_One         'End belt tracking
```

SySave

[Function]

Save the modifications made to the conveyor tracking function in the program.

[Syntax]

SySave

[Parameter Description]

Precautions:

1.All parameter modifications made to the conveyor tracking function in the program need to be saved using this instruction.

[Usage Example]

```
Recrt:
    SyRejectDis Cvt_One , 20
    SySave

    SyRejectDis Cvt_One

    SyStart Cvt_One '开启皮带跟踪

If SyGetPose (Cvt_One) > 700 Then
    Print "The materials are insufficient for the processing distance."
    SyEnd Cvt_One 'Disable belt tracking
    GoTo Recry
End If

SyStart Cvt_One       'Enable belt tracking to proceed with the production process.
.
.
.
.
SyEnd Cvt_One         'End belt tracking
```

Detailed Description of Mathematical Operation Instruction

+、-、*、/、**、Mod、And、Or、Xor、Not、>、>=、< 、<=、<>、=

[Function]

These include addition, subtraction, multiplication, division, exponentiation, modulus, AND, OR, XOR, NOT, greater than, greater than or equal to, less than, less than or equal to, not equal to, and equal to.

[Syntax]

运算符	格式示例	说明
+	A+B	加法
-	A-B	减法
*	A*B	乘法
/	A/B	除法
**	A**B	乘方
=	A=B	A 等于 B
>	A>B	A 大于 B
<	A<B	A 小于 B
>=	A>=B	A 大于等于 B
<=	A<=B	A 小于等于 B
<>	A<>B	A 不等于 B
And	A And B	逻辑与
Mod	A Mod B	整数的取模
Not	Not A	非
Or	A Or B	逻辑或
Xor	A Xor B	逻辑异或

[Parameter Description]

[Usage Example]

DegToRad

[Function]

Convert angles to radians.

[Syntax]

DegToRad(Angle)

[Parameter Description]

Angle	Specify the angle to be converted to radians by a numerical value, with the unit in degrees.

[Usage Example]

```
Function main
    Double rad_num , cos_num
    rad_num =  DegToRad (30)
    cos_num =  Cos (rad_num)
Fend
```

RadToDeg

[Function]

Convert radians to angles.

[Syntax]

RadToDeg(Radian)

[Parameter Description]

Radian	Specify the radians to be converted to angles by a numerical value.

[Usage Example]

```
Function main
    Double deg_num
    deg_num =  RadToDeg (1.21)
Fend
```

Sin、Cos、Tan、Asin、Acos、Atan、Atan2

[Function]

The sine function, cosine function, tangent function, arcsine function, arccosine function, arctangent function, and arctangent function 2.

[Syntax]

Sin(radian)
Cos(radian)
TaVn(radian)
Asin(numerical value)
Acos(numerical value)
Atan(numerical value)
Atan2(y, x)

[Parameter Description]

radian	Specify the radians for calculation as integers or floating-point numbers.
numerical value	Specify the numerical values for calculation as integers or floating-point numbers.

[Usage Example]

```
Function main
    Double num

    num = Sin(DegToRad(30))
    num = Cos(DegToRad(30))
    num = Tan(DegToRad(30))

    num = Asin(0.5)
    num = Acos(0.5)
    num = Atan(0.5)
    num = Atan2(1,2)
Fend
```

HexToFloat

[Function]

Hexadecimal to single-precision floating-point number, conforming to IEEE 754.

[Syntax]

HexToFloat(numerical value)

[Parameter Description]

numerical value It can be specified as a decimal integer or a hexadecimal number (&H value).

[Usage Example]

```
Function main
    Print HexToFloat(&H40490FDA)          "Output 3.1415926
Fend
```

FloatToHex

[Function]

Single-precision floating-point number to hexadecimal number, conforming to IEEE 754.

[Syntax]

FloatToHex(numerical value)

[Parameter Description]

numerical value It can be specified as a decimal integer or a single-precision floating-point number.

[Usage Example]

```
Function main
    Print Hex$ (FloatToHex (3.1415926))          ''Output 40490FDA
Fend
```

Abs

[Function]

Take the absolute value.

[Syntax]

Abs(numerical value)

[Parameter Description]

numerical value It can be specified as a decimal integer or a floating-point number.

[Usage Example]

```
Function main
    Abs(-1.234)
Fend
```

Sqr

[Function]

Take the square root.

[Syntax]

Sqr(numerical value)

[Parameter Description]

numerical value It can be specified as a decimal integer or a floating-point number.

[Usage Example]

```
Function main
    Sqr (4)
Fend
```

Sgn

[Function]

Return the sign of the numerical value.

[Syntax]

Sgn(numerical value)

[Parameter Description]

numerical value It can be specified as a decimal integer or a floating-point number.

[Usage Example]

```
Function main
    Sgn (4)
Fend
```

LShift

[Function]

Left shift.

[Syntax]

LShift(numerical value, number of bits)

[Parameter Description]

numerical value It can be specified as a decimal integer or a floating-point number.

number of bits Specify the number of bits to shift left as an integer.

[Usage Example]

```
Function main
    LShift (4433 , 3)
Fend
```

RShift

[Function]

Right shift.

[Syntax]

RShift(numerical value, number of bits)

[Parameter Description]

numerical value It can be specified as a decimal integer or a floating-point number.

number of bits Specify the number of bits to shift left as an integer.

[Usage Example]

```
Function main
    RShift (4433 , 3)
Fend
```

BClr

[Function]

Set the bit specified by the value to 0 and return the value.

[Syntax]

BClr(numerical value, number of bits)

[Parameter Description]

numerical value It can be specified as a decimal integer or a floating-point number.

number of bits Specify the bit position as an integer.

[Usage Example]

```
Function main
    BClr (123 , 2)
Fend
```

BSet

[Function]

Set the bit specified by the value to 1 and return the value.

[Syntax]

BSet(numerical value, number of bits)

[Parameter Description]

numerical value It can be specified as a decimal integer or a floating-point number.

number of bits Specify the bit position as an integer.

[Usage Example]

```
Function main
    BSet (123 , 2)
Fend
```

BTst

[Function]

Return the value of the bit specified by the number.

[Syntax]

BTst(numerical value, number of bits)

[Parameter Description]

numerical value It can be specified as a decimal integer or a floating-point number.

number of bits Specify the bit position as an integer.

[Usage Example]

```
Function main
    BTst (123 , 2)
Fend
```

Ubound

[Function]

Return the length of the array.

[Syntax]

Ubound(Array name{, dimension})

[Parameter Description]

Array name	It can be specified as a decimal integer or a floating-point number.
dimension	1 stands for one-dimensional, 2 for two-dimensional, and 3 for three-dimensional. It can be omitted, with the default being one-dimensional.

[Usage Example]

```
Function main
    Integer i , a(10)
    For i = 0 To UBound(a)
        a(i) = i
    Next
Fend
```

Dist

[Function]

Return the straight-line distance between two points.

[Syntax]

Dist(Point 1 , Point 2)

[Parameter Description]

Point 1 , Point 2	Specify the point data to be compared by the point table number.

[Usage Example]

```
Function main
        Print Dist (P1 , P2)        ' Check the straight-line distance from point P1 to point P2.
Fend
```

Detailed Description of String Operation Instructions

ParseStr

[Function]

String splitting.

[Syntax]

ParseStr String, string array$(), delimiter$
Length of the string array = ParseStr(String, string array$(), delimiter$)

[Parameter Description]

String	The string that needs to be split.
String array$()	Used to receive the strings split by the delimiter.
Delimiter$	The string is split according to this delimiter.
Length of the string array	The length of the array obtained after splitting the string.

[Usage Example]

```
Function main
    Str ing myStr$ , strArr$ (0)
     Integer i
    myStr$ = "1$2$3$4"
    ParseStr myStr$ , strArr$ () , "$"
    For i = 0 To UBound (strArr$ ())
        Print "Array content = " , strArr$ (i)
    Next

Fend
```

+

[Function]

String concatenation.

[Syntax]

string + string

[Parameter Description]

string The string to be used for concatenation.

[Usage Example]

```
Function main
    String myStr$
    myStr$ = "Robot" + "OS"
Fend
```

<>、=

[Function]

Not equal to, equal to, assignment.

[Syntax]

string <> string

string = string

String variable = string

[Parameter Description]

string The string to be used for the operation.

[Usage Example]

```
Function main
    String myStr$
    myStr$ = "stop"
     IF myStr$ = "Run" Then
        Print "Run"
    ElseIF myStr$ <> "Run" Then
        Print "stop"
    End IF
Fend
```

Val

[Function]

String to numeric conversion.

[Syntax]

Val(String)

[Parameter Description]

String	The string to be used for conversion.

[Usage Example]

```
Function main
    String myStr$
    myStr$ = "3.1415926"

    Double num
    num = Val(myStr$)
Fend
```

Str$

[Function]

Numeric to string conversion.

[Syntax]

Str$(numeric value)

[Parameter Description]

numeric value	The numeric value to be used for conversion.

[Usage Example]

```
Function main
    String myStr$
    Double num
    num    = Val(myStr$)
    myStr$ = Str$(num)
Fend
```

Len

[Function]

Get the length of the string.

[Syntax]

Len(String)

[Parameter Description]

String	The string to be used for conversion.

[Usage Example]

```
Function main
    String myStr$
    Print Len (myStr$)
Fend
```

Asc

[Function]

Convert string to ASCII code.

[Syntax]

Asc(String)

[Parameter Description]

String	The string to be used for conversion.

[Usage Example]

```
Function Main
    Integer AA1 , BB1 , CC1
    AA1 = Asc ("a")
    BB1 = Asc ("b")
    CC1 = Asc ("c")
    Print "The ASCII Value of AA1 is", AA1
    Print "The ASCII Value of BB1 is", BB1
    Print "The ASCII Value of CC1 is", CC1
Fend
```

Chr$

[Function]

Convert ASCII code to string.

[Syntax]

Chr$(ASCII code)

[Parameter Description]

ASCII code	Specified ASCII code from 1 to 255.

[Usage Example]

```
Function Main
    String Test$
    Test$ = Chr$ (&H41) + Chr$ (&H42) + Chr$ (&H43)
    Print "The value of Test= " , Test$
Fend
```

Left$

[Function]

Extract the specified string starting from the left side of the string.

[Syntax]

Left$(String, Extract the quantity)

[Parameter Description]

String	The string to be extracted.
Extract the quantity	The number of characters to be extracted from the left side of the string.

[Usage Example]

```
Function Main
    Print Left$ ("ABCDEFG" , 2)
    '>>> AB
    Print Left$ ("ABCDEFG" , 3)
    '>>> ABC
Fend
```

Mid$

[Function]

Extract the specified string starting from the specified range of the string.

[Syntax]

Mid$(String, Starting position{, Extract the quantity})

[Parameter Description]

String	The string to be extracted.
Starting position	The starting position of the extracted string.
Extract the quantity	If omitted, extract from the starting position to the end.

[Usage Example]

```
Function main
    Print Mid$ ("123456" , 3 , 3)
Fend
```

Right$

[Function]

Extract the specified string starting from the right side of the string.

[Syntax]

Right$(String, Extract the quantity)

[Parameter Description]

String	The string to be extracted.
Extract the quantity	The number of characters to be extracted from the right side of the string.

[Usage Example]

```
Function Main
    Print Right$ ("ABCDEFG" , 2)
    '>>> FG
    Print Right$ ("ABC" , 3)
    '>>> ABC
Fend
```

LSet$

[Function]

Extract a string of specified length starting from the left side of the string, and pad with spaces if the string length is insufficient.

[Syntax]

LSet$(String, Extract the quantity)

[Parameter Description]

String The string to be extracted.

Extract the quantity The number of characters to be extracted from the left side of the string.

[Usage Example]

```
Function Main
    String temp$
    temp$ = "123"
    temp$ = LSet$ (temp$ , 10)          'temp$ = "123"
Fend
```

RSet$

[Function]

Extract a string of specified length starting from the right side of the string, and pad with spaces if the string length is insufficient.

[Syntax]

RSet$(String, Extract the quantity)

[Parameter Description]

String The string to be extracted.

Extract the quantity The number of characters to be extracted from the right side of the string.

[Usage Example]

```
Function Main
    String temp$
    temp$ = "123"
    temp$ = RSet$ (temp$ , 10)          'temp$ = "123"
Fend
```

Space$

[Function]

Return a string of specified number of spaces.

[Syntax]

Space$(number of spaces)

[Parameter Description]

number of spaces	The number of spaces returned.

[Usage Example]

```
Function Main
    Print "XYZ" + Space$ (1) + "ABC"
    '>>>XYZ ABC
    Print Space$ (3) + "ABC"
    '>>>ABC
Fend
```

LCase$

[Function]

Return a string of lowercase letters.

[Syntax]

LCase$(String)

[Parameter Description]

String	The string to be used for conversion.

[Usage Example]

```
Function Main
    String Test$
    Test$ = "Data"
    Test$ = LCase$(Test$)          'Test$ = "Data"
Fend
```

UCase$

[Function]

Return a string of uppercase letters.

[Syntax]

UCase$(String)

[Parameter Description]

String The string to be used for conversion.

[Usage Example]

```
Function Main
    String Test$
    Test$ = "Data"
    Test$ = UCase$(Test$)          'Test$ = "Data"
Fend
```

LTrim$

[Function]

Remove the spaces on the left side of the string and return it.

[Syntax]

LTrim$(String)

[Parameter Description]

String The string to be used for conversion.

[Usage Example]

```
Function Main
    String Test$
    Test$ = "    Data    "
    Test$ = LTrim$ (Test$)          'Test$ = "Data    "
Fend
```

RTrim$

[Function]

Used to remove spaces on the right side of the string and return it.

[Syntax]

RTrim$(String)

[Parameter Description]

String	The string to be used for conversion.

[Usage Example]

```
Function Main
    String Test$
    Test$ = "    Data    "
    Test$ = RTrim$ (Test$)              'Test$ = "    Data"
Fend
```

Trim$

[Function]

Used to remove spaces on both sides of the string and return it.

[Syntax]

Trim$(String)

[Parameter Description]

String	The string to be used for conversion.

[Usage Example]

```
Function Main
    String Test$
    Test$ = "    Data    "
    Test$ = Trim$ (Test$)               'Test$ = "Data"
Fend
```

InStr

[Function]

Retrieve the position of the specified character in the string and return it.

[Syntax]

InStr(String, retrieve a string)

[Parameter Description]

String The string to be used for retrieval.

retrieve a string The character to be retrieved.

[Usage Example]

```
Function main
    Print InStr("ABCDEF", "C")
Fend
```

Tab$

[Function]

Used to return a string of specified number of tab characters.

[Syntax]

Tab$(number of tab characters)

[Parameter Description]

number of tab characters The specified number of tab characters to return.

[Usage Example]

```
Function Main
    Integer i
    Print "X", Tab$(1), "Y"
    For i = 1 To 10
        Print x(i), Tab$(1), y(i)
    Next i
Fend
```

Hex$

[Function]

Used to convert a hexadecimal number to a string.

[Syntax]

Hex$(numerical value)

[Parameter Description]

numerical value The numerical value to be converted to a string.

[Usage Example]

```
Function Main
    Integer stat
    stat = 10
    Print Hex$(stat)
    '>>>A
    Print  Hex$(255)
    '>>>FF
Fend
```

Detailed explanation of bitwise operation instructions

+、-

[Function]

Relative coordinate incremental movement.

[Syntax]

Motion command: Point + Component

Motion command: Point - Component

[Parameter Description]

Component	Direction component.

[Usage Example]

```
Function main
    Go P1 + X (10)
    Move P2 - U (5)
Fend
```

/

[Function]

Change the hand coordinate system, change the user coordinate system.

[Syntax]

Motion command /User coordinate system number

Motion command /[R/L]

[Parameter Description]

User coordinate system number	Specify the user coordinate system number to switch to, with a range of 1 to 64.
R/L	Specify the hand coordinate system to switch to: R for right hand, L for left hand.

[Usage Example]

```
Function main
    Go P1 + X (10) /R        'Move to the position 5mm in the X+ direction from point P1
                              in the right-hand coordinate system.
    Go P1 + X (10) /3        'Move to the position 5mm in the X+ direction from point P1
                             in user coordinate system 3.
Fend
```

:

[Function]

Absolute coordinate movement.

[Syntax]

Motion command Position : Component

[Parameter Description]

Component	Direction component.

[Usage Example]

```
Function main
    Go P1  :X (10)      'Move to the position of point P1 with X=10mm.
Fend
```

=

[Function]

Assignment

[Syntax]

Point = Instruction

[Parameter Description]

[Usage Example]

```
Function main
    P1 = XY (10 , 20 , 30 ,40)
Fend
```

@

[Function]

Switch to the specified user coordinate system.

[Syntax]

Motion command @User coordinate system number

[Parameter Description]

User coordinate system number	Specify the user coordinate system number to switch to, with a range of 1 to 64.

[Usage Example]

```
Function main
    GO XY (10 , 20 , 30 ,40)  @2
Fend
```

X、Y、Z、U、V、W

[Function]

Direction component.

[Syntax]

+/-/:Direction component.

[Parameter Description]

[Usage Example]

```
Function main
    Go P1 + X (10)
    Move P2 - U (5)
    Go P1 +X (5) /R          'Move to the position 5mm in the X+ direction from point P1 in the right-hand coordinate system.
    Go P1 +X (5) /3          'Move to the position 5mm in the X+ direction from point P1 in user coordinate system 3.
    Go P1 :X (10)            'Move to the position of point P1 with X=10mm.
Fend
```

RX、RY、RZ

[Function]

Specify the rotation components of the user coordinate system around the X, Y, and Z axes.

[Syntax]

+/-[RX/RY/RZ]

[Parameter Description]

[Usage Example]

```
Function main
      Go P1 + X (10) + RX (10)
    'Rotate the user coordinates of point P1 around the X-axis by 10°, and move to the position 5mm in the X+ direction from point P1.
Fend
```

TLX、TLY、TLZ、TLU、TLV、TLW

[Function]

Specify the rotation components of the tool coordinate system around the X, Y, Z, U, V, and W axes.

[Syntax]

+/-[TLX/TLY/TLZ/TLU/TLV/TLW]

[Parameter Description]

[Usage Example]

```
Function main
        Go P1 + X (10) + TLX (10)
    'Rotate the user coordinates of point P1 around the X-axis by 10° , and move to the position 5mm in the X+ direction from point P1.
Fend
```

SavePoints

[Function]

Save the point data in the current program memory to a point file. If the file does not exist, a new file will be created.

[Syntax]

SavePoints filename

[Parameter Description]

filename Name the point file with a string.

[Usage Example]

```
Function main
    P1 = XY (10 , 20 , 30 , 40)
    P2 = XY (40 , 30 , 20 , 10)
    SavePoints "ABCD"      'Save P1 and P2 to the ABCD point file.
Fend
```

LoadPoints

[Function]

Load the point file.

[Syntax]

LoadPoints filename {, Merge}

[Parameter Description]

filename	Name the point file with a string.
Merge	Set this when you do not want to clear the current points before reading in new points. If set, new points will be added to the existing points. When the points to be added already exist in the file, they will be overwritten, and this can be omitted.

[Usage Example]

```
Function main

    'Read the universal points into the current robot.
    LoadPoints "R1Common.pts"

    'Merge the points of component model 1.
    LoadPoints "R1Model1.pts", Merge

    'Read in the point file of Robot 2.
    LoadPoints "R2Model1.pts"

Fend
```

ClearPoints

[Function]

Used to clear the point data stored in the program memory.

[Syntax]

ClearPoints

[Parameter Description]

[Usage Example]

```
Function main
    P1 = XY (10 , 20 , 30 , 40)
    P2 = XY (40 , 30 , 20 , 10)
    ClearPoints        'The previously defined points P1 and P2 will no longer exist.
Fend
```