

FATEK

M Series

Programmable Controller

M-Series PLC Structured Language ST User Manual



NEXT Level SOLUTION

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	1
1-1 Features of ST Language	12
1-2 Adding ST Language Program.....	14
User Interface of Uperlogic ST	1
2-5 滑鼠懸浮提示	錯誤! 尚未定義書籤。
2-6 加入觀測變數.....	10
2-7 快捷滑鼠鍵盤操作	11
Basic Program Structure of Uperlogic ST	1
3-3-1 Operation Expression	7
3-3-2 Arithmetic (+、-、*、/)	7
3-3-5 Advanced Operation (Exponents)	8
ST	8
LD	9
3-4 Operand and Operator	10
3-4-1 ()	12
3-4-2 Arithmetic +, -, *, /	12
3-4-3 Quick Addition and Subtraction ++, --	12
3-4-4 Value compare >=, <=, >, <	12
3-4-5 Equal to/Not equal to =, <>	13
3-4-6 Bit shift left and right <<, >>	13
3-4-7 Negation of Operand NOT	13
3-4-8 Negative Sign of Operand -	13
3-4-9 Assign Value :=	13
3-4-10 Logic Operand AND (&)、OR ()、XOR、XNOR	14
3-4-11 Remainder MOD	14

3-5-1	Single-Line Comment //	15
3-5-2	Multiline Comment (* *)	15
3-6	Flow Control and Loop	16
3-6-1	IF ELSE	16
3-6-2	FOR	17
3-6-3	Case Of	18
3-6-4	WHILE LOOP & REPEAT	19
3-6-5	BREAK / EXIT	20
3-6-6	LBL, JMP, CALL	21
3-7	Variables and Data Type	23
	標籤建立範例	23
3-7-1	Bool / Bit	26
3-7-2	Integer Type	26
3-7-3	Floating Point	26
3-7-4	Constant	30
3-8	Using PLC Register and Memory	32
3-9	Calling System Built-In Functions	33
3-9-1	Timer	34
3-9-2	Counter / Counter_L	34
3-9-3	R_TRIG / F_TRIG	36
3-9-4	TARESUB and TAREZEOFFSET	36
3-9-5	Functions with multiple calling modes	37
3-10	複週期指令集	38
3-11	具有特殊意義的變數名稱	51
3-12	Calling FCM Function	55
3-13	函數注意事項	57

Manual for the FATEK M-Series Structured Language ST

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.	
<p>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.</p> <p>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.</p> <p>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.</p>	
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 matters 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

<u>1-1</u>	<u>Features of ST Language</u>	錯誤! 尚未定義書籤。
<u>1-2</u>	<u>Adding ST Language Program</u>	錯誤! 尚未定義書籤。

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.

$$R3=(R0+R1+R2)\div 3$$

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

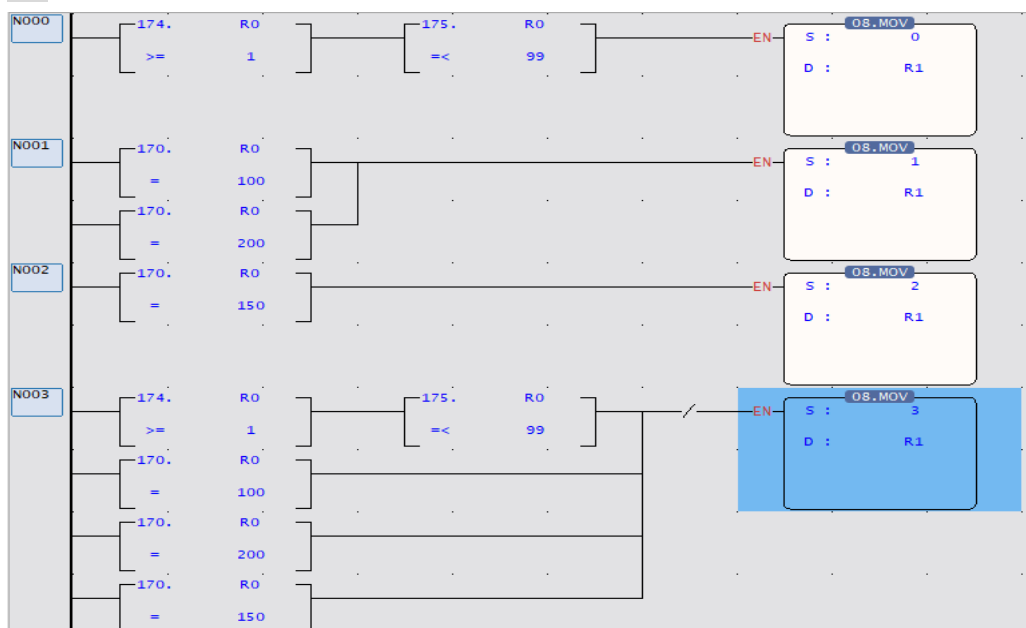
R1:=2;

ELSE

R1:=3;

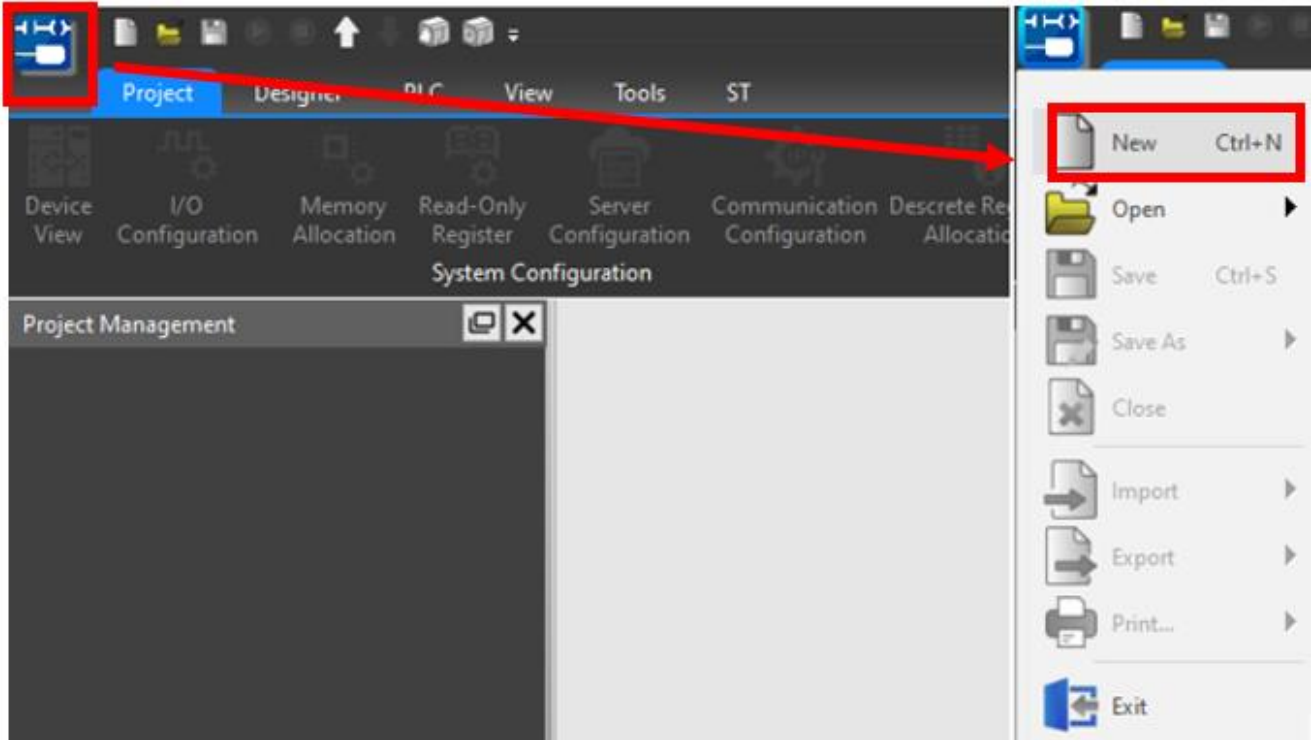
END_IF

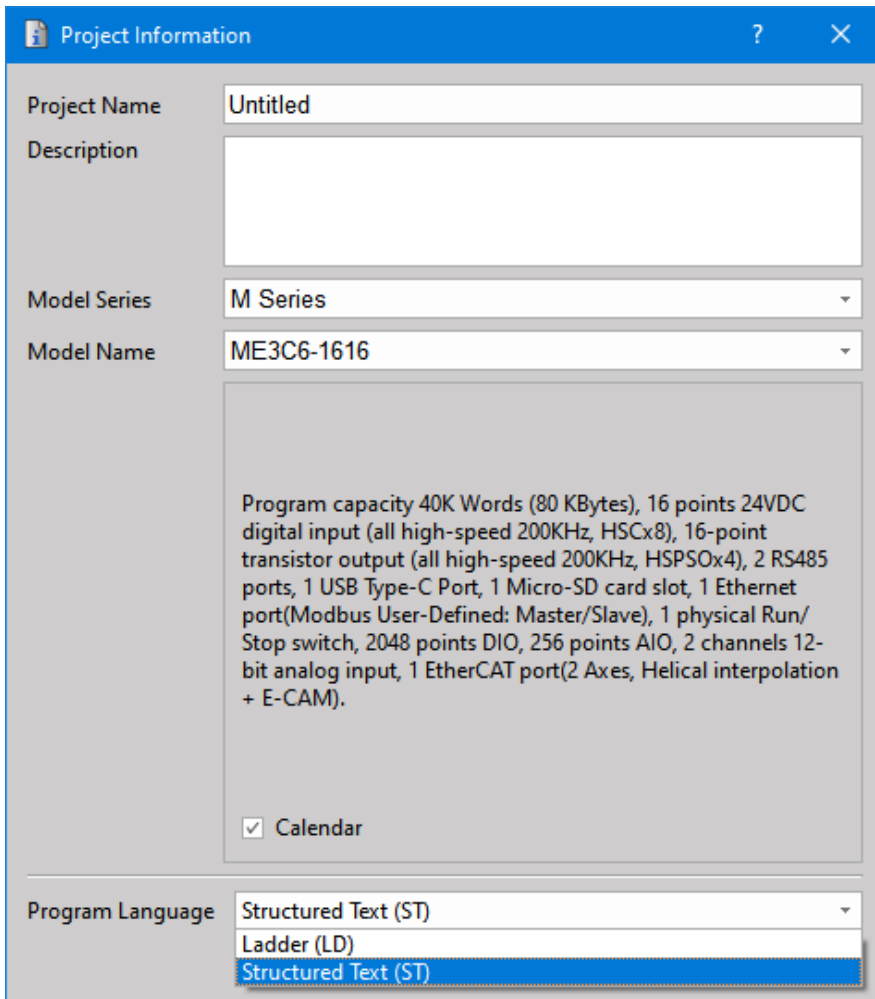
LD



1-2 Adding ST Language Program

Establishing a new Program

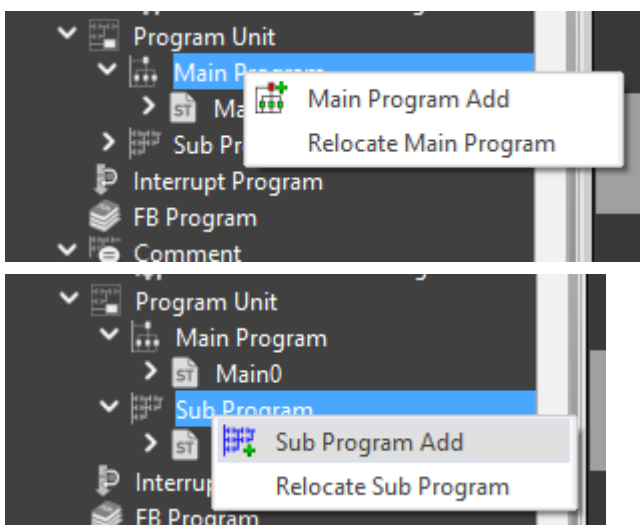




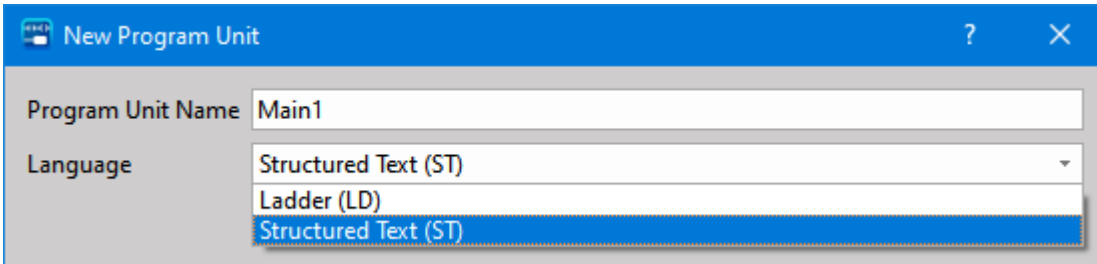
Example of creating MainUnit/SubUnit(ST):

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

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



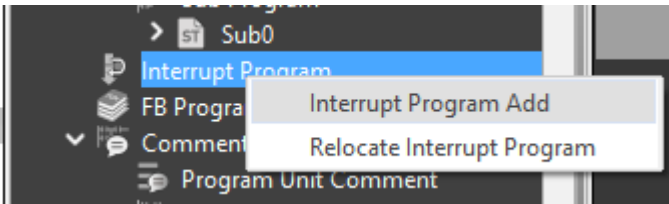
Select [Structured Text(ST)] and add the corresponding program.



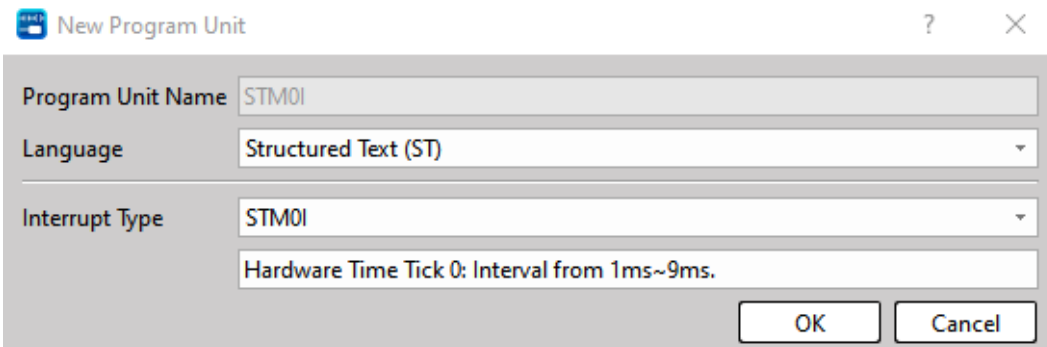
Example of creating an interrupt program (ST):

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

Interrupt Program Add



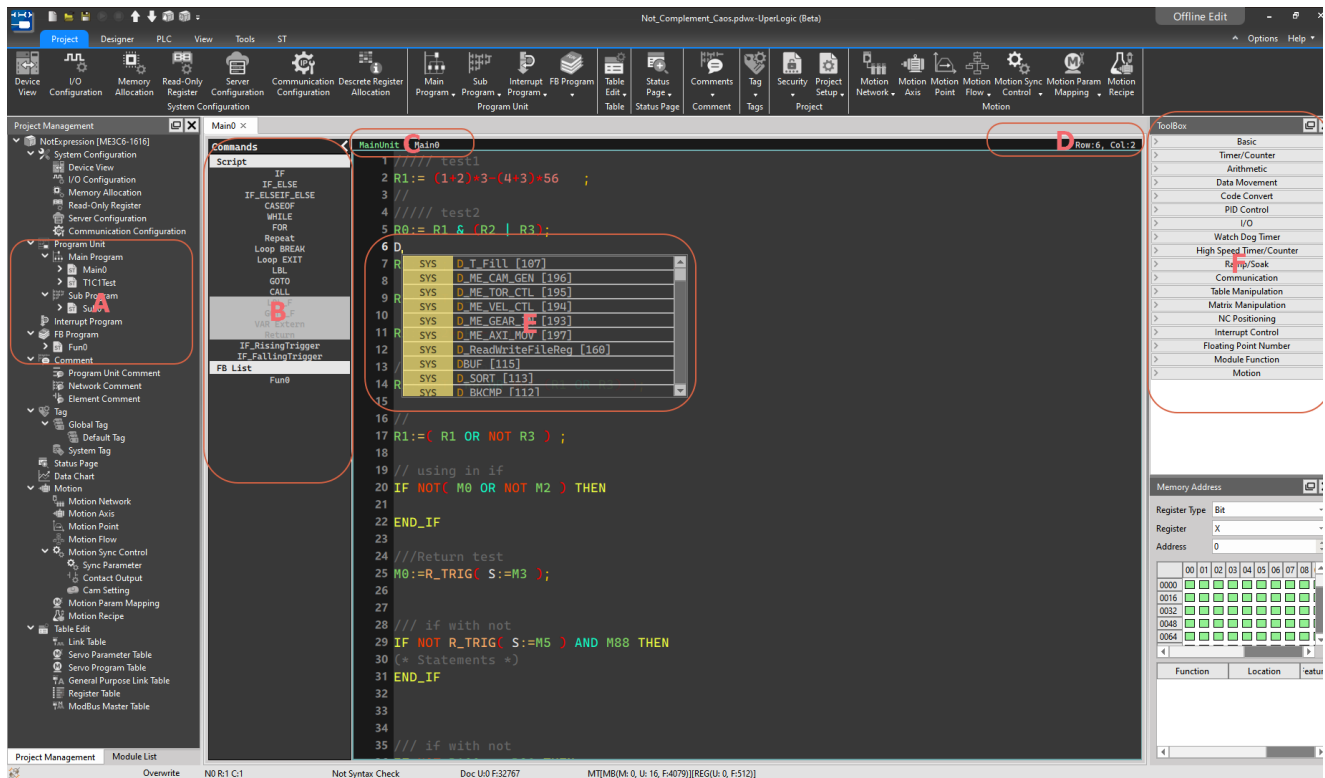
Select the interrupt signal and the program type (ST) to be processed.



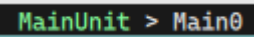
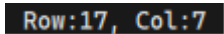


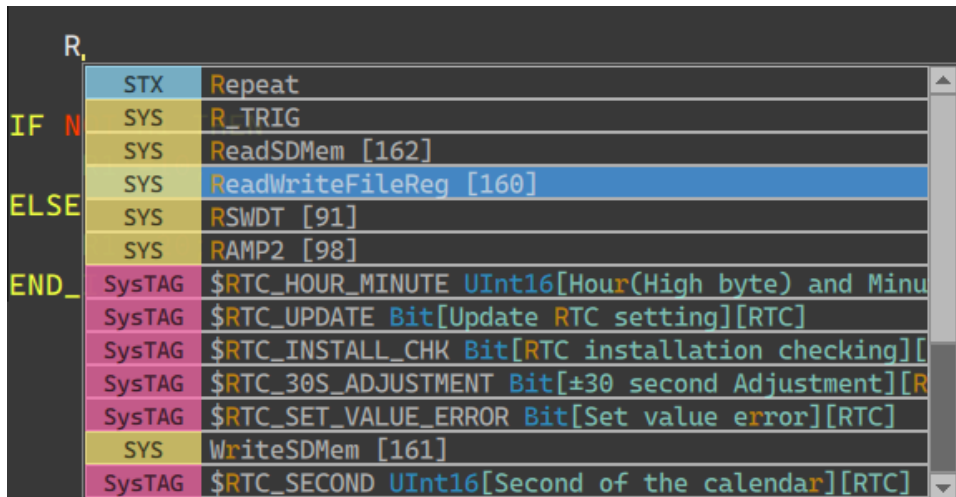
2

User Interface of Uperlogic ST

2-1 Interface Overview

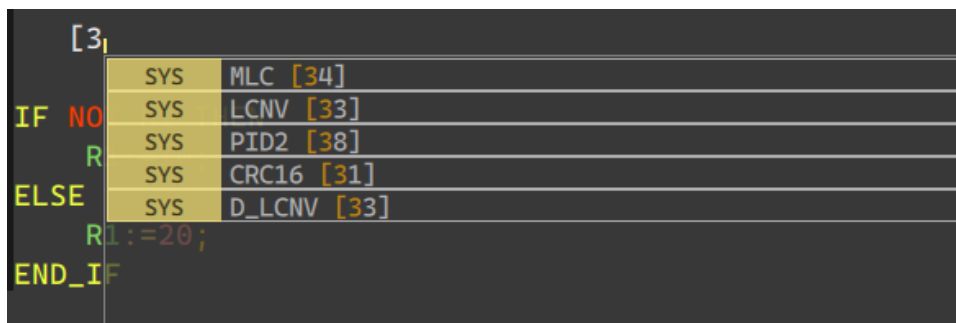


- A. For all ST programs in the current project, double-click the name with the left mouse button, and a new editing screen will be selected in the middle.
- B.
 1. Display the currently available ST syntax and FCM-related information. Double-click the field and the corresponding template will be inserted 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. 
- D. The current row and column information of the cursor. 
- E. Smart pop-up reminder window, which is convenient for users to prompt when typing.



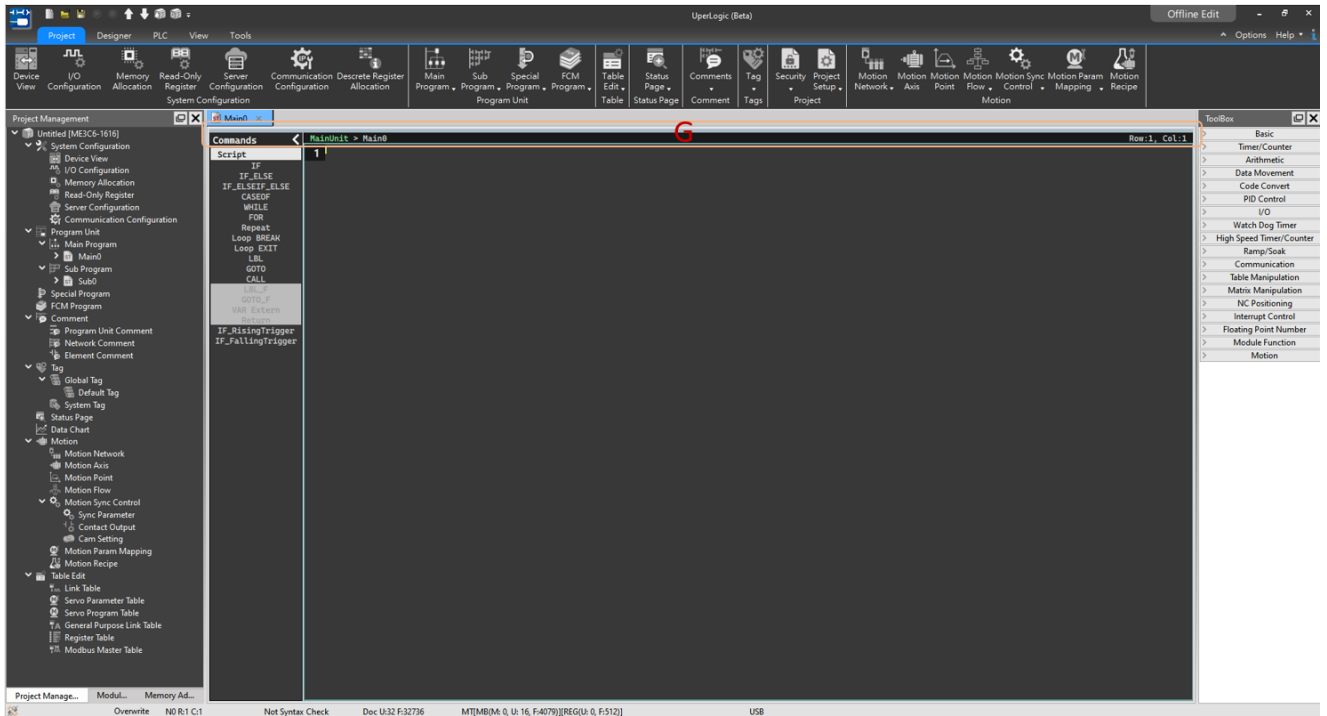
This pop-up window will follow the user's input and display the automatically detected prompt program fragments in real time, including available labels, variables, calling functions, etc. Matched strings will be displayed in orange in the window (as shown above).

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).

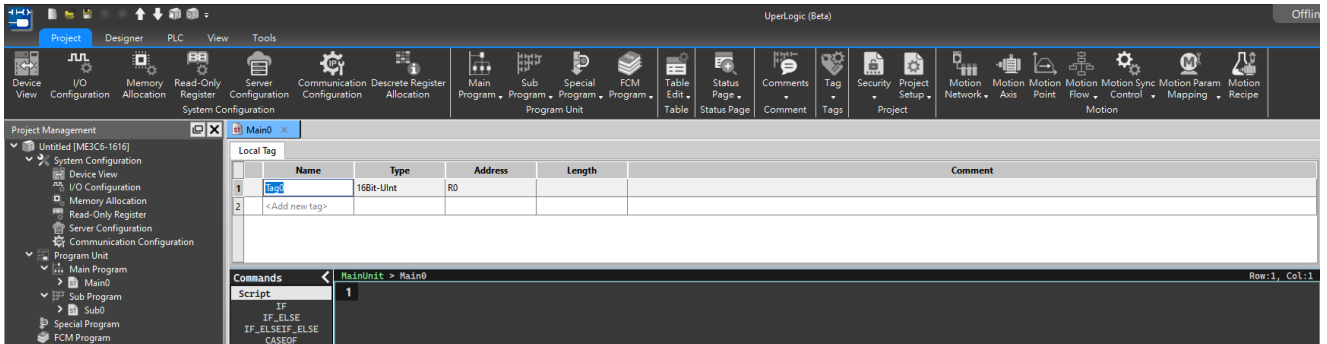


In this way, you can use the mouse to double-click, or use the up and down keys to select the program fragment you want to insert.

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

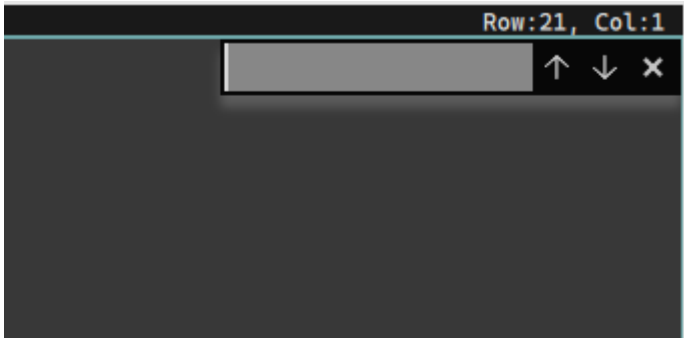


- G. Drag this area to display the area label. It will be more convenient to modify the program based on the label when writing ST programs.



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

2-2 Supportive Keyboard Instructions

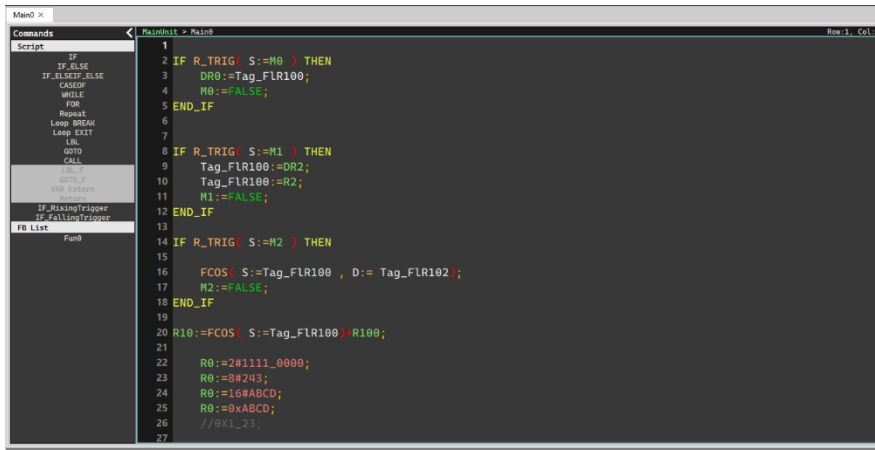
Key	Function
Ctrl + C	Copy
Ctrl + V	Paste
Ctrl + A	Select All
Ctrl + X	Cut
Tab	Insert Tab
Multiple Select+Tab	Select multiple lines and add Tab at the same time.
Shift + Tab	Back Tab simultaneously according to the cursor or the number of selected rows.
Ctrl + F	<p>A search window appears at the top right of the screen, and the current text can be searched.</p> 
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 System Mode

There are currently three modes of the system software:

ST Editor displays three status correspondingly.

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



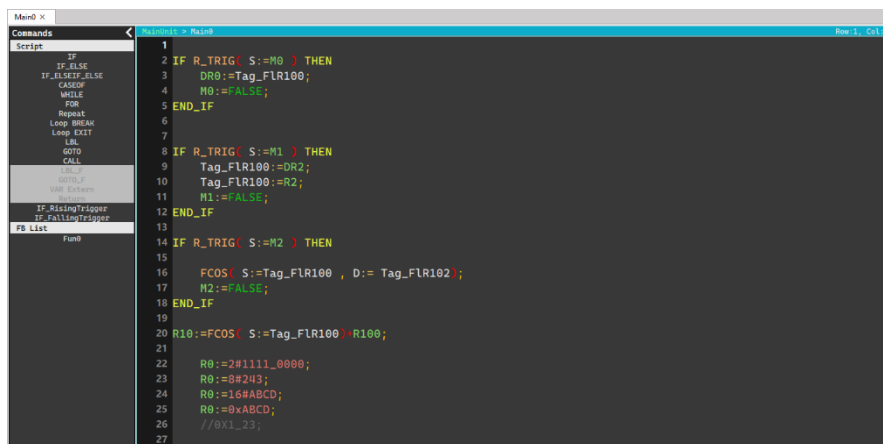
```

1
2 IF R_TRIG( S:=M0 ) THEN
3   DR0:=Tag_FLR100;
4   M0:=FALSE;
5 END_IF
6
7
8 IF R_TRIG( S:=M1 ) THEN
9   Tag_FLR100:=DR2;
10  Tag_FLR100:=R2;
11  M1:=FALSE;
12 END_IF
13
14 IF R_TRIG( S:=M2 ) THEN
15
16   FCOS( S:=Tag_FLR100 , D:= Tag_FLR102);
17   M2:=FALSE;
18 END_IF
19
20 R10:=FCOS( S:=Tag_FLR100)*R100;
21
22 R0:=2#1111_0000;
23 R0:=8#243;
24 R0:=16#ABCD;
25 R0:=0xABCD;
26 //0X1_23;
27

```

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

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

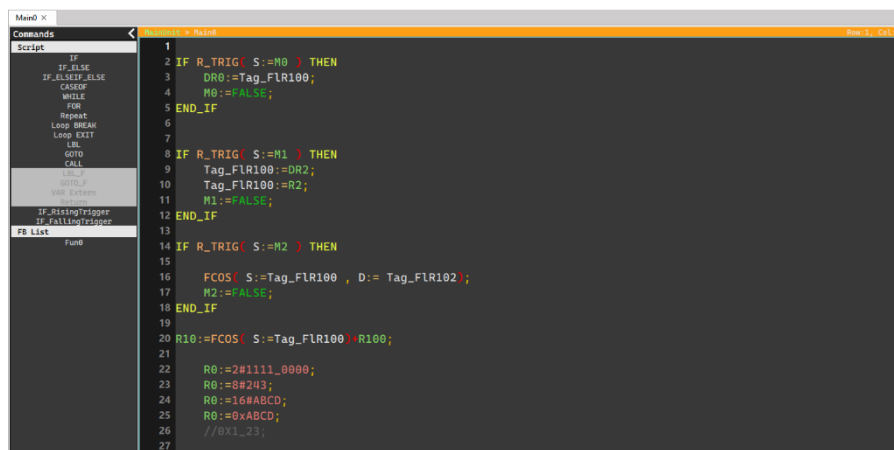


```

1
2 IF R_TRIG( S:=M0 ) THEN
3   DR0:=Tag_FLR100;
4   M0:=FALSE;
5 END_IF
6
7
8 IF R_TRIG( S:=M1 ) THEN
9   Tag_FLR100:=DR2;
10  Tag_FLR100:=R2;
11  M1:=FALSE;
12 END_IF
13
14 IF R_TRIG( S:=M2 ) THEN
15
16   FCOS( S:=Tag_FLR100 , D:= Tag_FLR102);
17   M2:=FALSE;
18 END_IF
19
20 R10:=FCOS( S:=Tag_FLR100)*R100;
21
22 R0:=2#1111_0000;
23 R0:=8#243;
24 R0:=16#ABCD;
25 R0:=0xABCD;
26 //0X1_23;
27

```

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



```

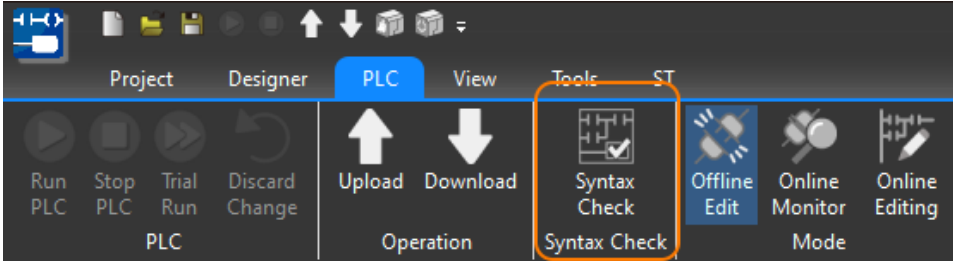
1
2 IF R_TRIG( S:=M0 ) THEN
3   DR0:=Tag_FLR100;
4   M0:=FALSE;
5 END_IF
6
7
8 IF R_TRIG( S:=M1 ) THEN
9   Tag_FLR100:=DR2;
10  Tag_FLR100:=R2;
11  M1:=FALSE;
12 END_IF
13
14 IF R_TRIG( S:=M2 ) THEN
15
16   FCOS( S:=Tag_FLR100 , D:= Tag_FLR102);
17   M2:=FALSE;
18 END_IF
19
20 R10:=FCOS( S:=Tag_FLR100)*R100;
21
22 R0:=2#1111_0000;
23 R0:=8#243;
24 R0:=16#ABCD;
25 R0:=0xABCD;
26 //0X1_23;
27

```

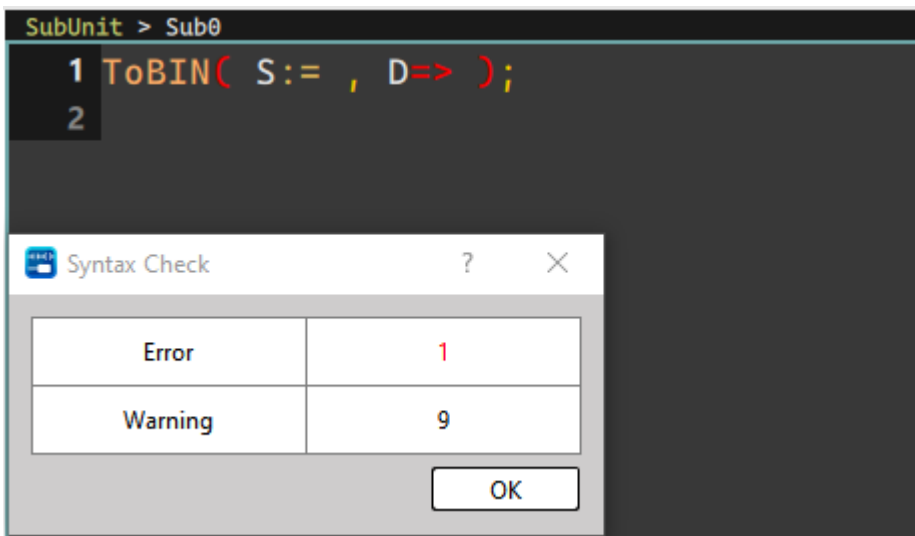
2-4 Syntax Check

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

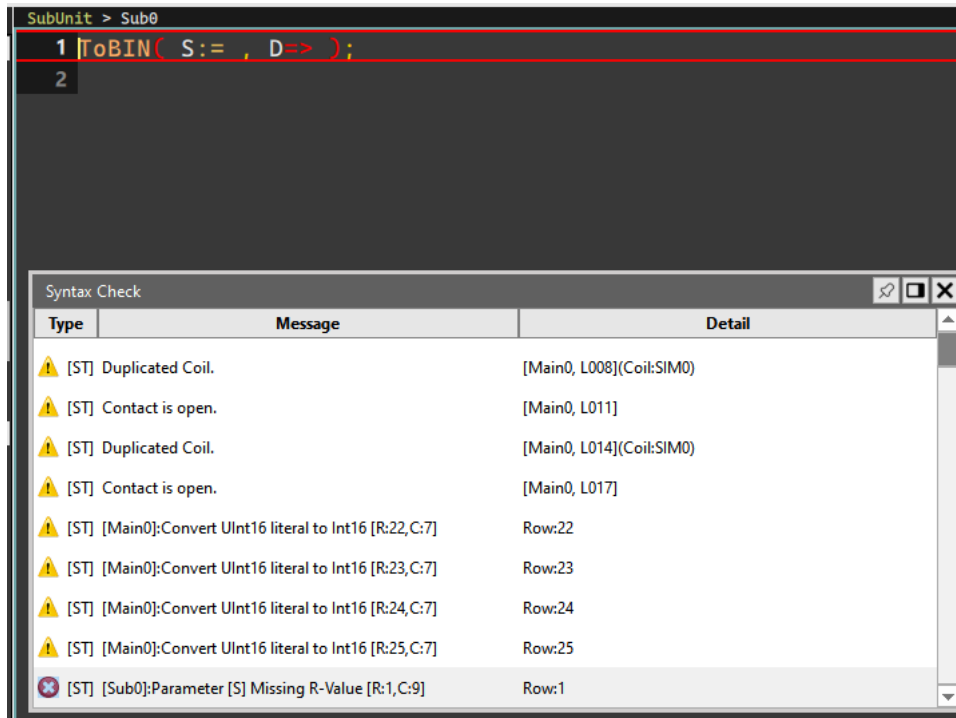
Usually when writing, users can click the button shown below to manually compile to see if it is correct:



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

為了除錯以及撰寫程式方便，支援滑鼠在變數或是函式文字上面懸浮停留會出現提示視窗的介面，給予使用者撰寫程式或除錯的提示

A. 一般函式

會顯示該函示的完整呼叫參數 如：

```

23 H[HSCTW] : HSCTW( S:= , CN:= , D:= )
24 HSCTW( S:=R0 , CN:=HSC_HSC0 , D:=HSC_PV );

```

B. 變數

會顯示該變數的型態跟數值 如：

```

2 IF M1 THEN
3   $STM1_PV:=500;
4   $STM1_PV[UInt16][R35437] : 0
5 END_IF

```

後面顯示的數值，可以直接用滑鼠點擊後面的數值進行修改

Bool 型別點擊 On/Off 切換

數字型別：則是直接輸入對應的數字修改

```

2 IF M1 THEN
3   $STM1_PV:=500;
4   $STM1_CTRL:=TRUE;
5 END_IF $STM1_CTRL[Bit][M9167] : 1
6

```

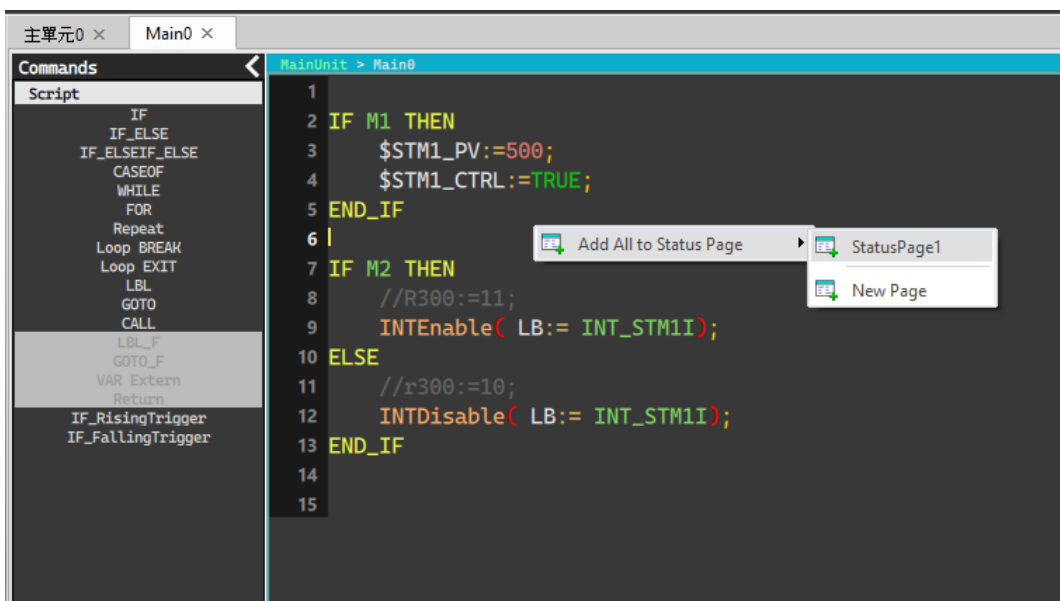
!注意! 要得到上述 B.變數功能的提示，該程式文本需要編譯成功過後才能取得正確提示。

2-6 Add observed variables

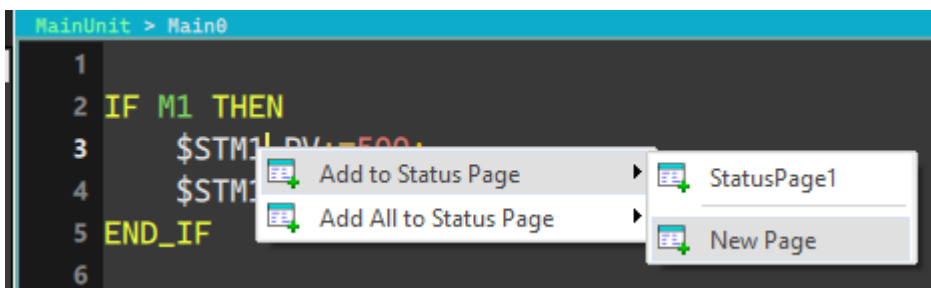
在線上編輯跟線上監測模式中，在文字編輯器上面按滑鼠右鍵會跳出選單可以將滑鼠游標對準的變數，或是目前文本所有變數自動加到狀態觀測的頁面中。

如下圖：

在空白區域點擊滑鼠右鍵會跳出自動加入所有變數到監視頁中的選項，並且可以在子選單選擇加入到哪一頁，或是創建新的一頁



在變數上面按滑鼠右鍵則會多出新增當前變數到監視頁面中



2-7 Quick mouse and keyboard operations

ST 文字編輯提供許多提升程式編輯便捷性之常見功能操作

A. 選取整行

滑鼠點選行數顯示的地方會選取該行，並且可以往上或往下連續選取整行

```

8 //r300:=11;
9 INTEnable( LB:= INT_STM1I);
10 ELSE
11 //r300:=10;
12 INTDisable( LB:= INT_STM1I);
13 END_IF
14
15
16 ReadWriteFileReg( EN:= R_TRIG(M0)(* *),
17                  R_W:= 1(* *),
18                  INC:= M10(* *),

```

B. 批次加 Tab / 移除 Tab

單行起始加 Tab:

游標移到該行的起始按 Tab 按鍵

多行批次加 Tab:

選取多行程式碼後再按 Tab，則所有行都會統一加一個 Tab 在開頭

單/多 行移除 Tab:

游標選取該行任意位置，或是選取多行，再按 Shift + Tab 則會統一往左縮排(Back-Tab)

C. 註解反註解

多行註解除了可以使用 "(*", "*)" 這兩種符號來作之外，支援 Ctrl+/ 來針對游標所在的行，或是目前所選取的行來批次新增 // 註解在每一行文字開頭。

新增邏輯:

只要選取行數範圍內有一行開頭沒有// 符號，則所有行數統一新增//註解

取消邏輯:

所選取的行數每一行開頭都要有//符號，就會運行批次移除註解

3

Basic Program Structure of Uperlogic ST

<u>3-1</u>	<u>Introduction</u>	錯誤! 尚未定義書籤。
<u>3-2</u>	<u>Statement</u>	錯誤! 尚未定義書籤。
<u>3-3</u>	<u>Expression</u>	錯誤! 尚未定義書籤。
<u>3-4</u>	<u>Operand and Operator</u>	錯誤! 尚未定義書籤。
<u>3-5</u>	<u>Comment</u>	錯誤! 尚未定義書籤。
<u>3-6</u>	<u>Flow Control and Loop</u>	錯誤! 尚未定義書籤。
<u>3-7</u>	<u>Variables and Data Type</u>	錯誤! 尚未定義書籤。
<u>3-8</u>	<u>Using PLC Register and Memory</u>	錯誤! 尚未定義書籤。
<u>3-9</u>	<u>Calling Sytem Built-In Functions</u>	錯誤! 尚未定義書籤。
<u>3-10</u>	<u>Calling FCM Function</u>	錯誤! 尚未定義書籤。

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	Reference
Computing Symbols		+、-、*、/、AND、&、OR、 、XOR、MOD、<=、>=、<>、++、---、<<、>>、NOT、:=、(、)	3-4 運算元 (Operand) 與運算子 (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	3-6 流程控制 與迴圈
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)	3-x 元件
	Function Call (FB)	1. System built-in Function 2. (User-created FB Function library)	3-10 呼叫 FCM 函數
Constant		Integer: Defaulted as "signed int" Ex: R0:=1; R0:=-2;	3-7-4 常數

	<p>“Unsigned int”</p> <p>Ex: Tag0:=0xFF;</p> <p>String: Use only on Label</p> <p>Goto, LBL...etc</p> <p>Bit (Bool) Type: TRUE 、 FALSE</p> <p>Float: 32-bit Float Constant</p> <p>ex: TagFloat1:=1.1;</p>	
Delimiter	<p>“:” appears in CASE OF</p> <p>分隔 “;” 多個函式參數傳遞使用</p>	3-6-3 Case Of
Left/Right Round Brackets	<p>“(, ”)”</p> <ol style="list-style-type: none"> 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	<p>Used in Sub-program or FB.</p> <p>To immediately leave the program section.</p>	-
“;” the end of the statement	<p>To mark the end of a sentence running at one end.</p>	-

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

Type	Example	Reference
Space	Space (Halfwidth/Fullwidth), TAB	-
Newline	Newline Code	-
Comment	<i>// 、 (* *)</i>	

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.

```

1 IF M0 THEN
2   Statement
3
4 END_IF
  
```

R10 := R0 * (R1 + R2);

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 $R0 \cdot (R1 + R2)$ 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.

```

MainUnit > Main0
1 IF M0 THEN
2
3 R0 * (R1 + R2);
4
5 END_IF
  
```

Shown below are some statements of Uperlogic ST Language:

```

MainUnit > Main0
1 R0 = D0 - 100;
2
3 IF R0 < 0 THEN
4   M0 := TRUE;
5   ELSE
6     M0 := FALSE;
7 END_IF
8
9 WHILE D0 <> 100 DO
10   D0 := D0 + 1;
11
12 END_WHILE
13
14 Read_Bit(PA0:=R10,PA1:=R20);
15
16 IF R_TRIG ( S := X8 ) THEN
17   RegSwap ( PA0 := R10 , PA1 := R20 );
18   IncGenerator ( PA0 := 30 );
19 END_IF
20
  
```

- ← Distributive Statement
- ← Conditional Statement
- ← Do-While Loop Statement
- ← Function Block(FB) Call Statement

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-While 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	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-while statements))

```

MainUnit > Main0
1  WHILE R0 DO
2      R0 := R0 + 1 ;
3      IF  R0 > 0 THEN
4          M0 := TRUE;
5          ELSE
6              M0 := FALSE;
7          END_IF
8  END_WHILE

```

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 expression 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:

```
MainUnit > Main0
1 R0 = D0 - 100;
2
3 IF R0 < 0 THEN
4   M0 := TRUE;
5 ELSE
6   M0 := FALSE;
7 END_IF
8
9 WHILE D0 <> 100 DO
10
11   D0 := D0 + 1;
12
13 END_WHILE
14
15 Read_Bit(PA0:=R10, PA1:=R20);
16
17 IF R_TRIG ( S := X8 ) THEN
18   RegSwap ( PA0 := R10 , PA1 := R20 );
19   IncGenerator ( PA0 := 30 );
20 END_IF
```

Right side of Distributive Statement

Condition of Conditional Statement

Condition of Do-While Loop Statement

Setting Parameter of FB

Parameter Condition of Conditional Statement

The below table shows Expression types:

Type		Data Type of Expression (Computing Result)	Example
Operation Expression	Arithmetic Expression	Integer, 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 Operation 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:

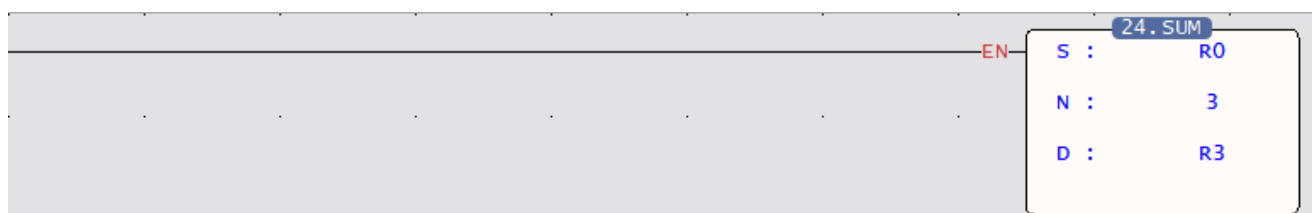
Substitute the adding result of R0-R2 in R3

R3=R0+R1+R2

ST

R3:=R0+R1+R2;

LD



- When adding multiple operation 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)

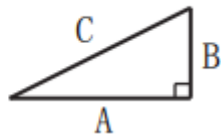
Exponential or trigonometric operations using general-purpose functions.

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:= (* *), MD:= (* *);
	TAN 、 ATAN	$B = \tan A$	FSIN(S:= , D:=);

Program Example

Find the length of the hypotenuse of a right triangle.

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



ST

(使用浮點數需先註冊 Tag 改變資料型別)

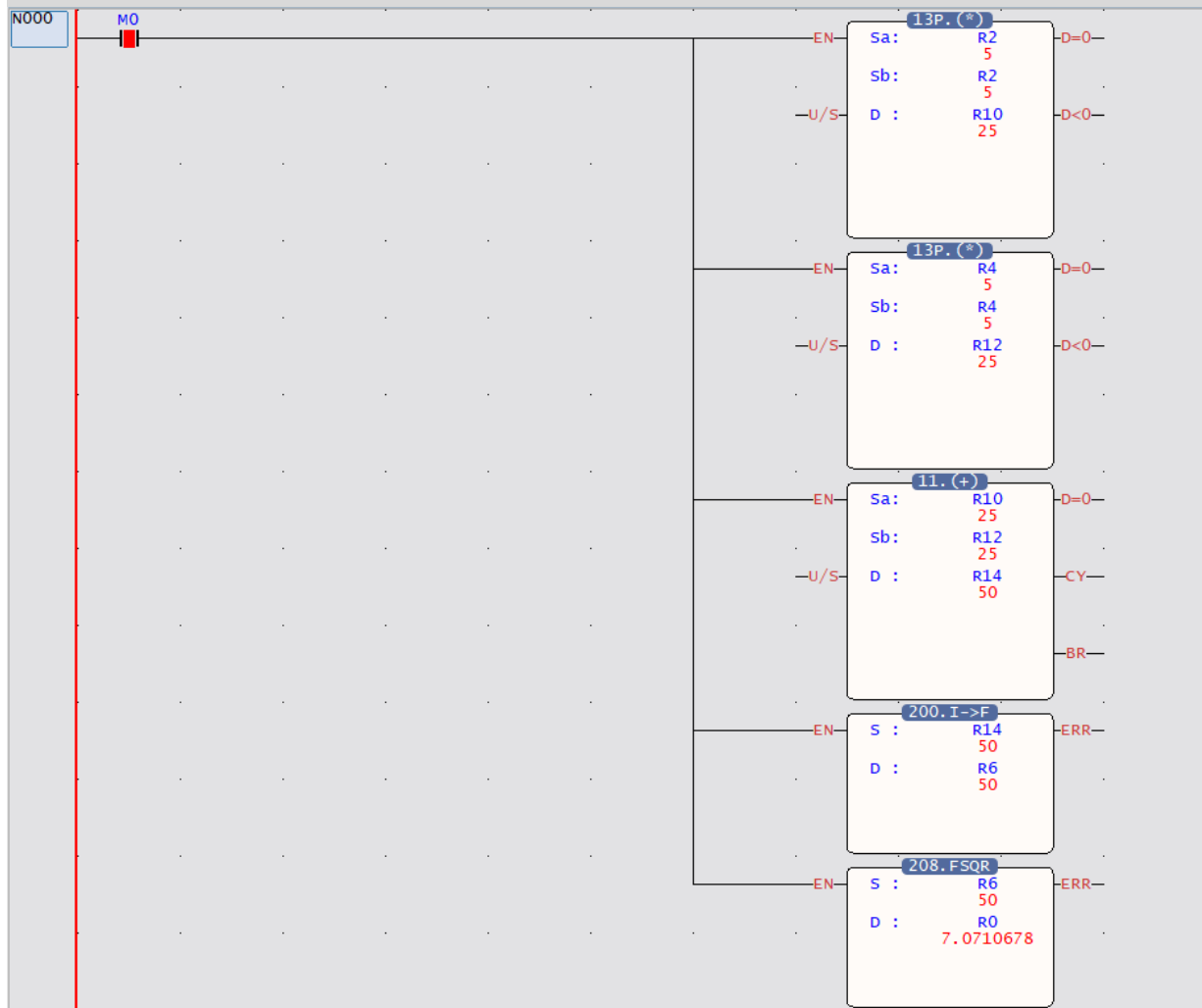
ex: Tag0= Float DR0 、 Tag1= Float DR6)

```
R6:=R2*R2+R4*R4;
```

```
ltoF( S:= R6, D=> Tag1 );
```

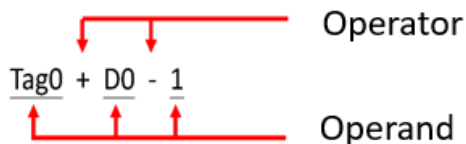
```
FSQR( S:= Tag1 , D:= Tag0 );
```

LD



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 + D1", 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 operands 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.

針對兩邊的運算元進行加減乘除的計算

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(00000001)
```

R0 value will be shifted right with 4 bits, and assign to R10 //R10=16(00010000)

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 complement" to the value in R2 and save it to R0
```

```
R2=1(00000001) => NOT R2= -2(11111110)
```

```
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 (0000000000000001)=> R0=-1(1111111111111111)
```

```
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; // 對 R1, R2 的數值進行位元 及 運算存到 R0
           // 假如 R1=1(00000001) R2=15(00001111) 則 R0 =1(00000001)
DR0:= R10 & DR12; // 對 R10, DR12 的數值進行位元 及 運算存到 DR0
           // 假如 R10=1118481(0x11_1111H) DR12=69905(0x1_1111H)
           // 則 DR0 =4369(0x1111H)
R0:= R1 | R2; // 對 R1, R2 的數值進行位元 或 運算存到 R0
           // 假如 R1=1(00000001) R2=15(00001111) 則 R0 =15(00001111)
DR0:= DR10 OR R12; // 對 R10, DR12 的數值進行位元 或 運算存到 DR0
           // 假如 DR10=4369(0x1111H) R12=69905(0x1_1111H)
           // 則 DR0 =4369(0x1111H)
R0:= R1 XOR R2; // 對 R1, R2 的數值進行位元 反或 運算存到 R0
           // 假如 R1=1(00000001) R2=15(00001111) 則 R0 =14(00001110)
R0:= R1 XNOR R2; // 對 R1, R2 的數值進行位元 反互斥或 運算存到 R0
           // 假如 R1=1(00000001) R2=15(00001111) 則 R0 =-15(11110001)
```

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 10/3 商數等於 3 餘數等於 1,R0=1
```

3-5 Comment

Comments are those used by program developers for easy maintenance 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 //

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

```
MainUnit > Main0
1  ///// test1
2  R1:= (1+2)*3-(4+3)*56 ;
3  //
4  ///// test2
5  R0:= R1 & (R2 | R3);
```

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.

```
29 IF NOT R_TRIG( S:=M5 ) AND M88 THEN
30 (* Statements *)
31 END_IF
32
33 (*
34 /// if with not
35 IF not R100 <> R39 THEN
36 END_IF
37 *)
```

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 conditions.

EX:

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

3-6-2 FOR

```
FOR (*Double Words (Register)*) := (*Initial value (int)*) TO (*target value(int)*)  
BY (*Increment value(int)*) DO  
(* Statements *)  
END_FOR
```

Repeat the instructions from **FOR** to **END_FOR** until the technology register reaches the target value. Each time the run is repeated, an incremental value is added to the target value

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

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

EX:

```
FOR DR0:=0 TO 30 BY 2 DO //DR0 from 0,2,4,...etc, until it reaches to 30  
    R10++; // R0=32,R10=16  
END_FOR
```

```
FOR DR0:=0 TO 30 DO //DR0 from 0,1,2,3....etc, until it reaches to 30  
    R10++; // R0=31,R10=31  
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 (only one description can be supported).

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:

```
SubUnit > Sub0
1 //Sub Program
2 //Sub do not need +RTS
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	SubUnit > Sub0	SubUnit > Sub1
1 CALL ("123")	1 LBL ("123")	1 LBL ("456")
2 CALL_67 ("456")	2 R2:=123;	2 R4:=456;
3	3	3

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:

```
FB > Fcm0
1 GOTO_F ("123")
2 FJMP_166 ("789")
3 LBL_F ("456")
4 PA0:=456;
5 R2:=PA0;
6 LBL_F ("123")
7 PA1:=123;
8 R0:=PA1*2;
9 FLBL_165("789")
10 R4:=789;
11
```

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.

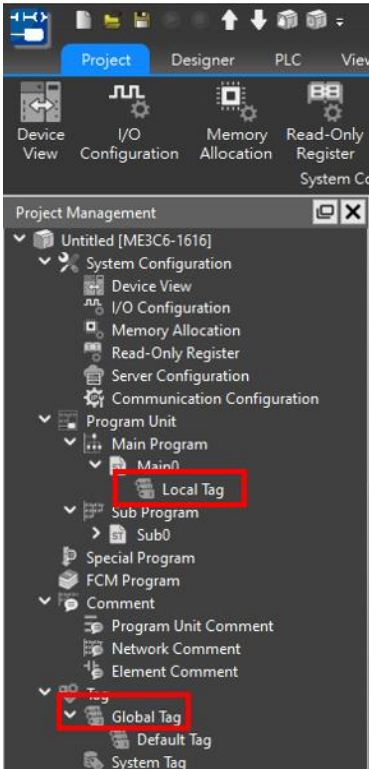
Precautions

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).

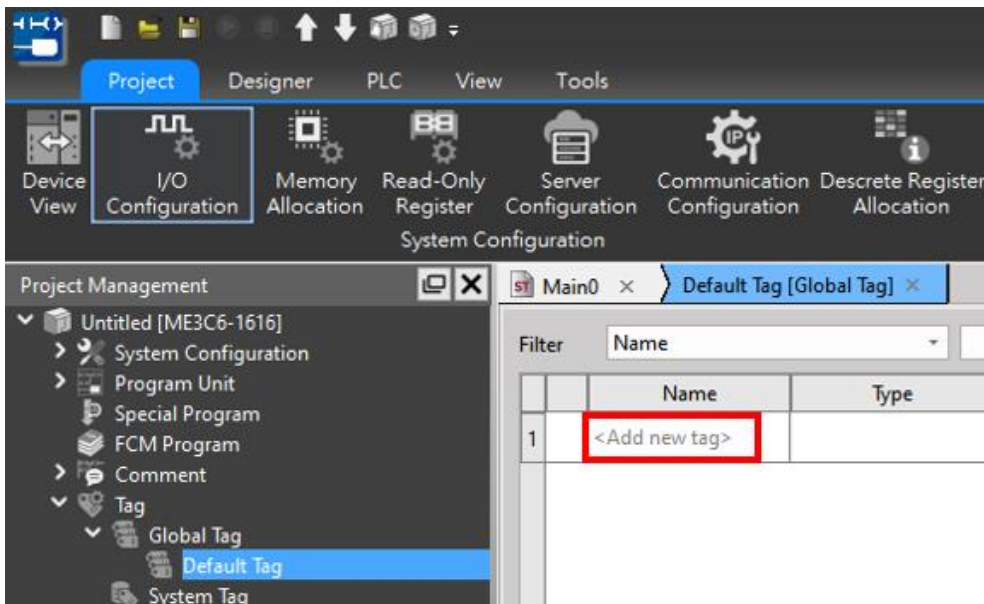


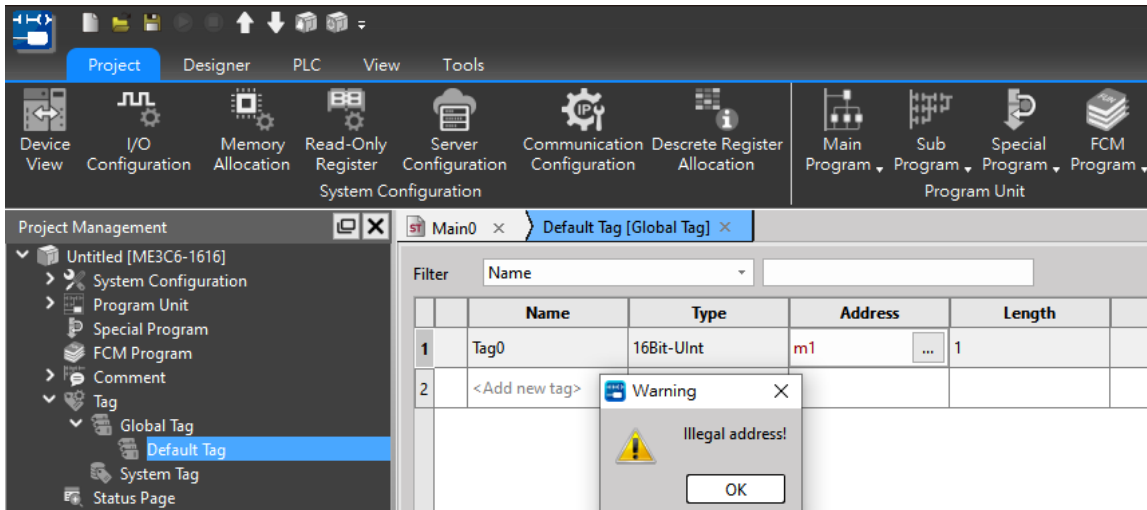
In order to cooperate with program operations of different data types, the following will demonstrate how to create different types of labels.

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:

The screenshot shows the following components:

- 區域標籤 (Area Labels):** A table with columns: 名稱 (Name), 類型 (Type), 位址 (Address), 長度 (Length). It contains one entry: 標籤0 (Label0) of type Float at address DR0 with length 1.
- 命令 (Commands):** A code editor showing five lines:
 - 1 標籤0 := 69905.14159;
 - 2
 - 3 DR10 := 標籤0;
 - 4
 - 5 R12 := 標籤0;
- 監視頁 (Monitoring Page):** A table with columns: 名稱 (Name), 狀態 (Status), 資料 (Data), 註解 (Comment). It shows two entries:

名稱	狀態	資料	註解	名稱	狀態	資料	註解
主單元0/標籤0	浮點數	69905.141	[R0]	DR10	十進制	69905	[R10]
				R12	十進制	4369	[R12]

Line 1: 表示將標籤 0 的浮點數數值賦值為 69905.14159

Line 3: 表示把標籤 0 的浮點數數值轉換成 int32 的整數數值
並且存在 DR10 的位置,小數點的部分則會被移除
(ex. 標籤 0 為 69905.141 則 DR10 內容則為 69905)

Line 5: 表示會把標籤 0 的浮點數數值轉換成 int16 的整數數值
並且存在 R12 的位置,超過 int16 的整數數值以及小數點的部分則會被
移除
(ex. 標籤 0 為 69905.141 則 R12 內容則為 4369)

主單元0

區域標籤

	名稱	類型	位址	長度
1	標籤0	Float	DR0	1

```

MainUnit > 主單元0
1 DR14:=123;
2 標籤0:=DR14;
3

```

監視頁

檢視 編輯 檔案

名稱	狀態	資料	註解	名稱	狀態	資料	註解
主單元0/ 標籤0	浮點數	123	[R0]	DR14	十進制	123	[R14]
主單元0/ 標籤0	十六進制	42F60000H	[R0]	DR14	十六進制	0000007BH	[R14]

Line 2: DR14(INT32)轉換成浮點數的資料型別，並且存進標籤 0 這個標籤中
(可觀察浮點數的數位表示跟整數不同，可當作 Fun 200 I -> F 運用)

EX:

主單元0

區域標籤

	名稱	類型	位址	長度
1	標籤0	Float	DR0	1
2	標籤1	Float	DR2	1

```

MainUnit > 主單元0
1 DR14:=123;
2 標籤0:=DR14;
3 FDIV( Sa:= 標籤0, Sb:=2, D:= 標籤1);
4

```

監視頁

檢視 編輯 檔案

名稱	狀態	資料	註解	名稱	狀態	資料
主單元0/ 標籤0	浮點數	123	[R0]			
主單元0/ 標籤1	浮點數	61.5	[R2]			

Line 3: Indicates that the floating-point value of Tag_FIR100 will be converted into an integer value of `int32`, and there will be a position of DR0, and the decimal point will be removed (ex. Tag_FIR100 is 3.14, and the content of DR0 is 3)

Line 5,7: It will convert DR2 (`INT32`) and R2 (`INT16`) into floating-point data types, and store them in Tag_FIR100 (ex. R2 is 3, and Tag_FIR100 will be converted to a floating-point type, resulting in 3.0.)

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_FIR100:=1E3; //1000 Tag_FIR100:=1.2E+3; //1200 Tag_FIR100:=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_FIR100:=1m; Tag_FIR100:=1n; Tag_FIR100:=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; //間接定址  
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:

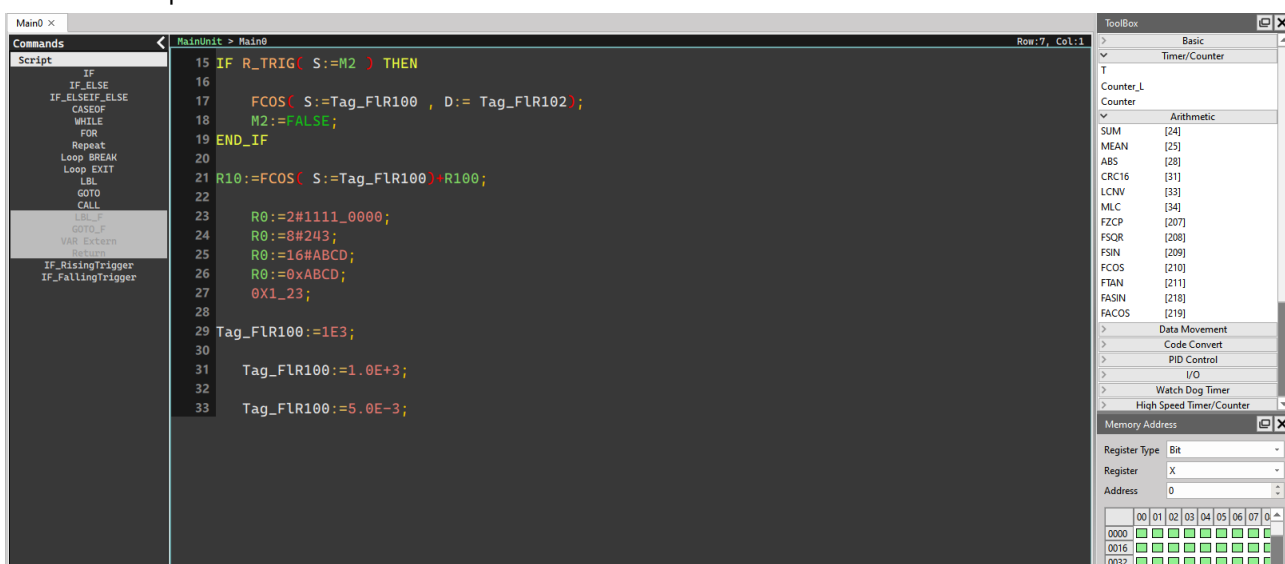
<Function> (<Parameter 1> := <Input Parameter 1>, ...);

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:二進制碼轉換格雷碼

錯誤轉換

正確轉換

MainUnit > 主單元1				MainUnit > 主單元1			
1 BintoGray(S:= DR300, D:= DR400);				1 BintoGray_D(S:= DR300, D:= DR400);			
監視頁				監視頁			
檢視 編輯 檔案				檢視 編輯 檔案			
名稱	狀態	資料	註解	名稱	狀態	資料	註解
DR300	十六進制	FFFFFFFH	[R300]	DR300	十六進制	FFFFFFFH	[R300]
DR400	十六進制	00008000H	[R400]	DR400	十六進制	80000000H	[R400]

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 TAREZOFFSET

These two functions are derived from the original [FUN258](#). 皮重偏移指令 模式拆出來的

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

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

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

3-10 Functions with multiple calling modes

Like the TRIG detection commands in Chapter 3-9-3 above, these commands support two types, one is directly complete call, the other is to omit the target value as the return value of the function, the following table shows the currently available function with the feature:

如前面 3-9-3 章節中的觸發偵測指令，這些指令都支援兩種型態，一種是直接完整呼叫，另一種是省略存放結果的暫存器位置(D)，下表則為目前具有該特性的函式

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:=)
Ftol	D	Ftol(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: (由於回傳數值及代表結果，則可以直接帶入一些四則運算或是條件是判斷)

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

3-11 複週期指令集

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:

```
SubUnit > Sub1
1 IF M0 THEN
2     M100:=1;
3 ELSE
4     M100:=0;
5 END_IF
6
7 HSPWM2( EN:= M100,Pw:= R2,Op:= 1,HZ:= R6,8,ACT=> M32,ERR=> M33);
8 |
```

↑
放在IF迴圈外,每次掃描都執行,依據EN訊號決定行為

這些指令的類別, 因為呼叫後, 執行時間通常都超過一個掃描週期。

The following table shows the 複週期指令 currently included:

Function	Function ID
LCNV	33
MLC	34
TPCTL2	99
RAMP2	98
HSPWM	139
HPSO	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	

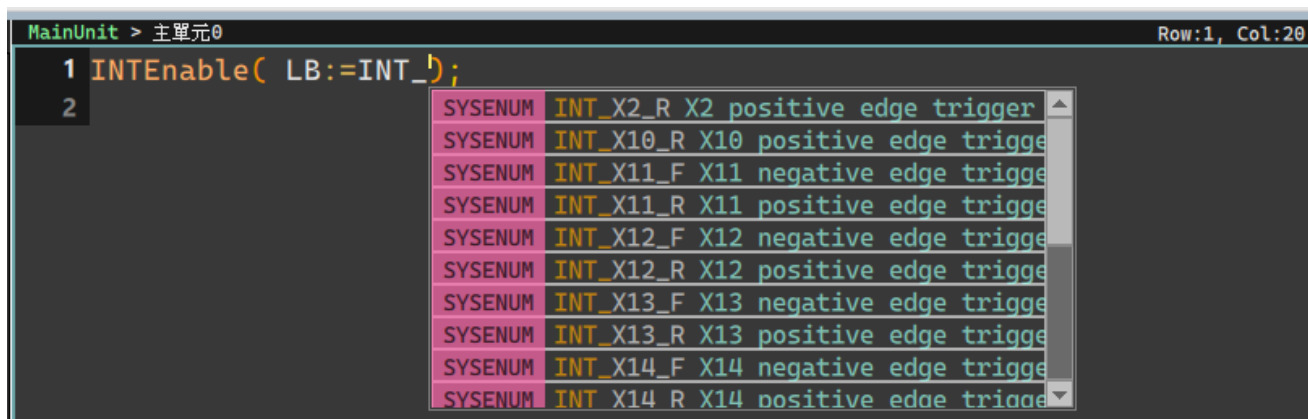
3-12 Enable/Disable Interrupt and Special Instructions

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

2			X0-I (INT0-)	X0 negative edge trigger	the priority of X0~X15.
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 unit 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	

29			HST4I	Interval from HST4 to (CV=PV)	Not supported yet
30			HST5I	Interval from HST5 to (CV=PV)	Not supported yet
31			HST6I	Interval from HST6 to (CV=PV)	Not supported yet
32			HST7I	Interval from HST7 to (CV=PV)	Not supported yet
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)	
41	External Module Event		COCPUI	Event from Co-processor (e.g. EtherCAT motion controller)	
42			LHMI	Event form left-side high-speed module	
43			RHM0I	Event form Right-side high-speed module 1	

44	RHM1I	Event form Right-side high-speed module 2	
45	RHM2I	Event form Right-side high-speed module 3	
46	RHM3I	Event form Right-side high-speed module 4	
47	RHM4I	Event form Right-side high-speed module 5	

48			RHM5I	Event form Right-side high-speed module 6	
49	Motion Control Cycle Interrupt		MSR	Periodic synchronization routine for motion	Motion Cycle = 1 ms
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

3-13 具有特殊意義的變數名稱

A. Timer, Counter

T0~TN, C0~CN

在 ST 環境中賦予這兩個變數同時具有 Bool 的特性跟 Integer 整數的特性
例如:

```
R0:=T1; //會把 Timer1 目前計算到的數值存到 R0(Integer)
```

```
M0:=T1; // 會把 Timer1 目前是否在運行的狀態存到 M0 (Bool)
```

```
M3:=C1>10; // 則會判斷 Counter 1 是否計數超過 10
```

```
IF R_TRIG( T1 ) THEN
```

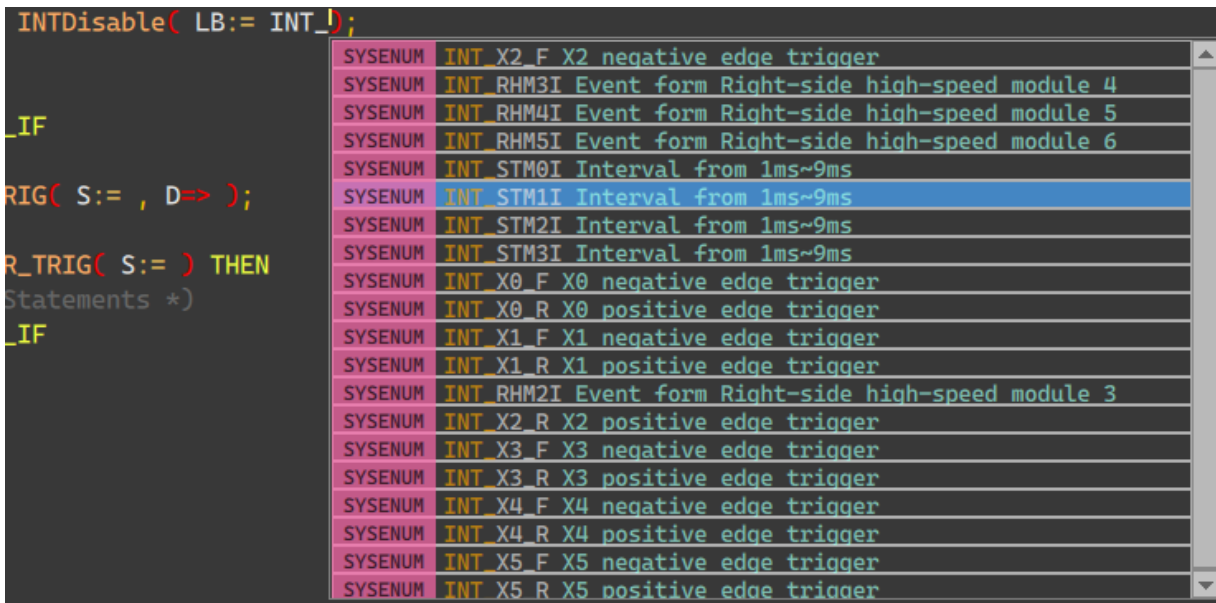
```
    /// 當 Timer1 從 0 變 1 的時候 (開始運行的瞬間),則會執行
```

```
END_IF
```

B. 中斷 Enum

原本 Enable / Disable 中斷只能輸入常數，為了讓程式看起來可讀性提高，修改為可以輸入系統預設的 Enum 參數

可以打 INT_ 的字樣就會顯示對應的一系列可用的 Enum



例如:

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

C. 高速計數 Enum “HSC_” Prefix

呼叫相關高速指令如

```
HSCTR( CN:= (* 0~7 *));
```

```
HSCTW( S:= , CN:= , D:= );
```

其中 CN, 跟 D 的參數式支援常數整數, 為了閱讀方便 也導入 HSC_ 開頭的 Enum

```
HSCTR( CN:= HSC_(* 0~7 *));  
HSCTW( S:= , CN:= , D:= );
```

SYSENUM	HSC_CV
SYSENUM	HSC_HSC0
SYSENUM	HSC_HSC1
SYSENUM	HSC_HSC2
SYSENUM	HSC_HSC3
SYSENUM	HSC_HSC4
SYSENUM	HSC_HSC5
SYSENUM	HSC_HSC6
SYSENUM	HSC_HSC7
SYSENUM	HSC_PV

讓程式更具可讀性

例如:

```
HSCTR( CN:= HSC_HSC0(* 0~7 *));
```

```
HSCTW( S:=R0 , CN:=HSC_HSC0 , D:=HSC_PV );
```

D. File Register Enum F0~F32767

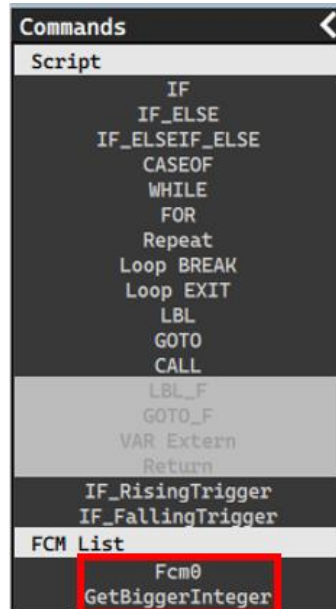
針對 FUN 160 的 Sb 參數，除了使用 0~32767 之外，可以使用 F0~F32767 的 Enum 來代表對應的位置。

例如：

```
15  
16 ReadWriteFileReg( EN:= R_TRIG(M0)(* *),  
17                   R_W:= 1(* *),  
18                   INC:= M10(* *),  
19                   Sa:= R11(* *),  
20                   Sb:= F10(* 0~32767 => F0~F32767 *),  
21                   Pr:= R100(* *),  
22                   L:= 10(* *));  
23
```

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.

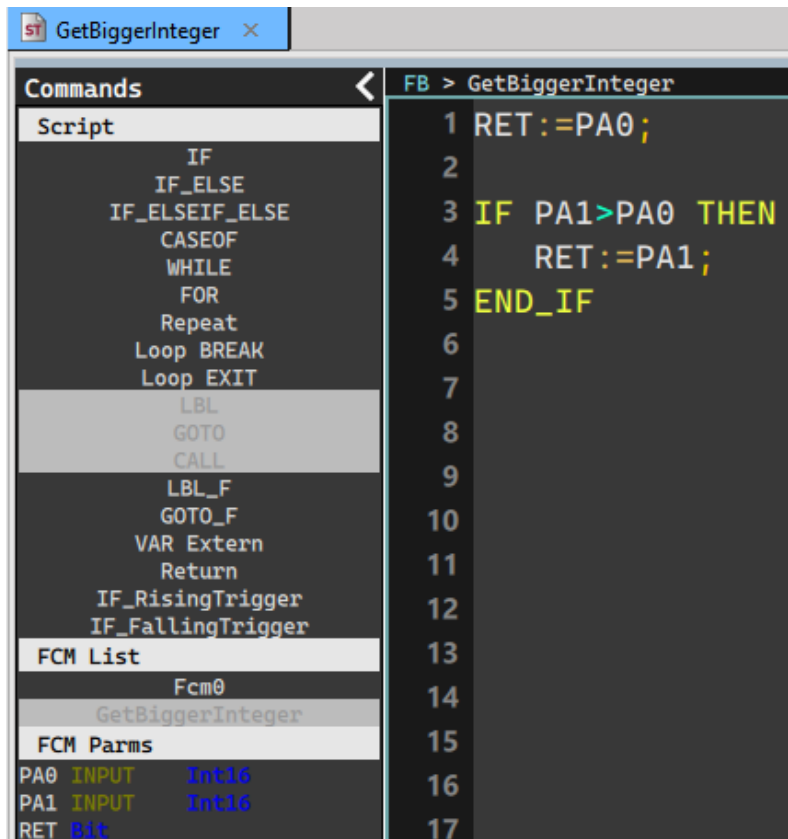


Users can double-click the section to insert the selected function to the text.

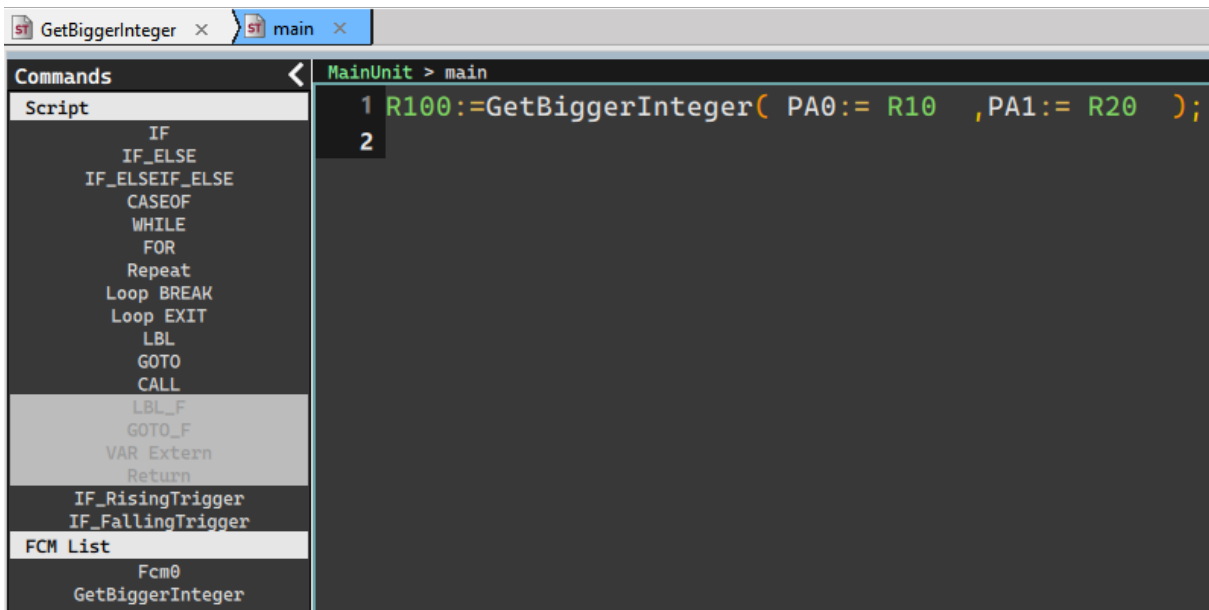
FCM can specify a Return Value, which can be call directly when programming.

EX:

FCM:



Calling side:



The screenshot shows a software interface with two tabs: 'GetBiggerInteger' and 'main'. The 'main' tab is active. On the left, there is a 'Commands' list with categories: 'Script', 'IF', 'IF_ELSE', 'IF_ELSEIF_ELSE', 'CASEOF', 'WHILE', 'FOR', 'Repeat', 'Loop BREAK', 'Loop EXIT', 'LBL', 'GOTO', 'CALL', 'LBL_F', 'GOTO_F', 'VAR Extern', 'Return', 'IF_RisingTrigger', 'IF_FallingTrigger', 'FCM List', and 'GetBiggerInteger'. The 'Script' category is selected. On the right, the script editor shows the following code:

```
MainUnit > main  
1 R100:=GetBiggerInteger( PA0:= R10 ,PA1:= R20 );  
2
```

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

3-15 函數注意事項

無論是呼叫系統內建的或是 FCM 均有以下共同特點

A. 呼叫方式：

1. 完整帶入參數名稱
2. 省略參數名稱
3. 輸出參數(=>)可以直接忽略不傳遞

範例:

以 R_TRIG 這個函式為例

完整表現：`R_TRIG(S:= M0, D=>M1);`

用參數帶入的話允許不按照預設的順序

上述的函式可以改成

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

`R_TRIG(D=>M1);`

都是合法輸入

省略參數名稱：`R_TRIG(M0, M1);`

帶有 Return 效果或是輸出的可以不傳遞：

```
R_TRIG( S:= M0 );
```

```
R_TRIG( M0 );
```

有 Return 特性的可以與其他變數或是描述句混合運用

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

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

！注意！ 帶入參數跟不帶入參數的呼叫方式不能混用如:

```
R_TRIG(S:=M0, M1);
```

```
R_TRIG(M1,D:=M2);
```

以上兩個都是不合法的輸入