FATEK

M Series

Programmable Controller

Structured Language ST User Manual

Since the content of the manual will be revised as the version changes, this version may not be the final version.
To download the latest version of the manual, please go to the technical support area of www.fatek.com

FATEK AUTOMATION CORP.

Index

Preface

Before using the product, be sure to read this Manual carefully in order to get familiar with and understand its content. Should you have any questions or comments, please contact the FATEK distributor for detailed warranty services and responsibility limit.

Precautions on Using the Product

Compliance with the application-related conditions

The user shall evaluate the suitability of FATEK product and shall install the product in the well-designed equipment or system.

The user needs to check if the system, machinery or device currently used is compatible with the FATEK product. If the user fails to confirm the compatibility or the suitability, then FATEK shall not be liable for the suitability of the product.

When required by the customer, FATEK shall provide correlated third party certification to define the value rating and the application restrictions that will be applicable for the product. However, the aforesaid certification message shall not be considered as sufficient to determine the suitability of the FATEK product, the final product, the machine, the system and other applications or relevant combinations. Described below are certain applications that should be cautiously treated by the user. In spite of this, the content described below shall neither be considered as having included all of the intended product purposes nor suggesting that all of the following purposes shall be entirely suitable for the product. For example, outdoors use, use in an area subjected to potential chemical contamination or electrical interference or used under conditions or functions not mentioned in this Manual or used with the system, machine and equipment that may create risks to life or properties.

Before working with the product, the user will be required to check if the entire system is marked with a hazard sign and shall select the design that can ensure the safety such as the backup design, etc. Otherwise, the user shall not be allowed to use the product in the application that will present personnel and the property safety concerns. In no event shall FATEK be liable for the specifications, statutory regulations or restrictions that will be used by the customer in the product combination or the product operations.

When using the CPU Module, FATEK shall not be liable for the programs edited by the user or the resulting consequences.

Disclaimers

Dimensions and weight

The dimensions and the weight specified in the manual are nominal values only. Even if provided with the tolerance, they cannot be used in the manufacturing purposes.

Performance data

The data specified in this Manual mean that the performance data obtained under FATEK' s test conditions are provided for the user to confirm its compliance only. Therefore, the user is also required to consider the actual application conditions. Therefore, actual performance shall be defined according to the content of the guarantee and the limit of responsibilities established by FATEK.

Errors and negligence

The content of this Manual is provided through careful checking process and is considered as correct. However, FATEK shall not be liable for the errors or the negligence that may be found in the text, printing content and proofreading.

Change of specifications

The product specifications and accessories may be subject to change along with the technical improvement or other reasons. In the event that the published specifications or performance need to be changed or where significant structural change is required, FATEK will change the model number of the product accordingly. If certain specifications of the product have changed, then FATEK will not give the notice under the following situation: when it is required to use a special model number or create particular specifications in order to support the customer' s application according to the instructions given by the customer. To confirm actual specifications of the product to be purchased, please contact the local FATEK distributor.

Precautions for safety

Signs and meaning of safety precautions

The following signs will be used in this Manual in order to provide precautions that will be required for using the M-Series PLC safely. These precautions are extremely important for using the product safely. Please read the safety precautions carefully in order to get familiar with and understand the content and the meaning of the aforesaid instructions.

 **Warning**	Means a potentially dangerous situation that will result in death or serious injury if not avoided. In the meantime, it may also lead to serious property losses.

 **Caution**	Means a potentially dangerous situation that may result in minor or medium level injury or property losses if not avoided.

	Means operations that must not be executed.
	Means operations that must be executed.
	Means general precautions.
	Means the precautions relating to hot surfaces.
	Means the precautions related to the wiring, grounding and electrocution of the electrical system.

Warning	
Do not attempt to dismantle any module or touch the internal side of the module when it is under energized status or it may lead to electrocution injury.	
Do not attempt to touch any terminal or terminal board when the module is under energized status, or it may lead to electrocution injury.	
To ensure the system safety in order to avoid abnormal actions that may be caused by man-made external factors or false actions resulting from the faulty PLC, it is required to install the following safety measures in the external circuit (not within the PLC procedure); otherwise, it may lead to serious accident. The externally controlled circuit must be provided with emergency stop switch, interlocking circuit, limit switch and similar safety measures. The PLC will stop outputting the signals when encountering major failure alarm during the operations. However, the errors in the I/O controller and the I/O register as well as other undetectable errors will still trigger unexpected actions. To deal with the aforesaid errors, you are required to install external safety measures to protect the system safety. If the output relay is jammed, burnt or if the output transistor is damaged, then the PLC may still maintain its output at the ON or OFF status. To solve the aforesaid issues, it is required to install external safety measures to protect the system safety. By installing the corresponding safety measures in the system and the equipment, it allows you to maintain the safety of the entire system in spite of the fact that communication errors or false actions have occurred during the operating process.	
The user must take corresponding failure preventive measures in order to ensure safety when the signal line is damaged or when the power is instantly disconnected or when the signal is wrong, missing or abnormal as may be caused by other reasons. If failing to taking the appropriate measures, it may lead to improper operations that may result in serious accidents.	

Precautions	
Do not touch the power module when the PLC is under energized status or when the power source is disconnected. At this time, the power module might still present extremely high temperature that can cause a scorching injury.	
When connecting with the terminal board of the power module, the cable should be secured with the appropriately sized Ferrule. If the cable is loose, it may lead to burning or the failure of the power module.	
The online editing shall be allowed only after confirming that the extended PLC cycle duration will not result in any adverse impact or the system may not be able to read the input signal.	
After confirming that the I/O terminal is safe, you may transmit the required parameters to other terminals such as PLC setting, I/O table and I/O register data, etc. Otherwise, it may lead to unexpected actions if transmitting or modifying the aforesaid data before that.	

Precautions for Use

When using the M-Series PLC, please observe the precautions provided below.

Using the power

- Please use the voltage specified in the Manual. Incorrect voltage will lead to false action or burning damage to the equipment.
- If the number of the module being connected exceeds the current rating of the power module, you may not be able to start the CPU module or other modules.
- Please use the designated power source and then supply the power according to the specified voltage and frequency rating. Special attention should also be given to the location subjected to unsteady power supply, as incorrect power supply may result in false action.
- Before starting any of the following operations, be sure to disconnect the PLC power; or it may lead to false action or electrocution injury.

  (1) When installing or dismantling power module, I/O module, CPU module or any other type of module.

  (2) When connecting cables or executing the system wiring.

  (3) When connecting or disconnecting the connector.
- When using the power module, be sure to observe following precautions.

  (1) The voltage applied at the equipment output point or the connected load shall not be higher than the rated specifications established for the power module.

  (2) If it is required to put aside the power module for over 3 months, it shall be stored in a cool and dry location in order to maintain its function at normal status.

  (3) If the power module is improperly installed, it will result in the accumulation of heat as to cause the aging or the damage of the component within. Therefore, it shall be properly connected and you are also required to use the standard installation method.

Installation

- Do not install the PLC at the location near a high frequency noise interfering source.
- Confirm that the terminal board, the connector, the memory card, the peripheral communication wires and other buckle-mounted devices are latched in position. Improper latching will result in false action.
- After connecting to the adjacent module, the buckle at the top or the bottom must be securely locked (*i.e.,* properly latched). If failing to lock the buckle tightly, the module may not be able to achieve the intended function.

Wiring

- Please follow the instructions provided in the Manual in order to execute the wiring operations correctly.
- Before connecting the power, please check the setting status of all wires and switches. Incorrect wiring may result in burning damage to the equipment.
- After checking the installation position, you may start installing the terminal board and the connector.
- During the wiring process, the label should be tagged on the module. If you tear off the label, foreign mattes may get into the module as to cause a false action.
- To ensure normal heat dissipating function, please tear off the label after completing the wiring operations. If retaining the label, it may lead to false action.
- Please use an EU-standard terminal to execute the wiring operations. Do not connect the terminal with bare stranded wires. The aging or the breaking of wires may result in burning damage to the equipment.
- The voltage applied to the input module shall not be higher than the input voltage rating or it may result in burning damage to the equipment.
- The voltage or the load applied to the output module shall not be higher than the maximum switch capacity. The over-voltage or the overload may result in burning damage of the equipment.
- Do not drag or bend the cable excessively. Such action may cause the breaking of the cable.
- Do not place any objects on the cable or other type of wires or it may cause the breaking of the cable.
- Please set the grounding wire correctly for the power module and communication port to avoid communication error and equipment malfunction caused by noise interference.
- It is recommended to use M series dedicated AC power modules to supply power to MPLC related modules.
- It is recommended to use twisted-pair shielded cables for communication cables and ground them properly.

Operating

- Before supplying power to the MPLC to start the operations, ensure that the setting of the data register is correct without any mistakes.
- Before executing any of the following tasks, confirm that it will not bring about any adverse impact on the system; otherwise, it may result in unexpected action.

  (1) When changing the operating mode of the PLC (RUN Mode/STOP Mode).

  (2) When executing compulsory enable/ compulsory disable for any of the data retained in the register.

  (3) When changing the present value of any bit or setting that has been logged in the register.
- Do not attempt to dismantle, repair or modify any module; or it may result in false action, fire or electrocution.
- It is required to protect the PLC from falling or from excessive vibration or impact.
- If the I/O is located at the "ON" position, when switching the "RUN Mode" to the "STOP Mode," the system will set the PLC output at the "OFF" position and then all output actions will be disabled. Please ensure that the external load will not generate hazardous factors during the aforesaid process.
- If the CPU module stops running due to catastrophic error, please set all of the output points on the output module at the "OFF" position. The output status will be retained after being set as the holding-type memory configuration parameters.
- If the status monitoring pages or the parameters are improperly set, it may result in unexpected action. Even though the status monitoring pages or the parameters are correct, it is also required to confirm that the controlled system will not be subject to adverse impact before starting.
- When applying maximum level of voltage or when the power supplied to the operating switch is interrupted suddenly during the Insulation Strength Test, it may result in the damage of the CPU module. In this case, please use the variable resistor to increase or reduce the voltage level gradually.
- Before conducting the Withstand Voltage Test or the Insulation Resistance Test, please separate the wire grounding terminal of the power module from the functional grounding terminal. Otherwise, it may result in burning damage to the equipment.

Precautions for the Application Environment

- Please follow the instructions described in this Manual for carrying out the installation activities correctly.
- Do not operate the control system in any of the following locations:
  (1) The location exposed to direct sunlight.
  (2) The location with temperature or humidity exceeding the specified range.
  (3) The location vulnerable to dewing effect due to abrupt temperature changes.
  (4) The location exposed to corrosive or combustible gases.
  (5) The location exposed to dust (especially iron chips) or smoke.
  (6) The location exposed to water, oil or chemicals.
  (7) The location vulnerable to impact or vibration.
- When installing the system in any of the following locations, appropriate and effective preventive measures should be taken:
  (1) The location exposed to electrostatic or other type of noise.
  (2) The location exposed to strong electromagnetic field.
  (3) The location that may be exposed to radioactive pollution.
  (4) The location near the power supply source.

1

Understand Structured Language ST

1-1 Features of ST Language

In the early days of automation control, when editing the logic of the programmable logic controller, it was necessary to insert the program short code (Mnemonic) similar to the combination language into the controller through the writer, and the action required by the project has been achieved. Following the evolution of the industrial environment, The control loop Ladder Diagram (LD) was developed to express the logic of the program, so that operators who are not good at writing programs can write and control PLC in a graphical way.

The logic that the current PLC can control and run is becoming more and more complex. In addition, writing programs has become more and more popular. PLC programs written in text have gradually become popular, which has led to the creation of programming syntax similar to Pascal and C. As long as learned, people in the information field can easily start programming.

Nowadays, more and more people use structured language (ST) to develop programmable logic controllers, making structured language (ST) one of the most popular automation development tools today.

Mathematical Processing

Mathematical instructions and comparison instructions can be described like general expressions.

■Multiple operations can be written on the same line

It can be described concisely using arithmetic expressions (+, -, etc.), so ST programs are easier to understand than ladder diagrams.

Program Example:

Substitute the average value of R0~R2 in R3.

R3=(R0+R1+R2)÷3

ST

```
R3:=(R0+R1+R2)/3;
```

LD

Complex Information Processing

The control program can be written through syntax such as "if" or "for.while." Compared with the ladder diagram, it can more concisely and clearly describe the complex branch or loop processing of the execution content according to different conditions.

Program Example

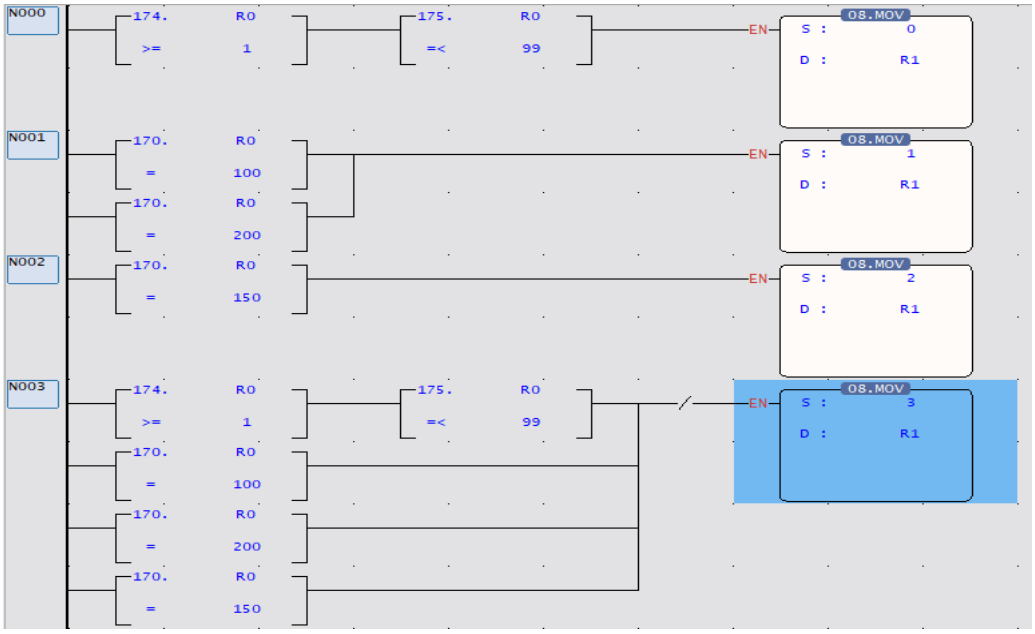
Set 0 to 3 in R1 according to the value of R0

- When 1 ~ 99: R1=0
- When 100 or 200: R1=1
- When 150: R1=2
- In case other than above: R1=3

ST

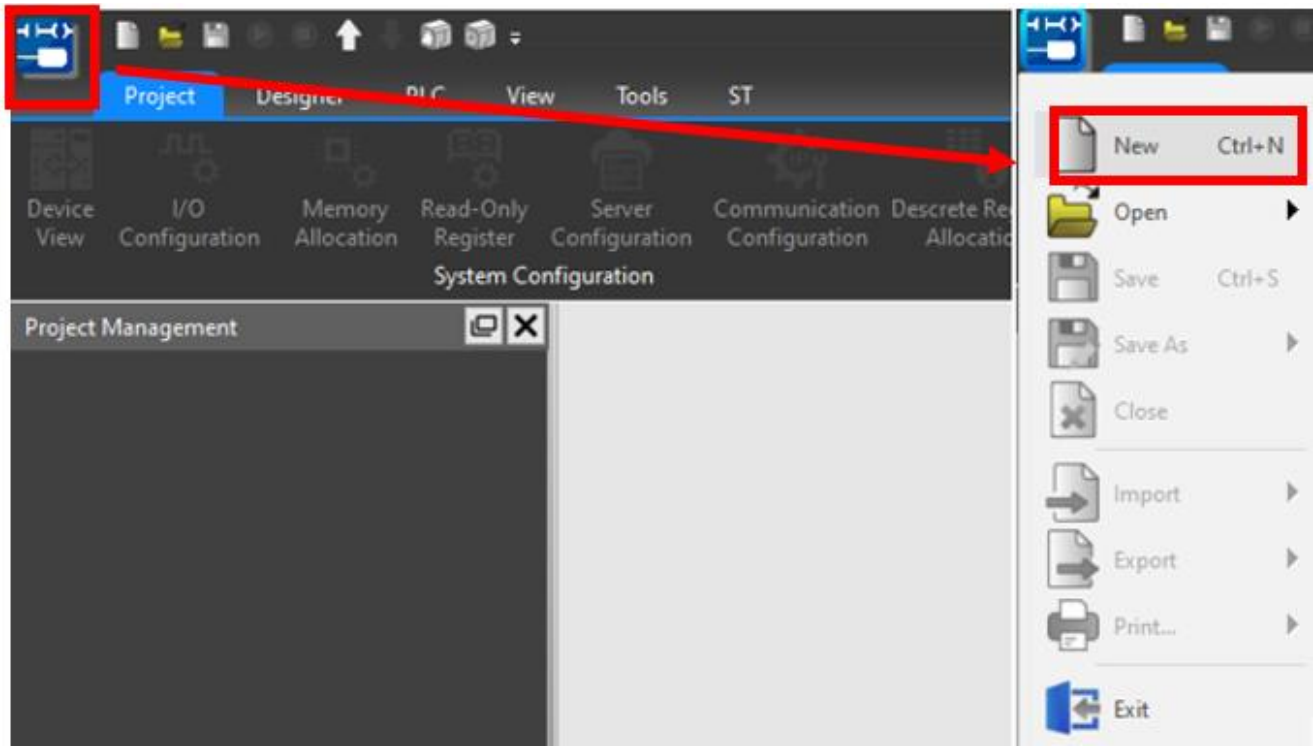
```
IF R0>=1 &R0 <=99 THEN
    R1:=0;
ELSEIF R0=100 or R0=200 THEN
    R1:=1;
ELSEIF R0=150 THEN
    R1:=2;
ELSE
    R1:=3;
END_IF
```

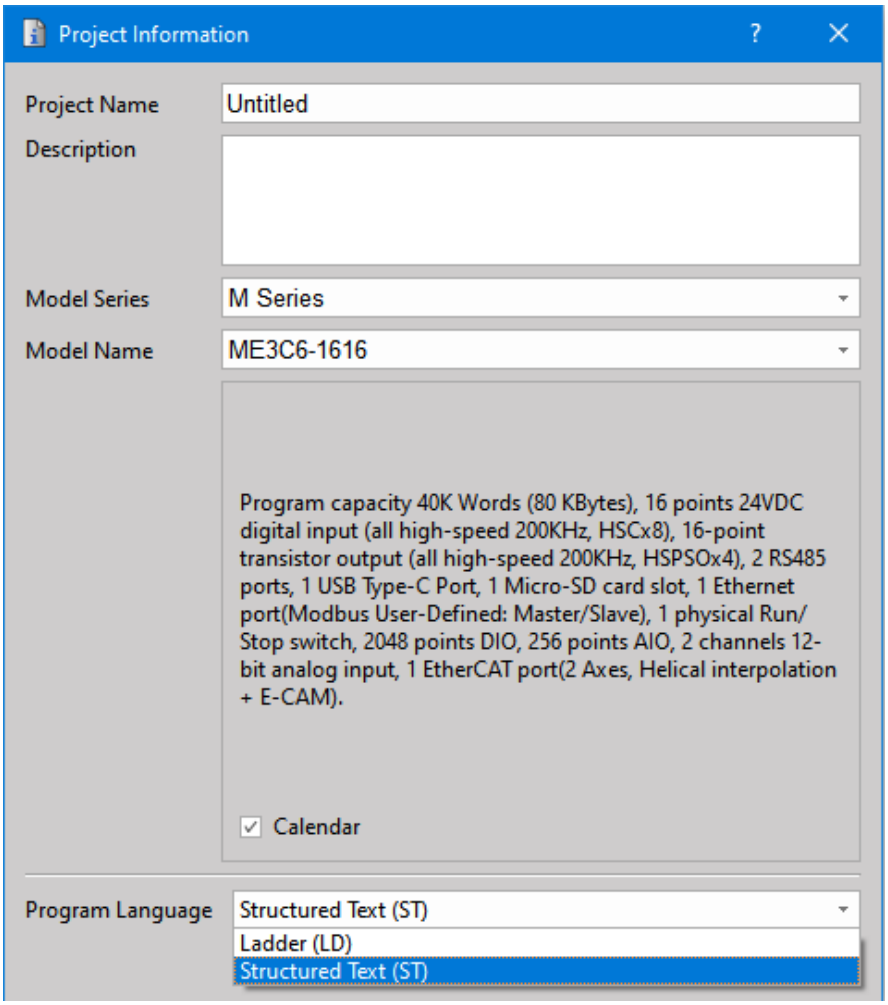
LD



1-2 Adding ST Language Program

Establishing a new Program

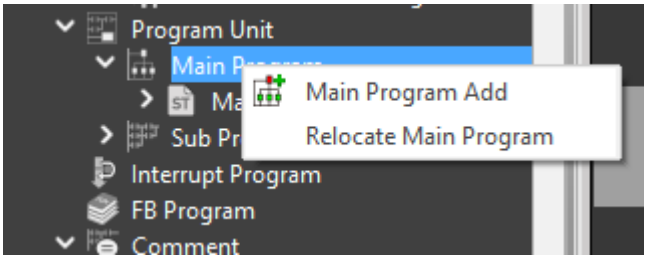


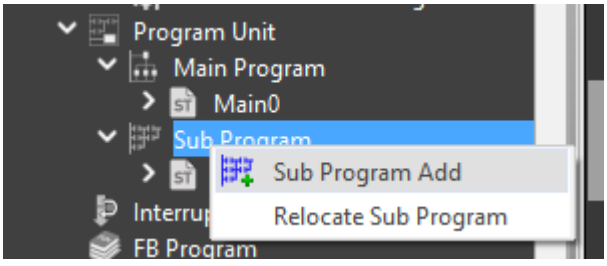


Example of creating MainUint/SubUnit(ST):

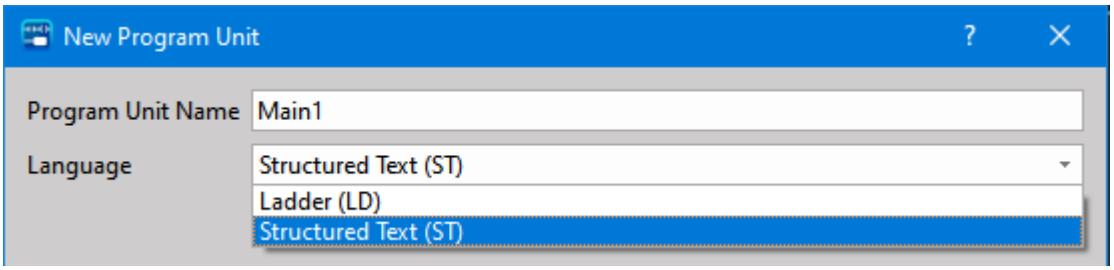
Right click on [Main Program] , or [Sub Program].

Select [Main Program Add] or [Sub Program Add] and a dialog box will pop up.





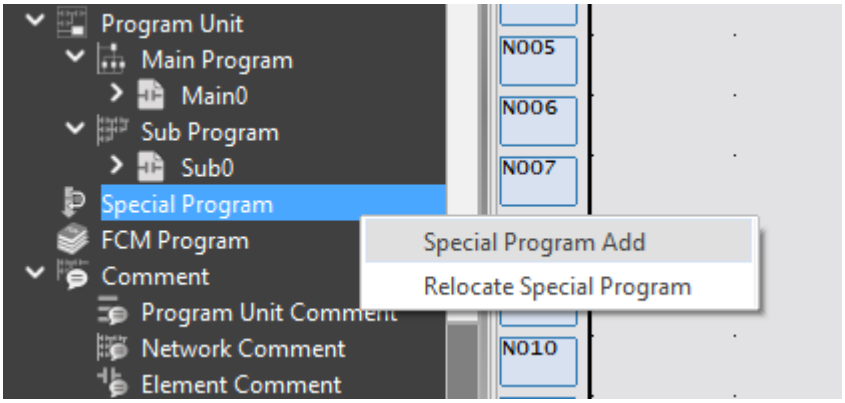
Select [Structured Text(ST)] and add the corresponding program.



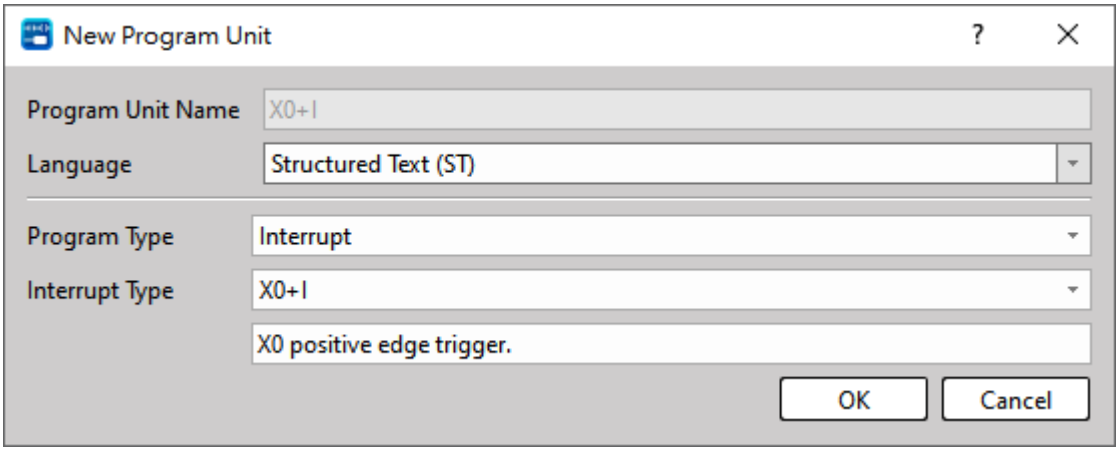
Example of creating an interrupt program (ST):

Click right mouse button on the node of the special program.

Interrupt Program Add



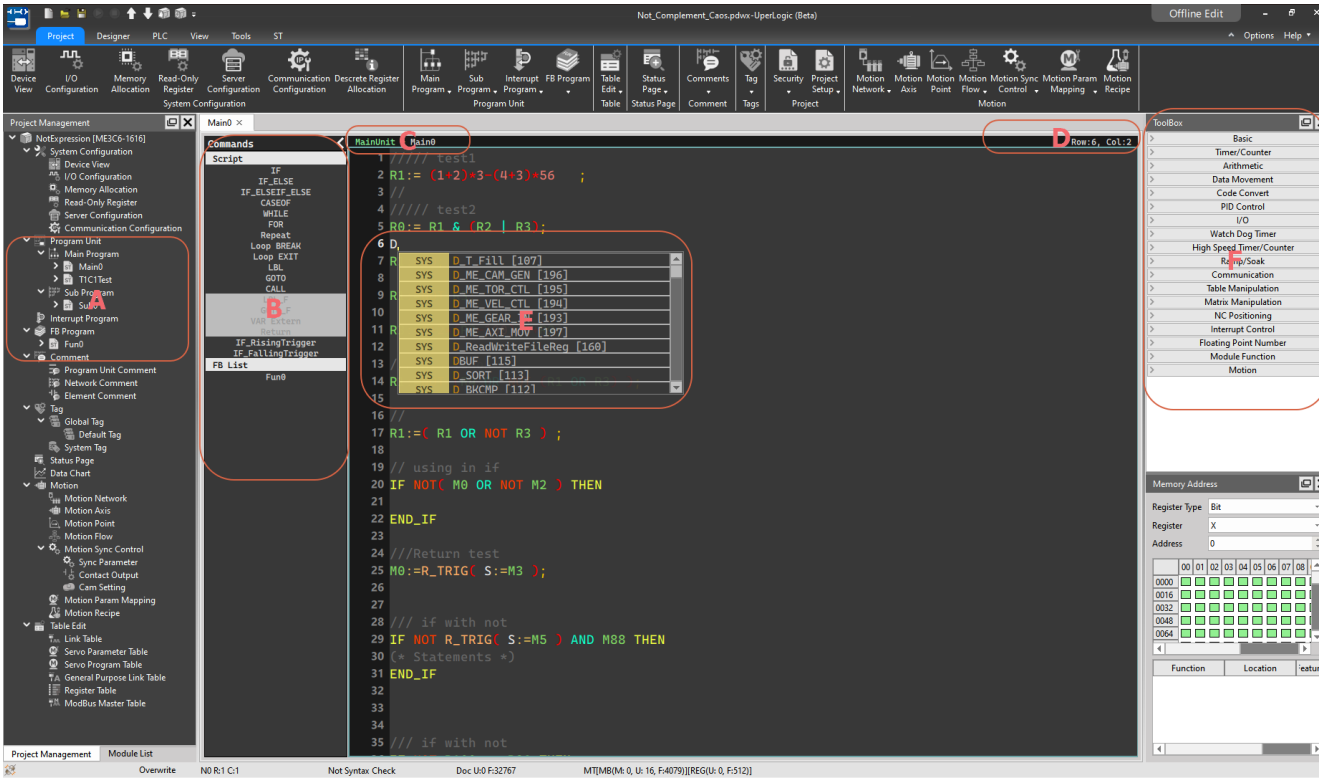
Select the interrupt signal and the program type (ST) to be processed.



2

User Interface of Uperlogic ST

2-1 Interface Overview



A. For all ST programs in the current project, double-click the name with the left mouse button, and a new editing screen will be selected in the middle.

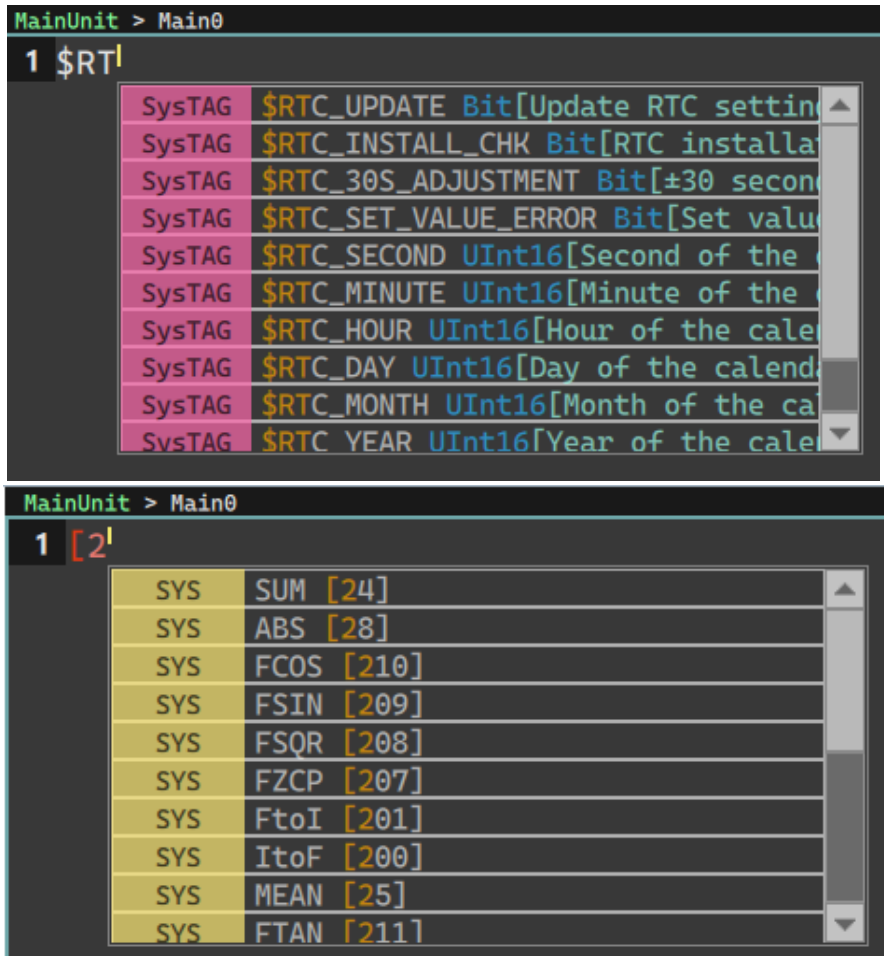
B.

1. Display the currently available ST syntax and FCM-related information. Double-click the field and the corresponding template will be inserted into the program.
2. If the display is grayed out, it means that the command cannot be used in the current text
3. The right border can be dragged to adjust the width, or click the button in the upper right corner to minimize or restore.

C. The current category and name of ST. MainUnit > Main0

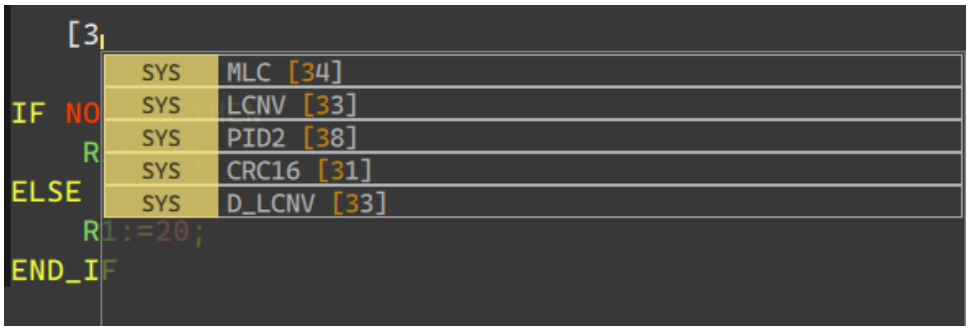
D. The current row and column information of the cursor. Row:17, Col:7

E. Smart pop-up reminder window, which is convenient for users to prompt when typing.



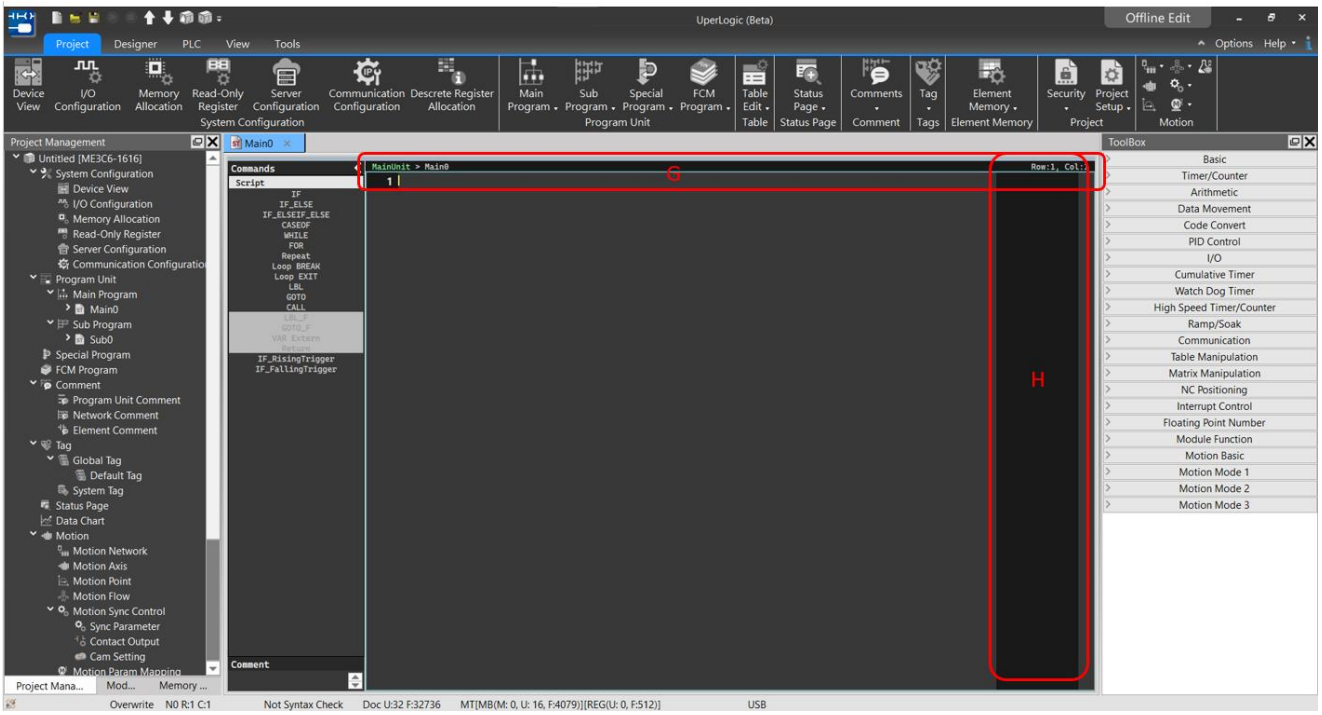
This pop-up window will follow the user’ s input (Match than 3 characters or [+number]) and display the automatically detected prompt program fragments in real time, including available labels, variables, calling functions, etc. Matched strings will be displayed in orange in the window (as shown above).

Since not only the name will be searched, but also the annotations behind the search (the display priority is lower), so in addition to directly searching the name of the function, you can also enter the relevant text of the annotation, such as the function ID (as shown below).

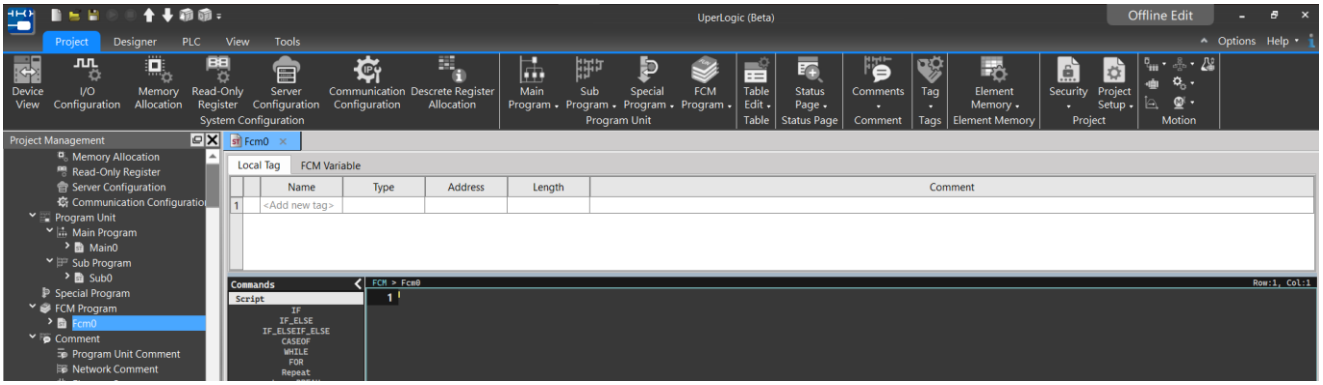


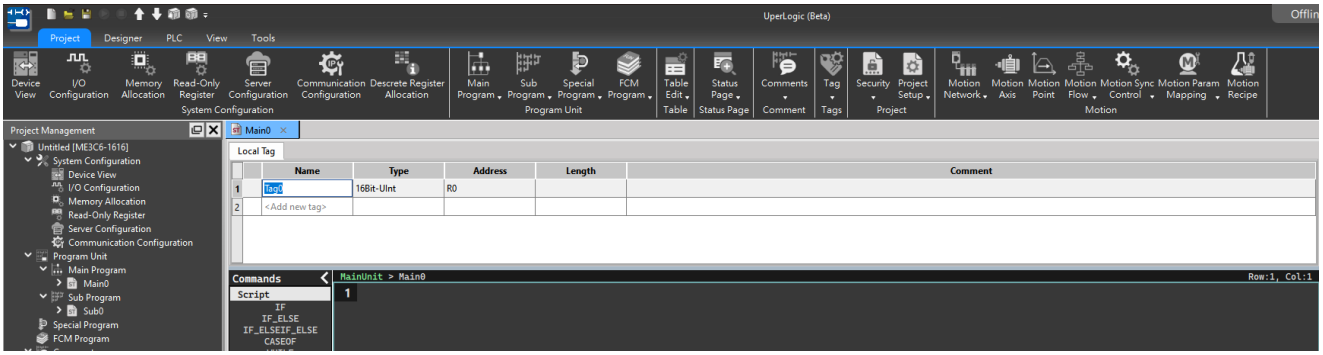
In this way, you can use the mouse to double-click, or use the up and down keys to select the program fragment you want to insert.

F. Callable system commands can be double-clicked or dragged into a text editor



G. Drag this area to display the area label. It will be more convenient to modify the program based on the label when writing ST programs.

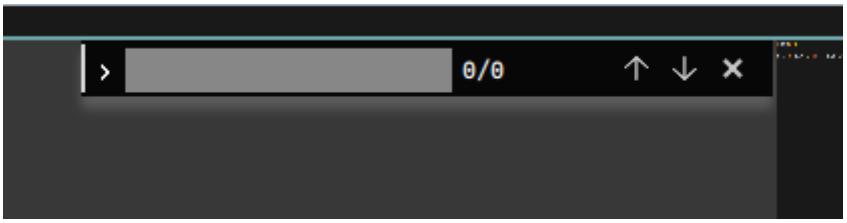
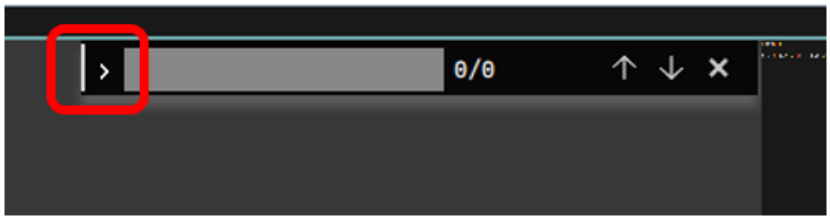
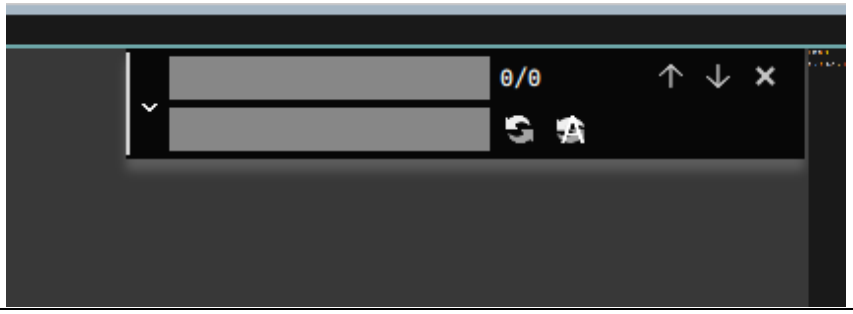




This area supports online editing mode and can be used for program debugging and correction.

H. BirdView ScrollBar, convenient for briefly viewing the program overview

2-2 Supportive Keyboard Instructions

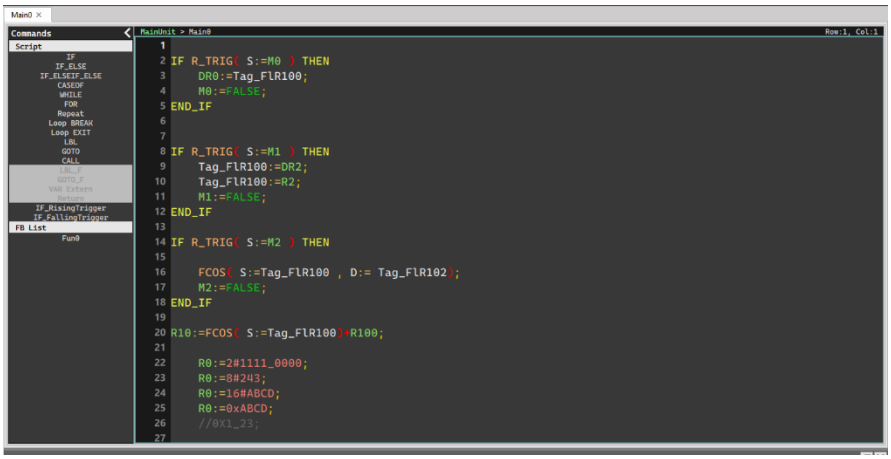
Key	Function
Ctrl + C	Copy
Ctrl + V	Paste
Ctrl + A	Select All
Ctrl + X	Cut
Tab	Insert Tab
Multiple Select+Tab	Select multiple lines and add Tab at the same time.
Shift + Tab	Back Tab simultaneously according to the cursor or the number of selected rows.
Ctrl + F	A search window appears at the top right of the screen, and the current text can be searched.
	Press the " > " button to use the text replace function   The replacement functions are "single replace" and "all replace" respectively. (can only replace the current program unit) 
Ctrl + /	Comment/Inverse Comment in batches according to the number of lines selected.
Ctrl + '+'	Enlarge the whole ST fonts
Ctrl + '-'	Shrink the whole ST fonts

2-3 Sytem Mode

There are currently three modes of the system software:

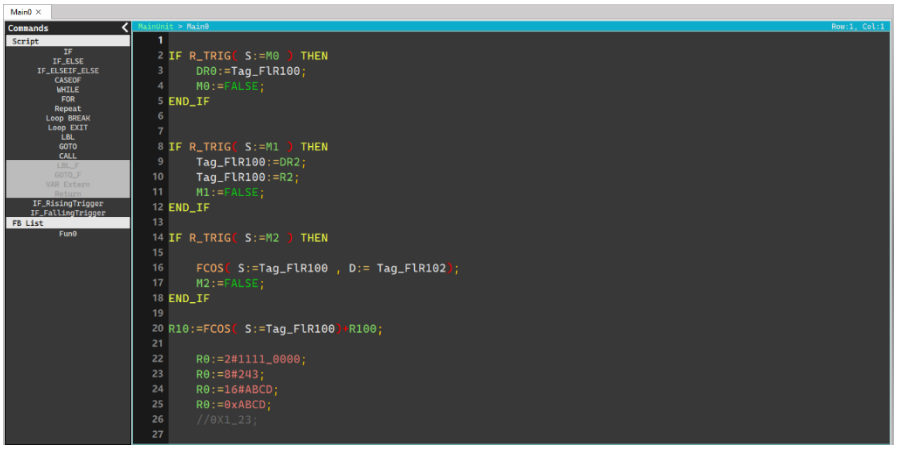
ST Editor diaplays three status correspondingly.

1. Offline Edit(The top of the ST editing window displays a black background)



2. Online Monitor (Read-Only, not for editing)

   (The top of the ST editing window displays a cyan background)

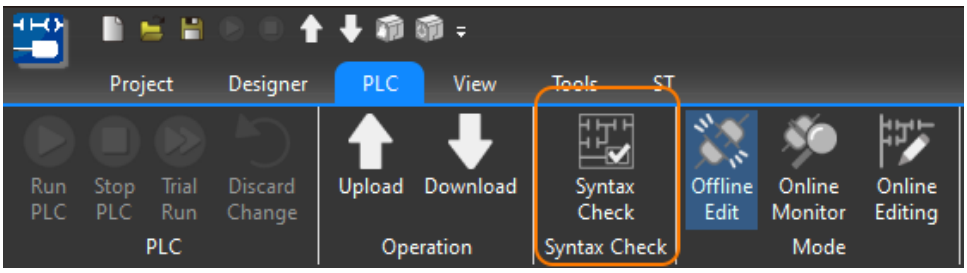


3. Online Edit(The top of the ST editing window displays an orange background)

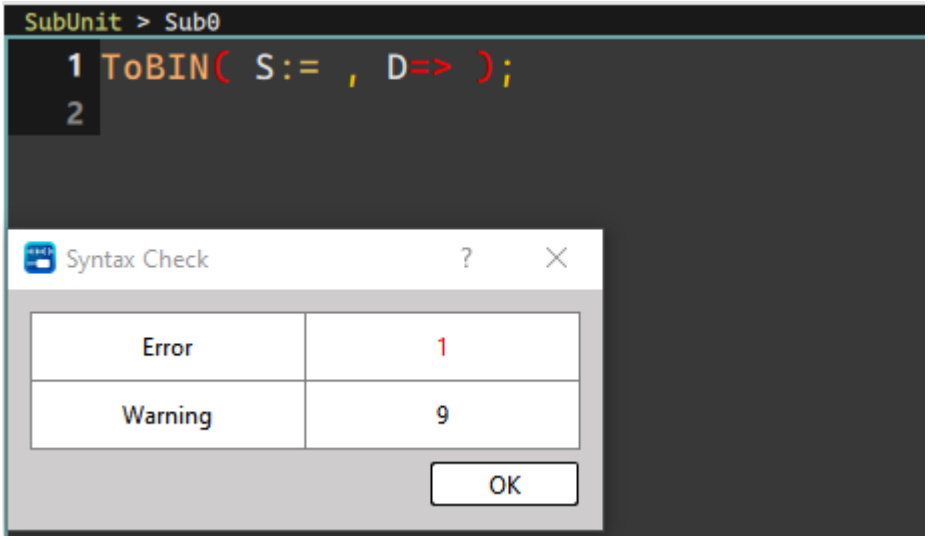
2-4 Syntax Check

ST files must be transferred into programs available for PLC running through syntax check.

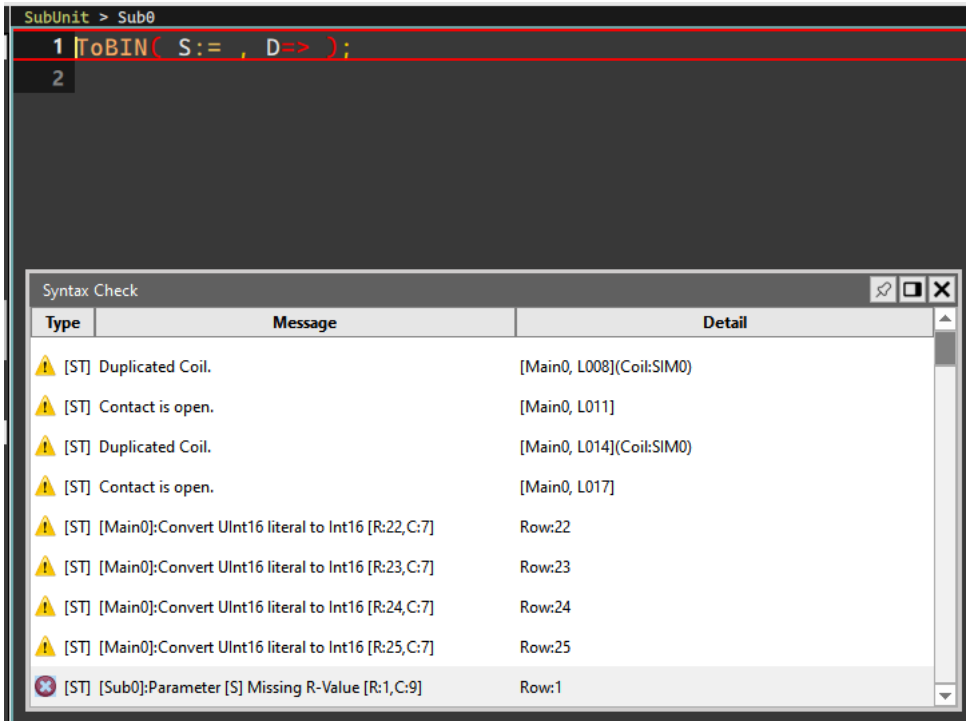
Usually when writing, users can click the button shown below to manually compile to see if it is correct:



If there is an error or a warning prompt, there will be a pop-up window display:



The detailed content will be displayed in the window as shown in the figure below, double-clicking the field will jump and prompt the wrong line numbers.



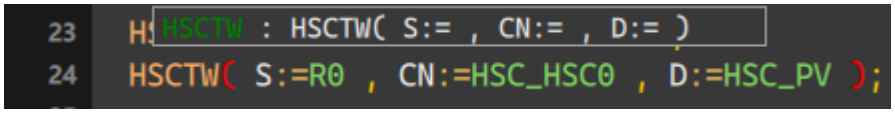
Every time the system downloads, it will automatically run the checking procedure of the current text during the trial run to ensure that the syntax is correct. When there is an error, it will not be able to download or trial run.

2-5 Mouse hover prompt

In order to facilitate debugging and writing programs, the interface supports a prompt window that will appear when the mouse is hovering over variables or function text, giving users tips for writing programs or debugging.

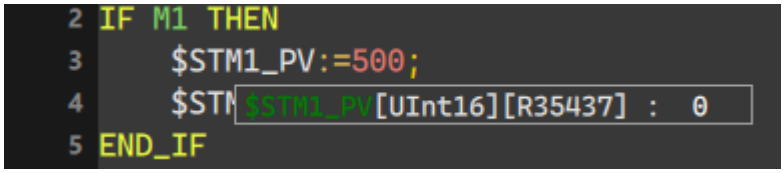
A. General functions

The complete call parameters of the function will be displayed, such as:



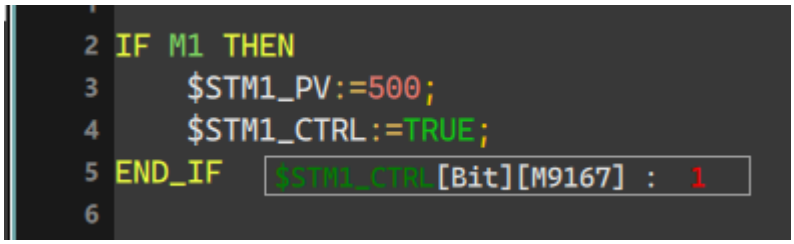
B. **Variables**

The type and value of the variable will be displayed, such as:



The value displayed later can be modified directly by clicking the value behind it with the mouse. Bool type click On/Off switch.

Numeric type: directly enter the corresponding number to modify.



!Attention!

To get the prompt of the above B. variable function, the program text needs to be compiled successfully to get the correct prompt.

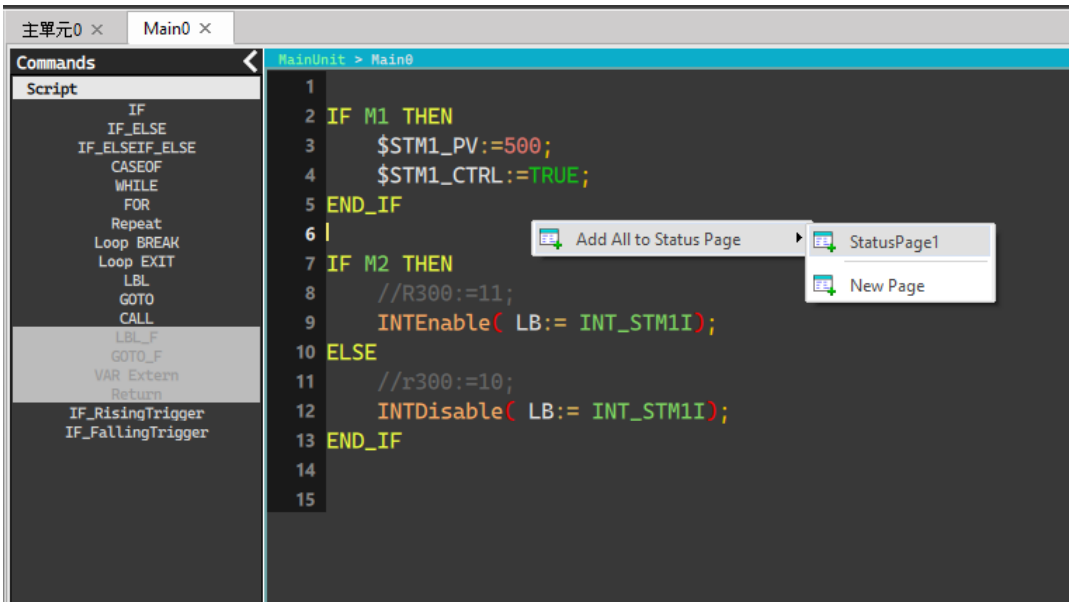
2-6 Add observed variables

In online editing and online monitoring modes, right-clicking on the text editor will pop up a menu where you can add the variable that the mouse cursor is pointing to, or automatically add all variables of the current text to the status observation page. It is convenient for designers to monitor the current value and debug it.

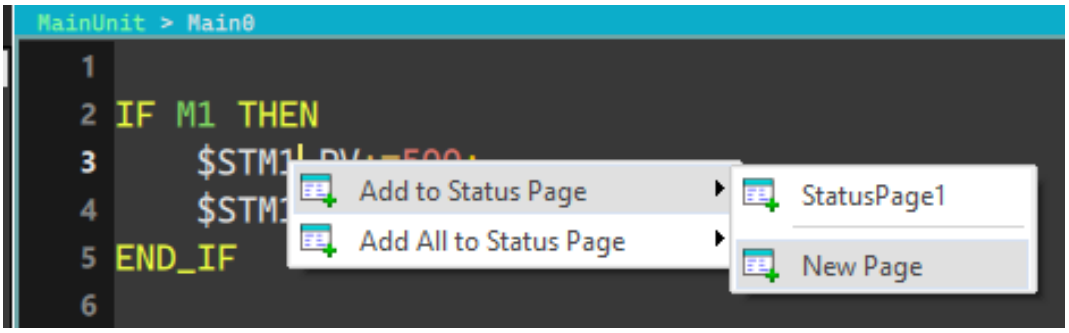
As shown below:

Right-clicking the mouse in a blank area will pop up the option to automatically add all variables to the monitoring page.

And you can choose which page to join in the sub-menu, or create a new page



Right-click on the variable and the current variable will be added to the monitoring page.

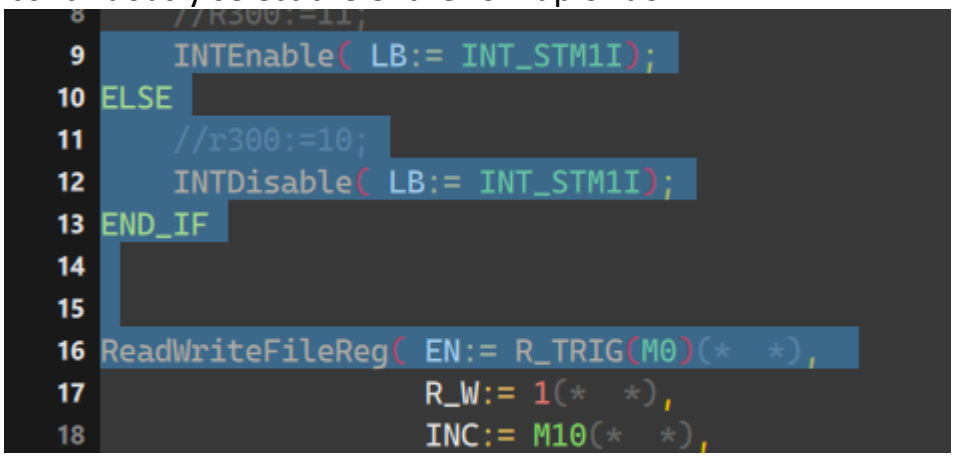


2-7 Quick mouse and keyboard operations

ST text editor provides many common functions to improve the convenience of program editing.

A. Select the entire row

Clicking the mouse where the row number is displayed will select the row, and you can continuously select the entire row up or down.



B. Add Tab/Remove Tab in Batch

Add Tab at the beginning of a single line:

Move the cursor to the beginning of the line and press the Tab key.

Add Tab to multiple lines in batches:

After selecting the multi-line code and then pressing Tab, all lines will have a Tab at the beginning.

Remove Tab from single/multiple lines:

Use the cursor to select any position in the row, or select multiple rows, and then press Shift + Tab to shrink the entire row to the left.

Back-Tab

C. Annotation and anti-annotation

In addition to using the two symbols "(*" and "*)" to make multi-line comments, Ctrl+/ is supported for the game.

The line where the mark is located, or the currently selected line to batch add // comments at the beginning of each line of text.

Added logic:

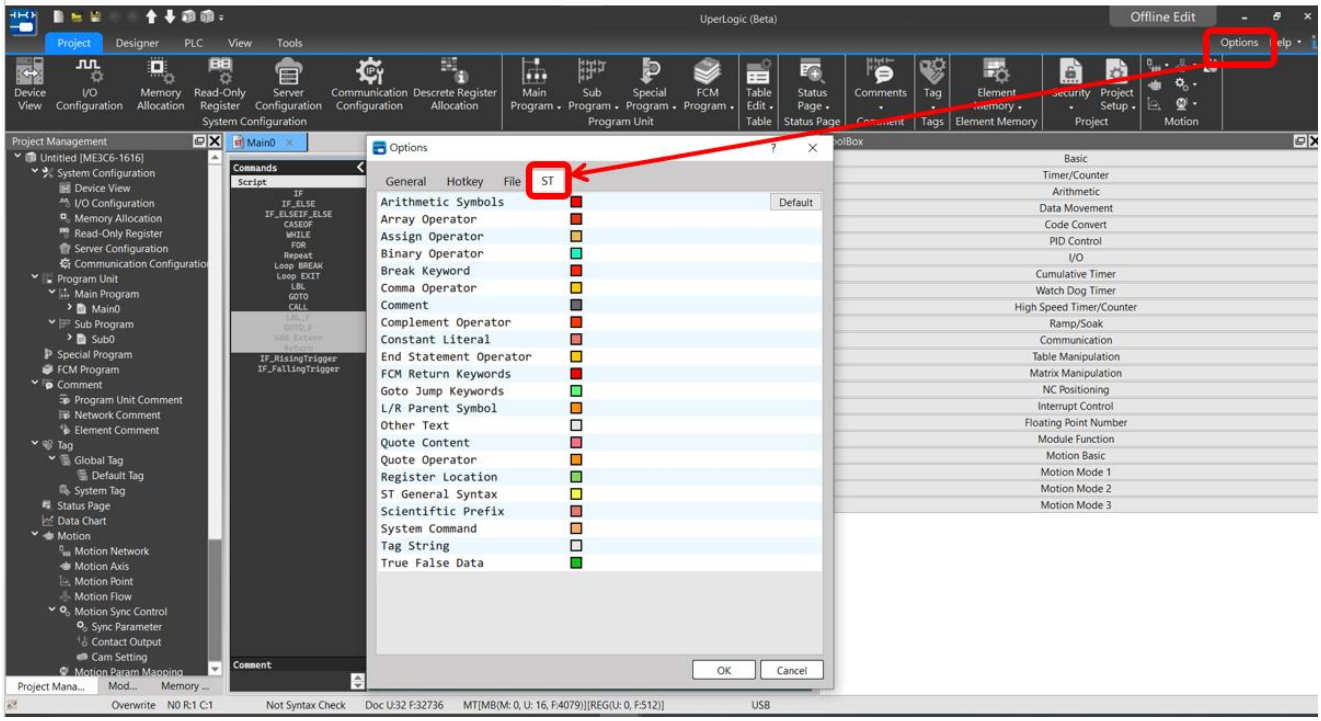
As long as there is a line in the selected line range that does not have a // symbol at the beginning, all lines will be added with // annotations.

Cancellation logic:

The selected number of lines must have a // symbol at the beginning, and batch removal of annotations will be run.

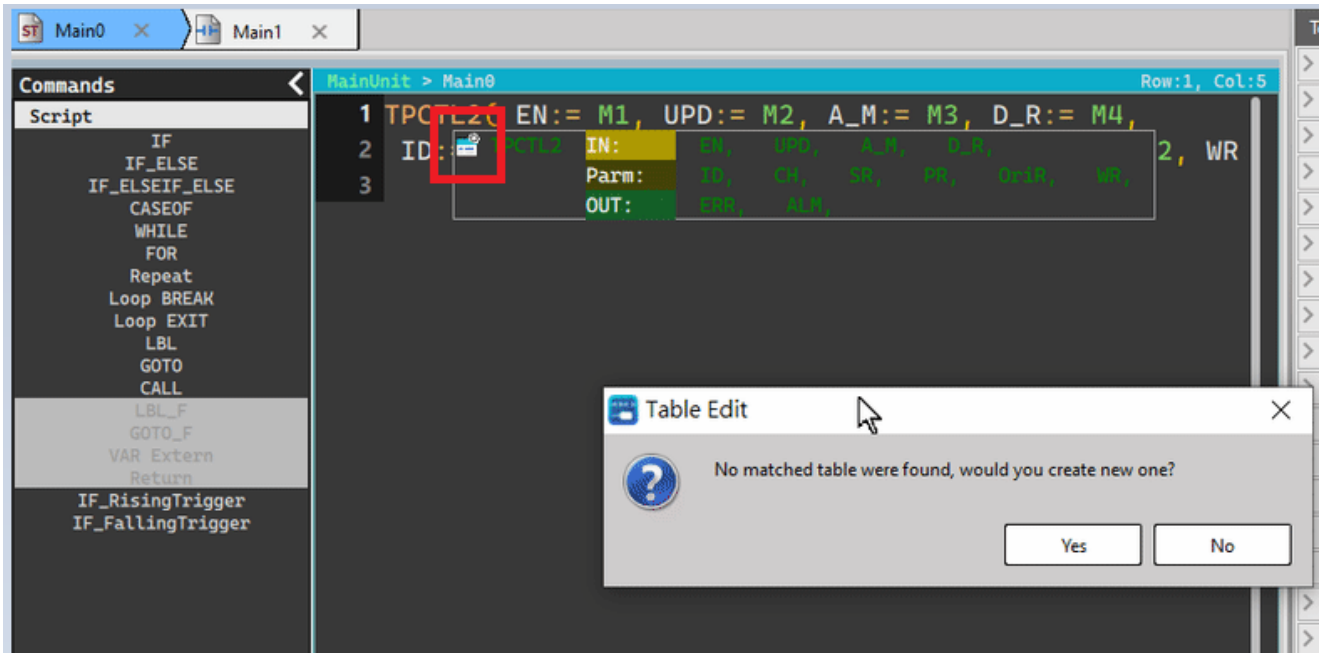
2-8 Structured text color adjustment

The font color of the ST editing environment can be adjusted to suit user habits.



2-9 Quick call/create table

For some commands, click the button in the upper left corner of the pop-up window to call/create the corresponding table.



Support List	Instruction ID
PID2	38
ASCWR	94
TPCTL2	99
HSPSO	140
MPARA	141
MHSPO	147
AHSPO	149
ModBUS	150
CLINK	151
NCR	152
NSSEND	154
NSRCV	155

3

Basic Program Structure of Uperlogic ST

3-1 Introduction

This chapter will explain the methods of applying basic ST Language programming.

3-1-1 Character Encoding

ST editor supports Unicode(encode in UTF-8). Supports basic characters and most symbols in Japanese, English, Chinese and other languages that appear in program editing. In addition to being used for comments, they can also be used in labels or program and table names.

3-1-2 Composing Units

The ST language uses the combination of the following symbols to describe the program. Later chapters will explain in detail.

Type		Example	Referrence
Computing Symbols		+ 、- 、* 、/ 、AND 、& 、OR 、| 、XOR 、MOD 、 <= 、 >= 、<> 、++ 、-- 、<< 、>> 、NOT 、:= 、(、)	3-4 Operand and Operator
Keywords that control syntax (Standard identifier being defined)		IF 、ELSEIF 、END_IF CASE 、OF 、END_CASE WHILE 、FOR 、END_WHILE END_FOR LBL GOTO 、CALL 、 LBL_F 、GOTO_F IF_RISINGTRIGGER IF_FALLINGTRIGGER	3-6 Flow Control and Loop
Identifier	Variables, Hardware Register, IO Discretes, Indirect Address...etc. (Labels, Elements...etc.)	X0 、Y0 、M100 、R0 、DR0 、 、D0 、DD0 、IM0...etc Tag (Register any name)	
	Function Call (FB)	1. System built-in Function 2. (User-created FB Function library)	3-14 Calling FCM Function modes

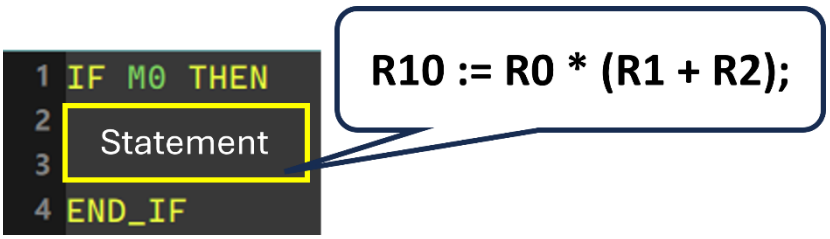
Constant	Interger: Defaulted as "signed int" Ex: **R0**:=1; **R0**:=-2; "Unsigned int" Ex: **Tag0**:=0xFF; String: Use only on Label Goto, LBL...etc Bit (Bool) Type: TRUE、FALSE Float: 32-bit Float Constant ex: TagFloat1:=1.1;	3-7-4 Constant
Delimiter	":" appears in **CASE OF** 分隔 "," Passing multiple function parameters	3-6-3 Case Of
Left/Right Round Brackets	"(" , ")" 1. The start and end of Function Call ex: Fun0 (S:=R0, D:=R1); 2. The enforced priority of a general operator R0:= (1+R0)*R2;	3-4-1 ( )
RETURN	Used in Sub-program or FB. To immediately leave the program section.	3-6-6 Label & Jump/Call
";" the end of the statement	To mark the end of a sentence running at one end.	-

Spaces, newlines and comments can be freely inserted between each symbol.

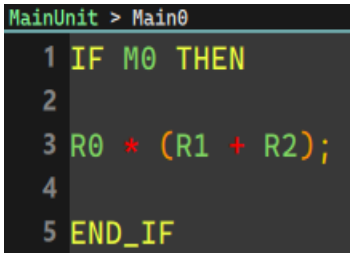
Type	Example	Referrence
Space	Space (Halfwidth/Fullwidth), TAB	-
Newline	Newline Code	-
Comment	//、(* .... *)	

3-2 Statement

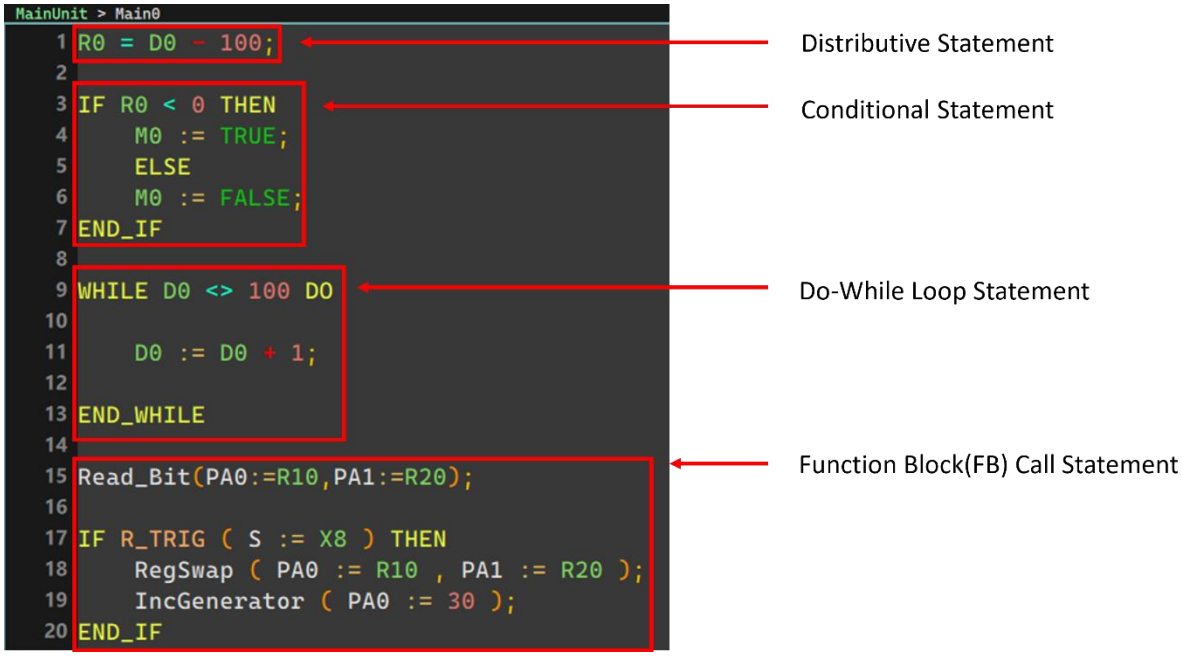
"Statement" is the most basic execution unit in ST Language, which represents a complete work to be executed. A complete statement is not limited to the same line of words; however, it must be ended with " ; ." In addition, a statement is also allowed to contain multiple or multi-level sub-statement, and regardless of the position of the statement, it must be followed by a ";" symbol at the end.



A complete statement is equivalent to a ladder diagram section (NETWORK) with complete functions, and it must be able to clearly express one work. Take the statement in the below program as an example, the work it performs is to compute the content value of each device according to the order R0*(R1+R2) expressed in the mathematical formula, and assign the result of the operation to the R10 device. However, although the content in the red frame in the figure below is legal, it is not a complete statement but an Expression, which represents only a value of a mathematical computation, but not a specific work.



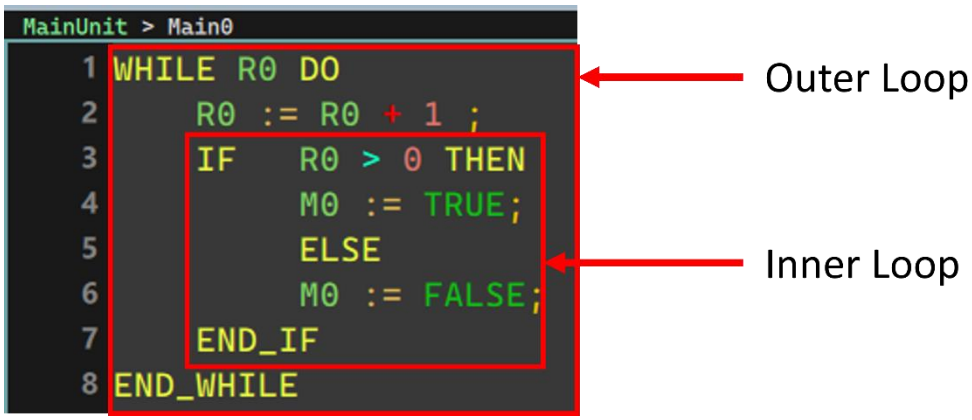
Shown below are some statements of Uperlogic ST Language:



Type		Content	Example
Distributive Statement		Substitute the result on the right into the variable on the left.	:=
Flow Control Syntax	Conditional Statement (IF, CASE)	Select the execution syntax based on the condition.	3-6-1
	Do-Whlie Statement (FOR, WHILE)	Execute multiple times depending on the end condition.	3-6-2 3-6-4
Sub-program Statement	Function Block Call Statement	Call FB(FCM)	3-14
	Label Statement	Call LBL	3-6-6
Toolbox Statement		Convenient statement to execute toolbox instructions	

Flow control syntax can be layered.

(It can be used in combination with Conditional statements and do-whlie statements))

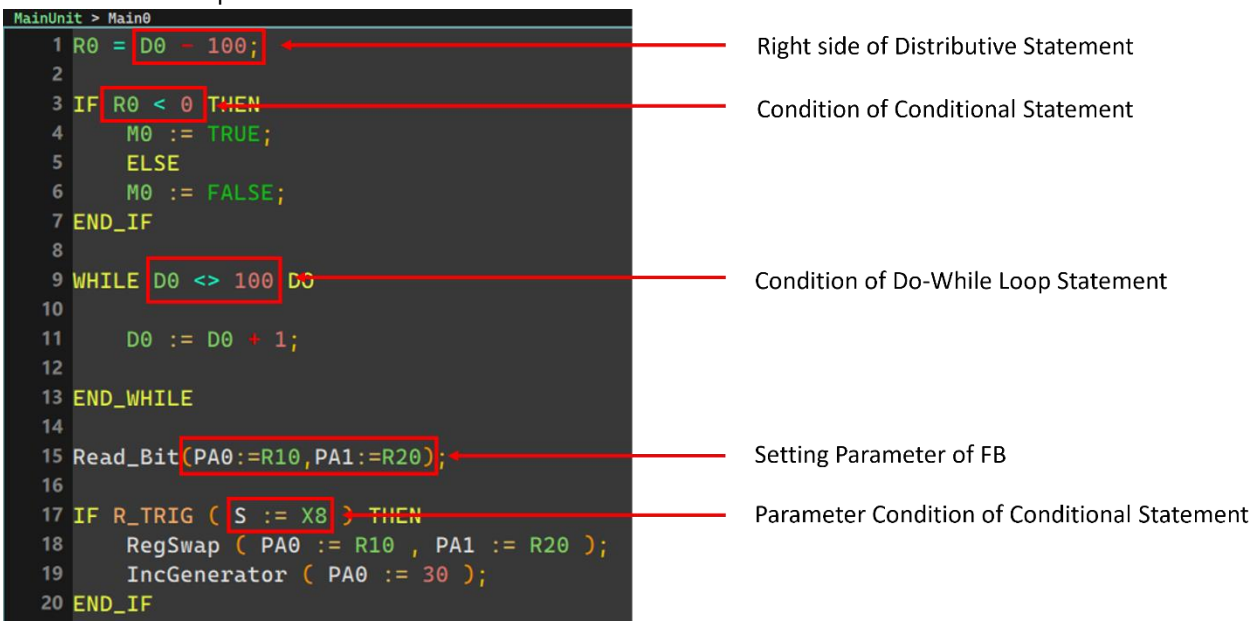


3-3 Expression

"Expression" is a very important element in the structure of a statement and it represents a "value", such as a Boolean value of TRUE or FALSE, or an integer value of 20 or -5. It can be an operation expreesion or a constant, and of course it can also be a variable symbol or device, depending on occasions. The following are some examples of expressions:

- M0 & M1 (Expression of Boolean) represents the Boolean value between the computation of M0 and M1.
- M0 = FALSE (Expression of Boolean) represents whether the condition "M0=FALSE" is true. When the value of M0 is ON, the Boolean value represented by this expression is FALSE; but when M0 is OFF, the Boolean represented by the expression will be TRUE because the condition is established.
- M0 (Expression of Boolean) directly takes the value of M0 as its representative Boolean value. When the value of M0 is ON, the Boolean value represented by the expression will be TRUE, and when M0 is OFF, the Boolean represented by the expression will be FALSE.
- D1 + D2 (Expression of Value) represents the result of adding D1 and D2.
- D0 (Expression of Value) directly takes the current value of D0 as its representative value.
- D2 = D0 + D1 (Expression of Boolean) is a relatively confusing Boolean expression, which represents whether the condition "D2=D0+D1" is true. When the result of adding D0 and D1 is equal to the current value of D2, the expression is TRUE; if the result of adding of D0 and D1 is not equal to the current value of D2, the expression represents FALSE.
- D2 := D0 + D1; (Statement, not Expression) is a complete statement rather than an expression, which represents the meaning of the work "Assigning the result of adding D0 and D1 to D2"; but this statement is also composed of the two expressions "D2" and "D0+D1."

Position of the Expression used:



The below table shows Expression types:

Type		Data Type of Expression (Computing Result)	Example
Operation Expression	Arithmetic Expression	Interger, real number...etc. (according to computing elements)	R0+R2
	Logic Expression	Boolean (TRUE/FALSE)	R0 AND R2
	Compare Expression	Boolean (TRUE/FALSE)	R0 > R2
Basic Expression	Variables, Constants	Defined data type	M0.R0.123.TRUE
	Function Call Expression	Data type of return value	Fcm0(PA0:= ,OUT0=>);

3-3-1 Opearation Expression

This section uses examples to explain how operation expressions are presented in the ST environment and the LD environment.

3-3-2 Arithmetic (+、-、*、/)

The four operation symbols are described using the same operation symbols (+, -, *, /) as the general arithmetic symbols.

For operations that cannot be described in LD diagrams, can be described concisely through single-line expressions.

Program Example:

Substitute the adding result of R0-R2 in R3

R3=R0+R1+R2

ST

```
R3:=R0+R1+R2;
```

LD

- When adding multiple operration expressions with one statement, the operation symbol with the highest priority will be processed. For the priority of the four operation symbols, please refer to the chapter on operands. When there are multiple operation symbols with the same priority, the operation starts from the leftmost operation symbol.。

Advanced Operation (Exponents)

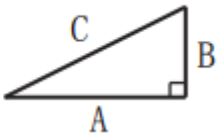
Exponential or trigonometric operations using general-purpose functions.

Type	Function Name	Example	
		General Expression	ST
Absolute Value	ABS	|X|	ABS(D:=);
Square Root	SQRT	$\sqrt{X}$	FSQR(S:= ,D:=);
Trigonometric Functions	SIN、ASIN	B=SIN A	FSIN(S:= , D:=);
	COS、ACOS	B=COS A	FACOS(S:= , D:=);
	TAN、ATAN	B=TAN A	FTAN(S:= , D:=);

Program Example

Find the length of the hypotenuse of a right triangle.

$$C = \sqrt{(A^2 + B^2)}$$



ST

(To use floating point numbers, you need to register Tag first to change the data type.

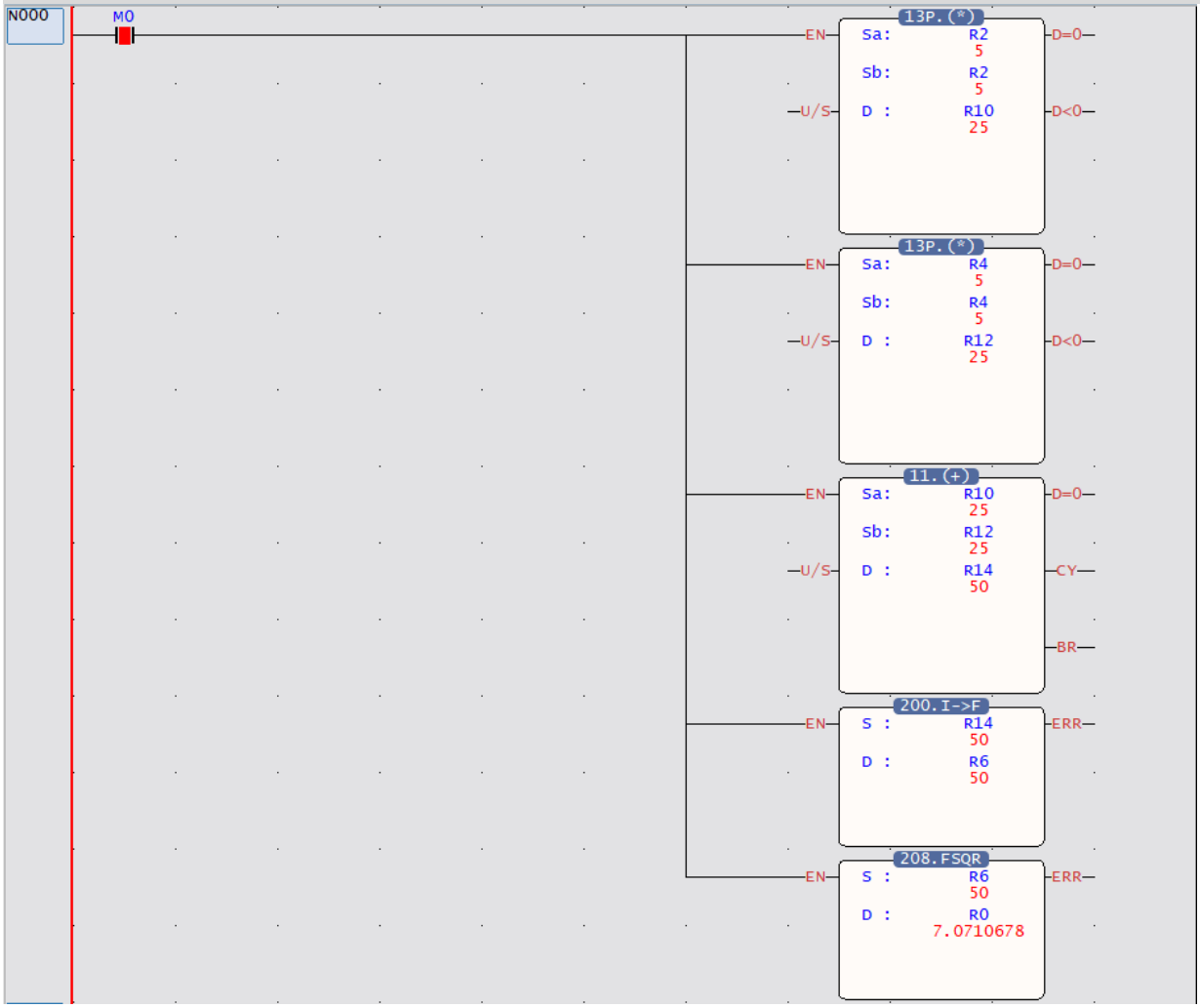
ex: A= Float DR0、B= Float DR2、C= Float DR4)

FSQR(S:= A*A+B*B, D:= C);

Local Tag

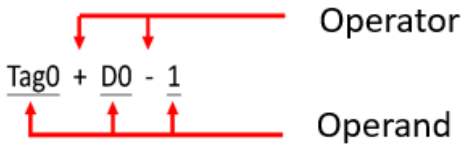
		Name	Type	Address	Length
1		A	Float	DR0	1
2		B	Float	DR2	1
3		C	Float	DR4	1

LD


N000
M0
13P.(*)
EN
Sa:
R2
5
Sb:
R2
5
U/S
D :
R10
25
D=0
D<0
13P.(*)
EN
Sa:
R4
5
Sb:
R4
5
U/S
D :
R12
25
D=0
D<0
11.(+)
EN
Sa:
R10
25
Sb:
R12
25
U/S
D :
R14
50
D=0
CY
BR
200.I->F
EN
S :
R14
50
D :
R6
50
ERR
208.FSQR
EN
S :
R6
50
D :
R0
7.0710678
ERR

3-4 Operand and Operator

Operands and operators are the basic elements that make up an expression. The operand refers to the object involved in the operation, and the operator represents the operation performed. For example, in the expression “D0 + D” 1, both “D0” and “D1” are operands, and the “+” sign is the operator. As seen from the examples in the previous section, an expression can be a combination of a group of operands and operators, but it can also be represented by an operand alone; while the operand can be a device, a variable symbol or a constant



Like mathematical forms, operators themselves have a priority order for performing operations. When the priority levels are the same, the order of the operation will be from left to right. The following table is a list of operators in ST syntax in Uperlogic

Symbol	Function	Data Form		Example of Expression		Priority Level
		Operand	Operation Result (Expression Value)	Expression	Value	Highest
( )	Prior block	Not limited	Not limited	(D0 + 6) * 3		
++,--	To quickly add or subtract 1	Any value	Any value	++D0 D0++		
-	Numerical negative sign	Any value	Any value	-D0		
NOT	Logic inversion	Boolean	Boolean	NOT M0	TRUE	
*	Multiplication	Any value	Any value	D0*3		
/	Division	Any value	Any value	15/D0		
+,-	Addition, Subtraction	Any value	Any value	D0+3		
< , >,<=, >=	Value compare	Any value	Boolean	D0>2		
=,<>	Equal, Not Equal	Any value	Boolean	D0<>2		
		Boolean		M0=TRUE		
AND,&	"and" operation	Boolean or Value	Boolean or Value	M0&M1		
OR, |	"or" operation	Boolean or Value	Boolean or Value	M0 OR M1		
XOR XNOR	"exclusive or" operation, "exclusive nor" operation	Boolean or Value	Boolean or Value Integer	M0 XOR M1		
MOD		Value	Value	D0 MOD 3		
>>,<<	Shift right/left 1	Any value	Any value	D0>>1	2	
R_TRIG, F_TRIG	Rising Edge Falling Edge					Lowest

■ The part with the same background color have the same priority.

■ AND, OR, XOR, XNOR basically do the role of BitWise (ex R0 AND R1) and the result will be a value.

Unless the left and right sides are both Bool(bit) (ex. M0 AND M1) types, logic operations will be performed and the result will be a Bool(bit) value.

■ No support &&, ||, == syntax.

The individual Operator will be introduced later.

3-4-1 ( )

The function is the same as a mathematical formula, and the expression in the round brackets is prioritized for operation.

EX:

R0:= (R0+2)*3-(R2+3)*56 ; // R0=1;R2=2
// R0= - 271

3-4-2 Arithmetic +, -,* , /

Operate addition, subtraction, multiplication and division for the operators on both sides.
Calculate addition, subtraction, multiplication and division for the operands on both sides

EX:

R0:= 5+4-3*2/1 ;
// R0=3

3-4-3 Quick Addition and Subtraction ++, --

Specifically for operand + 1 or -1

Prefix Syntax:

++(Register)
--(Register)
The same result as (Register):=(Register)+1, but the speed is faster.

PostFix Syntax:

(Register)++
(Register)--
First take (Register) value → return → operate (Register):=(Register)+1

ex:

R0:=10;
R2:=10;
R100:=++R0; //// R100 is 11, R0 is 11
R102:=R2++; //// R102 is 10, R2 is 11

3-4-4 Value compare >=,<=,>,<

Compare the values on the left and right sides of the symbol. The left and right sides of two symbols need to be of the same type to be able to compare; otherwise there will be truncate or digital missing (float <->int)

The result will be TRUE/ FALSE (1/ 0)

EX:

M0:=R1 >= R2; /// If R1 is greater than or equal to R2, M0 will be TRUE (1)

3-4-5 Equal to/Not equal to = , <>

Compare whether the left operator is equal to the right operators.

The result will be TRUE/ FALSE (1/ 0)

EX:

M0 := R0=100; // If R0 value=100, M0 will be TRUE (1)

3-4-6 Bit shift left and right <<, >>

For the left-side value, shift left or right by the number of bits on the right side.

EX:

R10:=R0<<4; **//R0=1(0b0_0001)**

// R0 value will be shifted right with 4 bits, and assign to R10

// R10=16(0b1_0000)

P.S. R0 will not be changed

3-4-7 Negation of Operand NOT

Perform Bitwise NOT operation on the right operand.

EX:

R0:=NOT R2; // Do "one's complmenet" to the value in R2 and save it to R0

R2=1(0b0_0001) => NOT R2= -2(0b1111_1110)

M0:= NOT (M0 OR M2); // M0 OR M2 result is negated and stored in M0

3-4-8 Negative Sign of Operand -

Perform negation (sign reversal) on the right operand.

EX:

```
IF R_TRIG( S:=M1 ) THEN
R0:=-R2;      //R2=1 0b0_0000_0000_0001)=> R0=-1(0b1111_1111_1111_1111)
END_IF
```

3-4-9 Assign Value :=

Assign the right value (variables) to the left variable of the symbol.

EX:

R1:=R0; // Move R0 to R1

R1:=1; // Set R1 as integer 1

3-4-10 Logic Operand AND (&)、OR (|)、XOR、XNOR

Perform logic operations (Bitwise) on the left and right operators, and the result of the value will be the same as the type of the operator.

EX:

```
R0:= R1 AND R2; // Perform bitwise AND operation on the values of R1 and R2 and store them in R0
                // If R1=1(0b0_0001) R2=15(0b0_1111) then R0 =1(0b0_0001)
DR0:= R10 & DR12;
// Perform bitwise AND operation on the values of R10 and DR12 and store them in DR0
// If R10=1118481(0x11_1111H) DR12=69905(0x1_1111H) Then DR0 =4369(0x1111H)
R0:= R1 | R2; // Perform bitwise OR operation on the values of R1 and R2 and store them in R0
              // If R1=1(0b0_0001) R2=15(0b0_1111) then R0 =15(0b0_1111)
DR0:= DR10 OR R12;
// Perform bitwise OR operation on the values of R10 and DR12 and store them in DR0
// If DR10=4369(0x1111H) R12=69905(0x1_1111H) Then DR0 =4369(0x1111H)
R0:= R1 XOR R2;
// Perform bitwise inverse OR operation on the values of R1 and R2 and store them in R0
// If R1=1(0b0_0001) R2=15(0b0_1111) then R0 =14(0b0_1110)
R0:= R1 XNOR R2;
// If R1=1(0b0_0001) R2=15(0b0_1111) then R0 =-15(0b1111_0001)
```

3-4-11 Remainder MOD

Operates the remainder of the left and right operands which are equal to C’s “%” symbol

EX:

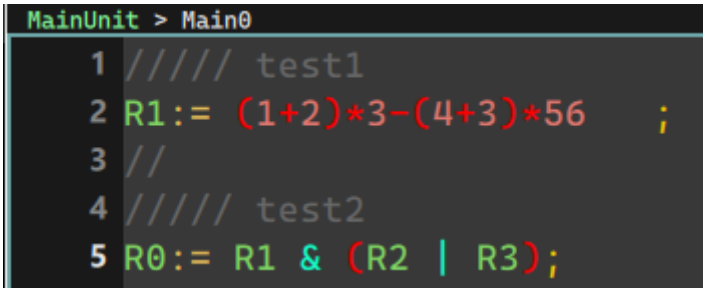
```
R0:= R1 MOD R2 ;   // R0 is the remainder after dividing R1 by R2
                   //   R1=10,R2=3   The quotient of 10/3 is equal to 3, the remainder is equal to 1, R0=1
```

3-5 Comment

Comments are those used by program developers for easy maintainance in the future. There are single-line or multiline comments appeared with light gray text, and these parts will be ignored by the editor and will not generate operating data.

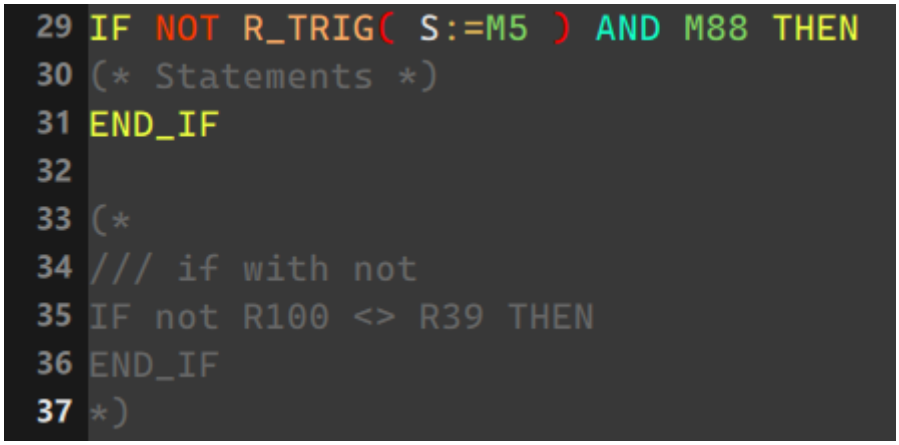
3-5-1 Single-Line Comment //

Ligh gray will appear after the symbol, and will not generate the operating program codes.



3-5-2 Multiline Comment (* *)

A continuous comment that can be multiple-line (or a single line), and the text between these two symbols will be seen as a comment.



P.S. Users can also use quick keys (Ctrl+ /) to comment on the selected rows and columns in batches (add/remove).

3-6 Flow Control and Loop

When writing ST, some conditions or loop control are usually required for easier design.

Introduction as shown below:

3-6-1 IF ELSE

```
IF (* bool exp *) THEN
(* Statements *)
ELSE
(* Statements *)
END_IF
```

When the expression after IF is TRUE or 1 at the end, the description after IF will be operated immediately, otherwise the description after ELSE will be operated.

EX:

```
IF R_TRIG( S:=M5 ) AND M88 THEN
    R100:=10;
    R10:=12;
END_IF
```

If M5 is Rise Trigger, and M88 is TRUE (1) , R100 = 10, and R10 = 12

Use ELSEIF if there are multiple comditions.

EX:

```
IF (* bool exp *) THEN
(* Statements *)
ELSEIF (* bool exp *) THEN
(* Statements *)
ELSE
(* Statements *)
END_IF
```

3-6-2 For Loop

```
FOR (*Reg*) := (*Start*) TO (*End*) BY  (*<inc>*) DO
(* Statements *)
END_FOR
```

Parameters：

(* Reg *)：Counter register

Repeat the statement from FOR to END_FOR until the counter register reaches the target value.

Each time it is repeated, an incremental value or decremental value is added to the target value

(* Start *)：Initial value

(* End *)：Target value

(* < inc > *)：Incremental value or decremental value

(* Statements *)：Program to be executed in a loop

P.S. The BY *field can be omitted (the default is 1) as shown below*

```
FOR (*Reg*) := (*Start*) TO (*End*)  DO
(* Statements *)
END_FOR
```

	Counter register	Initial value	Target value	Incremental value or decremental value
Single word	○	○	○	
Double word	○			
Constant		○	○	○

P.S. If the loop count is incrementing, then initial value < target value.

If the loop count is incrementing, then initial value > target value.

EX:

```
FOR DR0:=R2 TO R4 BY 2 DO     //DR0 from R2, R2+2, R2+4,....etc, until it reaches to R4
    R10++;                                  // DR0=R4,R10=R0+(R4-R2)/2
END_FOR
```

3-6-3 Case Of

Selection of Integer conditions:

```
CASE (* Integer target *) OF
(* int literal *) : (* single statement *)
ELSE
(* Statements *)
END_CASE
```

It will read the value of the target register, and run after the specified conditions are fully read: the following description .

If none are satisfied, the description after ELSE will be run

Conditions only support integer constants or constants within the range .. symbols

EX:

```
CASE R0 OF
    1:R100:=10;
    3..9:R100:=11;
END_CASE
```

R0 = 1 //R100 = 10

//3, 4, 5, 6, 7, 8, 9 ,R100 = 11

3-6-4 WHILE LOOP & REPEAT

Repeat the run description until the condition is (not) met

WHILE LOOP:

```
WHILE (* bool exp *) DO
(* Statements *)
END_WHILE
```

EX:

```
WHILE R0<10 DO
    ++R0;
END_WHILE
```

REPEAT:

```
REPEAT
(* Statements *)
UNTIL (* Stop Condition *)
END_REPEAT
```

EX:

```
REPEAT
    ++R0;
UNTIL R0=10
END_REPEAT
```

NOTICE: Because of the logic of hardware running, if the WHILE does not jump out of the loop for a long time, the PLC device will not be able to handle other IO states, resulting in system errors. Users should be careful when using it.

3-6-5 BREAK / EXIT

In the loop situation of FOR, WHILE, REPEAT, you can leave the loop early with BREAK or EXIT at the appropriate time

EX:

```
WHILE M0 DO
   ++R0;
   IF R0>=100 THEN
    BREAK;
   END_IF

END_WHILE
```

When R0 is greater than or equal to 100, it will run BREAK and jump out of the WHILE loop.

NOTICE: Because of the logic of hardware running, if the WHILE does not jump out of the loop for a long time, the PLC device will not be able to handle other IO states, resulting in system errors. Users should be careful when using it.

3-6-6 LBL, JMP, CALL

The LBL command can achieve the same effect as LD FUN_65, with a Label of up to 6 ASCII characters.

EX:

```
3 LBL ( "R65" )
4     R65 := R65 +1;
5 LBL ( "123456" )
6     R66 :=R66 + 1;
7 LBL_65 ( "ABCD" )//Equivalent to LBL
8     R67 := R67 +1;
```

The keywords that can be used to call Label in the ST environment are as follows:

JMP_66, GOTO (for detailed description, please refer to FUN 66)

EX:

```
JMP_66 ("123")

GOTO ("456")
R66:=R68+1;

LBL ("123")
R2:=123;

LBL ("456")
R4:=456;
```

CALL、CALL_67 (for detailed description, please refer to FUN 67)

EX:

MainUnit > Main0

```
1 CALL ("123")
2 CALL_67 ("456")
3
```

SubUnit > Sub0

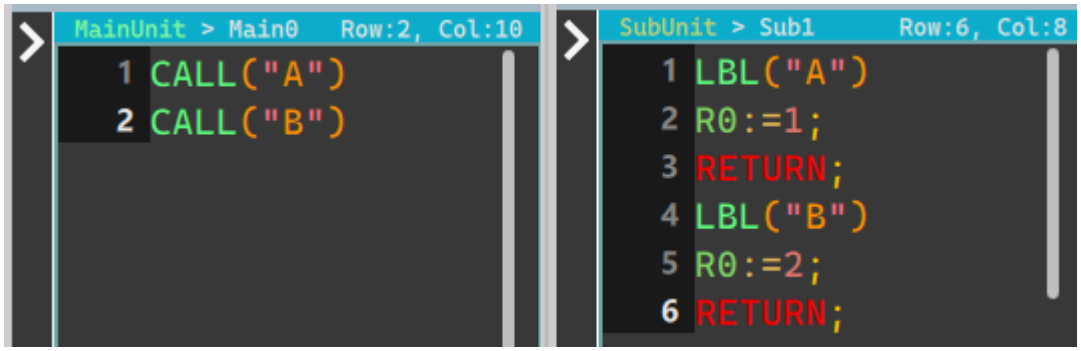
```
1 LBL ("123")
2 R2:=123;
3
```

SubUnit > Sub1

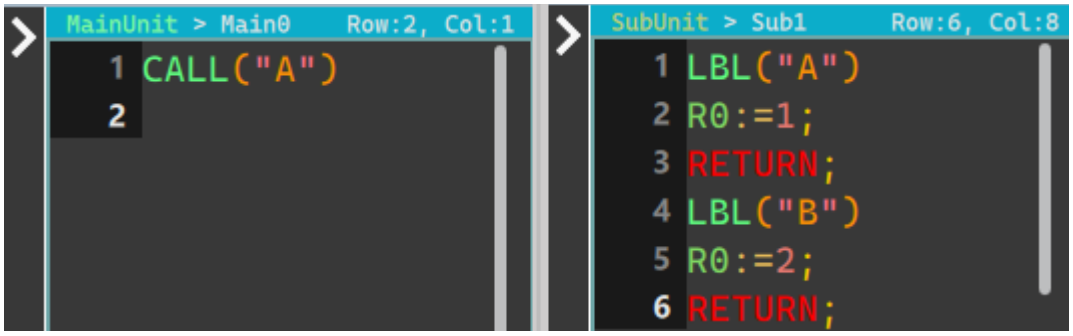
```
1 LBL ("456")
2 R4:=456;
3
```

When using the Label function in a subroutine, the LBL instruction combined with RETURN will have different results.

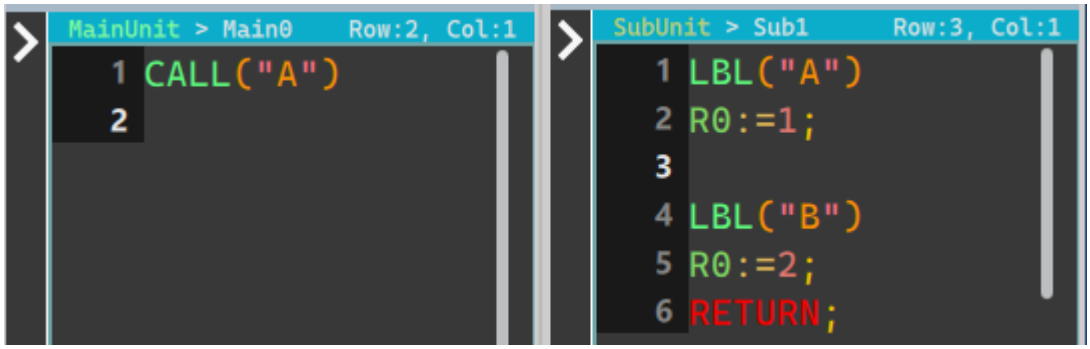
EX:



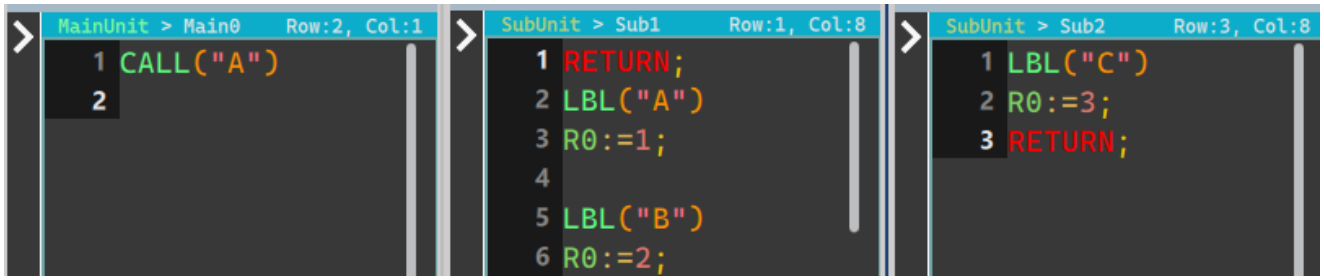
R0=2, because the Label of A.B is called at the same time and the result is RETURN.



R0=1, because only the Label of A is called, and the result is RETURN



R0=2, because the Label of A is called, but RETURN is after the Label of B



R0=3, because the Label of "Sub-unit 0" A is called, but RETURN is after the Label of "Sub-unit 1" C (special attention is required)

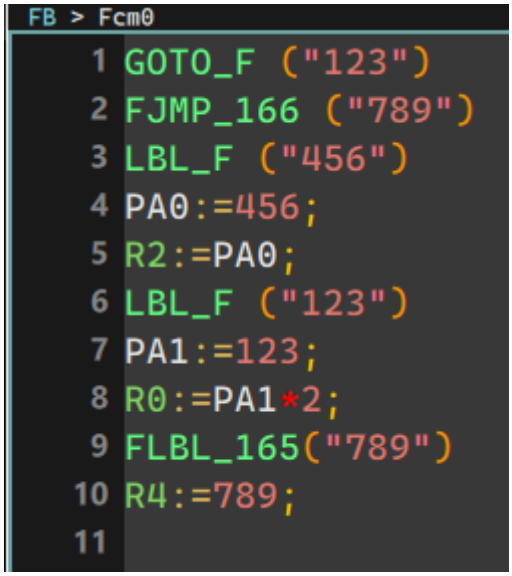
If you need to use tags in the FCM environment, you need to use the LBL_F、FLBL_165 instructions.

The Label keyword can only be called in FCM

GOTO_F (Label that can only be used in FB)

FJMP_166

EX:



3-7 Variables and Data Type

In the programming language, the use of variables and data types are an important part. In order to ensure that variables have typed characteristics, it is convenient for programmers to view and debug. ST uses Tag (Global, Local) to give variables a specific type.

The following introduces the data types and usage supported by ST.

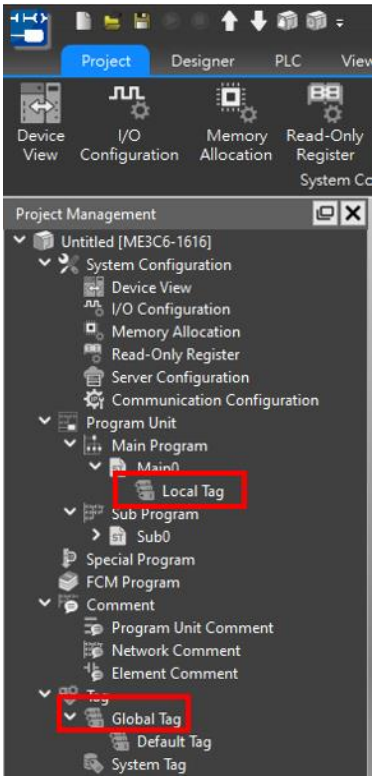
Precautions

If the operation result exceeds the value range that the data type can handle, the correct result (numeric value) will not be reflected in subsequent processing.

The data type of the operation object variable should be converted in advance into a data type within the range that can process the operation results.

Tag creation example

Labels are divided into global labels and regional labels. The label name created by the global label cannot be shared with the regional label. It is a label name that can be shared by the entire project. The label name created by the regional label can use the same name between different main programs, sub-programs, and Fcm programs without affecting each other (but the addresses must be different).

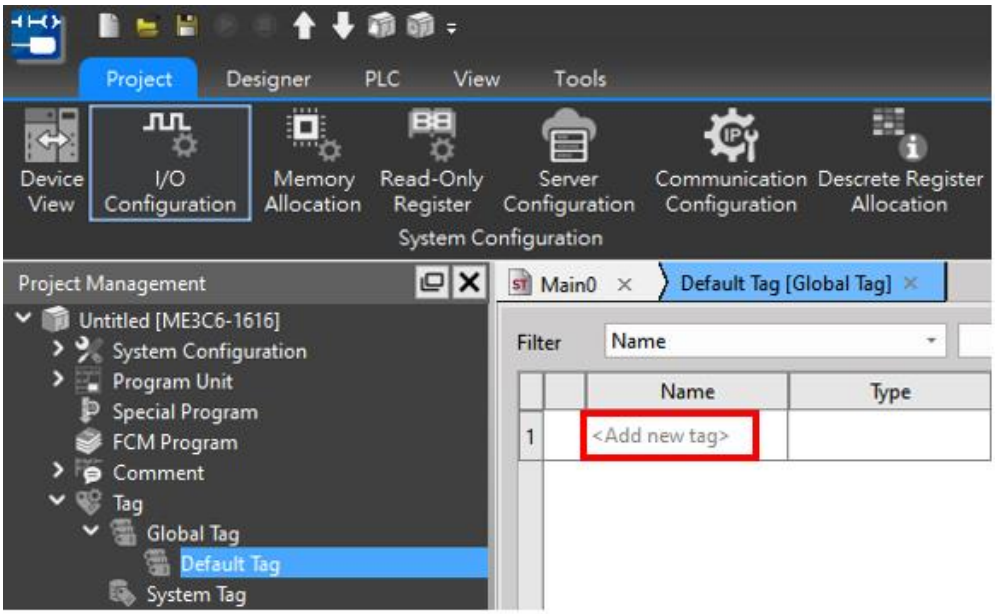


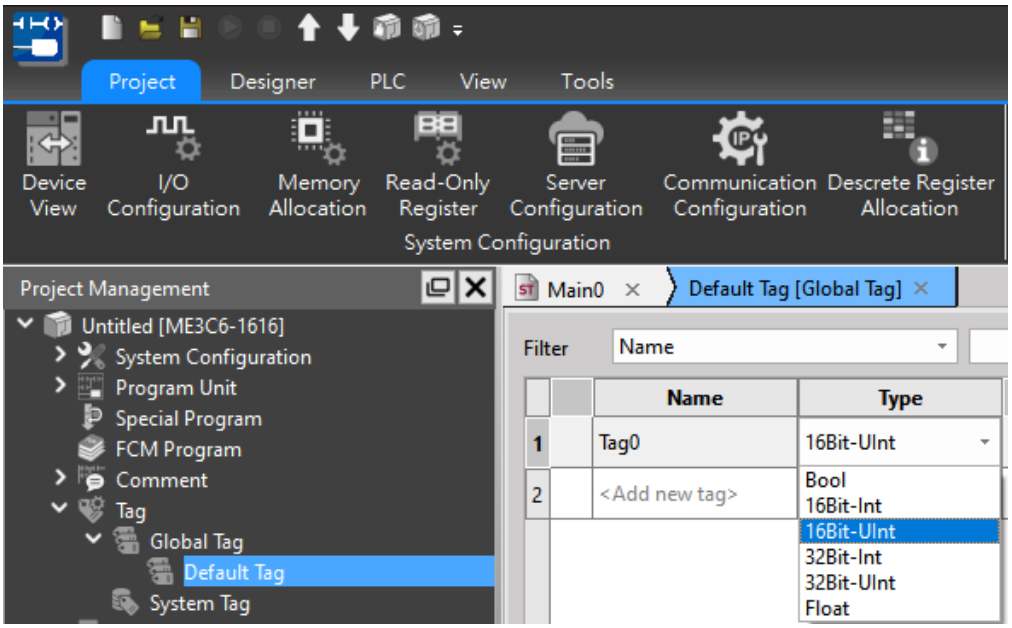
In order to cooperate with program operations of different data types, the following will demonstrate how to create different types of labels.

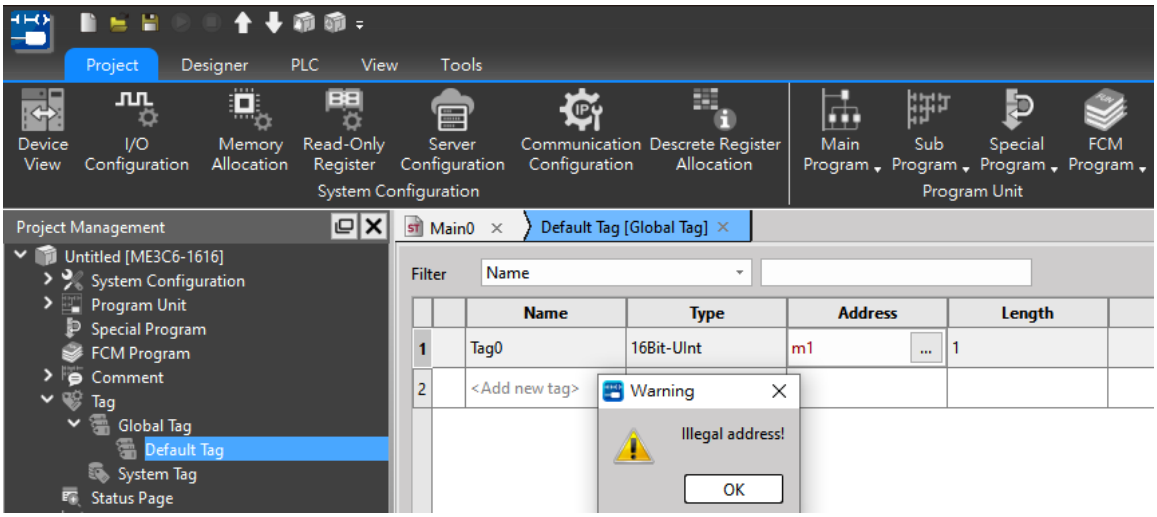
The Rx.Dx register is defaulted to INT16, and the DRx.DDx register is defaulted to INT32 integer variable.

EX

Double-click to select the Global Tag->Add new Tag







If the selected address and data type are not supported, a reminder will pop up to block it.

3-7-1 Bool / Bit

As long as the data represented is a bit or a value comparison operation (ex. <, >, <> ... etc.), these results are all Bool (Bit) type, and 0 or 1, TRUE, FALSE can be used here accepted as a constant representing type Bool.

Among them, the switch of the relay is also represented by Bool/Bit (ex. M0, X0, Y0...etc).

EX:

```
1 M0:=1;
2
3 M0:=TRUE;
4
5 M1:=R0<10;
6
```

3-7-2 Integer Type

There are four integer types of ST: INT16, UINT16, INT32, UINT32

Type	Description
INT16	16-bit integer
UINT16	16-bit positive integer
INT32	32-bit integer
UINT32	32-bit positive integer

If a 16-bit register position (ex. R0, R1..etc) is used, the system will default to INT16 data type to process; if it is a 32-bit register (ex. DR0, DR2... etc), it will be seen as INT32

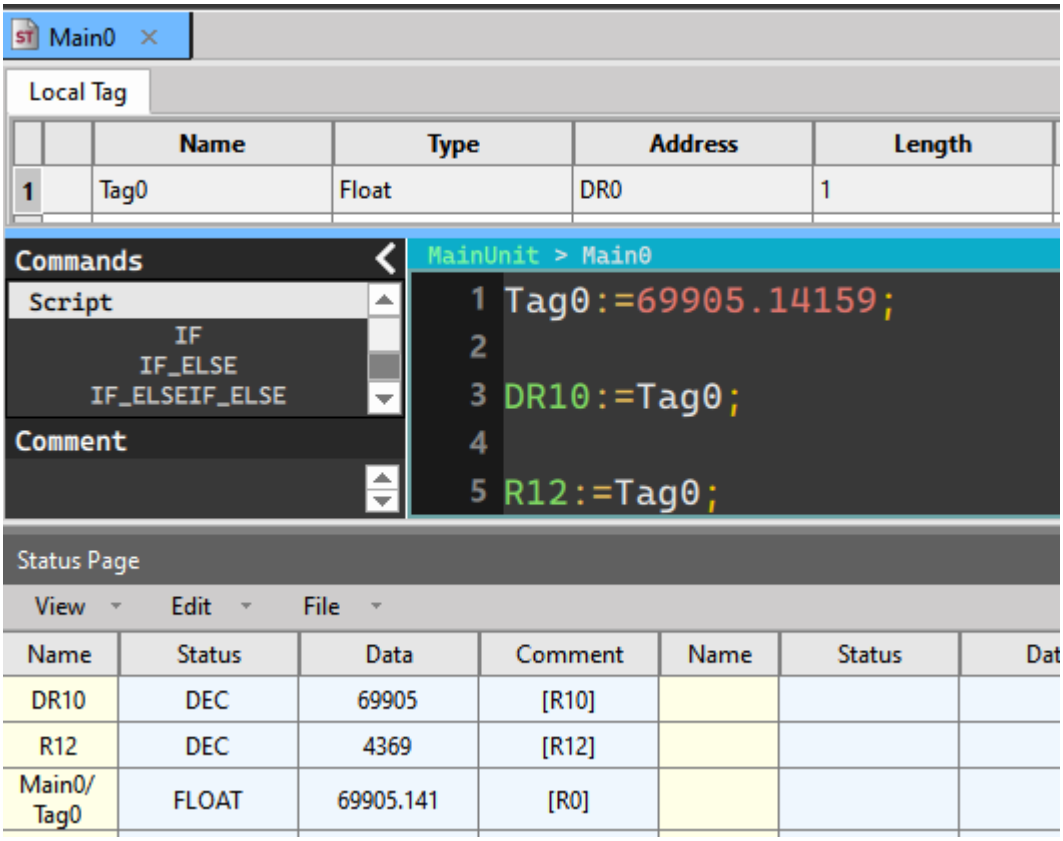
.

3-7-3 Floating Point

Floating-point defined for IEEE-754.

For ST operations on floating point, it is necessary to generate the corresponding Tag first, so that the system can know which variable refers to the register position representing a floating point.

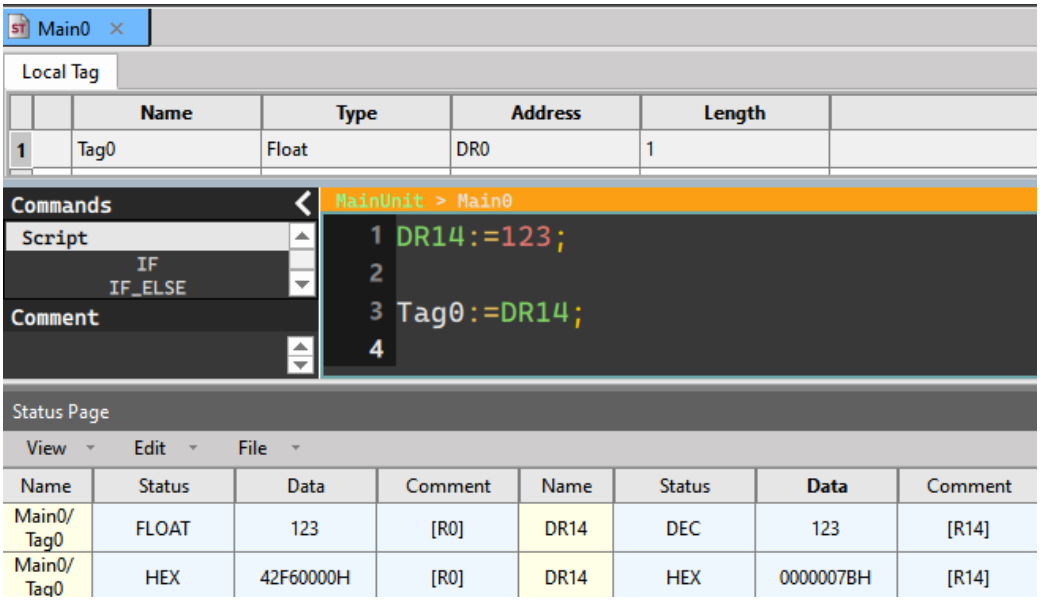
Among them, when using floating-point operations, there will be some defaulted implicit behaviors. The descriptive example is in the figure below:



Line 1: Indicates that the floating point value of label 0 is assigned to 69905.14159

Line 3: Indicates converting the floating point value of label 0 into an int32 integer value And there is a position of DR10, and the decimal point will be removed.
(ex. Tag 0 is 69905.141, then the DR10 content is 69905)

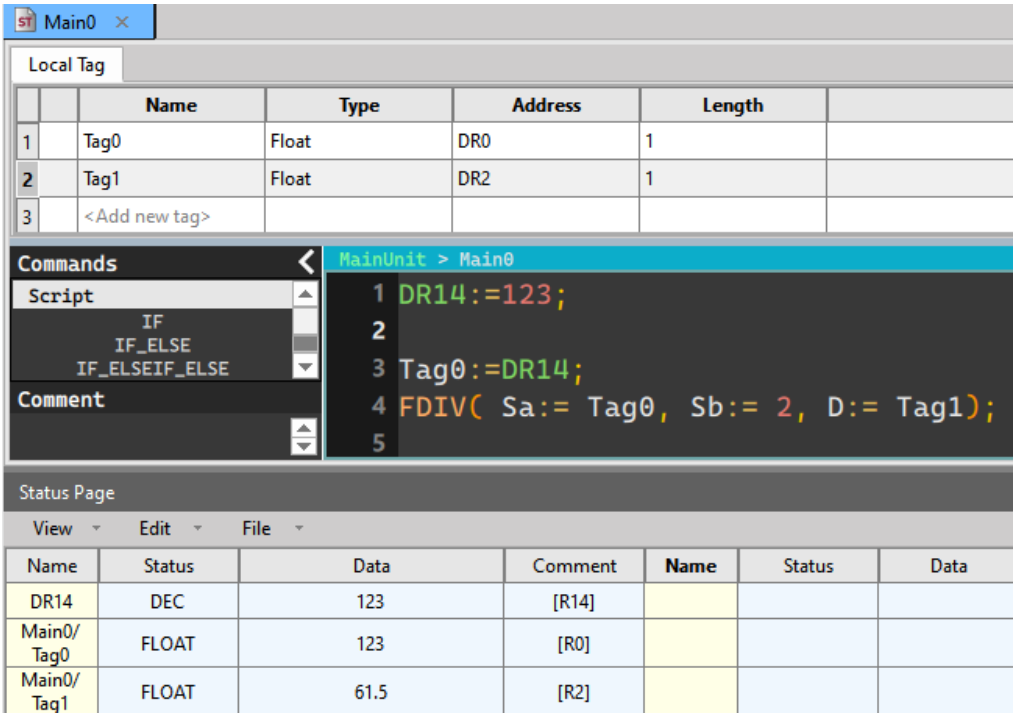
Line 5: Indicates that the floating point value of label 0 will be converted into an int16 integer value.
And there is a position of R12, the integer value exceeding int16 and the decimal point will be removed.
(ex. Tag 0 is 69905.141, then R12 content is 4369)



Line 2: DR14 (INT32) is converted into a floating-point data type and stored in the label 0

(The digital representation of observable floating point numbers is different from that of integers and can be used as Fun 200 I -> F)

EX:



3-7-4 Constant

The table below shows the ST-supported constant types:

Type	Format andnd Sample
Bool (Bit)	0, 1, TRUE, FALSE ---------- M0:=1; M0:=TRUE;
Integer	Integer: R0:=1; R0:=-10; Binary: 2# followed by the value of 0,1, underscores can be used to divide groups R0:=2#1111_0000; Octal: 8# followed by the value, underscores can be used to divide groups R0:=8#243; Hexadecimal: 16# followed by the value, underscores can be used to divide groups 0x followed by the value, underscores can be used to divide groups

Scientific Tag	Tag_FlR100:=1E3; //1000
	Tag_FlR100:=1.2E+3; //1200
	Tag_FlR100:=1E-3; //0.001

Unit Symbol	G: 1e+9 M: 1e+6 K,k: 1e+3 m: 1e-3 u: 1e-6 n: 1e-9 ---------- DR0:=1G; R0:=1K;

	R0:=1k; DR0:=1M; Tag_FlR100:=1m; Tag_FlR100:=1n; Tag_FlR100:=1u;
String (Currently only the Label command can be used)	Use ASCII string quoted by " symbol ---------- LBL ("my_lab")

3-8 Using PLC Register and Memory

In addition to using the created Tag to call the corresponding register, users can also directly enter the name of the register to perform operations or flow control, as long as the data type of the register is single word 16bit ---> INT16 double word 32bit ---> INT32

EX:

```
R29:=100;
  V:=19; // index register
  R200:= R10V+100;       R200=200
```

Among them, the special T and C bits will only have the characteristics of bit or int according to the logic of the program

EX:

```
  /// T1 indicates bit type
  /// whether timeout
  IF T1 THEN
     R10:=20;

/// T1 represents the value of current timer
/// is the integer of int16
ELSEIF T1>100 THEN
   R10:=30;
END_IF

///At this time, T1 will be regarded as a Bit type for negation
M0:=NOT T1;
```

3-9 Calling System Built-In Functions

The description of the built-in functions will be shown as below:

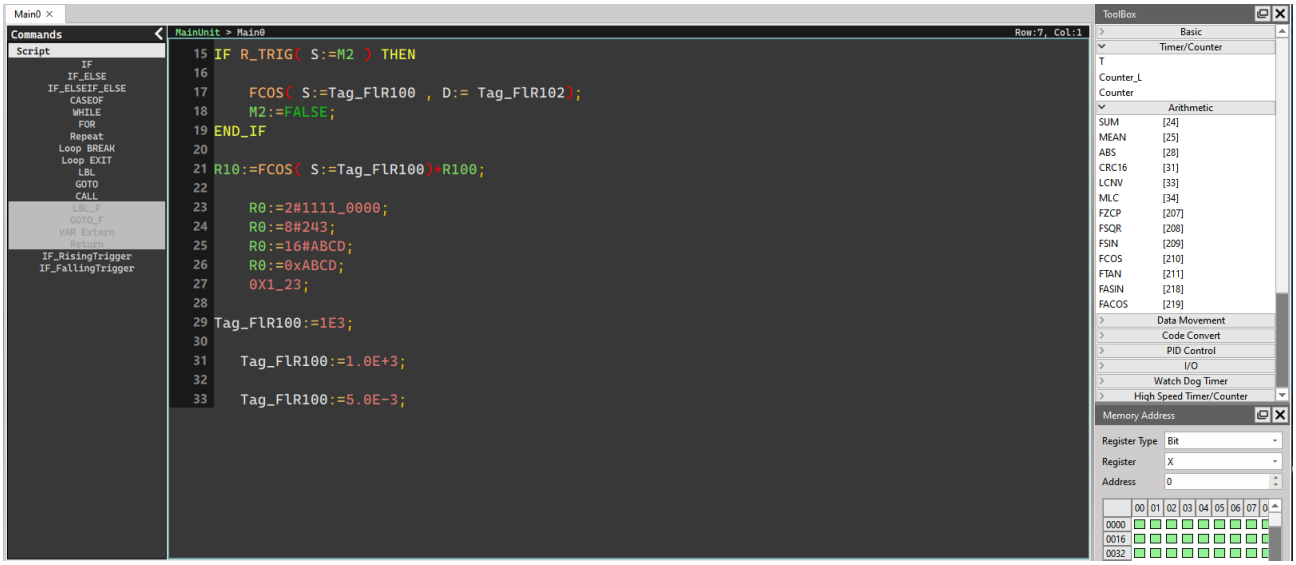
<Function> (<Parameter 1> := <Input Parameter 1> , ...);

Like the ladder diagram, ST has some built-in functions for users to call, which will appear in the toolbox column on the right.

You can drag and drop from the toolbox or double-click the field and it will be added to the text.

When calling, the order of parameters can be changed at will, but it cannot be omitted.

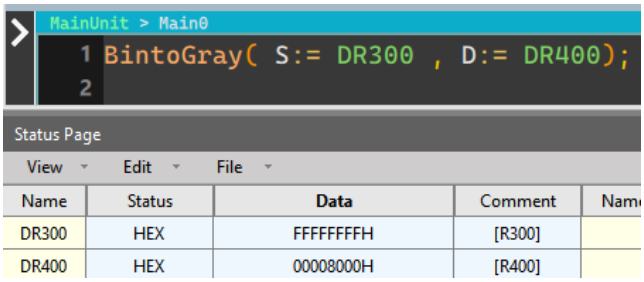
Unless some special values are allowed to be omitted



Some of the functions will be divided into 32 bit mode (the default is 16 bit mode), you will need to add D_ in front of the function to clearly indicate that the function is in 32 bit mode. You can directly input D_ and the corresponding will appear the list of hints indicates those with 32 bit mode.

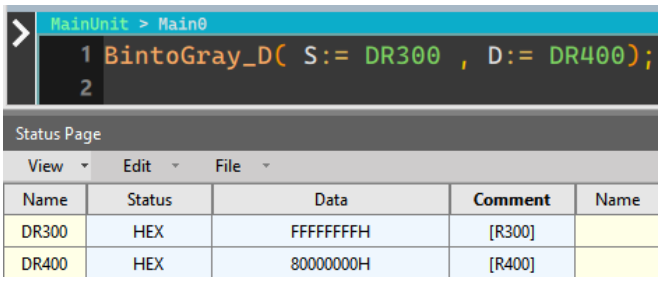
EX: Binary code conversion to Gray code

Error conversion



Name	Status	Data	Comment	Nam
DR300	HEX	FFFFFFFFH	[R300]	
DR400	HEX	00008000H	[R400]	

Correct conversion



Name	Status	Data	Comment	Name
DR300	HEX	FFFFFFFFH	[R300]	
DR400	HEX	80000000H	[R400]	

Most functions are executed as soon as they are called. If you want to execute the action only once after the condition is met, please refer to the writing method.

EX:
IF R_TRIG(S:="condition") then
"Function"
END_IF

!Note! The "Motion Basic"."Motion Mode1". "Motion Mode2". "Motion Mode3" functions in the toolbox are only executed once unless the conditional state changes.

For the detailed parameter content of each call function, please refer to FUN file. The following is an introduction for different FUN.

3-9-1 Timer

Prototype:

```
Timer( Trigger:=   (* Trigger Bit Rising Edge ->ON, Falling Edge ->OFF *),
       T:=         (* Timer 0~1023 *),
       PV:=        (* preset value (0~32767) *),
       IsTimeout=>    (* time out bit *));
```

Parameter:

Trigger	(bool)	:	Rising Edge Start Timer
			Falling Edge Stop run
T	(int)	:	Integer from 0~1023 represent Timer T0~T1023
PV	(int)	:	Timer threshold
IsTimeout	(bool)	:	Whether Timer reaches the target

For detailed description, please refer to corresponding LD Timer files.

3-9-2 Counter / Counter_L

There are two groups of Counter functions here, so that users can easily distinguish whether they are currently calling the counter of single word (Counter) or the version of double words (Counter_L)

Prototype:

```
Counter( Pulse:= (* Pulse Signal *),
         Clr:= (* CLR Signal *),
         C:= (* counter number (0~1023) *),
         PV:= (* Preset value (Single Word 0~65535) *),
         IsUp=> (* Counter Is Up *));
Counter_L( Pulse:= (* Pulse Signal *),
           Clr:= (* CLR Signal *),
           C:= (* counter number (1024~1279 Long counter) *),
           PV:= (* Preset value (Double Words) *),
           IsUp=> (* Counter Is Up *));
```

Parameter:

```
Pulse    (bool)   :  Off->On  Count once        //M0,M9129...etc
Clr      (bool)   :  Whether to clear counter
C        (int)    :  Counter id 0~1279
PV       (int)    :  Counter  Default value
IsUp     (bool)   :  Whether Counter reaches the target   //M0,C0...etc
```

3-9-3 R_TRIG / F_TRIG

Rising Edge and Falling Edge’ s detecting function

Prototype:

Mode 1:

R_TRIG(S:= , D=>)

F_TRIG(S:= , D=>)

Mode 2:

R_TRIG(S:=)

F_TRIG(S:=)

Implied with Return Value D

Parameter:

S (bool) : Rising / Falling Edge Detecting source

D (bool) : Detecting result

Example:

Mode 1: R_TRIG(S:=M0, D:=M5);

Mode 2:

```
IF R_TRIG(S:=M0) THEN
    ++R100;
END_IF
```

3-9-4 TARESUB and TAREZEOFFSET

These two functions are derived from the original FUN258. Removed from Tare Weight Deduction Command mode .

For detailed parameter settings from MODCONF, please refer to the original FUN258 file.

Splitting into two independent functions is also a function that is convenient for users to call at a glance when writing.

Prototype

TAREZEOFFSET(EN:= , MD:= , ID:= , CH:= , WR:= , ERR=> , DN=>)

TARESUB(EN:= , RST:= , ID:= , CH:= , SB:= , ERR=>)

3-10 Functions with multiple calling modes

Such as the trigger detection instructions in the previous chapter 3-9-3. These instructions support two types, one is a direct complete call, and the other is to omit the register location (D) where the result is stored. The following table is is a function that currently has this feature.

Function	Parameter specialized in Return	Function Format
FSQR	D	FSQR(S:= ,D:=)
FSIN	D	FSIN(S:= ,D:=)
FCOS	D	FCOS(S:= ,D:=)
FTAN	D	FTAN(S:= ,D:=)
FASIN	D	FASIN(S:= ,D:= ,MD:=)
FACOS	D	FACOS(S:= ,D:= ,MD:=)
R_TRIG	D	R_TRIG(S:= ,D:=)
F_TRIG	D	F_TRIG(S:= ,D:=)
ItoF	**D**	**ItoF(S:= ,D:=)**
FtoI	**D**	**FtoI(S:= ,D:=)**

That is to say, the parameters of the above-mentioned functions with return characteristics can be omitted when calling again, and directly use another parameter to undertake or directly operate.

EX:

Without omission:

FSIN(S:=Tag_Float_DR100, D:=Tag_Float_DR102);

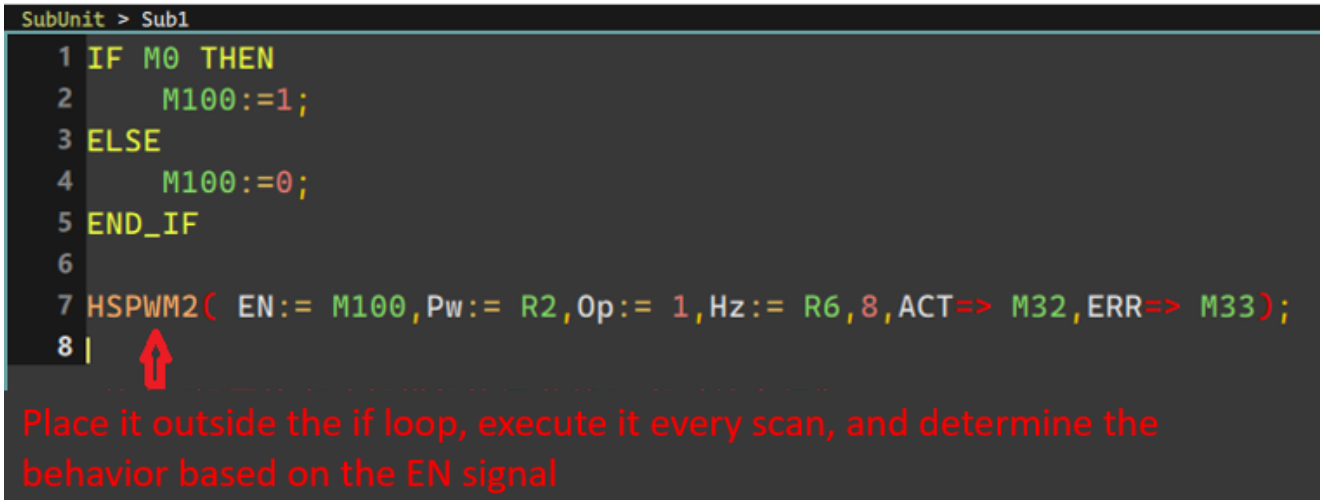
Omit the return parameter: (Since values and representative results are returned, some four arithmetic operations or conditional judgments can be directly introduced)

Tag_Float_DR200 := FSIN(S:=Tag_Float_DR100) + FSIN(S:=Tag_Float_DR102);

3-11 Multi-cycle instruction

The following instruction list shows the instructions operating continuously in the background. The way to use it is to place it outside the IF loop to execute it every scan cycle, and determine the operating state of the function according to the EN signal.

As the example shown below:



Because these instruction classes are called, the execution time usually exceeds one scan cycle.

The following table shows the Multi-cycle instruction currently included:

Function	Function ID
LCNV	33
MLC	34
TPCTL2	99
RAMP2	98
HSPWM	139
HSPSO	140
MPARA	141
PSOFF	142
PSCNV	143
MPG	148

ModBUS	150
CLINK	151
ReadWriteFileReg	160
WriteSDMem	161
ReadSDMem	162
PID2	38
DBUF	115
ICA	137
ICF	138
NCR	152
CMCTL	156
ME_START	176
ME_SYSSTOP	177
ME_HOME	178
ME_POS	179
ME_JOG	180
ME_CHGPRM	181
ME_PAUSE	182
ME_RESUME	183
ME_HALT	184
ME_RSTALM	185
ME_STOP	186
ME_SYSINIT	187

ME_RCPR	188
ME_RCPW	189
ME_CAMR	191
ME_CAMW	192
ME_GEAR_IN	193
ME_VEL_CTL	194
ME_TOR_CTL	195
ME_CAM_GEN	196
ME_AXI_MOV	197
ME_SET_MAP	198
ME_VIR_AXI	235
HSPWM2	144
TARESUB	
TAREZEOFFSET	

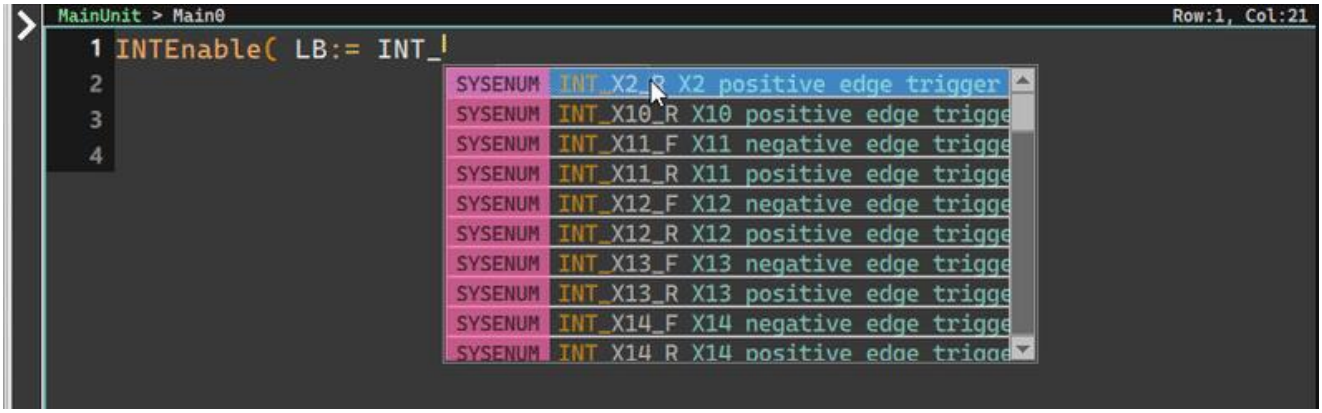
3-12 Enable/Disable Interrupt and Special Instructions

Please refer to FUN 145, 146

```
INTEnable( LB:= (* Interrupt number (Range 1~48) *));
INTDisable( LB:= (* Interrupt number (Range 1~48) *));
```

LB parameters input integers from 1 to 49, corresponding to the types of interrupts.

For all supported types, please refer to the chapter of special instructions.



INT No. UserView	Interrupt Source	Priority	Interrupt Label	Condition	Note
1	Build-in Digital Inputs		X0+I (INT0+)	X0 positive edge trigger	The software high speed counter HSC4~HSC7 can be assigned as the trigger source of any interrupt X0~X15. Therefore, the interrupt priority of the software high speed counter depends on the priority of X0~X15.

2			X0−I (INT0-)	X0 negative edge trigger	
3			X1+I (INT1+)	X1 positive edge trigger	
4			X1−I (INT1-)	X1 negative edge trigger	
5			X2+I (INT2+)	X2 positive edge trigger	
6			X2−I (INT2-)	X2 negative edge trigger	
7			X3+I (INT3+)	X3 positive edge trigger	
8			X3−I (INT3-)	X3 negative edge trigger	
9			X4+I (INT4+)	X4 positive edge trigger	
10			X4−I (INT4-)	X4 negative edge trigger	

11			X5+I (INT5+)	X5 positive edge trigger	
12			X5−I (INT5-)	X5 negative edge trigger	
13			X6+I (INT6+)	X6 positive edge trigger	
14			X6−I (INT6-)	X6 negative edge trigger	
15			X7+I (INT7+)	X7 positive edge trigger	
16			X7−I (INT7-)	X7 negative edge trigger	
17	Hardware Time Tick	3	STM0I	Interval from 1ms~60000ms	Tick unit 1ms，Value=1~60000
18		3	STM1I	Interval from 1ms~60000ms	
19		3	STM2I	Interval from 1ms~60000ms	

20		3	STM3I	Interval from 1ms~60000ms	
21		3	LTM0I	Interval from 10ms~60000ms	Tick unit 10ms，Value=1~6000
22		3	LTM1I	Interval from 10ms~60000ms	
23		3	LTM2I	Interval from 10ms~60000ms	
24		3	LTM3I	Interval from 10ms~60000ms	
25	HST	1	HST0I	Interval from 0.1ms to 6000ms	Tick uint 100us，Value=1~60000
26		1	HST1I	Interval from 0.1ms to 6000ms	
27		1	HST2I	Interval from 0.1ms to 6000ms	
28		1	HST3I	Interval from 0.1ms to 6000ms	

33	HSC	2	HSC0I	Interval from HSC0 to (CV=PV)	
34		2	HSC1I	Interval from HSC1 to (CV=PV)	
35		2	HSC2I	Interval from HSC2 to (CV=PV)	
36		2	HSC3I	Interval from HSC3 to (CV=PV)	
37		2	HSC4I	Interval from HSC4 to (CV=PV)	
38		2	HSC5I	Interval from HSC5 to (CV=PV)	
39		2	HSC6I	Interval from HSC6 to (CV=PV)	
40		2	HSC7I	Interval from HSC7 to (CV=PV)	

66	Build-in Digital Inputs (MA Series)		X8+I (INT8+)	X8 positive edge trigger	The software high speed counter HSC4~HSC7 can be assigned as the trigger source of any interrupt X0~X15. Therefore, the interrupt priority of the software high speed counter depends on the priority of X0~X15.
67			X8−I (INT8-)	X8 negative edge trigger	
68			X9+I (INT9+)	X9 positive edge trigger	
69			X9−I (INT9-)	X9 negative edge trigger	
70			X10+I (INT10+)	X10 positive edge trigger	
71			X10−I (INT10-)	X10 negative edge trigger	

72			X11+I (INT11+)	X11 positive edge trigger	
73			X11−I (INT11-)	X11 negative edge trigger	
74			X12+I (INT12+)	X12 positive edge trigger	
75			X12−I (INT12-)	X12 negative edge trigger	
76			X13+I (INT13+)	X13 positive edge trigger	
77			X13−I (INT13-)	X13 negative edge trigger	
78			X14+I (INT14+)	X14 positive edge trigger	
79			X14−I (INT14-)	X14 negative edge trigger	
80			X15+I (INT15+)	X15 positive edge trigger	

81			X15−I (INT15-)	X15 negative edge trigger	
82	MQ model DI Plug-in		X16+I (INT15+)	X16 上沿觸發	
83			X16−I (INT15-)	X16 下沿觸發	
84			X17+I (INT15+)	X17 上沿觸發	
85			X17−I (INT15-)	X17 下沿觸發	
86			X18+I (INT15+)	X18 上沿觸發	
87			X18−I (INT15-)	X18 下沿觸發	
88			X19+I (INT15+)	X19 上沿觸發	
89			X19−I (INT15-)	X19 下沿觸發	

90			X20+I (INT15+)	X20 上沿觸發	
91			X20−I (INT15-)	X20 下沿觸發	
92			X21+I (INT15+)	X21 上沿觸發	
93			X21−I (INT15-)	X21 下沿觸發	
94			X22+I (INT15+)	X22 上沿觸發	
95			X22−I (INT15-)	X22 下沿觸發	
96			X23+I (INT15+)	X23 上沿觸發	
97			X23−I (INT15-)	X23 下沿觸發	

3-13 Variable names with special meanings

A. Timer, Counter

T0~TN, C0~CN

In the ST environment, these two variables are given the characteristics of Bool and the characteristics of Integer.

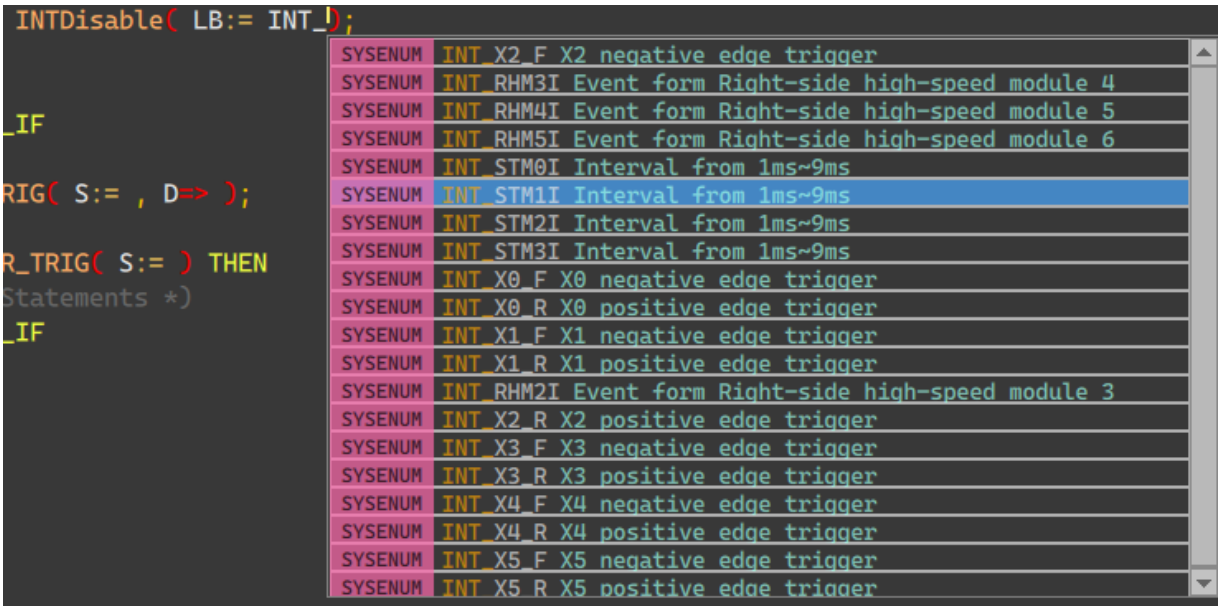
EX:

```
R0:=T1;
//The currently calculated value of Timer1 will be stored in R0(Integer)
M0:=T1;
// Will save the status of whether Timer1 is currently running to M0 (Bool)
 M3:=C1>10;
 // Will determine whether Counter 1 counts more than 10

 IF R_TRIG( T1 ) THEN
  /// When Timer1 changes from 0 to 1 (the moment it starts running), it will be executed
END_IF
```

B. Interrupt (Enum)

Originally, only constants could be input for Enable/Disable interrupts. In order to improve the readability of the program, it was modified to allow the input of system default Enum parameters. If you type INT_, the corresponding series of available Enums will be displayed**.**

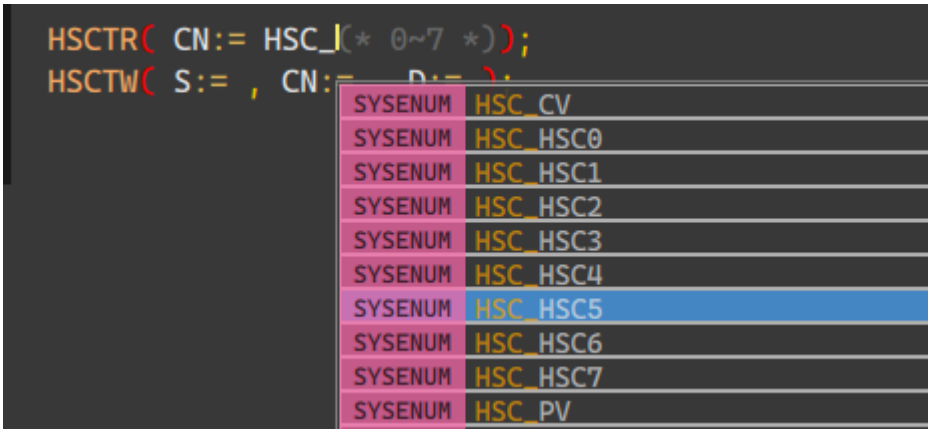


EX:

```
IF M2 THEN
    INTEnable( LB:= INT_STM1I);
ELSE
    INTDisable( LB:= INT_STM1I);
END_IF
```

C. High-speed counting Enum “HSC_” Prefix
Call related high-speed instructions such as
HSCTR(CN:= (* 0~7 *));
HSCTW(S:= , CN:= , D:=);;

Among them, the parameter expressions of CN, and D support constant integers. For the convenience of reading, Enum starting with HSC_ is also imported.



Make programs more readable
EX:

HSCTR(CN:= HSC_HSC0(* 0~7 *));
HSCTW(S:=R0 , CN:=HSC_HSC0 , D:=HSC_PV);

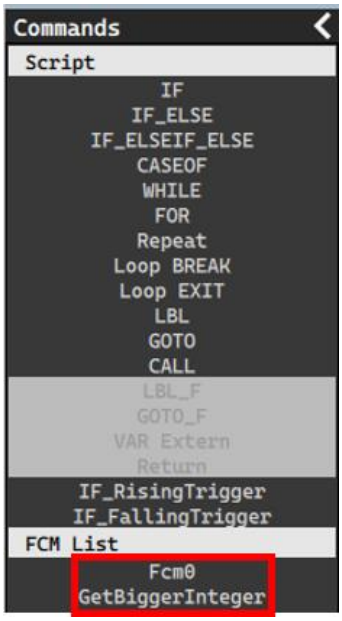
D. File Register Enum F0~F32767

For the Sb parameter of FUN 160, in addition to using 0~32767, you can use the Enum of F0~F32767 to represent the corresponding position.

EX:

3-14 Calling FCM Function

The calling method is similar to calling system function, while the functions are built by users themselves. The built functions will be placed in FCM List in the command column on the left.

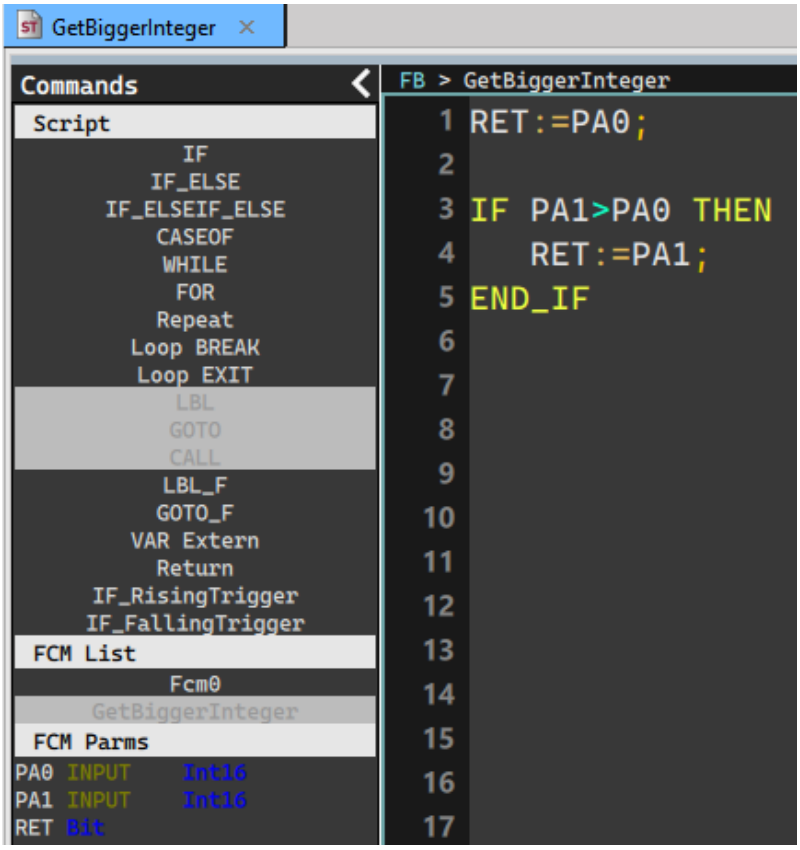


Users can double-click the section to insert the selected function to the text.

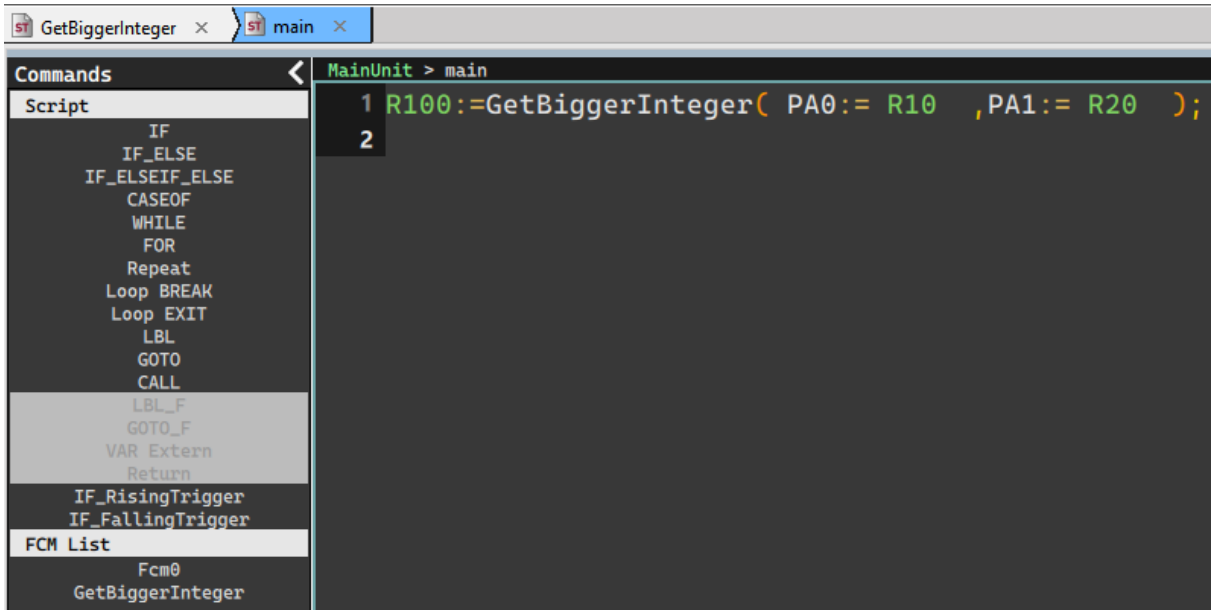
FCM can specify a Return Value, which can be call directly when programming.

EX:

FCM:



Calling side:



In this way, some temporarily unnecessary variable declarations can be reduced in a timely manner.

3-15 Function notes

Whether it is built-in call system or FCM, they all have the following common characteristics:

A. Calling method:

1. Enter the complete parameter name

2. Omit parameter names

3. Output parameters (=>) can be directly ignored and not passed.

example:

Take the R_TRIG function as an example

Complete performance: R_TRIG(S:= M0, D=>M1);

If you use parameters to bring it in, it is not allowed to follow the preset order.

The above function can be changed to

R_TRIG(D=>M1, S:= M0);

R_TRIG(D=>M1);

All are legal inputs

Omit parameter name: R_TRIG(M0, M1);

Those with Return effect or output do not need to be passed :

```
R_TRIG( S:= M0 );

R_TRIG( M0 );
```

Those with the Return attribute can be mixed with other variables or descriptive sentences.

```
IF R_TRIG(M0) THEN
   //// some statements
END_IF

M10:= R_TRIG(M0) AND R_TRIG(M1);
```

! Note! Calling methods with parameters and without parameters cannot be mixed. For example:

R_TRIG(S:=M0, M1);

R_TRIG(M1,D:=M2);

Both of the above are illegal inputs.

4

ST Editing Environment Support Function List

4-1 Name comparison between Ladder and ST editing environment instructions

ST function name	Parameter	Instruction manual corresponding information		Remark
		Number	Name	
ToBCD ToBCD_D	S,D	Fun20	BCD	
ToBIN ToBIN_D	S,D	Fun21	BIN	
F_TRIG	S,D	Fun4	DIFU	
R_TRIG	S,D	Fun5	DIFD	
Timer	EN,T,PV,IsTimeout	T	TIMER	
Counter Counter_L	Pulse,clr,C,PV,IsUp	C	COUNTER	
ABS ABS_D	D	Fun28	ABS	
CRC16	S,N,D	Fun31	CRC16	
DIV	Sa,Sb,D	Fun14	DIVISION	
FtoI FtoI_D	S,D	Fun201	F->I	
FACOS_deg	S,D	Fun219	FACOS	
FACOS_rad	S,D	Fun219	FACOS	
FASIN_deg	S,D	Fun218	FASIN	
FASIN_rad	S,D	Fun218	FASIN	
FCOS	S,D	Fun210	FCOS	
FDIV	Sa,Sb,D	Fun205	FDIV	
FMUL	Sa,Sb,D	Fun204	FMUL	
FSIN	S,D	Fun209	FSIN	
FSQR	S,D	Fun208	FSQR	
FTAN	S,D	Fun211	FTAN	
FZCP	S,Su,S1,INZ,ERR S_Gt_U, S_Less_L	Fun207	FZCP	
ItoF ItoF_D	S,D	Fun200	I->F	
LCNV	EN,Md,S,Ts,D,L	Fun33	LCNV	

LCNV_D				
MEAN MEAN_D	S,N,D	Fun25	MEAN	
MLC	EN,XY,Rs,Sl,Tx,Ty,Tl,D,OVR	Fun34	MLC	
MUL	Sa,Sb,D	Fun13	MULTIPLICATION	
SUM SUM_D	S,N,D	Fun24	SUM	
BITRD BITRD_D	S,N,OTB,ERR	Fun40	BITRD	
BITWR BITWR_D	INB,D,N,ERR	Fun41	BITWR	
DBUF	EN,ID,CH,D,DN	Fun115	DBUF	
ReadSDMem	EN,INC,Bk,Os,Pr,L,D,ERR	Fun162	RD-MP	
ReadWriteFileReg ReadWriteFileReg_D	EN,R_W,INC,Sa,Sb,Pr,L	Fun160	RW-FR	
SWAP	D	Fun46	SWAP	
WriteSDMem	EN,INC,S,Bk,Os,Pr,L,WR,ACT,ERR,DN	Fun161	WR-MP	
ToASCII	S,N,D	Fun64	->ASCII	
ToBCD ToBCD_D	S,D	Fun20	->BCD	
ToBIN ToBIN_D	S,D	Fun21	->BIN	
ToHEX	S,N,D	Fun63	->HEX	
ToHMS	S,D	Fun62	->HMS	
ToSEC	S,D	Fun61	->SEC	
BintoGray BintoGray_D	S,D	Fun55	B->G	
DECOD	S,Ns,Nl,D,D,ERR	Fun57	DECOD	
ENCOD	H_L,S,Ns,Nl,D,D_0,ERR	Fun58	ENCOD	
GraytoBin GraytoBin_D	S,D	Fun56	G->B	
PID	AM,BUM,D_R,Ts,	Fun30	PID	

PID_D	SR,O_R,PR,WR,CC ERR,HAL,LAL			
PID2	EN,UPD,AM,D_R, ID,CH,SR,OutR,PR, WR,ERR	Fun38	PID2	
TPCTL2	EN,UPD,A_M,D_R, ID,CH,SR,PR,OriR, WR,ERR,ALM	Fun99	TPCTL2	
IMDIO	D,N	Fun74	IMDIO	
SPD	EN,S,TI,D,OVF	Fun83	SPD	
ACTimer_10MS ACTimer_10MS_D	TIM,EN,CV,PV,TUP, NUP	Fun87	T.01S	
ACTimer_100MS ACTimer_100MS_D	TIM,EN,CV,PV,TUP, NUP	Fun88	T.1S	
ACTimer_1S ACTimer_1S_D	TIM,EN,CV,PV,TUP, NUP	Fun89	T1S	
RSWDT		Fun91	RSWDTF	
WDT	N	Fun90	WDT	
HSCTR	CN	Fun92	HSCTR	
HSCTW	S,CN,D	Fun93	HSCTW	
RAMP2	EN,Om,Ta,Td,Rt,Rc, WR,ACC,DEC	Fun98	RAMP2	
CLINK	EN,PAU,ABT,Pt,MD ,WR,ERR,DN	Fun151	CLINK	
CMCTL	EN,PAU,ID,Pt,Ts, MD,WR,ERR	Fun156	CMCTL	
ModBUS	EN,A_R,ABT,PT,SR WR,ACT,ERR,DN,	Fun150	M-Bus	
NCR	NE,SR,MD,WR,AC T,ERR,DN	Fun152	NCR	
BKCMP BKCMP_D	Rs,Ts,L,D	Fun112	BKCMP	
BT_M BT_M_D	Ts,Td,L	Fun103	BT_M	
QUEUE QUEUE_D	InOut,IW,Qu,L,Pr, OW,EPT,FUL	Fun110	QUEUE	
T_Search	FHD,DS,Rs,Ts,L,Pr,F	Fun105	R-T_S	

T_Search_D	ND,END,ERR			
SORT SORT_D	AD,S,D,L	Fun113	SORT	
STACK STACK_D	InOut,IW,ST,L,Pr, OW,EPT,FUL	Fun111	STACK	
T_Compare T_Compare_D	FHD,DS,Ta,Tb,L,Pr, FND,END,ERR	Fun106	T-T_C	
T_Fill T_Fill_D	Rs,Td,L	Fun107	T_FIL	
ZoneWR	Val,D,N	Fun114	Z-WR	
MAND	Ma,Mb,Md,L	Fun120	MAND	
MBCNT	OnOff,Ms,L,D	Fun130	MBCNT	
MBRD	EN,INC,CLR,Ms,L, Pr,OTB,END,ERR	Fun126	MBRD	
MBROT	L_R,Ms,Md,L,OTB	Fun129	MBROT	
MBSHF	INB,L_R,Ms,Md,L, OTB	Fun128	MBSHF	
MINV	Ms,Md,L	Fun124	MINV	
MOR	Ma,Mb,Md,L	Fun121	MOR	
MXNR	Ma,Mb,Md,L	Fun123	MXNR	
MXOR	Ma,Mb,Md,L	Fun122	MXOR	
HSPSO	EN,PAU,ABT,Ps,SR, WR,ACT,ERR,DN	Fun140	HSPSO	
HSPWM	EN,Pw,Op,Rs,Pn, OutR,WR,ACT	Fun139	HSPWM	
HSPWM2	EN,Pw,Op,Hz,O_R, ACT,ERR	Fun144	HSPWM2	
ICA	EN,D_R,Ps,Is,Fo,Ag ,ACT,ERR,DN	Fun137	ICA	
ICF	EN,D_R,Ps,Is,Fo,Fd ,ACT,ERR,DN	Fun138	ICF	
MHSPO	EN,PAU,ABT,Gp,SR ,WR,ACT,ERR,DN	Fun147	MHSPO	
MPARA	EN,Ps,SR	Fun141	MPARA	
MPG	EN,Sc,Ps,Fo,Mr,WR ,ACT	Fun148	MPG	
PSCNV	EN,Ps,D	Fun143	PSCNV	

PSOFF	EN,Ps	Fun142	PSOFF	
INTDisable	LB	Fun146	DIS	
INTEnable	LB	Fun145	EN	
TARESUB	EN,RST,ID,CH,SB, ERR	Fun258	MODCONF	
TAREZEOFFSET	EN,MD,ID,CH,WR, ERR	Fun258	MOD	
MFSysCAMR	EN,Md,D,ID,L,ACT, ERR,DN	Fun191	MFSysCAMR	
MFSysCAMW	EN,Md,D,ID,L,ACT, ERR,DN	Fun192	MFSysCAMW	
MFSysInit	EN,ACT,ERR,DN	Fun187	MFSysInit	
MFSysRCPR	EN,Md,D,Gp,ACT,E RR,DN	Fun188	MFSysRCPR	
MFSysRCPW	EN,MD,D,GP,ACT, ERR,DN	Fun189	MFSysRCPW	
MFSysRstAlm	EN,ACT,ERR,DN	Fun185	MFSysRstAlm	
MFSysSetVirt	EN,AX,ACT,ERR,D N	Fun235	MFSysSetVirt	
MFSysStop	EN,ACT,ERR,DN	Fun177	MFSysStop	
MFChgTbPrm	EN,GP,ID,N,D,ERR, DN	Fun181	MFChgTbPrm	
MFFlowHalt	EN,ID,ACT,ERR,DN	Fun184	MFFlowHalt	
MFFlowPause	EN,ID,ACT,ERR,DN	Fun182	MFFlowPause	
MFFlowStart	EN,ID,ACT,ERR,DN	Fun176	MFFlowStart	
MFFlowStop	EN,ID,ACT,ERR,DN	Fun186	MFFlowStop	
MFMapTbPrm	EN,GP,ID,N,D,ERR, DN	Fun198	MFMapTbPrm	
MFHome	EN,AX,ACT,ERR, DN	Fun178	MFHome	
MFJog	EN,D_R, AX, MD,ACT,ERR ,DN	Fun180	MFJog	
MFPointMov	EN,PT,AX,ACT,ERR, DN	Fun179	MFPointMov	
MFAxCirMov	EN,UPD,TAX,MD, DR,BF,D,EC,ACT,ER R,DN,UPD	Fun239	MFAxCirMov	

MFAxLMov	EN,UPD,TAX,MD, DR,BF,D,EC,ACT, ERR,DN,UPD	Fun238	MFAxLMov	
MFAxMov MFAxMov_D	EN,UPD_in,AX,MD ,Ps,V_n,A,D,SA,SD, DR,BF,ACT,ERR,DN ,UPD_out	Fun197	MFAxMov	
MFGearMPG MFGearMPG_D	EN,UPD_in,M,S,N, D,T,ACT,ERR,DN,U PD_out	Fun193	MFGearMPG	
MFPathMov	EN,ACT,UPD,AX1, AX2,AX3,PT,V,TA,T D,SA,SD,EC,ACT,ER R,DN,UPD	Fun240	MFPathMov	
MFTorqCtl MFTorqCtl_D	EN,UPD_in,AX,T,M X,ACT,ERR,DN, UPD_out	Fun195	MFTorqCtl	
MFVelCtl MFVelCtl_D	EN,UPD_in,AX,T,M X,ACT,ERR,DN, UPD_out	Fun194	MFVelCtl	

4-2 Functions with multiple calling modes

ST function name	Parameter	Parameter description
ToBCD ToBCD_D	S,D	S: The data being converted or its register number. D: The register number that stores the conversion result (BCD code)
ToBIN ToBIN_D	S,D	S: The data being converted or its register number. D: The scratchpad number where the conversion result (BIN code) is stored.
F_TRIG	S,D	S: The data being converted or its register number. D: The relay coil number where the derivative result is stored
R_TRIG	S,D	S: The data being converted or its register number. D: The relay coil number where the derivative result is stored
Timer	EN,T,PV,IsTimeout	Tn: The timer number, which is the stored accumulated timekeeping time (i.e. the current value CV). /b14> PV :(Preset Value) is the set value of the timer.
Counter Counter_L	Pulse,clr,C,PV,IsUp	CN: The counter number, which is the accumulated count value (i.e. the current value CV). PV:(Preset Value) is the setting of the counter.
ABS ABS_D	D	D: Take the register number of the absolute value
CRC16	S,N,D	MD:0, when calculating CRC, only the low group of the scratchpad is calculated, and the bytes of the scratchpad are not counted :1, reserved S: The starting register number of the CRC needs to be calculated N: The data length of the CRC needs to be calculated, and the unit is bytes D: The register number that stores the CRC calculation result,

		and the register D stores the Upper Byte register D of the CRC calculation result +1 Lower Byte S,N,D of the stored CRC resultThe operator can be combined with V, Z, P0~P9 indicators for indirect addressing applications
DIV	Sa,Sb,D	Sa: The dividend number or its scratchpad number. SB: The divisor or its register number. D: The(register number where the result (quotient and remainder) is stored. Sa,Sb,D can be combined with V,Z,P0~P9 for indirect addressing applications.
FtoI FtoI_D	S,D	S: The starting number of the source register D: The register used by the register start number operator to store the result (integer) must be an even address, e.g. R8 is valid, R7 is not. /b110> The S and D operators can be combined with V, Z, P0~P9 indicators for indirect addressing with
FACOS_deg	S,D	S: Find the source value or the register number of the arcsine function value. /b11> The register used by the operator must be an even address, e.g. R8 is valid and R7 is not. /b17> D : The) number of the register that holds the result (arcsine value). S and D can be combined with V, Z, P0~P9 for indirect addressing applications.
FACOS_rad	S,D	
FASIN_deg	S,D	S: Find the source value or the register number of the arcsine function value. /b11> The register used by the operator must be an even address, e.g. R8 is valid and R7 is not. /b17> D : The) number of the register that holds the result (arcsine value). /b113> S and D can be combined with V, Z, P0~P9 for indirect addressing applications.
FASIN_rad	S,D	
FCOS	S,D	S: Find the source value or register number of the COS

		value. D : The number of the scratchpad where the result is stored. /b12> The register used by the operator must be an even address, e.g. R8 is valid and R7 is not. The S and D operators can be combined with V, Z, P0~P9 indicators for indirect addressing applications
FDIV	Sa,Sb,D	Sa: The dividend number or its scratchpad number. SB: The divisor or its register number. D: The register start number of the result (quotient) The register used by the operator must be an even address, e.g. R8 is legitimate, R7 /b110>It's not legal. The S and D operators can be combined with V, Z, P0~P9 indicators for indirect addressing applications
FMUL	Sa,Sb,D	Sa: The multiplier or its scratchpad number. SB: Multiplier or its scratchpad number. D: The register start number of the result (product) The register used by the operator must be an even address, e.g. R8 is legitimate, R7 /b110>It's not legal. The S and D operators can be combined with V, Z, P0~P9 indicators for indirect addressing applications
FSIN	S,D	S: The source value or scratchpad number for which the SIN value is found. D : The number of the register where the result is stored. /b11> The register used by the operator must be an even address, e.g. R8 is valid and R7 is not. The S and D operators can be combined with V, Z, P0~P9 indicators for indirect addressing applications

FSQR	S,D	S: Find the source value or scratchpad number of the square root. D : The register number that holds the result (square root value) /b15> The register used by the operator must be an even address, e.g. R8 is valid and R7 is not. The S and D operators can be combined with V, Z, P0~P9 indicators for indirect addressing applications
FTAN	S,D	S: The source value or scratchpad number of the TAN value. /b13> D : The number of the register where the result is stored. /b15> The register used by the operator must be an even address, e.g. R8 is valid and R7 is not. The S and D operators can be combined with V, Z, P0~P9 indicators for indirect addressing applications
FZCP	S,Su,S1,INZ,ERR S_Gt_U, S_Less_L	S :) The number of the register that holds the comparison data (floating-point number). SU: The upper limit value of the area or the upper limit value of the register number. SL: The lower limit value of the zone or the number of the lower limit value scratchpad. /b11> The register used by the operator must be an even address, e.g. R8 is valid and R7 is not. The S and D operators can be combined with V, Z, P0~P9 indicators for indirect addressing applications
ItoF ItoF_D	S,D	S : The starting number of the source register D: The register start number of the result (floating-point) The register used by the operator must be an even address, e.g. R8 is valid, R7 /b110>It's not legal. The S and D operators can be combined with V, Z, P0~P9 indicators for indirect addressing applications

LCNV LCNV_D	EN,Md,S,Ts, D,L	Md: operation mode selection, 0~3 S: the start number of the source register to be converted TS: the start number of the start register of the conversion table D: The starting register number where the conversion result is stored L: The length to be converted, 1~64
MEAN MEAN_D	S,N,D	S: The starting number of the source register N: The number of registers to be averaged (N consecutive registers starting from S). D: The register number where the result (average) is stored S,N,D can be combined with V,Z,P0~P9 for indirect addressing applications
MLC	EN,XY,Rs,Sl,T x,Ty,Tl,D,OVR	Rs: Source data start register number Sl: The length of the source data to be converted, 1~64 Tx:X converts the table start register start number Ty:Y converts the start number of the table start register Tl: Conversion table length, 2~255 D: The starting register number where the conversion result is stored
MUL	Sa,Sb,D	Sa: The multiplier or its scratchpad number. SB: Multiplier or its scratchpad number. D: The register number where the result (product) is stored. Sa, Sb, D can be combined with V, Z, P0~P9 for indirect addressing.
SUM SUM_D	S,N,D	S: The starting number of the source register N: The number of registers to be summed (N consecutive registers starting from S). D: The register number where the result (sum) is stored S,N,D can be combined with V,Z,P0~P9 for indirect addressing applications
BITRD BITRD_D	S,N,OTB,ERR	S : The data of the bit to be read or its register number N: Specifies that the Nth digit of data in the S data is read

		out S,N can be combined with V, Z, P0~P9 for indirect addressing applications
BITWR BITWR_D	INB,D,N,ERR	D: The register number of the bit to be written N: Specifies that the state of the write bit INB is written to the Nth bit D in D, and N can be combined V, Z, P0~P9 for indirect addressing applications
DBUF	EN,ID,CH,D, DN	ID: Extension module ID number (0~N). CH : Extended Module Channel Index (0 ~ N) D : The start register that holds the cache data
ReadSDMem	EN,INC,Bk,Os,Pr,L,D,ERR	BK: SD card block number, 0~1 OS: the starting location of the partitioned data PR: Indicator scratchpad number L: Read data length 1~128 D: The start number of the register that stores the read data
ReadWriteFileReg ReadWriteFileReg_D	EN,R_W,INC,Sa,Sb,Pr,L	SA: The starting register number of the scratchpad list SB: The starting number of the file register PR: Indicator scratchpad number L: The length of the list is 1~511 Sa can be combined with V, Z, P0~P9 for indirect addressing applications
SWAP	D	D: The register number that performs byte data swapping D can be combined with V, Z, P0~P9 for indirect addressing applications
WriteSDMem	EN,INC,S,Bk,Os,Pr,L,WR,ACT,ERR,DN	S: The number of the source from which the data is written BK: the block number of the SD card, 0~1 OS: the starting position of the partition data Pr: Indicator register number L: Write data length 1~128 WR: The starting number of the working register, which occupies a total of 2 registers S can be combined with V, Z, P0~P9 for indirect addressing

		applications
ToASCII	S,N,D	S : The starting number of the source register N: The number of ASCII codes to be converted into hexadecimal values D: The start number of the register where the result (ASCII code) is stored S,N,D can be combined with V,Z,P0~P9 for indirect addressing applications
ToBCD ToBCD_D	S,D	S: The data being converted or its register number. D: The register number that stores the conversion result (BCD code) S, D can be combined with V, Z, P0~P9 for indirect addressing applications.
ToBIN ToBIN_D	S,D	S: The data being converted or its register number. D: The scratchpad number where the conversion result (BIN code) is stored. S,D can be combined with V,Z,P0~P9 for indirect addressing applications
ToHEX	S,N,D	S : The starting number of the source register N: The number of hexadecimal values to be converted to ASCII codes D: The start number of the scratchpad where the result (hexadecimal value) is stored S,N,D can be combined with V,Z,P0~P9 for indirect addressing applications
ToHMS	S,D	S: The number of seconds to be converted, the starting number of the data register D: The first number of the register stored in the transformation result (hours:minutes:seconds).
ToSEC	S,D	S: The starting number of the time data register to be changed D: The starting number of the register where the result is stored
BintoGray	S,D	S : The starting number of the source register

BintoGray_D		D: The start number of the scratchpad where the result (Gray code) is stored The S,D operators can be combined with V, Z, P0~P9 indicators for indirect addressing applications
DECOD	S,Ns,Nl,D,D,ERR	S: The decoded source data register number (16 bits). NS: The start bit to be decoded in S NL: The length of the decoded value (1~8 bits) D : The starting number of the register (2~256 points=1~16 words) where the decoding result is stored S, NS, NL, D can be combined with V, Z, P0~P9 for indirect addressing
ENCOD	H_L,S,Ns,Nl,D,D_0,ERR	S : The first digit of the encoded register NS: Specify one of the points in S as the starting point of the encoding NL: the number of single points of the encoding (2~256 points) D : The register number (1 word) S,NS,NL that holds the encoding result, D can be combined with V, Z, P0~P9 for interconnected addressing applications
GraytoBin GraytoBin_D	S,D	S : The starting number of the source register D: The start number of the scratchpad where the result (Gray code) is stored The S,D operators can be combined with V, Z, P0~P9 indicators for indirect addressing applications
PID PID_D	AM,BUM,D_R,Ts,SR,O_R,PR,WR,CCERR,HAL,LAL	Ts: The interval between PID operations SR : Programmable setpoint start register number, occupies a total of 8 registers OR: PID output register number PR: Parameter setting start register number, 7 registers occupied WR: The starting number of the working register required for this instruction occupies a total of 5 scratchpads, It cannot be reused elsewhere.
PID2	EN,UPD,AM,D_R,ID,CH,SR,Out	ID: The number of the extension module used as the input signal

	R,PR,WR,ERR	CH : The channel number used as the input signal SR : Programmable setpoint start register number, occupies a total of 8 registers OR: PID output register number PR: Parameter setpoint start register number, 7 registers occupied in total WR: The working instantial memory start number required for this instruction occupies a total of 5 scratchpads, and cannot be reused elsewhere.
TPCTL2	EN,UPD,A_M ,D_R, ID,CH,SR,PR, OriR, WR,ERR,ALM	ID: The number of the extension module used as the input signal CH: The channel number used as the input signal SR : Programmable start register PR : Gain setting start register OR: Output start register WR: Working Start Register
IMDIO	D,N	D: The I /O number to be updated N: The number of I/O points to be updated
SPD	EN,S,TI,D,OV F	S : To detect the pulse of velocity input point TI: Detection sampling time (in mS). D: The register number where the result is stored
ACTimer_10MS ACTimer_10MS _D	TIM,EN,CV,P V,TUP,NUP	CV: The number of the register that holds the timed time (i.e. the current value). PV: A timer setting or a register number range operator that stores the set value WX WY WM WS TMR CTR HR IR OR SR ROR DR
ACTimer_100M S ACTimer_100M S_D	TIM,EN,CV,P V,TUP,NUP	
ACTimer_1S ACTimer_1S_D	TIM,EN,CV,P V,TUP,NUP	
RSWDT		There are no operators in this command
WDT	N	N: The set time of the monitoring timer. /b11> Its value can only be 50, 60, 70... ... 120, the unit is 10mS, that is, the set time range is (50~ 120)×10mS, that

		is, 0.5 seconds ~ 1.2 seconds
HSCTR	CN	CN: Hardware high-speed counter number 0: HSC0 1: HSC1 2: HSC2 3: HSC3 4：HSC4 5：HSC5 6：HSC6 7：HSC7
HSCTW	S,CN,D	CN: Hardware high-speed counter number 0: HSC0 1: HSC1 2: HSC2 3: HSC3 4：HST4 2：HSC2 3：HSC3 4：HST4 D: Write object (0: for CV, 1: for PV)
RAMP2	EN,Om,Ta,Td,Rt,Rc,WR,ACC,DEC	OM: Maximum output setting; Set the range 0~65535 TA: Rise time from 0 to maximum output; The setting range is 0~65000, and the unit is MS TD: The fall time from the maximum output to 0 is set; The setting range is 0~65000, and the unit is MS RT: Target output setting scratchpad; Set the range 0~65535 RC: The current output register, which is used as a D/A output WR: The starting address of the working register, which occupies a total of 4 registers
CLINK	EN,PAU,ABT,Pt,MD,WR,ERR,DN	Pt: Specify the port, 1, 2 MD: Communication mode selection (MD0~MD3). SR : Holds the start register of the communication program WR: The instruction operates the start register, which occupies a total of 8 registers, and other programs cannot be reused.
CMCTL	EN,PAU,ID,Pt,Ts,MD,WR,ERR	ID: The number of the module used Pt: Specifies COMA/COMB (A= 0; B=1) TS: Communication Form Blocking Bit 1: Form 0, Bit 2: Form 1 ~ Bit 15: Form 15 Bit 16~Bit 31: Reserved, not usable MD: Setup Mode 0: RUN ONCE 1: RUN CYCLING 2: STOP WR: Store working status Bit0~Bit1: Table 0 state Bit2~Bit3: Table 1 state ~ Bit30~Bit31: Table 15 state (= 0:RUN_ONCE = 1:RUN_CYCLING = 2:STOP)

ModBUS	EN,A_R,ABT,PT,SRWR,ACT,ERR,DN,	PT: 1, 2, through this port,, the MODBUS RTU communication protocol is used for data transmission SR: Communicator start register (see example). WR: The instruction operation start register (see example), which occupies a total of 8 registers, and cannot be reused by other programs
NCR	NE,SR,MD,WR,ACT,ERR,DN	SR: Table start register address. MD: Modbus TCP active communication (=1). WR: Working Register.
BKCMP BKCMP_D	Rs,Ts,L,D	RS : the data being compared or its register number TS : The starting number of the upper and lower limit register blocks L: The number of groups of upper and lower limits D: The starting number of the relay where the comparison result is stored
BT_M BT_M_D	Ts,Td,L	TS: The register number at the beginning of the source list TD: The register number at the beginning of the list of destinations L: The length of the list of sources and destinations TS, TD can be combined with V, Z, P0~P9 for indirect addressing applications
QUEUE QUEUE_D	InOut,IW,Qu,L,Pr,OW,EPT,FUL	IW: The information or its register number that is squeezed into the list QU: The number of the starting register of the store L: The length of the storage column PR: Indicator scratchpad number OW: The number of the register that receives the data removed from the memory QU can be combined with V, Z, P0~P9 for indirect addressing applications
T_Search T_Search_D	FHD,DS,Rs,Ts,L,Pr,FND,EN	Rs: The source (sample) information sought or its scratchpad number

	D,ERR	TS: The register number at the beginning of the searched list L: the length of the list PR: Indicator, which is used to record the value of the location of the target Rs,Ts can be combined with V, Z, P0~P9 for indirect addressing applications
SORT SORT_D	AD,S,D,L	S : The register number at the beginning of the source data to be sorted D: The registered number at the beginning of the sorted data L: The length of the sorted data
STACK STACK_D	InOut,IW,ST,L,Pr,OW,EPT,FUL	IW: Stuffed data or its scratchpad number ST: The register number at the beginning of the stack L: The length of the stack PR: Indicator scratchpad number OW: The register number that receives the stack outgoing data ST can be combined with V, Z, P0~P9 for indirect addressing applications
T_Compare T_Compare_D	FHD,DS,Ta,Tb,L,Pr,FND,END,ERR	TA: The starting number of the register of List A TB: The starting number of the register in List B L: The length of lists a and b PR: Indicator, which is used to record the value of the location of the target Ta,Tb can be combined with V, Z, P0~P9 for indirect addressing applications
T_Fill T_Fill_D	Rs,Td,L	RS: The source information or its register number to be included in the list TD: The register number at the beginning of the list L : The length of the list RS, TD can be combined with V, Z, P0~P9 for indirect addressing applications

ZoneWR	Val,D,N	D: The starting address of the region to be written or cleared N: The length of the area to be written or cleared: 1~511 D, N can be combined with V, Z, P0~P9 for indirect addressing applications
MAND	Ma,Mb,Md,L	MA: The starting register number of Source Matrix A MB: The register number at the beginning of Source Matrix B MD: The number of the scratchpad at the beginning of the matrix for which the results are stored L: The length of the matrix (Ma, Mb, and Md). Ma, Mb, Md can be combined with V, Z, P0~P9 for indirect addressing applications
MBCNT	OnOff,Ms,L, D	MS: The register number at the beginning of the matrix L : The length of the matrix D : The register number where the quantity result is stored MS can be combined with V, Z, P0~P9 for indirect addressing applications
MBRD	EN,INC,CLR, Ms,L, Pr,OTB,END,ERR	Ms: The scratchpad number at the beginning of the matrix L : The length of the matrix PR: Indicator register number Ms can be combined with V, Z, P0~P9 for indirect addressing applications
MBROT	L_R,Ms,Md,L, OTB	Ms : The register number at the beginning of the source matrix MD: The register number at the beginning of the destination matrix L: Matrix (Ms and Md) length MS, Md can be combined with V, Z, P0~P9 for indirect addressing applications
MBSHF	INB,L_R,Ms,	Ms : The register number at the beginning of the source

	Md,L,OTB	matrix MD: The register number at the beginning of the destination matrix L: Matrix (Ms and Md) length Ms, Md can be combined with V, Z, P0~P9 for indirect addressing applications
MINV	Ms,Md,L	Ms : The register number at the beginning of the source matrix MD: The number of the scratchpad at the beginning of the matrix for which the result is stored L: The length of the matrix (Ms and Md). Ms,Md can be combined with V, Z, P0~P9 for indirect addressing applications
MOR	Ma,Mb,Md,L	MA: The starting register number of Source Matrix A MB: The register number at the beginning of Source Matrix B MD: The number of the scratchpad at the beginning of the matrix for which the results are stored L: The length of the matrix (Ma, Mb, and Md). Ma, Mb, Md can be combined with V, Z, P0~P9 for indirect Addressing applications
MXNR	Ma,Mb,Md,L	MA: The scratchpad number at the beginning of Source Matrix A MB: The register number at the beginning of Source Matrix B MD: The number of the scratchpad at the beginning of the matrix for which the result is stored L : the length of the matrix (Ma, Mb, Md). Ma, Mb, Md can be combined with V, Z, P0~P9 for indirect Addressing applications
MXOR	Ma,Mb,Md,L	MA: The scratchpad number at the beginning of Source Matrix A MB: The register number at the beginning of

		Source Matrix B MD: The number of the scratchpad at the beginning of the matrix for which the result is stored L : the length of the matrix (Ma, Mb, Md). Ma, Mb, Md can be combined with V, Z, P0~P9 for indirect Addressing applications
HSPSO	EN,PAU,ABT, Ps,SR,WR,AC T,ERR,DN	Ps : The first set of Pulse outputs (0~7). 0：Y0 & Y1 1：Y2 & Y3 2：Y4 & Y5 3：Y6 & Y7 4：Y8 & Y9 5：Y10 & Y11 6：Y12 & Y13 7：Y14 & Y15 SR: Locator program start register WR: The instruction operates the start register, which occupies a total of 7 registers, and other programs cannot be reused
HSPWM	EN,Pw,Op,Rs ,Pn, OutR,WR,AC T	PW : High-speed pulse-width modulation output point (0=Y0, 1=Y2, 2=Y4, 3=Y6, 4=Y8, 5=Y10, 6=Y12, 7=Y14) Op : Output polarity; 0=Output non-inverted 1=Output inverted Rs : Resolution; 0=1/100 (1%) 1=1/1000 (0.1%) Pn : Output frequency parameter setting (0~255). OR : PWM output width setting register 0~100 or 0~1000 WR : Instruction operation work register, other programs cannot be reused
HSPWM2	EN,Pw,Op,Hz ,O_R, ACT,ERR	PW: Pulse width modulation output point (0=Y0, 1=Y2, 2=Y4, 3=Y6, 4=Y8, 5=Y10, 6=Y12, 7=Y14) Op: Output polarity (0=positive phase, 1=inverted phase) Hz: Output frequency (1~100000000 or 1~200000000, unit 0.001Hz) OR: Pulse output width (0~100, unit %)
ICA	EN,D_R,Ps,Is, Fo,Ag,ACT,E RR,DN	Ps: Which group of Pulse output (0~7) 0: Y0 & Y1 1: Y2 & Y3 2: Y4 & Y5 3: Y6 & Y7 4: Y8 & Y9 5: Y10 & Y11 6: Y12 & Y13 7: Y14 & Y15 Ls: External input X point index number (0~15) Fo: Target working speed (1~100000 or 1~200000) Ag: Fixed angle during interruption (0~36000)
ICF	EN,D_R,Ps,Is, Fo,Fd,ACT,ER R,DN	Ps: Which group of Pulse output (0~7) 0: Y0 & Y1 1: Y2 & Y3 2: Y4 & Y5 3: Y6 & Y7 4: Y8 & Y9 5: Y10 & Y11 6: Y12 & Y13 7: Y14 & Y15 Ls: External input X point index number (0~15) Fo: Target working speed (1~100000 or 1~200000) Fd: pulse movement amount output after interrupting capture
MHSPO	EN,PAU,ABT,	Gp: which group (0~1)

	Gp,SR,WR,ACT,ERR,DN	SR: Positioning program start register WR: Instruction operation starting register, occupying 9 registers in total, cannot be reused by other programs
MPARA	EN,Ps,SR	Ps: Which group of Pulse Output (0~3) SR: Parameter table starting register, a total of 18 parameters, occupying 24 temporary registers
MPG	EN,Sc,Ps,Fo,Mr,WR,ACT	Sc: Specify the source of the rocker to take over the high-speed counter; 0~7 Ps: Specify the pulse wave output axis of the reaction hand wheel; 0~3 Fo: Output frequency setting register (2 registers) Mr: Ratio setting register (2 registers) Mr+0: Multiplicand (Fa) Mr+1: Multiplier dividend (Fb) WR: Starting address of working register, occupying 4 registers in total. Output pulse number = (hand wheel input pulse number × Fa)/Fb * PLC OS V4.60 (inclusive) and later supports this command
PSCNV	EN,Ps,D	Ps: 0~3; Convert the pulse wave position (PS) of the group into mm (Deg, Inch, PS) with the same unit as the set value to display the current position. D: The register that stores the current position after conversion. A total of two registers are needed; for example, D10 represents the two registers D10 (Low Word) and D11 (High Word).
PSOFF	EN,Ps	Ps: 0~3 forces the Pulse Output group to stop outputting.
INTDisable	LB	LBL: The name of the interrupt mark that prohibits external input or peripheral action.
INTEnable	LB	LBL: Allows external input or peripheral tag name to interrupt the action.
TARESUB	EN,RST,ID,CH,SB,ERR	ID: expansion module ID number (0 ~ N) CH: channel index (0 ~ N) SB: 32BIT value of the tare weight to be deducted
TAREZEOFFSET	EN,MD,ID,CH,WR,ERR	
MFSysCAMR	EN,Md,D,ID,L,ACT,ERR,DN	Read E-Cam data Route 1: From PLC Md: 0 D: The initial address of the PLC register after the E-Cam data is read ID: E-Cam number L: E-Cam resolution Route 2: From FATEK SD card

		Md: 1 D: FATEK SD card E-Cam data's file number ID: E-Cam number L: E-Cam resolution
MFSysCAMW	EN,Md,D,ID,L ,ACT,ERR,DN	Write E-Cam data Route 1: From PLC Md: 0 D: The initial address of the written E-Cam data in the PLC register ID: E-Cam number L: E-Cam resolution Route 2: From FATEK SD card Md: 1 D: FATEK SD card E-Cam data's file number ID: E-Cam number L: E-Cam resolution
MFSysInit	EN,ACT,ERR, DN	Enable the Motion function / open the driver connection
MFSysRCPR	EN,Md,D,Gp, ACT,ERR,DN	Read the formula according to the content set by the sports formula Route 1: From PLC Md: 0 D: meaningless in mode 0 Gp: Read all set recipes = 0, Read first set recipe = 1, and so on Route 2: From FATEK SD card Md: 1 D: FATEK SD card recipe file number Gp: Read all set recipes = 0, Read first set recipe = 1, and so on
MFSysRCPW	EN,MD,D,GP, ACT, ERR,DN	Write the formula according to the content set by the sports formula Route 1: From PLC Md: 0 D: meaningless in mode 0 Gp: Write all set recipes = 0, write first set recipe = 1, and so on

		Route 2: From FATEK SD card Md: 1 D: FATEK SD card recipe file number Gp: Write all set recipes = 0, write first set recipe = 1, and so on
MFSysRstAlm	EN,ACT,ERR, DN	Remade motion control bug
MFSysSetVirt	EN,AX,ACT,E RR,DN	Translate Real axis to virtual axis It is used when the physical shaft is sent for repair or the amount of shafts of different project models is reduced AX: axis number
MFSysStop	EN,ACT,ERR, DN	Stop the Motion function / close the drive connection
MFChgTbPrm	EN,GP,ID,N,D ,ERR,DN	single data mapping TM: form type PN: form number S: project PV: mapped value
MFFlowHalt	EN,ID,ACT,ER R,DN	Immediately suspend the Motion Flow ID: Motion Flow number
MFFlowPause	EN,ID,ACT,ER R,DN	After completing the current action block, pause the Motion Flow ID: Motion Flow number
MFFlowStart	EN,ID,ACT,ER R,DN	Start Motion Flow ID: Motion Flow number
MFFlowStop	EN,ID,ACT,ER R,DN	Abort Motion Flow ID: Motion Flow number
MFMapTbPrm	EN,GP,ID,N,D ,ERR,DN	Map data according to the motion parameter mapping table Gp: Mapping table page number N: Starting item number L: amount of mapped items
MFHome	EN,AX,ACT,E RR, DN	Execute axis homing mode AX: axis number
MFJog	EN,D_R, AX, MD,ACT,ERR ,DN	Execute Jog mode AX: axis number MD: Continuous motion at start speed (0), jog at start speed (1), continuous motion at JOG speed (2), jog at JOG speed (3) D/R: positive direction (ON) / negative direction (OFF)D/R：

		Positive direction (ON) / Negative direction (OFF)
MFPointMov	EN,PT,AX,ACT,ERR,DN	Execution point table positioning mode Pt: point number AX: Axis number
MFAxCirMov	EN,UPD,TAX,MD, DR,BF,D,EC,ACT,ERR,DN,UPD	Total number of axes (TAX): Mode (MD): 0: absolute, 1: relative Direction (DR): 1: Forward, 2: Reverse
MFAxLMov	EN,UPD,TAX,MD, DR,BF,D,EC,ACT, ERR,DN,UPD	Continuous mode (BF): 0 = execute current instruction immediately 1 = wait for the previous instruction to complete 2 = Select lower speed continuous 3 = Select the previous command speed 4 = Select current command speed 5 = Select higher speed continuous Click Data (D): scratchpad starting position Error code (EC): scratchpad storage location
MFAxMov MFAxMov_D	EN,UPD_in,AX,MD,Ps,V_n,A,D,SA,SD,DR,BF,ACT,ERR,DN,UPD_out	Execute single-axis positioning mode S: spindle number MD: Absolute coordinates (0) / Relative coordinates (1) / Infinite distance mode (2) Ps: target coordinates V: Maximum linear speed A: Maximum acceleration D: Maximum deceleration SA: Acceleration section S curve proportion (unit: 0.1%) SD: S curve ratio of deceleration section (unit: 0.1%) DR: Positive direction (1) / Negative direction (2) BF: continuous point mode
MFGearMPG MFGearMPG_D	EN,UPD_in,M,S,N,D,T,ACT,ERR,DN,UPD_out	M: (Spindle) 1~16: axis number 100: Gray code (X8~15) 101~108: HSC0~7 S: (slave axis) 1~16: axis number

		N: (conversion numerator) Integer, including decimal point from axis D: (conversion denominator) Positive integer including main axis decimal point T: (smoothing time) positive integer Unit: ms
MFPathMov	EN,ACT,UPD, AX1,AX2,AX 3,PT,V,TA,TD, SA,SD,EC,AC T,ERR,DN,UP D	AX1: Axis 1 AX2: Axis 2 AX3: Axis 3 PT: path number V: Vel positive integer Includes spindle decimal point TA, TD: acceleration time Positive integer (ms) SA, SD: (S proportion %) 0.0% ~ 100.0% Unit: 0.1% EC: register storage location
MFTorqCtl MFTorqCtl_D	EN,UPD_in,A X,T,MX,ACT,E RR,DN, UPD_out	AX: 1~16: axis number T: Unit: 0.1% MX: Maximum speed
MFVelCtl MFVelCtl_D	EN,UPD_in,A X,T,MX,ACT,E RR,DN, UPD_out	AX: 1~16: axis number V: Unit: See the drive MX: Maximum torque