08 Macro Script

Last modified by Hunter on 2023/02/21 10:15

This chapter provides information about scripts in PIStudio.

Script type

Script is applied to realize complex control functions. HMI compile software provide powerful function, simple operation, reliable script system, the features of script are list as follow:

Similar to BASIC grammatical structure;

- BASIC work as the first computer language for the general public, it is easy and efficient to use.

Support all of program logic control structures;

- Software script supports three logic control structures: order, condition, loops. It could realize complexity procedures.

Powerful function; Functions of script are divided into two types: system and custom function.

- System function: the functions that system has been predefined for users.
- Custom function: users could define a function and apply to all scripts.

Support variety of data format;

- Script supports integer, floating, BCD code, byte, string and etc.

Scripts Running Method

1. **Background Script:** Run independently during start project, screen updates have no influence and valid of all scripts.
2. **Screen Script:** Only run under the designated screen. Screen script start running until screen is closed or switched.

And both screen and background have four modes for script

Property	Description

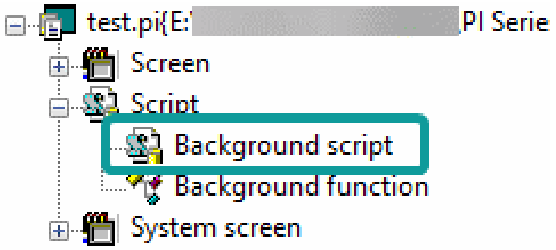
Initialize	The script would be executed once during loading project.
Close	The script would be run once during closing HMI project.
Timing	The script would run under certain conditions after the HMI is started, until the condition ends.
Bit trigger	Script would be repeat executed when meet the condition of bit trigger.

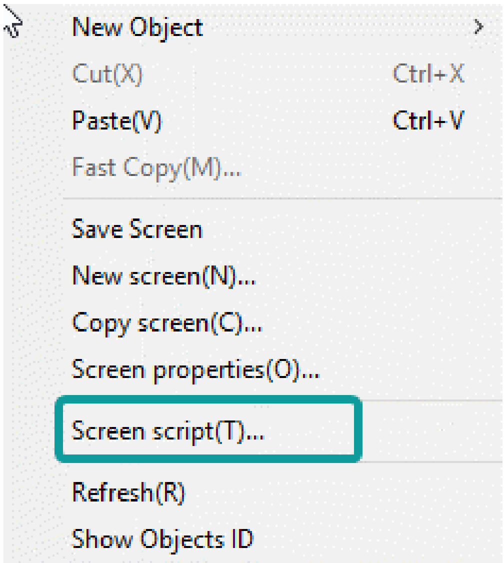
Initialize

Initialize script divided into screen initialize script and background initialize script. Screen initialize script runs once during the initialization of screen; background initialize script runs during the loading of project.

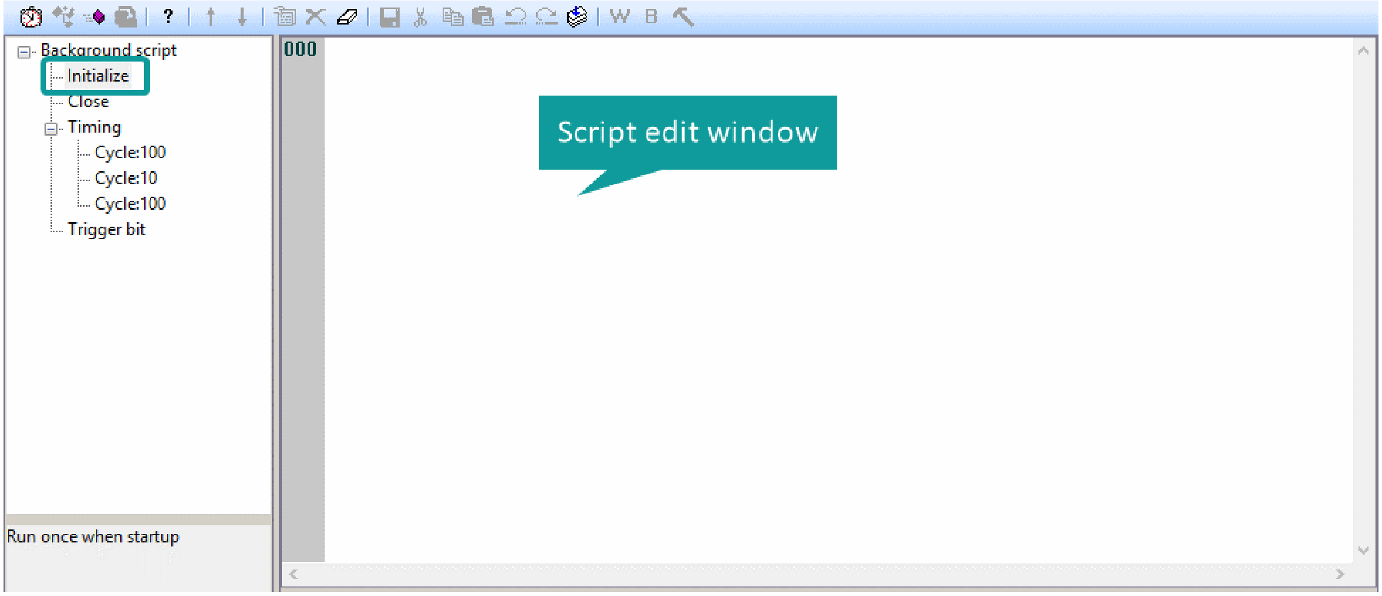
Operating procedures

Click "Background script" in project manager to enter script editor screen, or click "Screen script" in right click menu of screen to enter script editor screen;





Double click "initialize" to open script edit window, as below shows;



Enter scripts in edit window;

Close

Close script divided into screen close script and background close script. Screen close script runs once During the destroying of screen because of closing or switching; background close script runs during the closing of project (such as restart HMI, into HMI setup).

Operating procedures

1. Click "Background script" in project manager to enter script editor screen, or click "Screen script" in right click menu of screen to enter script editor screen;
2. Double click "Close" to open script edit window;
3. Enter scripts in edit window;

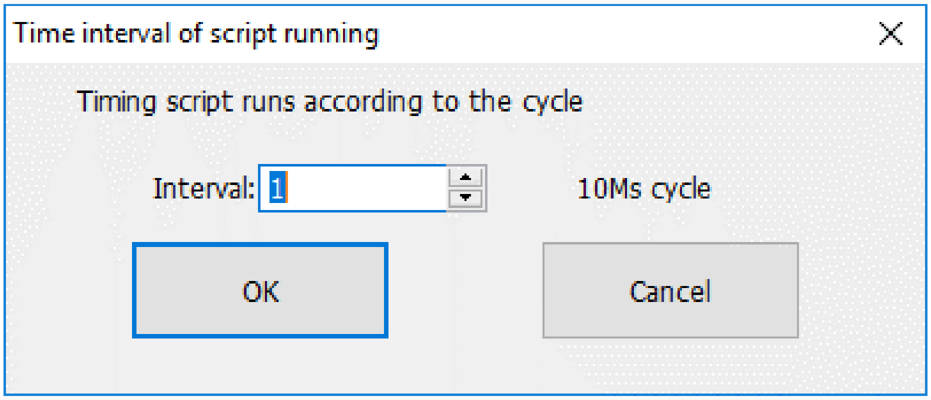
Timing

The script would run for a designated time interval.

Operating procedures of creating one

Click "Background script" in project manager to enter script editor screen, or click "Screen script" in right click menu of screen to enter script editor screen;

Double click "Timing", it would pop-up below setting window;


Time interval of script running

Timing script runs according to the cycle

Interval: 1 10Ms cycle

Property	Description
Cycle	Script runs at designated time interval, unit is 10 ms.
Ok	Script created.
Cancel	Cancel the current script setting.

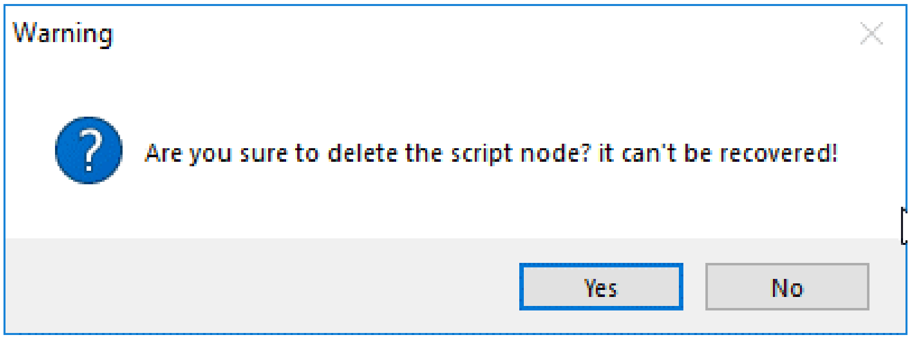
Enter scripts in edit window;

Operating procedures of editing

1. Click "Background script" in project manager to enter script editor screen, or click "Screen script" in right click menu of screen to enter script editor screen;
2. Select "Timing", and click to modify the script execution interval;
3. Double click selected "Timing" to open editing window;

Operating procedures of deleting

Click "Background script" in project manager to enter script editor screen, or click "Screen script" in right click menu of screen to enter script editor screen;


Warning

Are you sure to delete the script node? it can't be recovered!

Select "Timing", and click to change interval of script, it pops-up above window

Select "Yes" to execute operation or select "No" to cancel operation;

✎**Note:** The maximum number of timing script for each screen or background is 32.

Trigger bit

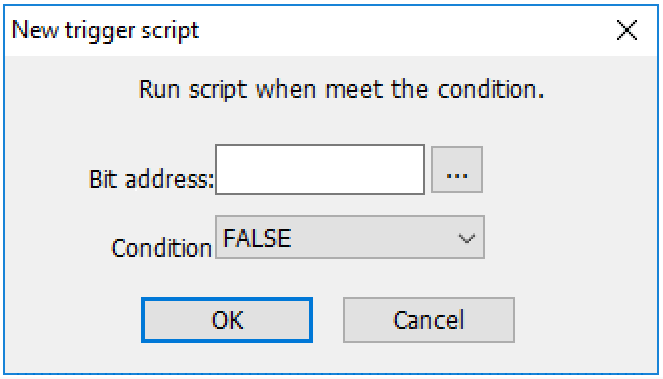
Introduction

Trigger control script is that software will check whether the designated bit meet trigger condition every 20ms. Script execute once condition is met until project closed.

Operating procedures of creating one

Click "Background script" in project manager to enter script editor screen, or click "Screen script" in right click menu of screen to enter script editor screen;

Double click "Trigger bit", it pops-up below setting window;


New trigger script

Run script when meet the condition.

Bit address:

Condition: FALSE

- Bit address: It sets trigger address for script;
- Condition: detailed information as below shows;

Condition	Description
TRUE	Script execute once the bit value is TRUE, detects the bit every per approximately 20 ms;
FALSE	Script execute once the bit value is FALSE, detects the bit every per approximately 20 ms;
Bit changed	Execute once the trigger bit switches state;
Rising	Script execute once the bit value from FALSE to TRUE, detects the bit every per approximately 20 ms;

Falling	Script execute once the bit value from TURE to FALSE, detects the bit every per approximately 20 ms;

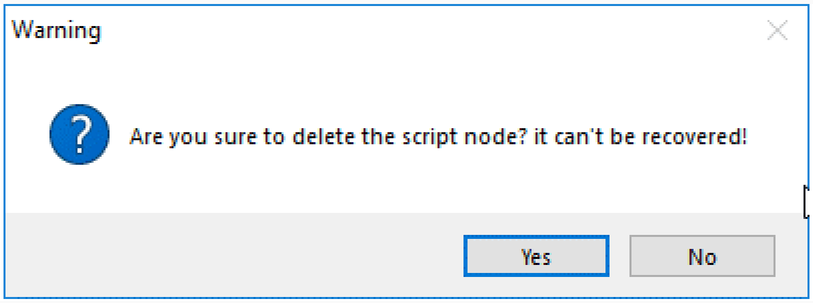
Set trigger bit and condition, click "OK" to open editing window.

Operating procedures of editing

- Click "Background script" in project manager to enter script editor screen, or click "Screen script" in right click menu of screen to enter script editor screen;
- Select "Trigger script", and click to change trigger bit and condition;
- Double click selected "Trigger script" to open editing window.

Operating procedures of deleting

- Click "Background script" in project manager to enter script editor screen, or click "Screen script" in right click menu of screen to enter script editor screen;
- Select "Trigger script", and click to change interval of script, it pops-up below window



Warning

Are you sure to delete the script node? it can't be recovered!

- Select "Yes" to execute operation or select "No" to cancel operation;

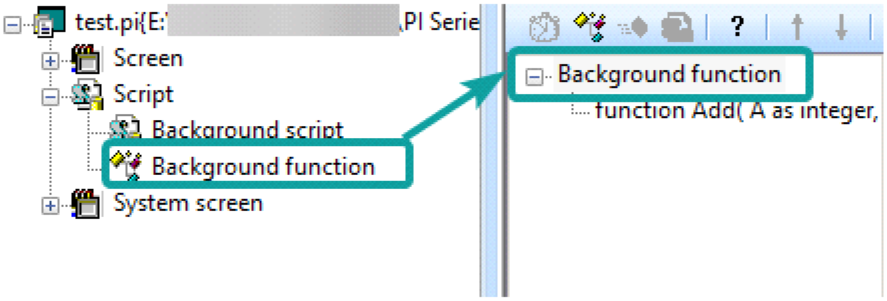
✎**Note:** The maximum number of trigger script for each screen or background is 32.

Background function

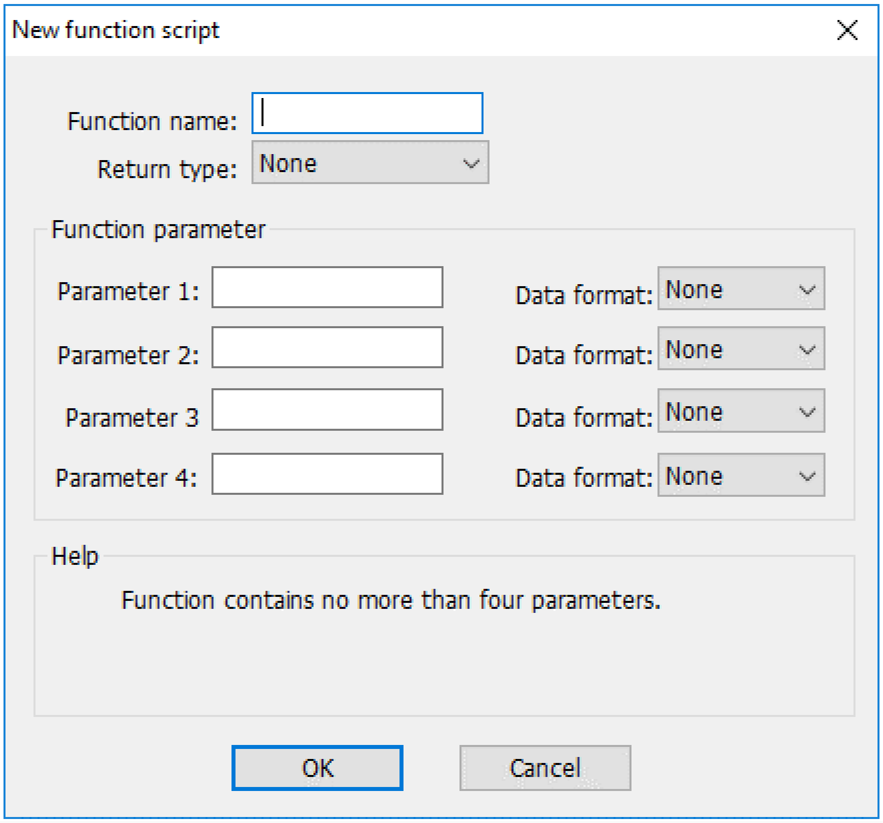
Global function is a collection of codes;it can be called in any script.The method reference system function;

Operating procedures

Double click "Background function" in project manager;



Set parameters;



New function script

Function name:

Return type: None

Function parameter

Parameter 1: Data format: None

Parameter 2: Data format: None

Parameter 3 Data format: None

Parameter 4: Data format: None

Help

Function contains no more than four parameters.

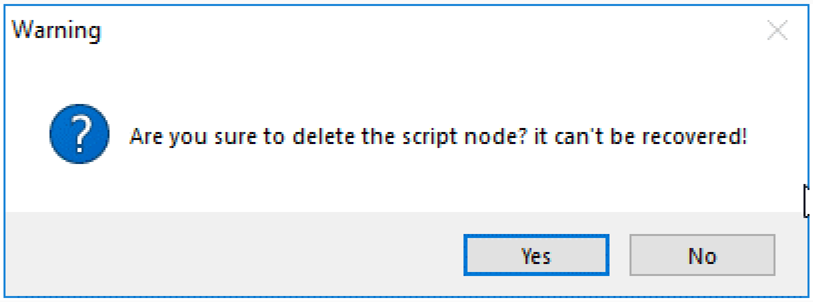
Property	**Description**
Function name	Function name could not be the same as existing.
Return type	None, string, integer, float.
Parameter 1	The name of parameter 1.

Operating procedures of editing

- Click "Background function" in project manager to enter script editor screen;
- Select Function name, and click to change parameters;
- Double click selected "Trigger script" to open editing window;

Operating procedures of deleting

- Click "Background function" in project manager to enter script editor screen;
- Select Function name, and click to change interval of script, it pops-up below window



Warning

Are you sure to delete the script node? it can't be recovered!

- Select "Yes" to execute operation or select "No" to cancel operation;

✎**Note:** The maximum parameters for each function are 4, and parameter name can't be unique;

Script usage

Script can make project more convenient and flexible. Script is useful in realizing complex HMI function. If the script is used improperly, it may affect the efficiency of entire project. So pay attention to the follow issues:

- Do not use too much script loops, if the script loops that execute too many times; it might influence the efficiency of HMI.
- In the cycle scripts, avoid using external register, due to the relatively slow serial communication, frequent access to external registers may cause the execution of scripts severely reduced, even influence the screen respond efficiently. There is fine to use internal register.
- The maximum script length is 512 rows.

This section describes how to edit scripts and use some of the accompanying tools and features of the script editor.

Script access to d evice

Software script supports an efficiency way to access the device address by using symbol @.

Writing	Meaning	Examples
@B_;@b_;	Access designated bit address	@B_I0.0:access bit address I0.0 @b_HDX0.0:access bit address HDX0.0
@W_;@w_;	Access designated word address HMI connect more than automatic control devices, "#"stands for choosing number before the symbol,":"stands for accessing the station number before symbol.	@W_IW0:access word address IW0 @b_HDW0:access word address HDW0
@B_(the number of protocol		@B_2#2:I0.0:access the bit address I0.0, with the

connection)#(station number):address	Access the first protocol without "#",access default station number1 without":".	connection number 2 and station number 2; @B_I0.0:access to bit address I0.0;

The script can access with the device though: write and read.

For example

```
If @B_HDX0.0 = 1 then 'read the value from address HDX0.0.

@B_HDX0.0 = 0 'write 0 to address HDX0.0

Else

@B_HDX0.0 = 1 'write 1 to address HDX0.0

Endif

@W_QW0 = @W_QW0 + 1 'read data from address QW0, add 1 to this value then write to address QW0
```

Grammar checking

Operating procedures

1. Select from script tool bar;
2. System does not prompt grammar error if grammar is correct, or system will list all errors;
3. Check error information, and modify errors;

Error information

1. Identifier *** contains invalid characters

2. Attempt to redeclare sub ***

3. Attempt to redeclare function ***

4. Attempt to use reserved word *** as identifier

5. Attempt to use type *** as identifier

6. Unexpected ')' while parsing arguments for function ***

7. Could not parse expression (one of the arguments of function ***)

8. Could not parse arguments of ***

9. Too many arguments for function ***

10. Not enough arguments for function ***

11. '(' expected after sub name ***

12. Unexpected '(' while parsing arguments for sub ***

13. Could not parse expression (one of the arguments of sub ***)
14. Could not parse arguments of ***
15. Too many arguments for sub/function ***
16. Not enough arguments for sub/function ***
17. Could not parse expression
18. '(' expected after function name ***
19. Unexpected use of sub *** as a part of expression
20. Illegal statements preceding subs/functions declaration
21. Unexpected end of file while looking for 'endsub'
22. End of line expected after 'else'.
23. End of line expected after 'endif'.
24. End of line expected after 'next'.
25. End of line expected after 'wend'.
26. 'while', 'until' or end of line expected afte r'do'.
27. Could not parse expression after 'while'
28. Could not parse expression after 'until'
29. 'do' without 'loop'
30. Sub *** contains invalid character'@'
31. Sub *** already declared
32. Function *** already declared
33. Sub name expected after 'sub'
34. Function name expected after 'function'
35. Variable name expected
36. Argument *** contains invalid character '@'
37. 'integer', 'floating' or 'string' expected
38. '",' or ')' expected
39. 'endsub' without 'sub'
40. 'end function' without 'function'
41. End of line expected after 'beep'
42. 'dim' unexpected here
43. Variable name expected after 'dim'
44. 'as' expected after variable name
45. 'integer' 'floating' or 'string' expected after 'as'

46. ',' or end of line expected after type in dim statement

47. Could not parse expression after 'while'

48. End of line expected after' while' condition

49. 'while' without 'wend'

50. End of line expected after 'wend'

51. 'wend' without 'while'

52. Variable name expected after' for'

53. '=' expected after variable name

54. Could not parse expression after 'for'

55. 'to' expected

56. Could not parse expression after 'to'

57. Could not parse expression after 'step'

58. End of line expected

59. 'for' without 'next'

60. End of line expected after 'next'

61. 'Next' without 'for'

62. Could not parse expression after'if'

63. 'then' expected

64. Unexpected end of file while looking for 'endif'

65. Unexpected end of file while looking for 'else' or 'endif'

66. 'else' without ‘if'

67. 'end if' without 'if'

68. Label name expected after 'goto'

69. Unexpected end of line while looking for ')' in function call

70. ',' expected

71. Missing ')'

72. Unexpected end of line in expression

73. Unexpected end of file in expression

Script function list

Mathematical

Function	Introduction
Abs	Get absolute value
ACos	Compute the inverse cosine value
Asc	Return the first character of the string in ASCII value
ASin	Calculate the arcsine value
ATan	Return an arctangent value,the radian ranges -pi/2 to pi/2
ATan2	Return the arctangent value
Cos	Return a cosine value of an angle
Exp	
Hypot	Calculate the value of the hypotenuse of a right triangle
Tan	Implement tan calculation to computing the value
Sin	Implement sin calculation to computing the value
Sqr	Assign a square root value
SignedInt16	Assign the value to [val] from address A1 which is signed integer
SignedInt32	Assign the value to [val] from address A1 which is signed even integer

Data migration

Function	Introduction
BMOV	Copy data with a designated length from source address
FILL	Write the same value to designated address constantly
SetKeyMap	The key values of the keyboard are mapped so that multiple keyboard buttons perform the same function

Process control

Function	Introduction
Goto	Go to the designated row unconditionally in a function body
FOR	Execute a command repeatedly for designated times
End	End the execution of script
while	If the condition is true, then all the commands before Wend in the statement will be executed and then recheck the condition, if the condition is false, the command after Wend will be executed

Data conversion

Function	Introduction
A2H	Convert string A1 to hexadecimal number
AsFloating	Convert parameter to a float
AsInteger	Convert parameter to a integer

AsString	Convert parameter to a string
B2W	Convert an array
BCD	Convert binary to BCD, save the result as return value
BIN	Convert BCDto binary, save the result in return value
Chr	Convert integer parameter into correspond ASCII character,return the character string
D2F	Convert the 32 bit integer format data to float then output the result
D2Float	Convert the designated value to floating then assign to variable
D2Int	Output the 32-bit integer in the form of integer
DegToRad	Convert the angle into correspond radian, and display
F2D	Convert a 32 bit floating to integer format, and then output the result
F2S	Output a format of floating that in the type of string
Float2D	Copy floating value to the address
H2A	of ASCII
Int2D	Write the 32-bit integer into the specified address
Lcase	Convert all parameters to lowercase strings

MAX	Compare the value of A2 and A3, assign the greater number to A1
MIN	Compare the value of A2 and A3, assign the smaller number to A1
RadToDeg	Convert radiant value to degree
S2F	Convert string to floating in the specified format
SWAP	Exchange the high and low bytes of the parameter
W2B	Combine the high byte of the value of two consecutive addresses into a new value
W2D	Convert the unsigned Word to unsigned Dword and save the result
W2F	Convert a 16bit integer to a 32bit floating, and then save to the next word address
W2S	Convert integer word text as S1 format, and then save

String processing

Function	Introduction
A2I	A string of length is intercepted from A1 and converted to a single/double word integer, and then this integer is assigned to A2
InStr	Return the position of str1 in str2
Left	Return a string of the specified length from the left side of parameter
Len	Return the string length

LTrim	Remove the left empty part of the string and return
Mid	Returns a string contain a specified characters length from a string
Right	Return a string of the specified length from the right side of parameter
RTrim	Clear the empty part on the right side of string [str], then assign the empty part to val
Trim	Return a value of an address without empty string next to it
UCase	Capitalize the string data, and then assign the value to val

Fixed constant

Function	Introduction
Pi	pi = 3.14159265358979321
True	TRUE = 1
False	FALSE = 0
Operator	Operator in scripts edit window
Variable	A variable is any factor, trait, or condition that can exist in differing amounts or types

Bit control

Function	Introduction

Clrb	Set the bit of A1 as FALSE
InvB	The state of inverse bit
SetB	Set the bit A1 ON

File operation

Function	Introduction
HmiRegoperator	Data of Upload/ Download address
CopyFile	Copy the A3 file from the A1 directory to the A2 directory according to the format of A4 and A5
DbToCSVFile	file to csv format and export it
FileCmpDir	The filename input and the filename in the folder whether is duplicate or not
MyDeleteFile	Delete the designated file
WriteWordToFile	Write data with designated length to designated file
ReadWordFormFile	Read data with designated length from designated file

Comparison

Function		Introduction
IF... THEN GOTO...	IF=	Execute correspond instruction when fulfill condition. Condition will be tested during executing if. it will

	IF<>	execute the next instruction block of then, if condition is true. Otherwise, execute the later of else. Complete the two instructions, next execute the later of End if.
	IF>	
	IF>=	
	IF<	
	IF<=	
	IF AND=0	
	IF AND<>0	
IF	IF=	
	IF<>	
	IF>	
	IF>=	
	IF<	
	IF<=	
	IF AND=0	
	IF AND<>0	
ELSE		

ENDIF	

Application type

Function	Introduction
AddrStringCompare	Compare the designated length of two character strings
Beep	Enable buzzer
IsFloating	whether a parameter is floating, return TRUE if it is floating, otherwise return FALSE
IsInteger	whether a parameter(A1) is integer, return TRUE if the parameter is integer, otherwise return FALSE
Log	Log function: return the natural logarithm of the value
Log10	Log function: return the natural logarithm of the value
MSeconds	Display the current microseconds of system
NewNoAddr	At the basic of source address A2,offset designated length,obtain a new address A1
NewStatAddr	At the basic of source address A2,offset the designated length,to obtain a new station A1.
NStringCompare	Compare whether the designated length of two strings is the same,return 1 to A1 if yes,otherwise return 0
Power	The value of [expr2] to the power of [expr1] will be assigned to Var

RAND	Generate a random number
ReadAddr	Assign the value in the specified address to [word]
SleepA	Wait time T(ms)
WriteAddr	Assign the value from A2 to address A1

Others

Function	Introduction
DIM...AS...	Declare a variable
do	Condition determent instruction
Function	Differ from internal function, need to declare the name, parameter, code of the function
Sub	Declare the name, parameters and codes of the Sub (sub function)
PrintText	Output the contents to the printer for printing
PI_GetTickCount	Write the starting time in the set address as a 32-bit integer
StAndFtChange	Calculate the number of seconds from January 1, 1970 to the current time
GetServerDelayInfo	Convert string to hexadecimal number

Function description

A2H

Function

Val = A2H(A1)

Description

Convert string in the specified address to a hexadecimal number;

Parameters

- **A1:** Source data, only convert the first four digits of string
- **Val:** The value is hexadecimal number.

Example

```
@W_HDW20=A2H(@W_HDW10) ' convert the string of HDW10 to hex then save in HDW20.
```

Input: @W_HDW10=255

Result: @W_HDW20=255

✎Note:

- A1 needs to be address (such as:@W_HDW000002).
- Only [0~1], [a~f] can be converted, others will be converted to 0

A2I

Function

A2I (A1, A2, length, S)

Description

Intercept a string of the specified length from A1,convert it to a single/double word integer and assign this integer to A2.

Parameters

- **A1:** String to be intercepted
- **A2:** The final integer value
- **Length:** intercepts the length of the string
- **S:** Control single or double words.
    - S=0, indicating an integer single word; S=1, indicating an integer double word.

Return value: none

Example

```
A2I("@W_HDW200","@W_HDW100",4,0) 'converts a string into a 16-bit (single word) decimal integer

A2I("@W_HDW600","@W_HDW500",4,1)' converts a string to a 32-bit (double word) decimal integer
```

Input: @W_HDW200="12345", **Result:** @W_HDW100=1234

Input: @W_HDW600="12345", **Result:** @W_HDW500=1234

✎**Note:** A1 and A2 need to be address (such as:@W_HDW000002).

Abs

Function

val = Abs(A1)

Description

The absolute value of A1.

Parameters

- **A1:** the data of absolute value, need to be variable.
- **Val:** it is absolute value that is address or variable.

Example

```
Dim a as integer 'a is defined as integer

a = SignedInt16("@W_HDW0")'convert the data of @W_HDW0 into signed data.

@W_HDW1 = Abs(a) 'assign the returned absolute value to @W_HDW1
```

Input: @W_HDW0=-6

Result: @W_HDW1=6

✎**Note: SignedInt16** function is designed to convert unsigned to signed.

ACos

Function

val = ACos(A1)

Description

To compute the inverse cosine value of A1.

Parameters

- **A1:** floating value, can be an address or variable.
- **Val:** return value is floating, can be an address or variable.

Example

```
Dim a ,b as floating'define two float variable a,b

a = 0.5 'assign the designated value to a

b = ACos(a) 'the inverse cosine value of "a" is a radian which assign to variable b.

'to add the following sentence if needed to view the return value:
```

```
float2d("@W_HDW200", b)'float b written into HDW200.
```

Result: @W_HDW200=1.047

✎**Note:** Please call **RadToDeg** function to convert radian into angle.

AddrStringCompare

Function

val = AddrStringCompare(A1, A2, length)

Description

It is designed to compare the designated length of two character strings. The string value is 1 when the two strings are the same.

Parameters

- **A1, A2:** character string, need to be an address (such as:"@W_HDW000002")
- **Length:** The length of character string.
- **Val:** Return value, 0 or 1.

Example

```
if AddrStringCompare("@W_HDW10","@W_HDW0",2)=1 then

'compare the character string of HDW10 and @W_HDW0 whether value are 1.

@W_HDW20=1 '@W_HDW20 display 1

else

@W_HDW20=0'@W_HDW20 display 0

Endif
```

Input: @W_HDW10="1a2 ", @W_HDW0="1a2 ",

Result: @W_HDW20=1

Input: @W_HDW10="ab2 ", @W_HDW0="12a ",

Result: @W_HDW20=0

Asc

Function

val = Asc(A1)

Description

Return the first character of the string in ASCII value.

Parameters

- **A1:**character string, it can be an address (such as: @W_HDW000002)
- **val:**return value,ASCII value,it could be an address or variable.

Example

```
@W_HDW10 = Asc("A") 'return the ASCII value of A to HDW10

@W_HDW11 = Asc("a")'return the ASCII value of a to HDW11

@W_HDW12 = Asc("Apple") 'return the first character A of string Apple to HDW12

@W_HDW13 = Asc("123") 'return the first character ASCII value 1 of string 123 to HDW13.
```

Result:

- @W_HDW10 = 65
- @W_HDW11 = 97
- @W_HDW12 = 65
- @W_HDW13 = 49

AsFloating

Function

val = AsFloating(A1)

Description

Convert parameter A1 to a float.

Parameters

- A1: integer variable.
- **val:** return float value, can be a variable or address.

Example

```
Dim a as integer 'define a integer variable {a}.

a = @W_HDW10 'assign @W_HDW10 to a

b = AsFloating(a) 'convert integer a to float then assign to b.

b = b/1.2 'add as following sentence when need to view the return value:

Float 2D("@W_HDW11",b) 'float variable b written into HDW11.
```

Input: @W_HDW10=24,

Result: @W_HDW11=20.00(set two decimals)

ASin

Function

val = ASin(A1)

Description

Calculate the arcsine value of A1.

Parameters

- **A1:** Float can be an address or variable.
- **Val:** Return float value, can be an address or variable.

Example

```
Dim a, b as floating'define two float variable a,b

a = 0.5'assign the designated value to a

b=ASin(a) 'calculate the arcsine value of a ,assign the radian to b.

'Add the following command if need to view the return value:

float2d ("@W_HDW200", b) 'float variable b written into HDW200
```

Result: @W_HDW200=0.524

✎**Note:** Pleasecall **RadToDeg** function to convert radian into angle.

AsInteger

Function

val = AsInteger(A1)

Description

Convert parameter A1 to integer value.

Parameters

- **A1:** floating need to be a variable.
- **Val:** the value returned could can be a variable or address.

Example

```
Dim a as floating 'define floating variable a

a = D2Float("@W_HDW0",a) 'use D2Float function to save the float date of HDW0 in a

b= AsInteger(a) 'convert the float a into integer, the return value assigned to b

@W_HDW10=b 'save b to HDW10
```

Input: @W_HDW0=20.12,

Result: @W_HDW10=20

AsString

Function

val = AsString(A1)

Description

Convert parameter A1 to a character string.

Parameters

- A1: not string parameter, it can be a variable.
- Val: return string value, variable or address.

Example

Script 1:

```
a=123 'assign a value to a

b=234 'assign a value to b

c=AsString(a)+AsString(b) 'convert a and b to string then add up the two strings ,assign the result to c.

@W_HDW0=c 'assign c to HDW0

d=a+b 'plus a with b

@W_HDW100=d 'assign d to (HDW100)
```

Result:

- @W_HDW0=123234
- @W_HDW100=357

Script 2:

```
W2S("@W_HDW200","@W_HDW300","02d")

W2S("@W_HDW210","@W_HDW400","02d")

W2S("@W_HDW220","@W_HDW500","02d")

@W_HDW0=AsString(@W_HDW300)+AsString(@W_HDW400)+AsString(@W_HDW500)
```

Input: @W_HDW200=12,@W_HDW210=34,@W_HDW220=56

Result :@W_HDW300=12,@W_HDW400=34,@W_HDW500=56,@W_HDW0=123456

✎**Note:** Ensure the data always is two words; otherwise occur error. reference the other chapter of W2S function

ATan

Function

var = ATan(A1)

Description

Return an arctangent value,the radian ranges -pi/2 to pi/2.

Parameters

- **A1:** Can be float, address or variable.
- **Val:** radian of return value.

Example

```
@W_HDW20= Atan(@W_HDW10)'save the arctangent value of (HDW10) to (HDW20)
```

Input: @W_HDW10=1.000,@W_HDW20=0.785

✎**Note:** Please call RadToDeg function convert radian to angle.

ATan2

Function

val = ATan2(A1,A2)

Description

Return the arctangent value of A1/A2,radian range

Parameters

- **A1, A2:** Address or variable.
- **Val:** returned value is a radian, range -pi to pi.
- **Notice:** ATan2 use sign of two parameters to define the quadrant of return value.

Example

```
@W_HDW20= ATan2 (@W_HDW10,@W_HDW12)'save the arctangent value of (HDW10/HDW12) to (HDW20).
```

Input:@W_HDW10=1.0,@W_HDW12=1.0,

Result: @W_HDW20=0.785

✎**Note:** Please call **RadToDeg** function convert radian to angle.

B2W

Function

B2W(A1, A2, length)

Description

Convert an array (begins with A2, unit: byte, to another array (begins with A1, unit: word).

Parameters

- **A1:** Saving address after converting
- **A2:** Address of the value be converted
- **Length:** The length of conversion

Return value: None

Example

B2W(@W_HDW100,@W_HDW10,2) 'convert (@W_HDW10) to the length of 2,save as the result that begins with @W_HDW100.

Input: @W_HDW10=1A2B

Result:

- @W_HDW100=2B
- @W_HDW101=1A

✎**Note:**

- A1 and A2 need to be address(e.g.: @W_HDW000002);
- Length could be address or variable;
- This is a subprogram; it has no returned value.

BCD

Function

val = BCD(A1)

Description

Convert A1(binary) to BCD, save the result as return value.

Parameters

- **A1:** The binary be converted; it can be an address or variable.
- **Val:** Return value, BCD code; it can be an address or variable.

Example

@W_HDW20=BCD(@W_HDW10) 'convert HDW10 (binary) to BCD code, then save in (HDW20)

Input: @W_HDW10=11111111(binary)

Result: @W_HDW20=255

✎**Note:** Return value is a word; it hexadecimal corresponds to BCD code.

Beep

Function

Beep()

Description

Enable buzzer

Parameters

None

Example

```
if @B_HDX100.0=1 then 'beep when the bit switch HDX100.0 set ON

beep()

endif
```

Result: HMI beep when bit switch HDX100.0 set ON.

BIN

Function

Val = BIN(A1)

Description

Convert A1 (BCD) into binary, save the result in return value.

Parameters

- **A1:** The BCD code is converted; it can be address or variable.
- **Val:** Return binary value,it can be address or variable.

Example

```
@W_HDW20=BIN(@W_HDW10) 'convert HDW10(BCD) to binary, save the result in (HDW20)
```

Input: @W_HDW10=255

Result: @W_HDW20=11111111 (binary)

BMOV

Function

BMOV(A1, A2,length)

Description

Copy data with a designated length from source address A2 to A1.

Parameter

- **A1:** saving address;
- **A2:** source address;

length: operating length;

Example

```
@W_HDW20 = 20 'assign value to HDW20

@W_HDW21 = 21 'assign value to HDW21

@W_HDW22 = 22 'assign value to HDW22
```

BMOV(@W_HDW10,@W_HDW20,3) 'assign the word address of HDW20, HDW21, HDW22 to HDW10, HDW11, HDW12
Result:

- @W_HDW10 = 20
- @W_HDW11 = 21
- @W_HDW12 = 22

✎**Note:**

- A1 and A2 need to be address
- Length can be an integer variable or an address. When destAddr and srcAddr are PLC (external device) addresses, the length range is 1-2048, and the 2049th address would not be operated when the range is exceeded.
- When destAddr and srcAddr are HMI addresses, the length range is 1-4096, and This function is invalid when out of range.

Chr

Function

val = Chr(A1, A2, ...)

Description

Convert integer parameter into correspond ASCII character, return the character string.

Parameters

- **A1, A2....:** converted integer; it can be an address or variable.
- **Val:** returned value, can be an address or variable.

Example

@W_HDW100=Chr(@W_HDW20, @W_HDW21, @W_HDW22, @W_HDW23, @W_HDW24)

'convert the value of(HDW20, HDW21 ,HDW22, HDW23, HDW24) to ASCII character, assign the value to (HDW100)
Input 72,69,76,76,79 step by step according to HDW20, HDW21, HDW22, HDW23, HDW24,

Result: returns HELLO to (@W_HDW100).

Clrb

Function

ClrB(A1)

Description

Set the bit of A1 as FALSE (0).

Parameters

A1: System address(bit)

✎**Note:** subprogram has no return value.

Example

```
ClrB(@B_HDX100.0) 'assign 0 to (@B_HDX100.0)
```

Constant

Description

Script supported constant, users can use on script:

pi = 3.14159265358979321

TRUE = 1

FALSE = 0

Example

```
Dim a as integer 'define integer a

a = RadToDeg(pi) 'convert radian pi to angle then assign to a,RadToDeg function is used to convert radian to angle.

@W_HDW11 = a 'assign a to (HDW11)
```

Result: @W_HDW11=180

CopyFile

Function

A6=CopyFile(A1,A2,A3,A4,A5)

Description

Copy the A3 file from the A1 directory to the A2 directory according to the format of A4 and A5, and write returned value status to A6.

Parameters

- **A1:** the source path of the file to be copied.
- **A2:** target path.
- **A3:** the name of the file to be copied.
- **A4:** copy type (0: copy file, 1: copy directory).
- **A5:** Whether to overwrite the file with the same name when copying (0: Yes, 1: No).
- **A6:** returned value.
  - 0: copy failed
  - 1: copy succeeded
  - 2: Parameter error
  - 3: U disk does not exist
  - 4: SD card does not exist
  - 5: Path error

Example

Copy a single file:

```
@W_HDW100 = " test.csv"

CopyFile("UDisk/Test","Flash/Test","test.csv",0,0) 'Copy the Test.csv file in the UDisk/Test directory to the Flash/Test directory.

Can also be written as CopyFile("UDisk/Test","Flash/Test",@W_HDW100,0,0)

(2) Copy the entire directory file:

CopyFile("UDisk/Test","Flash/Test","",1,0) ' Copy the files in the UDisk/Test

directory to the Flash/Test directory.
```

✎**Note:**

- Both source and destination paths need to begin with UDisk or Flash or SDCard;
- A1 and A2 could be string (requires double quotes) or variable, maximum character length 127 allowed in path;
- A3 could be a string, such as: [FileName] (requires double quotes); or address, such as: @W_HDW100 (no need to add double quotes);
- A4 and A5 could be values, addressesor variable;
- A6 could be address or variable.

Cos

Function

Val = Cos(A1)

Description

Return a cosine value of an angle.

Parameters

- **A1:** a float radian of angle, it can be an address or variable.
- **Val:** return float value, it can be an address or variable.

Example

```
Dim a, b as floating 'define float a, b

b = pi/3 'convert the value of HDW11 to float and assign to 'b'.

a=Cos(b) 'return the cosine value of'b' and assign the result to 'a'.

'to add the following sentence if need to view the return value:

Float2D("@W_HDW20",a) 'the float value of a written into HDW13.
```

Result: @W_HDW20=0.5

✎**Note:** Please call **RadToDeg** function convert radian to angle.

D2F

Function

D2F (A1, A2) or A1= D2F (A1, A2)

Description

Convert the 32-bit integer format data to float then output the result.

Parameters

- **A1:** required data, begin with"@";
- **A2:** source data, begin with"@";

Example

```
D2F(@W_HDW2,@W_HDW0) 'convert the double word (HDW0) to float, save the result to (HDW2).

@W_HDW2=D2F(@W_HDW2,@W_HDW0) 'convert the double word (HDW0) to float, save the result to (HDW2).
```

Result: HDW0=100, HDW2=100

D2Float

Function

F= D2Float("A1",F)

Description

Convert the designated value to floating then assign to variable.

Parameters

- **A1:** Source data;
- **F:** User-defined floating variable

Example

```
dim F as floating 'define F as floating

F=D2Float("@W_HDW10",F) 'assign the value of (HDW10) to F in floating

Float2D("@W_HDW12",F) 'copy the floating value of F to HDW12 register,use to display result.
```

Result:

- HDW10=200,
- HDW12=200.

✎**Note:** A1 needs to be address;

D2Int

Function

A2= D2Float("A1",A2)

Description

Output the 32-bit integer in the form of integer.

Parameters

- **A1:** Source data can only be the HMI internal or external register starting with "@".
- **A2:** Target data can only use theinteger variable defined by script.

Example

```
dim var1 as integer 'define var1 as integer

var1=D2Int("@W_HDW0",var1) 'Read out the 32-bit integer in HDW0 and save the result in var1.

HDW0=9999999
```

Result: Var1=9999999

DataLogToUorSD

Function

DataLogToUorSD (A1, A2, A3, A4, A5, A6, A7, A8)

Description

Exported Data record (database file) as db or csv format file, and register HSW1853 can change the exported file type.

(HSW1853 = 0: exported as db format, filename format: Start time-end time-group name.db)

(HSW1853 = 1: exported as csv format, filename format: Start time-end time-group name.csv)

Parameters

- A1: Data record group name address (string, must be address). For example: " @W_HDW30 "
- A2: Start time address (string, must be address). For example: " @W_HDW40 "
- A3: End time address (string, must be address). For example: " @W_HDW50 "
- A4: db file save location (integer value)
  - = 1: Saved on SD card
  - = 2: Save in USB flash drive
- A5: Return value(string, must be address). For example: " @W_HDW60 ", return result of the script will be written to HDW60; Meaning of value in address:
  - = 1: Script executed successfully
  - = 100: U disk or SD card unplugged
  - = 101: U disk or SD card does not exist
  - = 102: Script executed failed
  - = 201: db export failed
  - = 203: Script is being executed (previous executed script doesn't exit)
- A6: Script running mode (integer value)
  - = 1: Wait for script execution before exit
  - = Others: Exit without waiting for the result of script execution (when the data record volume is large, it takes some time to execute the script, and when wait for the result of script execution, it may affect the execution of other scripts)

- A7: Path name address (string, must be address). For example: " @W_HDW70 ", script will automatically prefix the folder name HMI_, for example: @W_HDW70 = "Log", the db file will be saved under the directory of HMI_Log in U disk or SD.
- A8: Reserved address (string, must be address). For example:" @W_HDW80 ". This address is temporarily useless but needs to be set.

Example

```
@W_HDW30 = "Record test" 'Data record group name
@W_HDW40 = 2021
@W_HDW41 = 04
@W_HDW42 = 30
@W_HDW43 = 09
@W_HDW44 = 0
@W_HDW45 = 0
'Start time 2021-04-30 09:00:00
@W_HDW50 = 2021
@W_HDW51 = 04
@W_HDW52 = 30
@W_HDW53 = 20
@W_HDW54 = 0
@W_HDW55 = 0
'End time 2021-04-30 20:00:00
@W_HDW70 = "Log" 'Directory name

DataLogToUorSD ("@ W_HDW30", "@ W_HDW40", "@ W_HDW50", 2, "@ W_HDW60", 1, "@ W_HDW70", "@ W_HDW80")

'Export the data records collected from 2021-04-30 09:00:00 to 2021-04-30 20:00:00 under the group name of the record test to the HMI_Log directory of the U disk with the file name being 20210430090000-20210430200000-record tests. db. (HSW1853 is 0).
```

DbToCSVFile

Function

A8=DbToCSVFile(A1,A2,A3,A4,A5,A6,A7)

Description

Convert db (database file) file to csv format and export it.

Parameters

- **A1:** db file save path (value is integer);
  - =0: Alarm record file in HMI flash;
  - =1: Alarm record file in UDisk (USB flash disk);
  - =2: Alarm record file in SD card;
  - =3: Data record file in HMI flash;
  - =4: Data record file in UDisk (USB flash disk);
  - =5: Data record file in SD card;
- **A2:** group numer of db file (value is integer);

    - Used during exporting data record file, the group number could be seen in the data record setting interface;
- **A3:** csv file save path (value is integer);
    - =0: Save in UDisk (USB flash disk);
    - =1: Save in SD card;
- **A4:** csv name;
- **A5:** start time of data recordin db file(string), consecutive 6 word addresses (the values in the address are year, month, day, hour, minute, second);
- **A6:** end time of data recordin db file(string), consecutive 6 word addresses (the values in the address are year, month, day, hour, minute, second);
- **A7:** csv encoding format;
    - =0 UTF8 format;
    - =1 GBK format;
- **A8:** returned value;
    - =0: Failed to export;
    - =1: Exported;
    - =2: db file path error;
    - =3: U disk or SD card does not exist;
    - =4: csv name error;
    - =5: db file does not exist;
    - =6: csv file already exists;

Example

- @W_HDW100 = 2018
- @W_HDW101 = 12
- @W_HDW102 = 25
- @W_HDW103 = 19
- @W_HDW104 = 10
- @W_HDW105 = 30
- @W_HDW200 = 2018
- @W_HDW201 = 12
- @W_HDW202 = 25
- @W_HDW203 = 20
- @W_HDW204 = 10
- @W_HDW205 = 30

Export data record file

```
DbToCsvFile(3,2,0,"123.csv","@W_HDW100","@W_HDW200",0)
'In the HMI flash, the group number is 2, and the data records collected in the time of 2010.12.25 19:10:30-2018.12.25 20:10:30
'are exported to the 123.csv file in the Udisk in UTF8 format.
```

Export alarm record file

```
DbToCsvFile(0,0,0,"456.csv","@W_HDW100","@W_HDW200",1)
'The alarm records generated in HMI Flash at 2018.12.25 19:10:30-2018.12.25 20:10:30
'are exported to the 456.csv file in the Udisk in GBK format
```

✎Note:

- A1 can be an address or a variable or a value, and the path need to start with UDisk or Flash or SDCard;
- A2 can be an address or a variable or a value;
- A3 can be an address or a variable or a value, and the path need to start with UDisk or Flash or SDCard;

- A4 can be an address or variables or string, and the length of the file name (sum of values, English, Chinese) could not exceed 127;
- A5 need to be an address;
- A6 need to be an address;
- A7 can be an address or a variable or a value;

DegToRad

Function

A2 = DegToRad(A1)

Description

Convert the angle into correspond radian, and display.

Parameters

- **A1:** inputting angle supports address, other variable or floating.
- **A2:** outputting radian supports address, other variable or floating.

Example

Script 1

```
@W_HDW12=DegToRad(@W_HDW10) 'input angle on (HDW10),convert to correspond redian and copy to (HDW12)
```

Result: HDW10=180; HDW12=3.14159

Script 2

```
dim a as floating 'set variable

dim b as floating 'set variable

b=30 'input angle

a=DegToRad(b) 'convert the length of radian and copy to variable {a}

float2d("@W_HDW0", a) 'display the value of floating on(HDW0)
```

Result: HDW0=0.52360

DIM ... AS ...

Function

Dim "variable" as "date type"

Description

Declare a variable, stable the type of data.

Parameters

- **Variable:** begin with letter, other character can be letter, numbers, underscores ('_'), need to begin with '@' if it is address;

- **Data type:** string,floating,integer;

Example

dim a as integer 'define a as integer

dim @W_HDW0 as floating 'define @W_HDW0 as floating

dim hi as string 'define "hi" as string

Result:

a is integer

@W_HDW0 is floating

hi is string

✎**Note:** Use the variable of Dim during running, could not change the type, Dim will be missed if the type of variable is no difined. Variable could be declared in once.

DO ... LOOP

Function

Do [While | Until condition]

[statements]

Loop

or

Do

[statements]

Loop [While | Until condition]

Description

Condition determent instruction.

- Do while.loop executes an instruction of block repeatedly when condition is true.
- Do until.loop executes an instruction of block repeatedly until condition is false.

Parameters

Condition: determine condition; obtain the expression of True or False.

Statements: execute one or more instructions repeatedly when condition is True or until condition is True.

If condition is true, all statements are executed until the Wend statement is encountered. Control then returns to the While statement and condition is again checked. If condition is still True, the process is repeated. If it is not true, execution resumes with the statement following the Wend statement.

Example

dim i as integer 'end DO loop when i=100

```
do while i<100

i=i+1

@W_HDW0=i

loop
```

Result: HDW0=100

End

Function

Terminates the script immediately.

Description

End the execution of script.

Parameters

Statement: Judging condition, use with IF together. end script when meet condition.

Example

```
If a = 10 Then End 'end script when a=10.
```

Result: End the script program.

Exp

Function

A1=Exp(A2)

Description

Returns the power value of e (natural logarithm), save the outputting result to A1, e=2.71828182846.

Parameters

- **A1:** the goal date: the power floating value of returning, need to begin with '@'(e.g.@W_HDW10);
- **A2:** Source data, natural exponential function, need to be integer or variable. Could not not begin with the address of "@"(e.g.@W_HDW10)

Example

```
dim a as integer'define a as integer

a = @W_HDW2 ' assign the value of (HDW2) to variable a

@W_HDW0= Exp(a)'exponential is the value of (HDW2),save result to(HDW0)
```

Result:

- HDW2=2,
- HDW0=7.38905600

F2D

Function

F2D (A1, A2)

Description

Convert a 32 bit floating to integer format, then output the result.

Parameters

- **A1:** Destination, the value can be an address(e.g.@W_HDW12).
- **A2:** source date, it can be an address or other variable.

Example

F2D(@W_HDW12,@W_HDW10) 'convert the floating of (HDW10) to integer, save in (HDW12).

Result:

- HDW10=200,
- HDW12=200

F2S

Function

F2S (A1,A2,s1)

Description

Output a format of floating that in the type of string.

Parameters

- **A1:** Source address, used to store floating, the value is an address(e.g.@W_HDW200);
- **A2:** Destination address, used to store string after converted, value is an address(e.g.@W_HDW100).
- **S1:** the format of displaying goal data. such as the format of 03.03f,f, used for outputting a single-precision in the form of decimal.m.nf:means m column and n decimals when outputting.

Example

F2S("@W_HDW200", "@W_HDW100", "03.03f") '(HDW200) is floating input,(HDW100) is text output;

Result:

- HDW200=1.22365,
- HDW100=1.224

FileCmpDir

Function

FileCmpDir(A1, A2, A3, A4, A5, A6, A7, A8, A9)

Description

The filename input and the filename in the folder whether is duplicate or not.

Parameters

A1: file path (value is integer);

In HMI:

- =0: Recipe folder in HMI flash;
- =1: Custom folder in UDisk (USB flash disk);
- =2: Custom folder in SD card.
- =3: Custom folder in HMI flash;

In simulator:

- = 0: D:/Recipe/
- = 1: C:/WECON/CustomFileDir/
- = 2: C:/WECON/CustomFileDir/
- = 3: C:/WECON/CustomFileDir/

✎**Note:**

A1 can be an address,variable or fixed value.

A2: file name (value is string);

The file name would be compared;

✎**Note:**

A2 can be an address, variable or string. The length of file name cannot exceed 32 characters.

A3: folder name (value is integer);

The folder name would be compared;

✎**Note:**

A3 can be an address, variable or string. The length of file name cannot exceed 32 characters.

A4:file name display address (value is a string);

Display the specified number of file names in the specified folder.

✎**Note:**

A4 Can only use address here. And the length of the file name cannot exceed 32 characters.

A5: Function type (value is an integer)

= 0: compare whether the file name in the folder and the file name input are duplicated or not.

= 1: return list of file names.

= 100: compare file names, without input suffix .csv

✎**Note:**

A5 can be an address, variable or value.

A6: Return value (value is an integer)

= 1: Successfully opened the folder.

= 2: Failed to open the folder.

= 3: The file has duplicate names

✎**Note:**

A6 must be an address.

A7: File number (value is an integer)

Display the number of files in the folder (up to 100)

✎**Note:**

A7 must be an address.

A8: File creation time (value is a string);

Display the time of file creation, time occupies 32 characters

A9: The maximum number of files to read (value is an integer)

Set the maximum number of files to read (up to 100 files)

✎**Note:** A9 can be an address, variable or value.

Example

Read the specified number of files in the folder.

FileCmpDir(@W_HDW1254, @W_HDW50, "File", "", 0, "@W_HDW0", "", "", 10)'Determine whether there is a file with the same name as @W_HDW50 in the file directory of HDW1254.@W_HDW0=3 is Yes.@W_HDW0 = 1 is No.@W_HDW0=2 is comparison failed.

Compare filename input and the filename in the folder whether is duplicate or not.

FileCmpDir(@W_HDW1254, "", "File", "@W_HDW200", 1, "@W_HDW0", "@W_HDW300", "@W_HDW400", 10)’The file name under the directory File in the flash is written to the continuous address begin from @W_HDW200, the file creation time is written to the continuous address begin from @W_HDW400, the number of files is written to the address @W_HDW300, the maximum number of files is 10

Compare the csv file name in the folder and the csv file name input whether is duplicated or not.

FileCmpDir(@W_HDW1254, @W_HDW50, "File", "", 100, "@W_HDW0", "", "", 10)' Judge whether there is a file in the directory File in the flash is same with following: @W_HDW50, @W_HDW50.csv, @ W_HDW50.CSV , @W_HDW0=3 is Yes, @W_HDW0 = 1 is No ,@W_HDW0=2 is comparison failed

FILL

Function

FILL (A1, A2, A3)

Description

Write the same value to designated address constantly.

Parameters

A1: The beginning address, it can be an address (e.g.@ W_ HDW25);

A2: Source data, it needs to be written in continuous value, the value can be an address, variable or constant;

A3: The number of operation, writing address number, it can be an address, variable or constant;

When the PLC (external device) address is used in this function, the length range is 1-2048, and only the 2048th address can be operated when the range is exceeded.

When the HMI internal address is used in this function, the length range is 1-4096. If the function is out of range, then the function is invalid.

Example

FILL (@W_HDW25, 10, 3) 'At the beginning three address of @W_ HDW25 is 10.

Result: At the beginning three address of @W_ HDW25 is 10, @W_ HDW25=10, @W_ HDW26=10, @W_ HDW27=10.

Float2D

Function

Float2D (A1,A2);

Description

Copy floating value to the address.

Parameters

- **A1:** Goal address, the value need to be address (e.g.@W_HDW102);
- **A2:** Source data, it can be floating;

Example

```
dim f as floating 'define f as floating

f=1.1 'assign a designated value to f

Float2D ("@W_HDW102",f) 'assign the value f to HDW102
```
Result: HDW102=1.1

For. to. step. next

Function

For counter = start to end Step

[Statements]

Next

Description

Execute a command repeatedly for designated times.

Parameters

- **counter:** Work as a variable for loop counter;
- **start:** The start value of counter, it could be any variable type or expression;
- **end:** The end value of counter, it could be any variable type or expression;
- **step:** Every loop, the changed value of counter is step value, step default if it is not designation. Step default as below:
    - if start>end, step default is 1;
    - if start< end. It could be any variable type or expression;
- **statements:** Between For with Next, execute instruction set of designated times;
    - Set a loop of for...Next in another loop, it can nest call the loop of for...Next. Different from a while, for only search once value from end. Empty for loop will be ignored, and could not delay time.

Example

```
'Use HDX2.0 to trigger the loop

for i=100 to 0 step -5 ' set the start is 100,end is 0,subtract 5 every time, execute 20 times totally.

@W_HDW100=@W_HDW100+1 ' execute (HDW100+1) 21 times totally, the final result is 21.

Next

@B_HDX2.0=0
```

Result: HDW100=21

Function

Function

- Function name (arglist)
- statements
- name = expression
- statements
- End Function

Description

Unlike internal function, it needs to declare the name, parameter, code of the function.

Parameters

- **Name:** function name.
- **arglist:** stands for the variable list of parameter, this parameter will be entered during calling function. use comma to separate.
- **statements:** a set of code in running function body.

Notice: it could not define a function program at any other program body. Write name first and then follow with parameter list, when calling function. Declare function need to before be calling. in the internal function body, it could be assigned to a function name from returned value at any place. Return value is 0 if not assign function name. Functions can recursive, but it may lead to stack overflow.

Example

```
Function sincos (angle as floating)

sincos = sin(angle) + cos(angle)

End Function<

........

@W_0002 = sincos(pi/2)

........
```

GetServerDelayInfo

Function

A3=GetSerVerDelayInfo(A1,A2)

Description

Convert string A1 to hexadecimal number.

Parameters

A1: The starting address, save the delay data of each server (10 consecutive addresses, the last 7 addresses are reserved), when the server testing fails, the value in the corresponding address is -1. The unit is ms (signed decimal number).

Address	Description
A1	Delay data of the server configured in [Project Settings]
A1+1	Delay data of Server 1
A1+2	Delay data of Server 2

A2: Server test result address

Address	Description
0 bit	Set ON, Server configured in [Project Settings] fails
1stbit	Set ON, Server 1 fails

2ndbit	Set ON, Server 2 fails
3rd~ 10thbits	Reserved
11thbit	Set ON, network error, network port interface failure

A3: Returns the preferred server number or network status (automatically selects the server with the best connection network status).

- =0: Server configured in [Project Settings];
- =1: Server 1;
- =2: Server 2;
- =3~10: Reserved;
- =100: Try again later (two test intervals need to wait 30 seconds);
- =101: Failed to test Server;

Example

@W_HDW200 = GetServerDelayInfo("@W_HDW100", "@W_HDW50")

Result:

- HDW100: Delay data of the server configured in [Project Settings]
- HDW101: Delay data of Server 1
- HDW102: Delay data of Server 2
- HDX50.0=1:Server configured in [Project Settings] fails
- HDX50.1=1: Server 1 fails
- HDX50.2=1:Server 2 fails;
- HDX50.11= 1: network error, network port interface failure;

When the server signal is stable, the optimal server is automatically connected according to the network status @W_HDW200. When the server connection is abnormal, @W_HDW200=101, the server test fails or the test is abnormal.

✎**Note:** The time interval between executions of this function could not be less than 30 seconds, otherwise an error will occur.

Goto

Function

Goto label

Description

Go to the designated row without any condition in a function body.

Parameters

- **Label:** target character, start with letter in row label, end with(:)of any string.row label has no sensitive to the format of letter.

Example

Goto sd 'go to the row which start with "sd";

Result: Go to sd row.

✎**Note:** Goto only can jump into the internal function that visible row.

H2A

Function

A1 = H2A (A2)

Description

Convert a binary (16-bit) to hexadecimals (4-bit) of ASCII.

Parameters

- **A1:** Returned value, string, it could be an address or variable.
- **A2:** Binary is needed to be converted, the value could be a address or variable.

Example

@W_HDW100= H2A (@W_HDW0) ' convert the binary of (HDW0) to character and save in (HDW100).

Result:

- HDW0=200,
- HDW=100

HmiRegoperator

Function

HmiRegoperator (A1,A2,A3,A4,A5,A6)

Description

Data of Upload/ Download address

Parameters

- **A1:** The start address of target
- **A2:** Length, address length, unit: word, range: 1~1000
- **A3: s**torage
    - A3=0, select USB flash disk as storage, and save files in Root directory;
    - A3=1, select SD card as storage, and save files in Root directory;
- **A4:** File name address, itneed to be address such as"@ W_HDW2"
- **A5**: Upload / download data
    - A5=0, save the data in the specified address as a file and store the file in the root directory;

    - A5=1, Read data from file and it woule be written into specified address;
- **A6:** State display, it needs to be address, such as "@ W_HDW2";
    - A6=1, Normal
    - A6=2, USB flash disk/ SD card does not exist
    - A6=3, File doesn’t exist;
    - A6=4, File name error;
    - A6=5, Check error
    - A6=6, Abnormal communication;
    - A6=7, HUW register is not allowed;
    - A6=8, Address length range error (address length range: 1-1000);

Example

Script 1

HmiRegoperator("@W_HDW0",10,0,"@W_HDW2000",0,"@W_HDW3000")’The data in HDW0-HDW10 is saved as a file, the file name is set by HDW2000 and stored in a USB flash disk.

Script 2

HmiRegoperator("@W_HDW0",10,1,"@W_HDW2000",1,"@W_HDW3000")

Read the values from files which stored in the SD card (files named by the HDW2000), and write these values to the HDW0-HDW10.

✎Note:

- The length of the file name is less than 32 characters, and the file name consists of numbers and letters (the file name does not meet this standard; WECON does not guarantee the accuracy of the data).
- The interval of download function operation should be 5s or more.
- A1 need to be address such as “@W_HDW2”.
- Please use “Character Input/Display” object for it, and the file name consists of numbers and letters, could not be punctuated characters, maximum character length 32 allowed.

Hypot

Function

Var = Hypot (expr1, expr2)

Description

Calculate the value of the hypotenuse of a right triangle.

Parameters

- **expr1, expr2:** Source data, the two sides of right triangle. it need to be address;
- **Var:** Destination data,it need to be address;

Example

@W_HDW200=Hypot (@W_HDW105,@W_HDW108) ' input the value of right-angle side at (HDW105) and (HDW108),and assign the result of the hypotenuse to (HDW200).

Result:

- HDW105=3,
- HDW108=4,
- HDW200=5

✎**Note:** hypot function could support integer and floating when the format of source data and target data are the same.

IF ... THEN ... ELSE ... END IF

Function

If condition Then

Statements

[Else

else statements]

End If

Description

Conditional judgments instruction. When the [Condition] defined by [IF] is TURN, the operation following [THEN] is performed. When [Condition] is FALSE, the operation after [ELSE] is executed.

Parameters

- **condition:** any expression, the value could be true or false.
- **statement:** execute the instruction block when condition is true.
- **else statement:** execute the instruction block when condition is false.

Example

```
if @W_HDW105=200 then ' judging condition: whether the value of (HDW105) is 200

@W_HDW108=1 'the value of (HDW108) is 1 if fulfil condition

else

@W_HDW200=1 ' the value of (HDW200) is 1 if not fulfil condition.

Endif
```

Result:

- HDW105=199;
- HDW108=0;
- HDW200=1

InStr

Function

var = InStr ("str1", "str2")

Description

Returned the position of str1 in str2(start with 0), set -1 if no result.

Parameters

- **str1:** source string, it could only be string, not address;
- **str2:** target string, it could only be string, not address;
- **var:** Returned value, the format of data needs to be string;

Example

```
dim a as floating

a = InStr ("Hello", "o") ' calculate the position of"o"in"hello".

float2d ("@W_HDW0",a) ' Returned value is 4.(start with 0)
```
Result: HDW0=4

Int2D

Function

Int2D("A1",A2)

Description

Write the 32-bit integer into the target address

Parameters

- **A1:** Source data could only be the HMI internal or external register starting with "@".
- **A2:** Target data could only use the integer variable defined by script.

Example

```
dim var1 as integer 'define var1 as integer

Int2D("@W_HDW0", var1)    'Read out the 32-bit integer in var1 and save the result in HDW0 .

var1=9999999,
```
Result: HDW0=9999999.

InvB

Function

InvB (A1)

Description

The state of inverse bit, it is a sub function, no returned value. Achieve the state of switching address constantly.

Parameters

- **A1:** it is an address.

Example

```
InvB (@B_HDX0.1) 'switch the state if (HDX0.1).
```

Result: Switch the state of (HDX0.1) constantly.

IsFloating

Function

A2=IsFloating (A1)

Description

Decide whether a parameter is floating, return true if it is floating, otherwise return FALSE.

Parameter

- **A1:** source data, variable;
- **A2:** target data, variable;

Example

```
dim a as integer

dim b as floating

b= D2float ("@W_HDW200",b) 'assign the value of (HDW200) to b

a = IsFloating (b) 'judge whether b is floating or not

@W_HDW300=a 'save the result to (HDW300)
```

Result: HDW300=1

IsInteger

Function

A2= IsInteger (A1)

Description

Determine whether a parameter(A1) is integer, return TRUE if the parameter is integer, otherwise return FALSE.

Parameter

- **A1:** Source date, it is variable or number;
- **A2:** Target date, need to be variable, it could not be system address ;

Example

```
dim a as integer

a = IsInteger (20) 'determine whether 20 is integer
```

@W_HDW300=a ' display the result on (HDW300)
Result: HDW300=1

IsString

Function

val = IsString(expr)

Description

Determine whether a parameter is string, return TRUE if it is string, otherwise return FALSE.

Parameters

- **Expr:** source string, it could be a variable or string, not address;
- **Val:** target date, the result need to be variable, could not be address;

Example

dim a as integer 'define variable, display the result;

a= isstring ("hello") 'determine whether"hello"is string;

@W_HDW0=a 'assign the result to (HDW0)
Result: HDW=1

Lcase

Function

A2 = LCase(A1)

Description

Convert all parameters to lowercase strings.

Parameters

- **A1:** source string, it could be an address or variable;
- **A2:** outputting string, it could be an address or variable;

Example

@W_HDW33 = LCase (@W_HDW25) 'input source sting on (HDW25), convert it to destination string and display the result on (HDW33);

Result: HDW25=HELLO

HDW33=hello

Left

Function

Val =Left (String, Length)

Description

Return a string of the specified length from the left side of parameter.

Parameters

- **String:** source string; it could be an address or string.
- **Length:** return the number of character. It could be an address, integer or variable. Return empty string if length<1. return the whole string if length not less than the character number of string.
- **Val:** destination string, outputting string, it could be an address or variable.

Example

@W_HDW30=Left (@W_HDW36, @W_HDW40) '(HDW36) used to input source string,(HDW30) used to display the string result;

Result:

- HDW36=hello,
- HDW40=2,
- HDW30=he

Len

Function

Length=Len(String)

Description

Return the string length.

Parameters

- **String:** source string, it could be a address or string;
- **Length:** target data, return value, it could be a address, variable, integer or floating;

Example

@W_HDW30=Len (@W_HDW36) 'count the character number of (HDW36), save the result to (HDW30);

Result:

- HDW36=hello
- HDW30=5

Log

Function

a= Logn (x)=Log(x)/Log(n)

Description

Log function:return the natural logarithm of the value.

Parameters

- **a:** source date,it could be a variable, but it could not be address;
- **x, n:** source date, it could be a variable, but it could not be address;

Example

```
Dim a as integer 'define a as integer;

Dim b as integer 'define b as integer;

Dim c as integer 'define c as integer;

b=@W_HDW10 'assigns a value to b

c=@W_HDW20 'assigns a value to c

a=Log (b)/Log(c) 'calculate logarithm

@W_HDW0

=a 'assign the result to (HDW0)
```

Result:

- HDW10=27,
- HDW20=3,
- HDW0=3

Log10

Function

a=Log10(x)= Log(x) / Log(10)

Description

Log function: return the natural logarithm.

Parameters

- **A:** target data, result could be variable, could not be address;
- **x:** source data, it could be variable that needs to be the multiples of 10, can’t be address

Example

```
dim a as integer 'define a as integer

dim b as integer 'define b as integer

b=@W_HDW10 'assign a value to b

a= Log (b)/Log(10) 'result
```

```
@W_HDW0=a 'assign the result to (HDW0)
```

Result:

- HDW10=100,
- HDW0=2

LTrim

Function

val=LTrim("string")

Description

Remove the left empty part of the string and return.

Parameters

- **Val:** Destination string, it could be either a variable or address;
- **string:** Source string, it could be either a variable or address;

Example

```
dim a as string

a=Ltrim("hello")

@W_HDW103=a
```

Result: HDW103=hello

MAX

Function

A1=MAX(A2,A3)

Description

Compare the value of A2 and A3, assign the greater value to A1.

Parameters

- **A1:** Return value (used to store the greater value between A2 with A3).
- **A2:** The first comparison value.
- **A3:** The second comparison value.

✎**Note:** A1,A2,A3 should be only used in unsigned integer or unsigned address.

Example

```
DIM A1 as integer

@W_HDW106=10 'assign the value to (@W_HDW106), unsigned decimal word.
```

```
@W_HDW107=5 'assign the value to (?@W_HDW107),unsigned decimal word.

A1 = Max(@W_HDW106,@W_HDW107)

@W_HDW105 = A1
```

Result: @W_HDW105 = 10

Mid

Function

A1=mid(A2, start, length)

Description

Returns a string contain a specified characters length from a string.

Parameters

- **A1:** string contains the selected characters, it needs to be a string
- **A2:** string to be selected, it needs to be a variable or address
- **Start:** the start position of string, it needs to be a variable or address, it means that count begin with 0.
- **Length:** the designated length of string, maximum character length 127 allowed

Example

```
DIM A1 as string

A1 = Mid("hellokitty",1,2) 'select the string of in "

@W_HDW106=A1
```

Result: @W_HDW106 'display "el" on text input and output window

MIN

Function

A1=MIN(A2,A3)

Description

Compare the value of A2 and A3, assign the smaller value to A1.

Parameters

- **A1:** Return value (used to store the snaker value between A2 with A3).
- **A2:** The first comparison value.
- **A3:** The second comparison value.

Example

```
DIM A1 as integer
```

```
@W_HDW106=10 'assign the value to (@W_HDW106), unsigned decimal word.

@W_HDW107=5 'assign the value to (@W_HDW107), unsigned decimal word.

A1 = Min(@W_HDW106,@W_HDW107)

@W_HDW105 = A1
```

Result: @W_HDW105 = 5

✎**Note:** A1, A2,A3 only used in unsigned integer or unsigned address.

MSeconds

Function

A1=MSeconds( )

Description

A1 is used to display the current milliseconds of system.

Parameters

- **A1:** used to store the current milliseconds of system.

Example

```
DIM A1 as integer

@W_HDW0= 10 'assign a value to (@W_HDW0), unsigned decimal word

A1=MSeconds() 'return the current milliseconds of system to A1

@W_HDW0=A1

@W_HDW1=A1>>16 'display milliseconds on screen, (HDW0) is an 32-bit unsigned decimal integer address
```

Result: @W_HDW0 will generate the time value of changing milliseconds unit.

✎**Note:**

- A1 is 32-bit unsigned integer variable or unsigned integer address;
- MSeconds() function rolls back over to zero once the maximum value has been reached (4294967295->0, 1,2,....4294967295->0, 1, 2,....4294967295->0, 1, 2);

MyDeleteFile

Function

MyDeleteFile (A1,A2,A3,A4,A5)

Description

Delete the specified file

Parameters

A1: File location (value is an integer);

In HMI:

- =0: Recipe folder in HMI flash;
- =1: Custom folder in UDisk (USB flash disk);
- = 2: Custom folder in SD card.
- =3: Custom folder in HMI flash;

In simulator:

- = 0: D:/Recipe/
- = 1: C:/WECON/CustomFileDir/
- = 2: C:/WECON/CustomFileDir/
- = 3: C:/WECON/CustomFileDir/

✎**Note:**

A1 can be an address, variable or value.

A2: filename (value is string);

Input the name of the file want to delete.

✎**Note:**

A2 can be an address , variable or a character string, and the length of the file name cannot exceed 32 characters.

A3: folder name (value is integer);

Input the folder where the file want to delete.

✎**Note:**

A3 can be an address , variable or a character string, and the length of the folder name cannot exceed 32 characters.

A4: Delete function type (value is integer)

- = 0: Delete the specified file.
- = 1: Delete all files.

✎**Note:**

A4 can be an address , variable or value.

A5: Return value (value is an integer)

- = 0: parameter error
- = 1: Delete successfully
- = 2: Delete failed
- = 3: Failed to open file

✎**Note:** A5 must be an address.

Example

```
MyDeleteFile(@W_HDW1254,@W_HDW4200,@W_HDW4300,@W_HDW1250,"@W_HDW1252")

‘According to the value of @W_HDW1250, delete the designated file @W_HDW4200 in the folder @W_HDW4300 or
delete all files in the folder @W_HDW4300.
```

NewNoAddr

Function

A1= NewNoAddr (A2, length)

Description

At the basic of source address A2, offset designated length, obtain a new address A1.

Parameters

- **A1:** address after offsetting, it must be String type variable.
- **A2:** source address, it must be an address(e.g.:"@W_HDW2")
- **Length:** offset length, it must be a constant or an integer variable.

Example

```
DIM A1 as string

A1=NewNoAddr("@W_HDW0",50) '(HDW0) offsets 50 words address (16 bit), and save the result to A1

@W_HDW1=A1 '(HDW50) save in A1
```

Result: (@W_HDW1) character input/display will show @W_HDW50

NewStatAddr

Function

A1= NewStatAddr (A2, length)

Description

At the basic of source address A2, offset the designated length, to obtain a new station A1.

Parameters

- **A1:** The address after offsetting, it needs to be variable.
- **A2:** Source station address, it needs to be address (e.g.:"@W_1:10").
- **Length:** offset length

Example

```
DIM A1 as string

A1=NewStatAddr("@W_1:10",2) 'address 10 of station address 1 that offset 2 station addresses, then save the result to
A1

@W_HDW1=A1 'address 3:10 is saved in A1
```

Result: @W_HDW1 character input/display will show @W_3:10

NStringCompare

Function

A1= NStringCompare (A2, A3, length)

Description

Compare whether the designated length of two strings is the same, return 1 to A1 if yes, otherwise return 0.

Parameters

- **A1:** Returned value (compare the designated length of two strings, display 1 when equal, else 0). It could be an address or variable.
- **A2:** the address of string to be compared, it needs to be address.
- **A3:** source string, it needs to be variable or constant string.
- **Length:** string length to be compared

Example

```
@W_HDW1= NStringCompare("@W_HDW0","87654",5)

if @W_HDW1=1 then

@B_HDX10.0=1 'result: HDX10.0 set ON 'when the two strings are the same.

endif

if @W_HDW1=0 then

@B_HDX10.0=0 'result:HDX10.0 set OFF 'when not equal.

Endif
```

Operator

Operation	Symbol	Example	Return type
Addition	+	A1=A2+A3	Return type depending on the type of variable or address of the addition
Subtraction	-	A1=A2-A3	Return type depending on the type of variable or address of the subtraction
Multiplication	*	A1=A2*A3	Return type depending on the type of variable or address of the multiplication

Division	/	A1=A2/A3	Return type depending on the type of variable or address of the division

Remainder	Mod (%)	A1=A2 mod A3
		A1=A2%A3
			Returns the remainder of the division of two numbers. The type of the return value is an integer

Logical OR	Or(|)	A1=A2 or A3
		A1=A2|A3
			Returns the result of a logical OR. The type of the return value is an integer.

Logic AND	And (&)	A1=A2 and A3
		A1=A2&A3
			Returns the result of a logical AND. The type of the return value is an integer.

Logical XOR	Xor (^)	A1=A2 xor A3
		A1=A2^A3
			Returns the result of a logical XOR. The type of the return value is an integer.

Logical reversal	Not (!)	A1=not A1
		A1=A2!A3
			Returns the result of a logical reversal. The type of the return value is an integer.

Left shift	<<	A1=A2<<A3	Shift the value of A2 to the left by A3 digits and return the displacement result. The type of the return value is an integer.
Right shift	>>	A1=A2>>A3	Shift the value of A2 to the right by A3 digits and return the displacement result. The type of the return value is an integer.
Bit reversal	~	A1=~A1	Perform a bit reversal on a value. The type of the return value is an integer.

PI_GetTickCount

Function

PI_GetTickCount (A1, A2)

Description

Writethe startingtime tothe set address asa 32-bit integer.

Parameters

- **A1:** Source data could only be the HMI internal or external register starting with "@".

- **A2:** =0: Unit of time for returning 0ms;(the value will become 0 after 49.7 days and so on)
    - =1: Unit of time for returning 10 ms;(the value will become 0 after 497 days and so on)
    - =2: Unit of time for returning 100 ms;(the value will become 0 after 4970 days and so on)
    - =3: Unit of time for returning 1000ms;(the value will become 0 after 49700 days and so on)

✎**Note:** If user restarts the hmi,all value will be 0.

Example

PI_GetTickCount("@W_HDW100",0)'save the starting time in HDW100 address as a 32-bit integer.

Result: HDW100=123456(different returned data for each moment)

Power

Function

var = power (expr1, expr2)

Description

The value of [expr2] to the power of [expr1] will be assigned to Var.

Parameters

- **var:** returned value.
- **expr 1:** base number.
- **expr 2:** power number.

Example

Dim a as floating

a=power (2, 3) 'the value of 3 to the power of 2 is assigned to a.

Float2D("@W_HDW10",a) 'assign the float value of a to @W_HDW10
Result: @W_HDW10=8

PrintText

Function

PrintText(A)

Description

Print the content of A or locates in A.

Parameters

A: source data. A could be a variable or a string (text information),not a register address.

Example

A is text information

PrintText("HMI 8070”)

Result: Printer will print out “HMI 8070”

- A is variable

Dim a as string

a= “HMI 8070”

PrintText(a)

Result: Printer will print out “HMI 8070”

✎**Note:**

source data length range:1-128 characters.

RadToDeg

Function

Var= RadToDeg(expr)

Description

Convert radiant value to degree, then assigned to Var.

Parameters

- **Var:** return degree value.
- **expr:** input radiant value.

Example

Dim a as floating

a = RadToDeg(pi)□ □'assignt the degree value of ? to a.

Float2D("@W_HDW4",a) 'assign the degree value to address "@W_HDW4".
Result: @W_HDW4=180

RAND

Function

Var = rand(expr1)

Description

Generate a random number.

Parameter

- **Var:** generated random number.
- **Expr1:** the base number.

Example

@W_HDW0=rand(@W_HDW0) 'Set the value of address@W_HDW0 as the base number to generate random number.

Result: @W_HDW0 random number.

ReadAddr

Function

Word = ReadAddr(A1)

Description

Assigned the red value from A1 to word.

Parameter

- **Word:** return value

Example

Dim word as integer

@W_HDW100=10

word = ReadAddr("@W_HDW100") 'Read the value of address @W_HDW100 and assign to word.

@W_HDW200=word
Result: @W_HDW200=10

ReadWordFormFile

Function

ReadWordFormFile (A1, A2, A3, A4, A5, A6)

Description

Read data of specified length from specified file

Parameters

A1: File location (value is an integer);

In HMI:

- =0: Recipe folder in HMI flash;
- =1: Custom folder in UDisk (USB flash disk);
- = 2: Custom folder in SD card.
- =3: Custom folder in HMI flash;

In simulator:

- = 0: D:/Recipe/
- = 1: C:/WECON/CustomFileDir/
- = 2: C:/WECON/CustomFileDir/

- = 3: C:/WECON/CustomFileDir/

✎**Note:** A1 can be an address, variable or value.

A2: File name (value is a string);

Input the file name want to write.

✎**Note:** A2 must be an address, and the length of the file name cannot exceed 32 characters.

A3: Data start address (value is a string);

Input the data want to write

✎**Note:** A3 must be the address.

A4: Data length (value is an integer)

Set the length of the data to be written (unit:word)

✎**Note:** A4 can be an address , variable or value.

A5: Return value (value is an integer)

= 1: Successfully read

= 2: Failed to open the folder

= 3: Read address error

= 4: File reading error

✎**Note:** A5 must be an address.

A6: Folder name (value is a string);

Input the folder where to be written the file.

✎**Note:** A6 can be an address or a variable or a character string, and the length of the folder name cannot exceed 32 characters

Example

ReadWordFormFile(@W_HDW1254,"@W_HDW4200","@W_HDW2000",@W_HDW4000,"@W_HDW4100", @W_HDW4300)'Write the @W_HDW4000 words in the file @W_HDW4200 in the directory @W_HDW4300 into the address @W_HDW2000 specified location by @W_HDW1254

Right

Function

val = Right (string, length)

Description

Return a string of the specified length from the right side of parameter.

Parameter

- **string:** the operated string.

- **length:** the designated number of byte required to return, count from the right side.

Example

@W_HDW103= Right("Hello", 3) 'return "llo"

Result: @W_HDW103="llo"

RTrim

Function

val = RTrim(str)

Description

Clear the empty part on the right side of string [str], then assign the empty part to val

Parameter

- **val:** returned value.
- **str:** the string needs to be operated.

Example

@W_HDW0 = RTrim("   -Hell o- ") 'retrun"   -Hell o-"

Result: @W_HDW0display "   -Hell o-"

S2F

Function

S2F (A1,A2,s1)

Description

S2F is used to translate the string stored in A1 to floating and store the floating number in A2 according to the data format shown in A2.

Parameters

- **A1:** initial data address, used to store the string data, it should be the internal address of HMI or external address that started with “@”,like @W_HDW0
- **A2:** destination address, used to store the floating number data. It should be the internal address of HMI or external address that started with “@”,like @W_HDW0
- **S1:** display format of target data, for example m.nf, m means the length of string is m, n means the decimal places, f is the format used to output single precision number. (Since the floating point number is up to 7 digits, the decimal point in the string is also a bit, so it is recommended that the length should not exceed 8 bits)

Return value: none

Example

The lengTh of string is 8

@W_HDW0="12345.67""Assign the string “12345.67" to HDW1

S2F("@W_HDW0","@W_HDW100","8.2f") 'read string "12345.67 "from HDW0and convert it into a floating point with 2 decimal places, store in the HDW100

Result: @W_HDW100 address displays “12345.67”.

The length of string is less than 8

@W_HDW0="1234.5 67""‘assign the string“1234.567”to HDW1

S2F("@W_HDW0","@W_HDW100","6.2f") ' read string "1234 .5"from HDW0and convert it into a floating point with 2 decimal places, store in the HDW100 .

Result: the floating value of @W_HDW100 is 1234.50

The length of string is more than 8

@W_HDW0="12345.6789""’assign the string “12345.6789” to HDW1

S2F("@W_HDW0","@W_HDW100","8.2f") ' read string "12345 .67" from HDW0 and convert it into a floating point with 2 decimal places, store in the HDW100 .

Result: the floating value of @W_HDW100 is 12345.67

SetB

Function

SetB(A1)

Description

Set the bit A1 ON.

Parameters

- **A1:**Bit address

Example

SetB(@B_HDX100.0) 'Set the address {@B_HDX100.0} ON

Result: @B_HDX100.0=1

SetKeyMap

Function

SetKeyMap(A1,A2,A3)

Description

The key values of the keyboard are mapped so that multiple keyboard buttons perform the same function.

Parameters

- **A1:** The starting address of the source key;It needs to be an address format;
- **A2:** The starting address of the mapped value; It needs to be an address format;

- **A3:** Mapping length (continuous length of mapped address); It needs to be a value, the maximum mapping range: 108 key values;

Example

@W_HDW3000 = 3 ' The starting address of the source key

@W_HDW3001 = 5

@W_HDW3002 = 7

@W_HDW3003 = 9

@W_HDW3004 = 61

@W_HDW4100 = 103 'The starting address of the mapped value

@W_HDW4101 = 105

@W_HDW4102 = 106

@W_HDW4103 = 108

@W_HDW4104 = 28

SetKeyMap("@W_HDW3000","@W_HDW4100",5) ' Map the values of the HDW4000~HDW4004 addresses to the HDW3000~HDW3004 addresses.

Result

Map the value of the HDW4000~HDW4004 address (mapped to 103 105 106 108 28) to the value of the HDW3000~HDW3004 address (source key value 3 5 7 9 61)

Button 2 (key value 3) is mapped to the direction key (key value is 103), button 4 (key value 5) is mapped to the left arrow key (key value is 105), and so on. When using the keyboard, the function of input 2 could be performed on both the button 2 and the direction button.

SignedInt16

Function

val = SignedInt16(A1)

Description

Assign the value to {val} from address A1 which is signed integer.

Parameters

- **A1:** contain signed integer as "@W_HDW000002"
- **Val:** returned value

Example

Dim a as integer 'Integer variable a

a = SignedInt16("@W_HDW0") 'read signed integer from HDW0 addresses **and** assign the value to a

@W_HDW2=a'assign the value a to HDW2

Input: @W_HDW0=-2:

Result: @W_HDW2=-2.

SignedInt32

Function

val = SignedInt32 (A1)

Description

Assign the value to {val} from address A1 which is signed even integer.

Parameters

- **A1:** the address contains signed even integer
- **Val:** Returned value

Example

```
Dim a as integer 'define {a} as a integer

a = SignedInt32("@W_HDW0") 'read signed even integer from HDW0, then assign this value to a.

@W_HDW2=a 'assign the value of a to HDW2

@W_HDW3=a>>16
```

Input: @W_HDW0=-2

Result:

- @W_HDW2=-2
- @W_HDW13=-1

Sin

Function

val = Sin(A1)

Description

Get the sine value of A1, and copy result to val.

Parameters

- **A1:** A1 needs to be an angle.
- **Val:** Returned value.

Example

```
Dim a as floating 'floating variable a,b

a=sin(pi/6) 'return sinb to a
```

Float2D("@W_HDW13",a) 'assign the value of the floating variable a to address HDW13.
Result: @W_HDW13=0.5

SleepA

Function

SleepA(T)

Description

Wait time T(ms).

Parameters

- **T:** wait time, the unit is [ms]

Returned value: none.

Example

SleepA(10) 'wait 10ms

Result: When the script runs to SleepA(10), it means the scripts would go running after waiting 10ms

Sqr

Function

val = Sqr(A1)

Description

Assign a square root value of A1 to val.

Parameters

- **A1:** the data need to be operated
- **Val:** Returned value

Example

@W_HDW0 = Sqr(4) 'calculate the square root of HDW0

Result: @W_HDW0=2

StAndFtChange

Function

StAndFtChange(A1,A2,A3)

Description

Calculate the number of seconds from January 1, 1970 to the current time, and also be invertible.

Parameters

- **A1:** The start address of curren t time (Enter or output year, month, day, minute, and second); It needs to begin with address"@", and occupies 6 addresses;
- **A2:** The number of seconds; It needs to begin with address “@", data format 32-bit unsigned.
- **A3:** conversion method;
  - A3=0, convert time to seconds;
  - A3=1, convert seconds to time;

Returned value: none;

Example

Script 1

StAndFtChange("@W_HDW10","@W_HDW20",0) ‘use HDW10 as start address, and enter year, month, day, hour, minute, second. The script calculates the number of seconds from January 1, 1970 to the time of the entry, and stores the result in HDW20

Input: HDW10 = 2017,HDW11 = 12, HDW12 = 9 , HDW13 = 15, HDW14 = 15, HDW15 = 0

Output: 1512832500

Script 2

StAndFtChange("@W_HDW30","@W_HDW20",1) ‘read number of seconds from HDW20, and the script calculates the date time, and stores the result start from HDW30

Input: 1512833760

Output: HDW30 = 2017, HDW31 = 12, HDW32 = 9, HDW33 = 15, HDW34 = 36, HDW35 = 0

Sub

Function

Sub name (arglist)

statements

End Sub

Description

Declare the name, parameters and codes of the Sub (sub function)

Parameters

- **Name:** naming rules refer to variable.
- **Arglist:** variable list.
- **Statements:** the code set of the sub function.

Example

```
sub samesub(a,b as integer) ' samesub and integer variable a,b

c=a+b
```

```
@W_HDW0=c

endsub

samesub(1,12) 'call function samesub
```

Result: @W_HDW0=13

SWAP

Function

SWAP(A1,length)

Description

Swap the big-endian with the little-endian from address A1, swap length is adjustable.

Parameters

- **A1:** the swapped high endian, need to be an address as HDW_000002.
- **Length:** swap length.

Returned value: None.

Example

```
@W_HDW103=0x1234 'assign value to HDW103

@W_HDW104=0x2345 'assign value to HDW104

@W_HDW105=0x2565 'assign value to HDW105

@W_HDW106=0x2675 'assign value to HDW106

SWAP(@W_HDW103,4) 'swap the high and low endian for the 4 adjacent addresses start with HDW103.
```

Result:

- @W_HDW103=0x3412
- @W_HDW104=0x4523
- @W_HDW105=0x6525
- @W_HDW106=0x7526

Tan

Function

val = Tan(A1)

Description

Get the returned tagent value of A1, and then assign to val.

Parameters

- **A1:** A1 needs to be an angle.
- **Val:** Returned value.

Example

```
Dim a as floating 'define a floating variable a

a=TAN(pi/3) 'calculate the tangent value of pi/3 and assign to a

Float2D("@W_HDW16",a) 'assign the value of a to HDW16
```

Result: @W_HDW13=1.732

Trim

Function

val = Trim(A1)

Description

Return A string in A1 without empty string next to it.

Parameters

- **A1:** The operated string
- **val:** Returned value

Example

```
@W_HDW1=Trim(" ab ")
```

Result: @W_HDW1="ab"

UCase

Function

val = UCase(A1)

Description

Capitalize the string data, and then assign the value to val.

Parameters

- **A1:** Operated string, address or variable.
- **Val:** Returned value

Example

```
@W_HDW1=ucase("abcd") 'Capitalize abcd then assign the value to HDW1
```

Result: @W_HDW1="ABCD"

Variable

Description

A variable is any factor, trait, or condition that could exist in differing amounts or types.

Define variable

Use Dim to define variable in script. The variable could be string, floating, integer.

Example:

```
Dim a as floating ‘define variable {a} as a floating.

Dim b,c,d as integer‘define variable {b,c,d} as integer
```

Naming rules

The first letter needs to be English letter.

No symbols.

Maximum character length 15 allowed.

W2B

Function

W2B(A1, A2, A3)

Description

Replace the high endian of [A2]+1 with the high endian of A2.

Parameters

- **A1:** operated address.
- **A2:** source address.
- **A3:** the conversion length.

Returned value: none.

Example

```
@W_HDW0 = 4660 'assign 16bit value 1234 to HDW0.

@W_HDW1=0x5678 'assign 16bit value 5678 to HDW1.

@W_HDW2 = 0x2425 'assign 16bit value 2425 to HDW1.

@W_HDW3 = 0x3536 'assign 16bit value 3536 to HDW0.

@W_HDW4 = 0x1415 'assign 16bit value 1415 to HDW0.

W2B(@W_HDW20,@W_HDW0, @W_HDW10)

@W_HDW10=1 ‘save the high endian {34} of HDW0 to HDW20.
```

Result: @W_HDW20=0x34, @W_HDW21=0, @W_HDW22=0

W2D

Function

W2D(A1, A2)

Description

Convert the unsigned Word to unsigned Dword and save the result in A1.

Parameters

- **A1:** operated address .
- **A2:** source address.

Returned value.

Example

Unsigned decimal word

@W_HDW0 = 1234 'assign 1234 to HDW0.

W2D(@W_HDW2, @W_HDW0) 'convert unsigned word {1234} from HDW0 to Dword and save in HDW2

Result: @W_HDW0=12345, @W_HDW2=12345, @W_HDW3=0

Signed decimal word

@W_HDW0 = -12345 'assign value to HDW0: convert {-12344} to unsigned decimal word is {53191}.

W2D(@W_HDW2, @W_HDW0) 'save unsigned Dword to HDW0

Result: @W_HDW0=-12345,@W_HDW2=53191,@W_HDW3=0

W2F

Function

A1 = W2F (A2)

Description

Convert a 16bit integer to a 32bit floating, and then save to the next word of A1.

Parameters

- **A1:** operated address.
- **A2:** source address.

Returned value: none.

Example

A1, A2 are addresses

@W_HDW0 = 1234 ' assign unsigned word {1234} HDW0 @W_HDW1=W 2F(@W_HDW0) ‘ Convert {1234} to a 32bit floating and then save to HDW1, HDW2.

Result: @W_HDW1=1234'32bit floating

A1 is an address,A2 is variable

```
dim a as integer

a=134 'define a integer 134 to a,

@W_HDW2=W2F (a) 'convert to 32bit floating save to HDW1, HDW2.
```

Result: @W_HDW1=134' 32bit floating

W2S

Function

W2S(A1,A2,S1)

Description

Convert integer word in address A1 text as S1 format, and then save to A2.

Parameters

- **A1:** operated address.
- **A2:** source address.
- **S1:** saving format.
  - d format: Decimal format.d: Real data length.Md: Designated data length. 0md: Designated data length if the length is shorter than m adds 0 at the left.
  - format: Unsigned octal format. Mo and 0mo is also applied.
  - x format: unsigned Hex integer format ?Mx and 0mx is also applied.
  - c format: ASCII format.

Example

Decimal format

```
@W_HDW1=1456'assign value {1456} to HDW1.

W2S("@W_HDW1", "@W_HDW10", "6d") ' convert{1456} to decimal text and save to HDW10.
```

Result: @W_HDW10 shown "1456"

0md

```
@W_HDW1=1456 ' assign value {1456} to HDW1

W2S("@W_HDW1", "@W_HDW10", "06d") ' convert{1456} to integer decimal text and add 2 {0} on the left of the data then save to HDW10.
```

Result: @W_HDW10 show text "001456"

WaitEthernetStart

Function

WaitEthernetStart (A1)

Description

Waiting for Ethernet to start, it will extend the HMI start up time (only added in PI i series, Ethernet start up is earlier than HMI in other PI series)

Parameters

- **A1:** Waiting timeout (1~20s)
  - If A1=0, the wait timeout is 10s;
  - If A1>20, the wait timeout is 20s;

Returned value: None

Example

WaitEthernetStart (15)

‘The maximum waiting time is 15 seconds. If Ethernet is not started within 15 seconds, HMI will start the system and no longer wait for ethernet.

WHILE ... WEND

Function

While condition

[statements]

Wend

Description

If the condition is true, then all the commands before Wend in the statement will be executed then recheck the condition, if the condition is false, the command after Wend will be executed.

Parameters

- **Condition:** Number or string, the result represents as True or False.

Returned value: None.

Example

```
while @W_HDW1>50 'the condition is the value of HDW1 bigger than 50.

@W_HDW1=@W_HDW1-1 'when the condition is true, execute subtract 1 from 1HDW.

wend

@W_HDW2=@W_HDW2+1 ' when the condition is false, execute add 1 from 1HDW.
```

Result: If HDW1=60, after executed; HDW1=50, if the condition is true.

WriteAddr

Function

WriteAddr(A1,A2)

Description

Assign the value from A2 to address A1.

Parameters

- **A1:** operated address
- **A2:** source address

Returned value: None.

Example

```
dim f as integer ' integer f

f=13 ' assign the value 13 to f

WriteAddr("@W_HDW1",f) ' write the value to HDW1.

WriteAddr("@W_HDW10",@W_HDW2) ' write the value from HDW2 to HDW10.
```

Result:

- HDW1=13
- HDW10= HDW2'IF HDW2=1456,Then HDW10=1456;IF HDW2=-123,Then HDW10=-123

WriteWordToFile

Function

WriteWordToFile (A1,A2,A3,A4,A5,A6)

Description

Write data of designated length to specified file

Parameters

A1: File location (value is an integer);

In HMI:

- =0: Recipe folder in HMI flash;
- =1: Custom folder in UDisk (USB flash disk);
- = 2: Custom folder in SD card.
- =3: Custom folder in HMI flash;

In simulator:

- = 0: D:/Recipe/
- = 1: C:/WECON/CustomFileDir/
- = 2: C:/WECON/CustomFileDir/
- = 3: C:/WECON/CustomFileDir/

✎**Note:** A1 can be an address, variable or value.

A2: File name (value is a string);

Input the file name want to write.

✎**Note:** A2 must be an address, and the length of the file name cannot exceed 32 characters.

A3: Data start address (value is a string);

Input the data want to write

✎**Note:** A3 must be the address.

A4: Data length (value is an integer)

Set the length of the data to be written (unit:word)

✎**Note:** A4 can be an address , variable or value.

A5: Return value (value is an integer)

- = 1: Successfully written
- = 2: Failed to open the folder
- = 3: Read address error
- = 4: File writing error
- = 5: The file already exists

✎**Note:** A5 must be an address.

A6: Folder name (value is a string);

Input the folder where to be written the file.

✎**Note:** A6 can be an address or a variable or a character string, and the length of the folder name cannot exceed 32 characters

Example

WriteWordToFile(@W_HDW1254,"@W_HDW4200","@W_HDW1000",@W_HDW4000,"@W_HDW4100", @W_HDW4300)'Write the @W_HDW4000 words in the address @W_HDW1000 to the file @W_HDW4200 in directory @W_HDW4300 specified location by @W_HDW1254