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

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.	⚠
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:

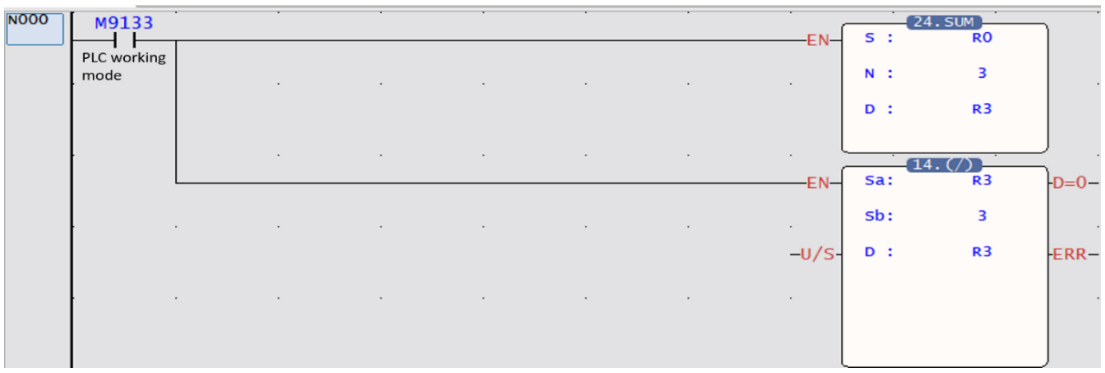
Substitute the average value of R0~R2 in R3.

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

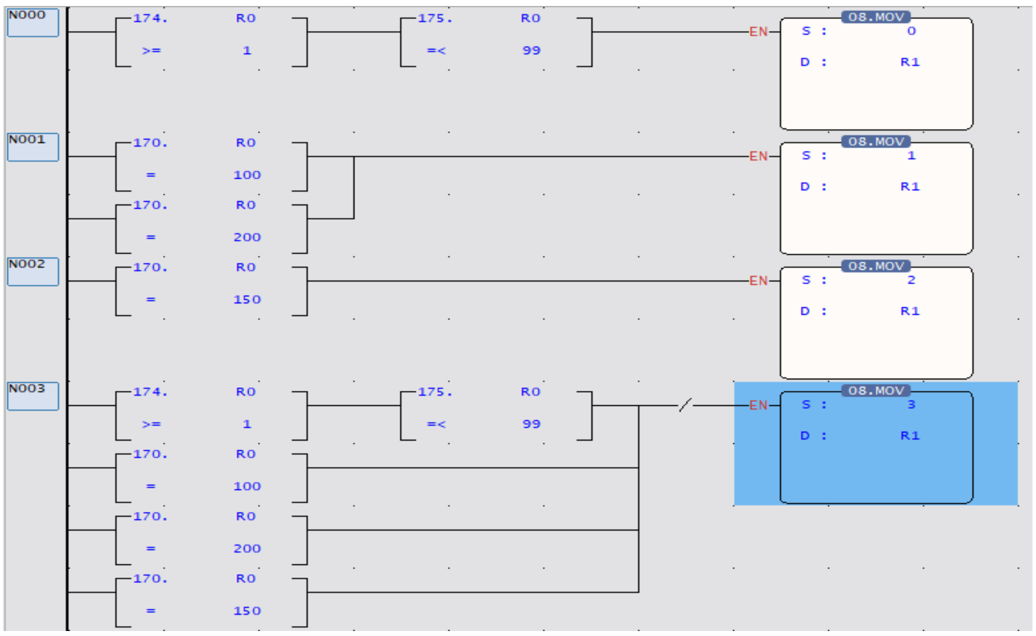
Set 0 to 3 in R1 according to the value of R0

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

ST

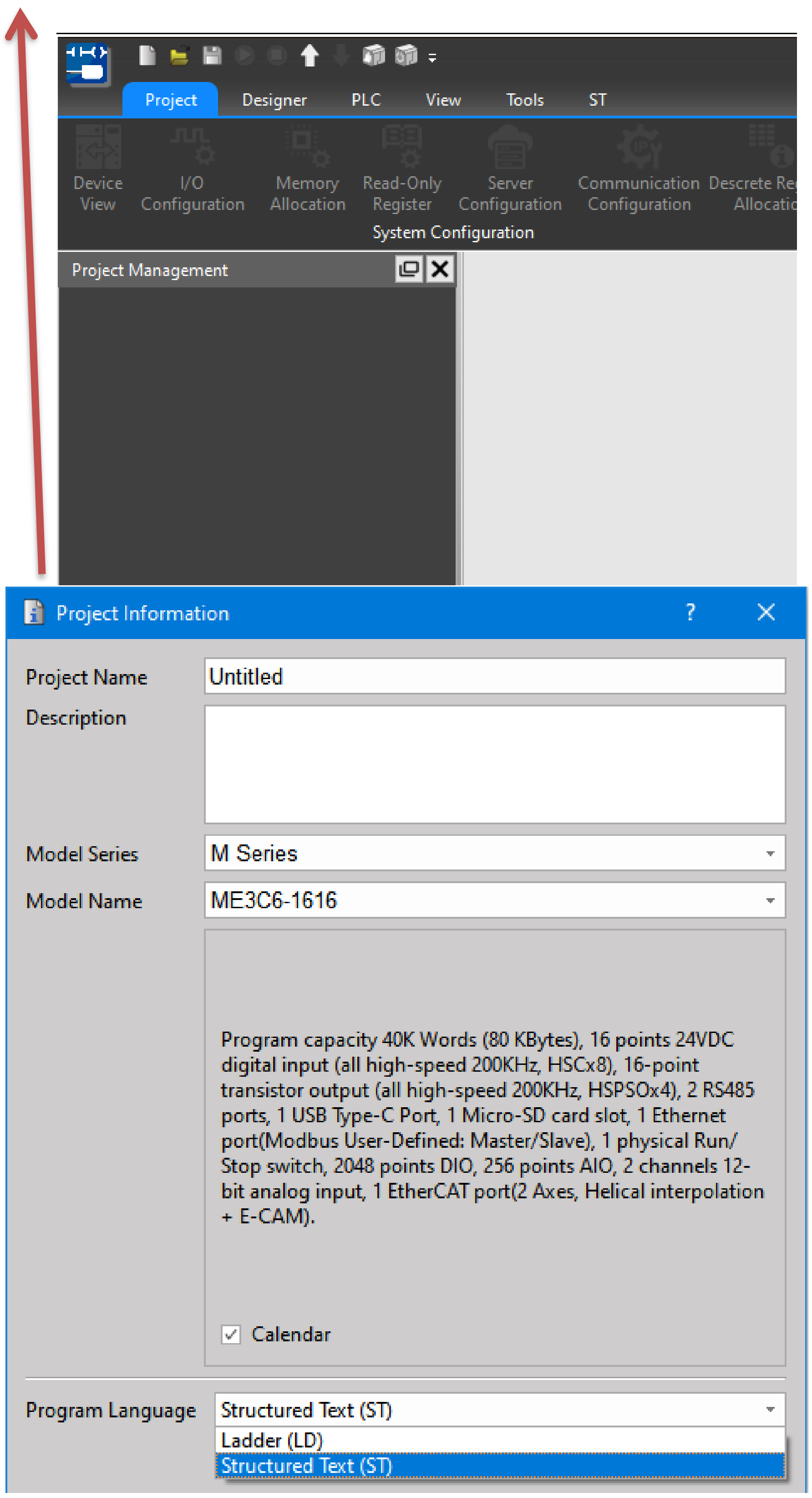
```
IF R0>=1 &R0 <=99 THEN
   R1:=0;
ELSEIF R0=100 or R0=200 THEN
   R1:=1;
ELSEIF R0=150 THEN
   R1:=2;
ELSE
   R1:=3;
END_IF
```

LD



1-2 Adding ST Language Program

Establishing a new Program



Project Name: Untitled

Description:

Model Series: M Series

Model Name: ME3C6-1616

Program capacity 40K Words (80 KBytes), 16 points 24VDC digital input (all high-speed 200KHz, HSCx8), 16-point transistor output (all high-speed 200KHz, HSPSOx4), 2 RS485 ports, 1 USB Type-C Port, 1 Micro-SD card slot, 1 Ethernet port(Modbus User-Defined: Master/Slave), 1 physical Run/Stop switch, 2048 points DIO, 256 points AIO, 2 channels 12-bit analog input, 1 EtherCAT port(2 Axes, Helical interpolation + E-CAM).

✓ Calendar

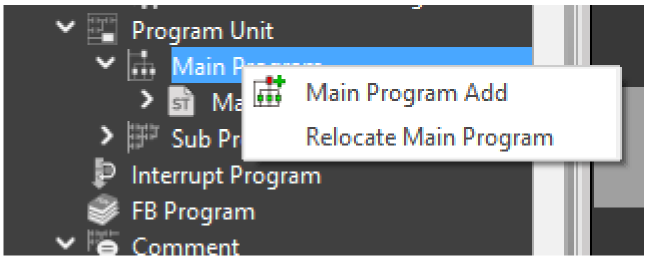
Program Language: Structured Text (ST)

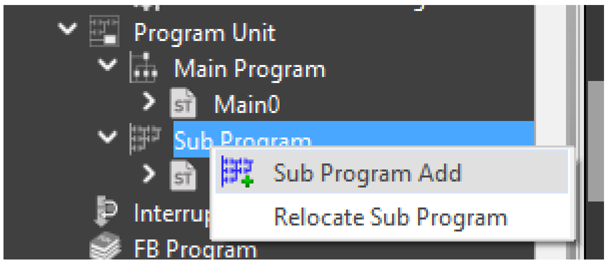
Ladder (LD)
[✓ Structured Text (ST)]

Example of creating MainUint/SubUnit(ST):

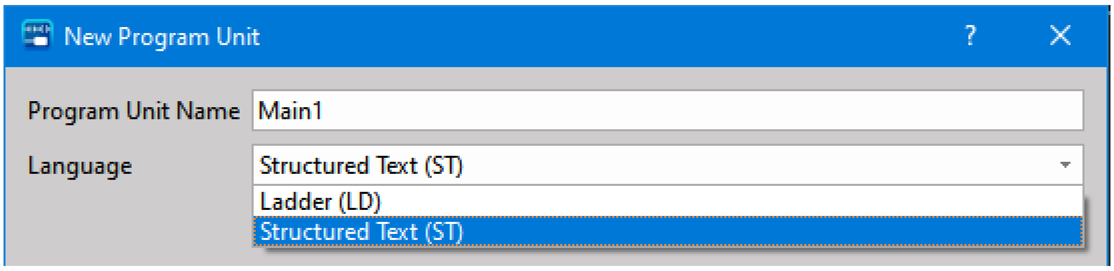
Right click on [Main Program] , or [Sub Program].

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





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

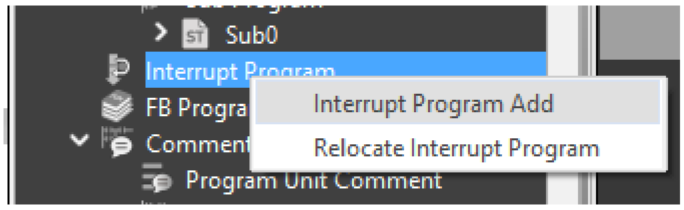


Program Unit Name: Main1
Language: Structured Text (ST)

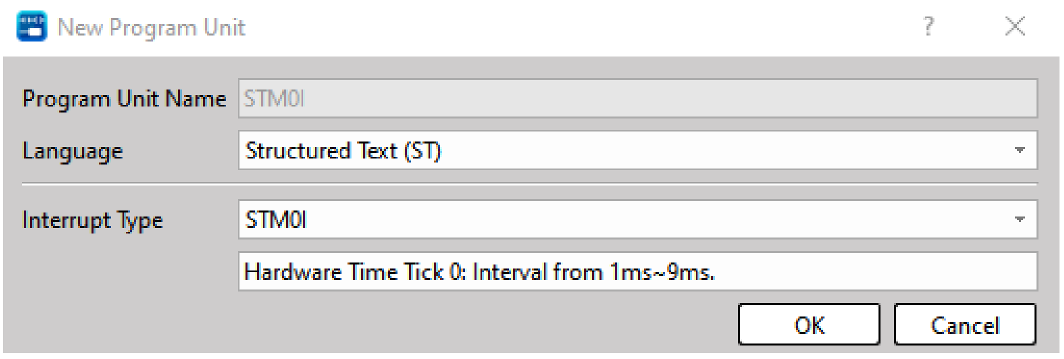
Example of creating an interrupt program (ST):

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.

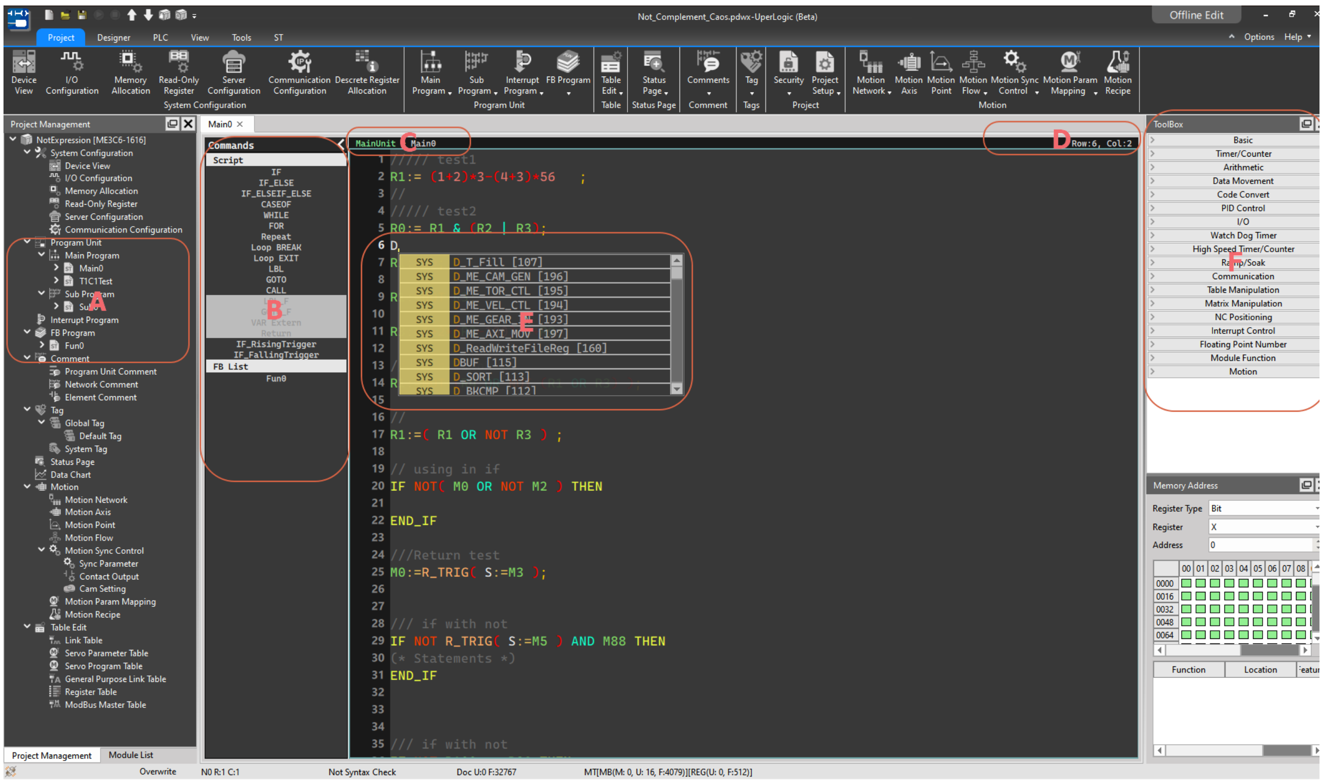


Program Unit Name: STM0I
Language: Structured Text (ST)
Interrupt Type: STM0I
Hardware Time Tick 0: Interval from 1ms~9ms.

2

User Interface of Uperlogic ST

2-1 Interface Overview



```
MainUnit  Main0                                             Row:6, Col:2
///// test1
R1:= (1+2)*3-(4+3)*56    ;
//
///// test2
R0:= R1 & (R2 | R3);
D
   SYS  D_T_Fill [107]
   SYS  D_ME_CAM_GEN [196]
   SYS  D_ME_TOR_CTL [195]
   SYS  D_ME_VEL_CTL [194]
   SYS  D_ME_GEAR_IN [193]
   SYS  D_ME_AXI_MOV [197]
   SYS  D_ReadWriteFileReg [160]
   SYS  DBUF [115]
   SYS  D_SORT [113]
   SYS  D_BKCMP [112]

//
R1:=( R1 OR NOT R3 ) ;

// using in if
IF NOT( M0 OR NOT M2 ) THEN

END_IF

///Return test
M0:=R_TRIG( S:=M3 );


/// if with not
IF NOT R_TRIG( S:=M5 ) AND M88 THEN
(* Statements *)
END_IF



/// if with not
```

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.

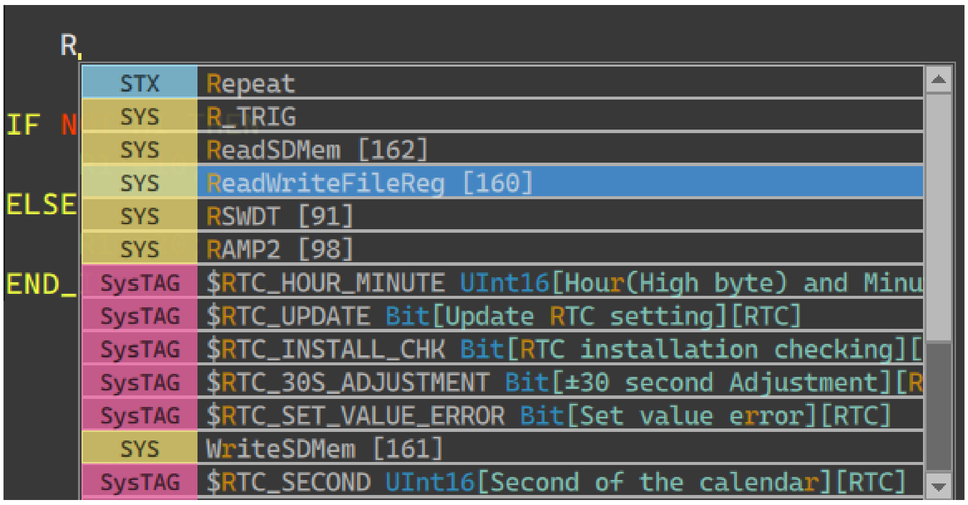
B.

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`

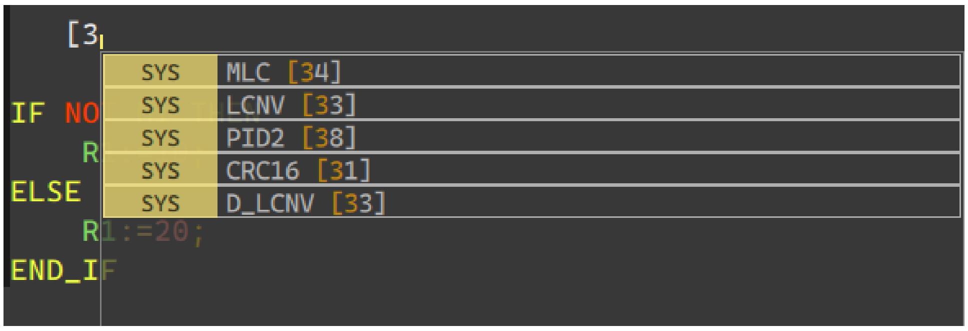
E. Smart pop-up reminder window, which is convenient for users to prompt when typing.



```
R
IF N
ELSE
END_

STX     Repeat
SYS     R_TRIG
SYS     ReadSDMem [162]
SYS     ReadWriteFileReg [160]
SYS     RSWDT [91]
SYS     RAMP2 [98]
SysTAG  $RTC_HOUR_MINUTE UInt16[Hour(High byte) and Minu
SysTAG  $RTC_UPDATE Bit[Update RTC setting][RTC]
SysTAG  $RTC_INSTALL_CHK Bit[RTC installation checking][
SysTAG  $RTC_30S_ADJUSTMENT Bit[±30 second Adjustment][R
SysTAG  $RTC_SET_VALUE_ERROR Bit[Set value error][RTC]
SYS     WriteSDMem [161]
SysTAG  $RTC_SECOND UInt16[Second of the calendar][RTC]
```

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).

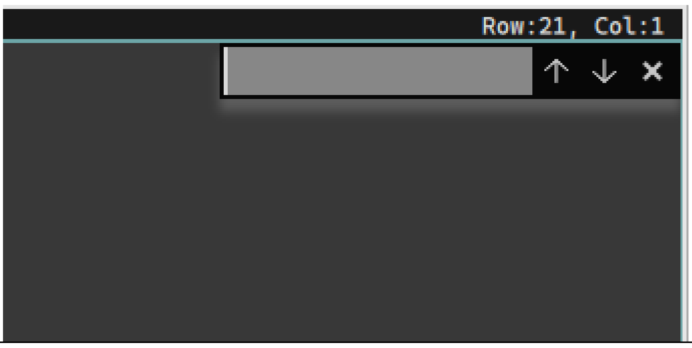


```
[3
IF NO
    R
ELSE
    R1:=20;
END_IF

SYS   MLC [34]
SYS   LCNV [33]
SYS   PID2 [38]
SYS   CRC16 [31]
SYS   D_LCNV [33]
```

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

2-2 Supportive Keyboard Instructions

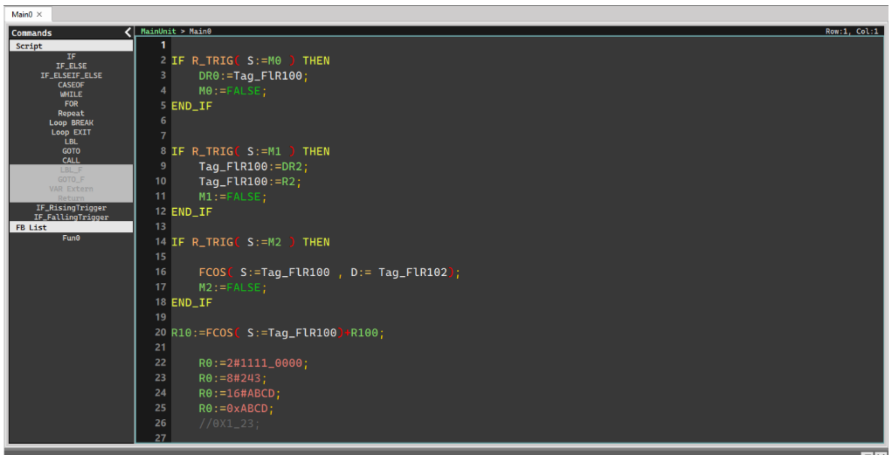
Key	Function
Ctrl + C	Copy
Ctrl + V	Paste
Ctrl + A	Select All
Ctrl + X	Cut
Tab	Insert Tab
Multiple Select+Tab	Select multiple lines and add Tab at the same time.
Shift + Tab	Back Tab simultaneously according to the cursor or the number of selected rows.
Ctrl　+ F	A search window appears at the top right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

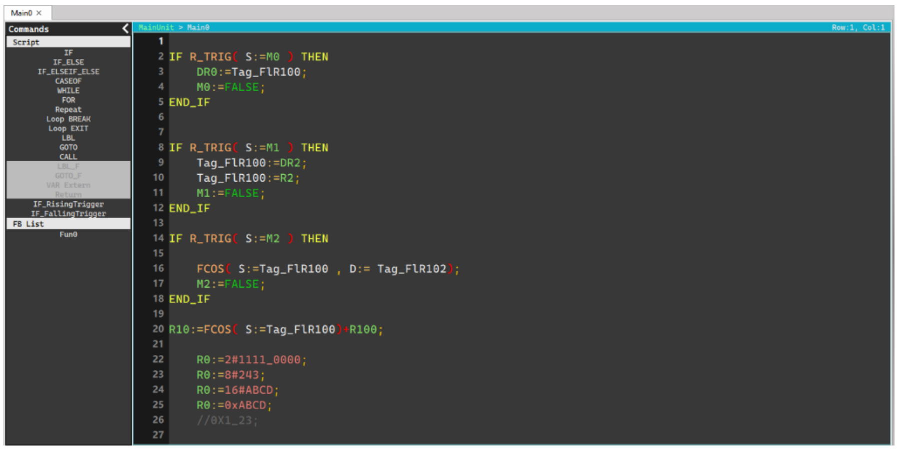
ST Editor diaplays three status correspondingly.

1. Offline Edit



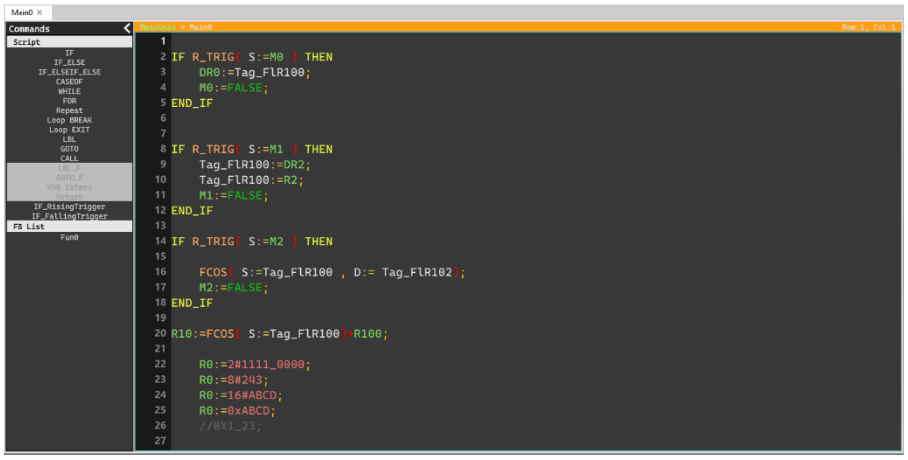
```

IF R_TRIG( S:=M0 ) THEN
    DR0:=Tag_FlR100;
    M0:=FALSE;
END_IF


IF R_TRIG( S:=M1 ) THEN
    Tag_FlR100:=DR2;
    Tag_FlR100:=R2;
    M1:=FALSE;
END_IF

IF R_TRIG( S:=M2 ) THEN

    FCOS( S:=Tag_FlR100 , D:= Tag_FlR102);
    M2:=FALSE;
END_IF

R10:=FCOS( S:=Tag_FlR100)+R100;

    R0:=2#1111_0000;
    R0:=8#243;
    R0:=16#ABCD;
    R0:=0xABCD;
    //0X1_23;

```

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



```

IF R_TRIG( S:=M0 ) THEN
    DR0:=Tag_FlR100;
    M0:=FALSE;
END_IF


IF R_TRIG( S:=M1 ) THEN
    Tag_FlR100:=DR2;
    Tag_FlR100:=R2;
    M1:=FALSE;
END_IF

IF R_TRIG( S:=M2 ) THEN

    FCOS( S:=Tag_FlR100 , D:= Tag_FlR102);
    M2:=FALSE;
END_IF

R10:=FCOS( S:=Tag_FlR100)+R100;

    R0:=2#1111_0000;
    R0:=8#243;
    R0:=16#ABCD;
    R0:=0xABCD;
    //0X1_23;

```

3. Online Edit

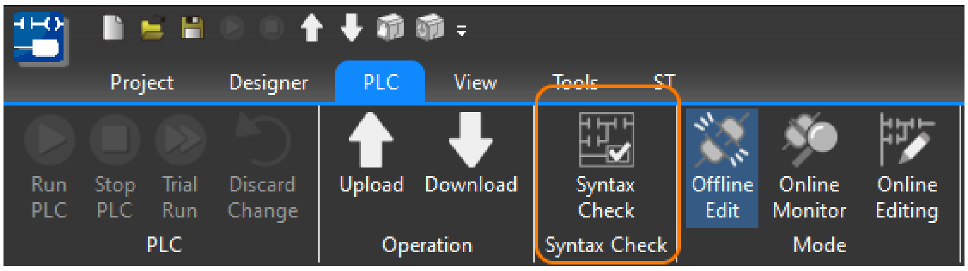


```

IF R_TRIG( S:=M0 ) THEN
    DR0:=Tag_FlR100;
    M0:=FALSE;
END_IF


IF R_TRIG( S:=M1 ) THEN
    Tag_FlR100:=DR2;
    Tag_FlR100:=R2;
    M1:=FALSE;
END_IF

IF R_TRIG( S:=M2 ) THEN

    FCOS( S:=Tag_FlR100 , D:= Tag_FlR102);
    M2:=FALSE;
END_IF

R10:=FCOS( S:=Tag_FlR100)+R100;

    R0:=2#1111_0000;
    R0:=8#243;
    R0:=16#ABCD;
    R0:=0xABCD;
    //0X1_23;

```

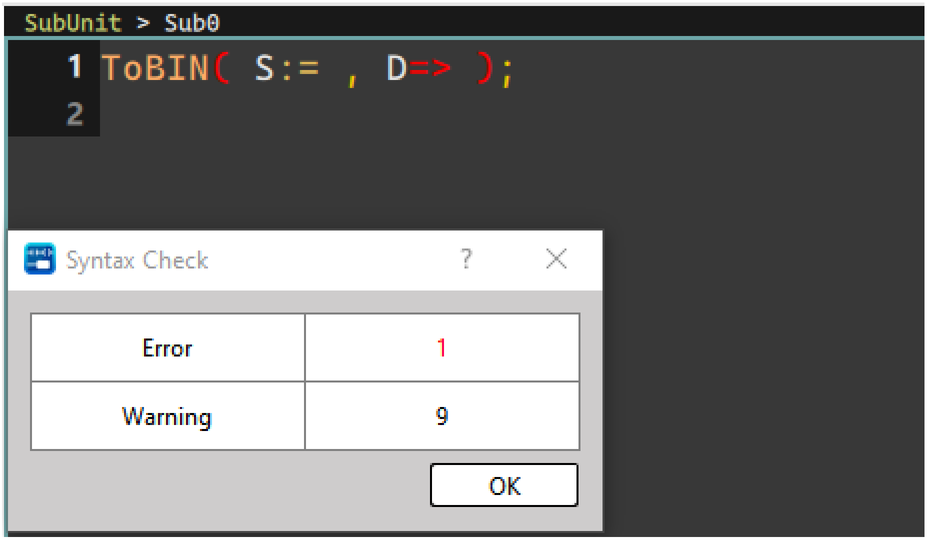
2-4 編譯文本 / Syntax Check

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:

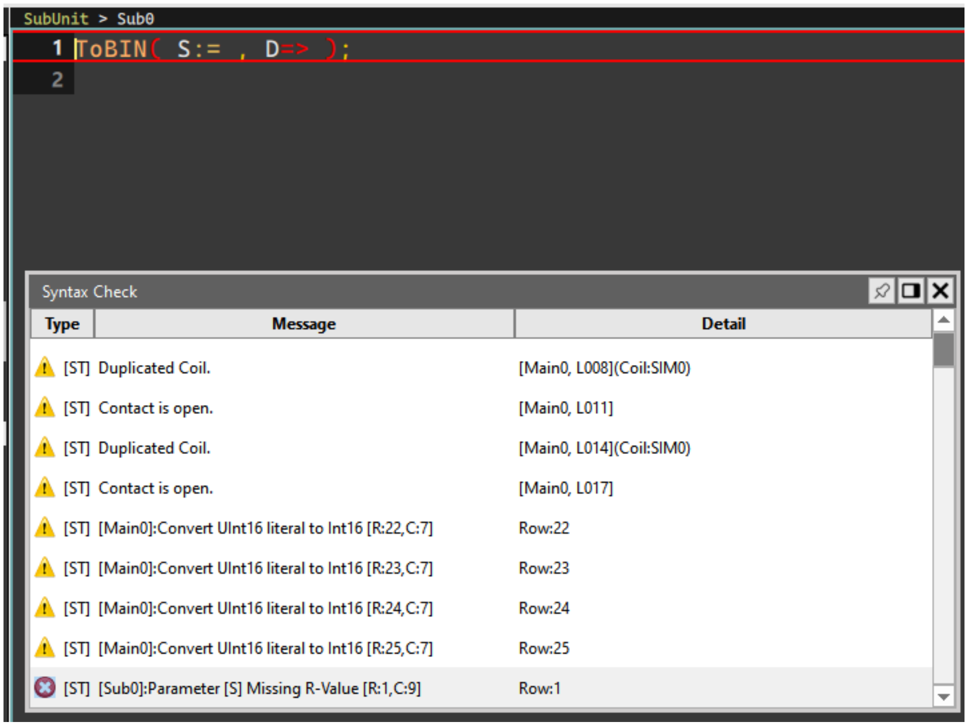

SubUnit > Sub0

```
ToBIN( S:= , D=> );
```

Syntax Check

Error	1
Warning	9

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.


SubUnit > Sub0

```
ToBIN( S:= , D=> );
```

Syntax Check

Type	Message	Detail
⚠	[ST] Duplicated Coil.	[Main0, L008](Coil:SIM0)
⚠	[ST] Contact is open.	[Main0, L011]
⚠	[ST] Duplicated Coil.	[Main0, L014](Coil:SIM0)
⚠	[ST] Contact is open.	[Main0, L017]
⚠	[ST] [Main0]:Convert UInt16 literal to Int16 [R:22,C:7]	Row:22
⚠	[ST] [Main0]:Convert UInt16 literal to Int16 [R:23,C:7]	Row:23
⚠	[ST] [Main0]:Convert UInt16 literal to Int16 [R:24,C:7]	Row:24
⚠	[ST] [Main0]:Convert UInt16 literal to Int16 [R:25,C:7]	Row:25
⊗	[ST] [Sub0]:Parameter [S] Missing R-Value [R:1,C:9]	Row:1

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.

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、:=、(、)	
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	
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)	
Constant		Interger: Defaulted as "signed int" Ex: **R0**:=1; **R0**:=-2; "Unsigned int" Ex: **Tag0**:=0xFF; String: Use only on Label	

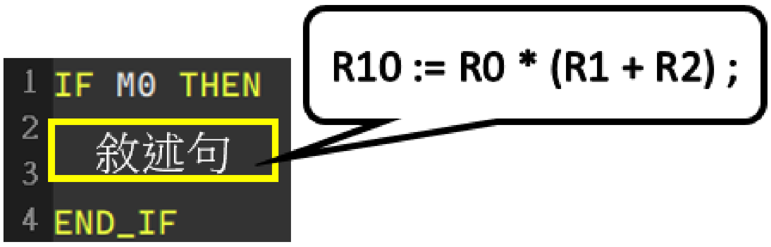
	Goto, LBL...etc Bit (Bool) Type: TRUE、FALSE Float: 32-bit Float Constant ex: TagFloat1:=1.1;	
Delimiter	":" appears in **CASE OF** 分隔 "," 多個函式參數傳遞使用	
Left/Right Round Brackets	"(", ")" 1. The start and end of Function Call ex: Fun0 (S:=R0, D:=R1); 2. The enforced priority of a general operator R0:= (1+R0)*R2;	
RETURN	Used in Sub-program or FB. To immediately leave the program section.	
";" the end of the statement	To mark the end of a sentence running at one end.	R0:=R1*2;

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

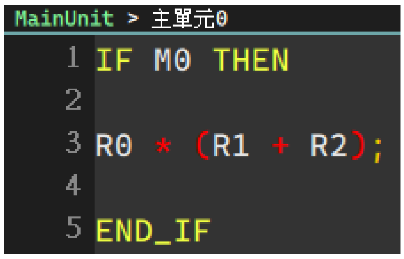
"Statement" is the most basic execution unit in ST Language, which represents a complete work to be executed. A complete statement is not limited to the same line of words; however, it must be ended with " ; ." In addition, a statement is also allowed to contain multiple or multi-level sub-statement,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



```
IF M0 THEN
    敘述句
END_IF
```

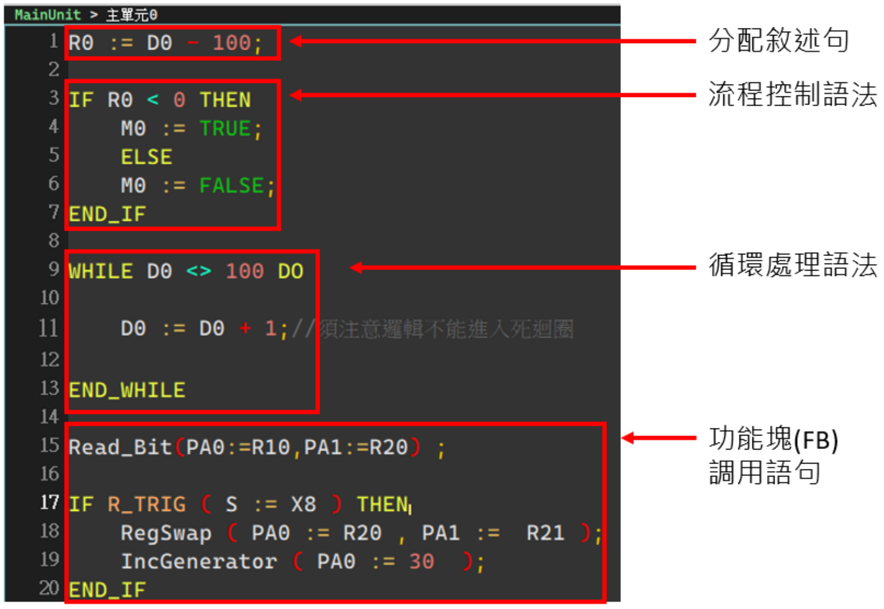
```
R10 := R0 * (R1 + R2) ;
```

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*(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.



```
IF M0 THEN

R0 * (R1 + R2);

END_IF
```

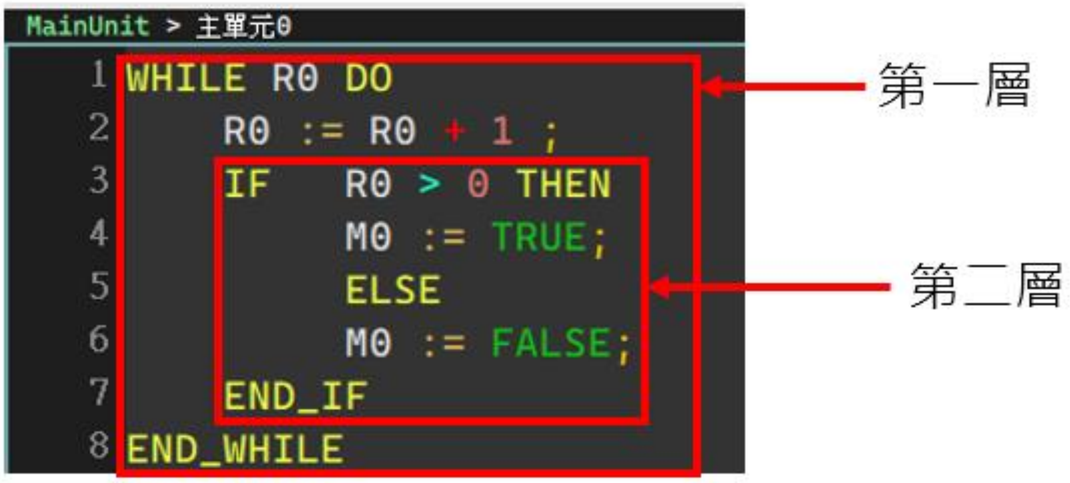
Shown below are some statements of Uperlogic ST Language:



```
R0 := D0 - 100;

IF R0 < 0 THEN
    M0 := TRUE;
    ELSE
    M0 := FALSE;
END_IF

WHILE D0 <> 100 DO

    D0 := D0 + 1;//須注意邏輯不能進入死迴圈

END_WHILE

Read_Bit(PA0:=R10,PA1:=R20) ;

IF R_TRIG ( S := X8 ) THEN
    RegSwap ( PA0 := R20 , PA1 :=  R21 );
    IncGenerator ( PA0 := 30  );
END_IF
```

Type		Content	Example
Distributive Statement		Substitute the result on the right into the variable on the left.	:=
Flow Control Syntax	選擇 Statement (IF, CASE)	Select the execution syntax based on the condition.	
	循環處理 Statement (FOR, WHILE)	Execute multiple times depending on the end condition.	
Sub-program Statement	FB 調用 Statement	調用 FB	
	Label Statement	調用 LBL	
Toolbox Statement		Convenient statement to execute toolbox instructions	

Flow control syntax can be layered.
(It can be used in combination with 選擇 statements and 循環處理 statements))



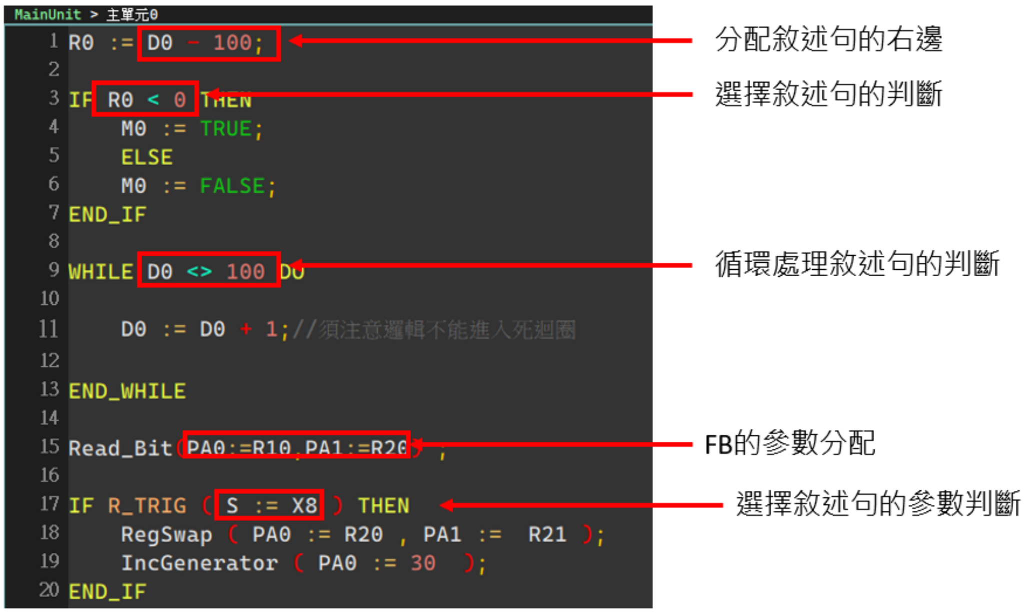
```
WHILE R0 DO
    R0 := R0 + 1 ;
    IF    R0 > 0 THEN
          M0 := TRUE;
          ELSE
          M0 := FALSE;
    END_IF
END_WHILE
```

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 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:



```
R0 := D0 - 100;

IF R0 < 0 THEN
    M0 := TRUE;
    ELSE
    M0 := FALSE;
END_IF

WHILE D0 <> 100 DO

    D0 := D0 + 1;//須注意邏輯不能進入死迴圈

END_WHILE

Read_Bit(PA0:=R10,PA1:=R20) ;

IF R_TRIG ( S := X8 ) THEN
    RegSwap ( PA0 := R20 , PA1 :=  R21 );
    IncGenerator ( PA0 := 30  );
END_IF
```

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)	
	Logic Expression	Boolean (TRUE/FALSE)	
	Compare Expression	Boolean (TRUE/FALSE)	
Basic Expression	Variables, Constants	Defined data type	
	函數調用表達式	Data type of return value	

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.

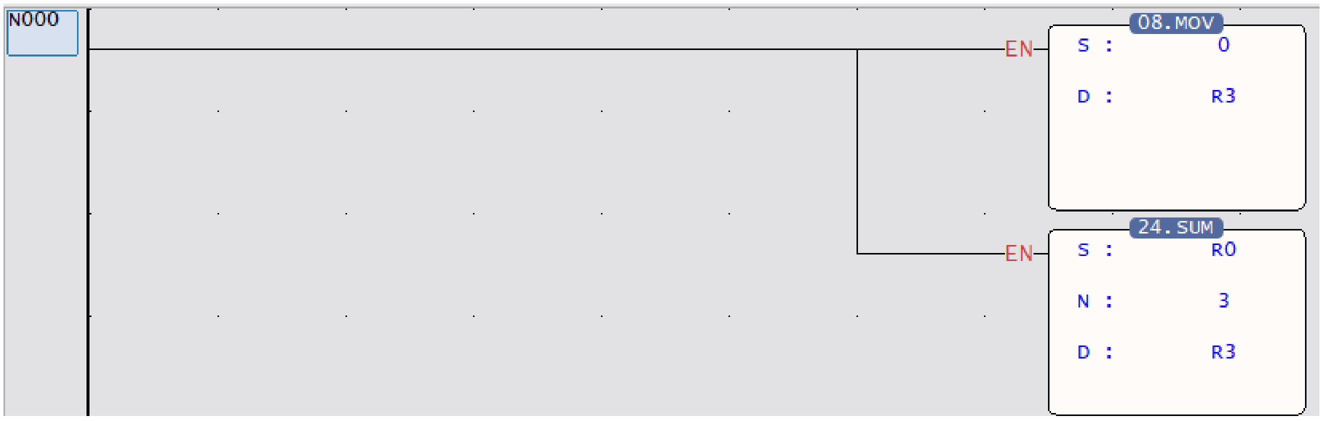
Program Example:

Substitute the adding result of R0-R2 in R3
R3=R0+R1+R2

3-3-3 ST

R3:=R0+R1+R2;

3-3-4 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.。

3-3-5 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:=);
Natural Logarithm	LN	e^{X}	
浮點數對數運算	LOG	10^{X}	
浮點數自然指數運算	EXP	e^{X}	
Trigonometric Functions	SIN、ASIN	B=SIN A	FSIN(S:= , D:=);

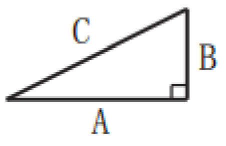
	COS、ACOS	B=COS A	FACOS(S:= (* *),
			D:= (* *),
			MD:= (* *);

	TAN、ATAN	B=TAN A	FSIN(S:= , D:=);

Program Example

Find the length of the hypotenuse of a right triangle.

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



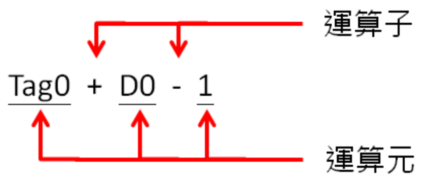
ST

DR0 := FSQR ((R1*R1)+(R2*R2));

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 + 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		
NOT	Logic inversion	Boolean	Boolean	NOT M0	TRUE	
++,--	To quickly add or subtract 1	Any value	Any value	++D0 D0++		
*	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	"互斥或" 運算 除法取整數	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.

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:

R1:= (R1+2)*3-(R4+3)*56

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

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

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;
R1:=10;
R100:=++R0; //// R100 is 11, R0 is 11
R101:=R1++; //// R101 is 10, R1 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 丟失調資料 (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 < <, > >

針對左側數值 左右 Shift 右側的位元數

EX:

R10:=R0< <4; R0 value will be shifted right with 4 bits, and assign to R10

P.S. R0 will not be changed

3-4-7 反向運算元 NOT

針對右邊的運算元進行 Bitwise 反向

EX:

R1:=NOT R3; // Do "one's complmenet" to the value in R3 and save it to R1

M0:= NOT (M0 OR M2); // M0 跟 M2 做完 or 之後 在反向 存到 M0

3-4-8 負號運算元 -

會針對右邊的 Operand 進行反向

EX: R0:=-R0;

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 0x01, R2 is 0x03, and R0 will be 0x01

3-4-11 Remainder MOD

Operates the remainder of the left and right operands which are equal to C's "%" symbol

EX:

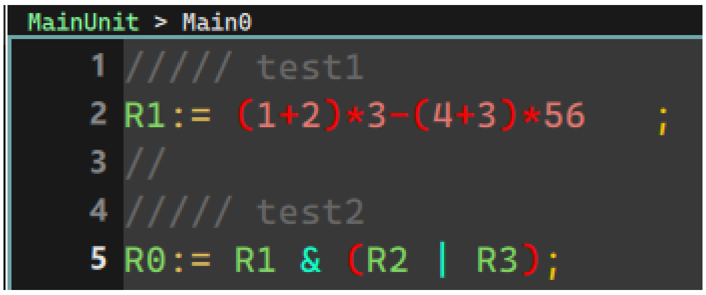
R0:= R1 MOD R2 ; // R0 is the remainder after dividing R1 by R2

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 //

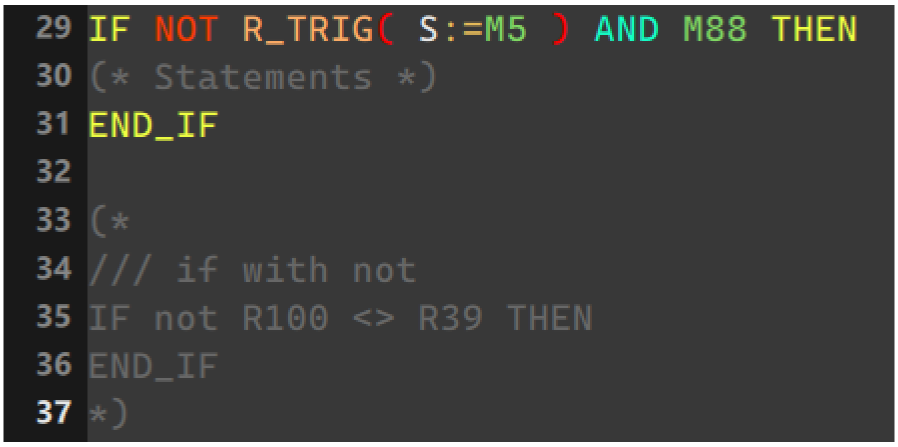
Ligh gray will appear after the symbol, and will not generate the operating program codes.


MainUnit > Main0

```
///// test1
R1:= (1+2)*3-(4+3)*56    ;
//
///// test2
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.



```
IF NOT R_TRIG( S:=M5 ) AND M88 THEN
(* Statements *)
END_IF

(*
/// if with not
IF not R100 <> R39 THEN
END_IF
*)
```

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

```
FOR (*Double Words計數暫存器 *) := (*初始值(常數)*) TO (*目標值(常數)*) BY 1 (*遞增值(常數)*) 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++;                               // R10++will be run repeatedly
END_FOR

FOR DR0:=0 TO 30 DO      //DR0 from 0,1,2,3....etc, until it reaches to 30
        R10++;                               // R10++ will be run repeatedly
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
```

REPEAT:

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

EX: The result of the two operation below are the same:

```
REPEAT
    ++R0;
UNTIL NOT M0
END_REPEAT

WHILE M0 DO
   ++R0;
END_WHILE
```

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.

P.S. For the loop series such as FOR, WHILE, REPEAT may cause an error when the PLC is running because the loop runs for too long. Please pay special attention when using it. You can use the interrupt signal to deal with the problem of loop processing for too long.

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.

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

JMP_66 , CALL (Please refer to FUN 66 for detailed description)

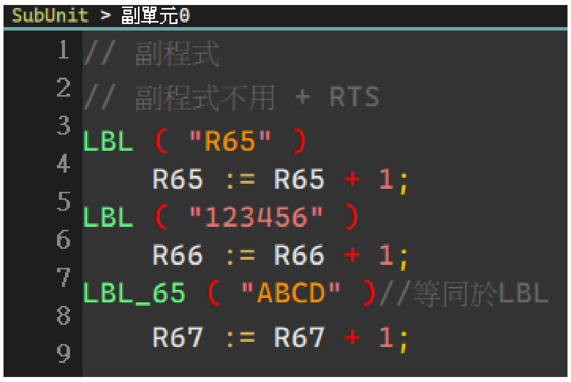
CALL_67 , GOTO (Please refer to FUN 67 for detailed description)

Use only in FB:

GOTO_F (Label that can only be used in FB)

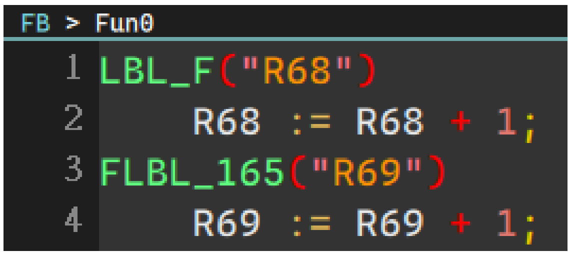
FJMP_166

EX:



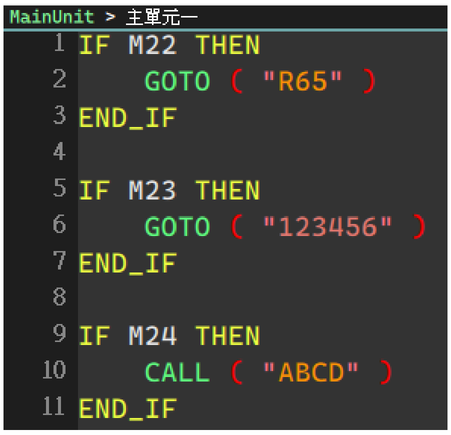
```
SubUnit > 副單元0
// 副程式
// 副程式不用 + RTS
LBL ( "R65" )
     R65 := R65 + 1;
LBL ( "123456" )
     R66 := R66 + 1;
LBL_65 ( "ABCD" )//等同於LBL
     R67 := R67 + 1;
```

If you need to use labels in the FB environment, you need to use the LBL_F instruction.



```
FB > Fun0
LBL_F("R68")
     R68 := R68 + 1;
FLBL_165("R69")
     R69 := R69 + 1;
```

Jump to such label:



```
MainUnit > 主單元一
IF M22 THEN
     GOTO ( "R65" )
END_IF

IF M23 THEN
     GOTO ( "123456" )
END_IF

IF M24 THEN
     CALL ( "ABCD" )
END_IF
```

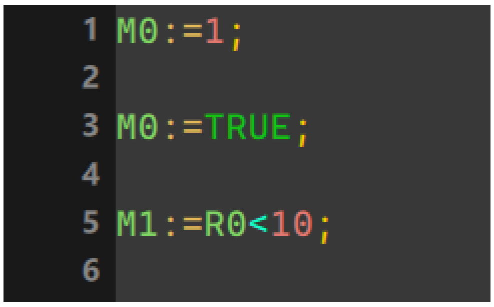
3-7 Variables and Data Type

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.

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:



```
M0:=1;

M0:=TRUE;

M1:=R0<10;

```

3-7-2 Integer Type

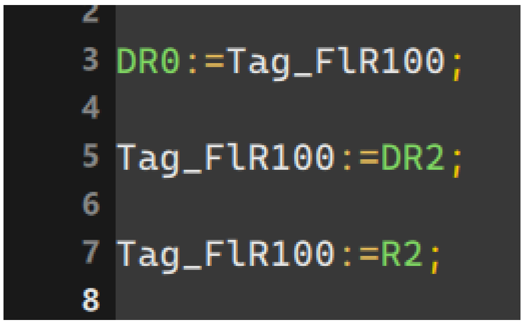
There are four integer types of ST: INT16, UINT16, INT32, UINT32

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:



```
3 DR0:=Tag_FlR100;
4
5 Tag_FlR100:=DR2;
6
7 Tag_FlR100:=R2;
8
```

Line 3: Indicates that the floating-point value of Tag_FlR100 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_FlR100 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_FlR100 (ex. R2 is 3, Tag_FLR100 裏頭則回 IEEE-754 的 3.0 浮點數)

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;

	Tag_FlR100:=1.2E+3;
	Tag_FlR100:=1E-3;

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 調用 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:

```
R0:=100;
X0:=TRUE;
Y0:=1;
V:=19; //indirect address
R200:= R10V+100;
```

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 反向
M0:=NOT T1;
```

3-9 Calling System Built-In Functions

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

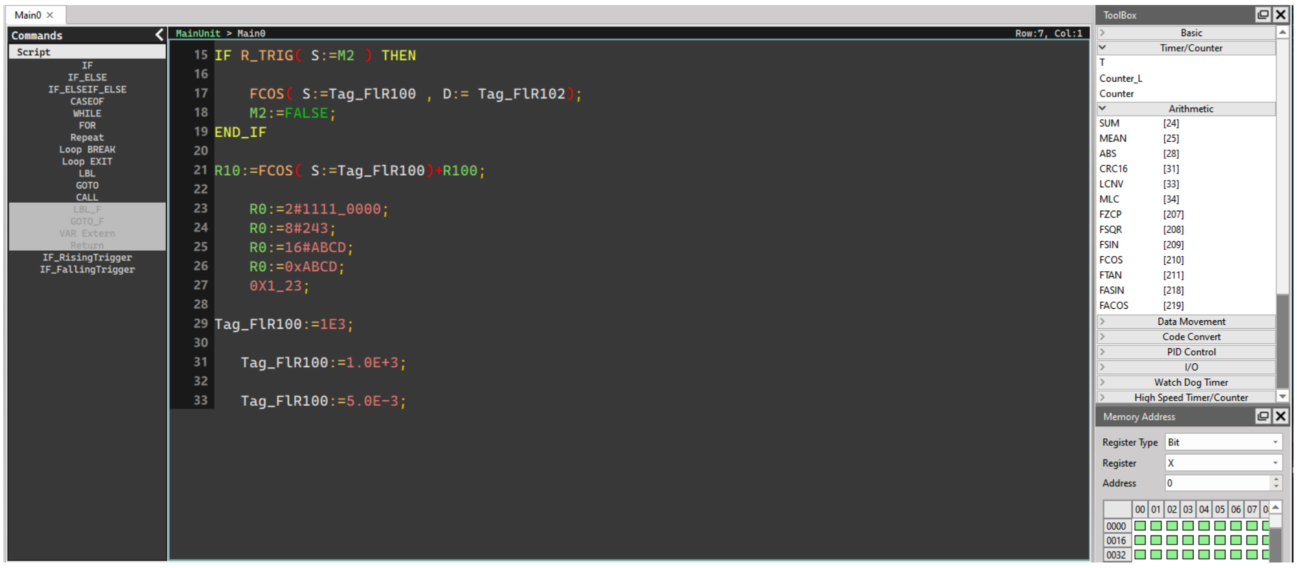
<Function> (　<Parameter 1> := <Input Parameter 1> ,　....);

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.

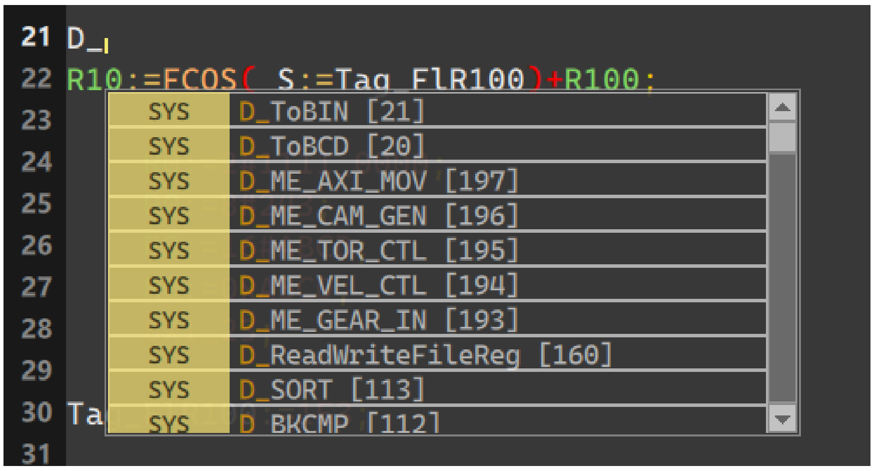
When calling, the order of parameters can be changed at will, but it cannot be omitted.

Unless some special values are allowed to be omitted



```
IF R_TRIG( S:=M2 ) THEN

    FCOS( S:=Tag_FlR100 , D:= Tag_FlR102);
    M2:=FALSE;
END_IF

R10:=FCOS( S:=Tag_FlR100)+R100;

    R0:=2#1111_0000;
    R0:=8#243;
    R0:=16#ABCD;
    R0:=0xABCD;
    0X1_23;

Tag_FlR100:=1E3;

   Tag_FlR100:=1.0E+3;

   Tag_FlR100:=5.0E-3;
```

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.



```
D_
R10:=FCOS( S:=Tag_FlR100)+R100;
```

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
Clr	(bool)	:	Whether to clear counter
C	(int)	:	Counter id 0~1279
PV	(int)	:	Counter Default value
IsUp	(bool)	:	Whether Counter reaches the target

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.

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=> )
```

3-9-5 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:

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:=)

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)

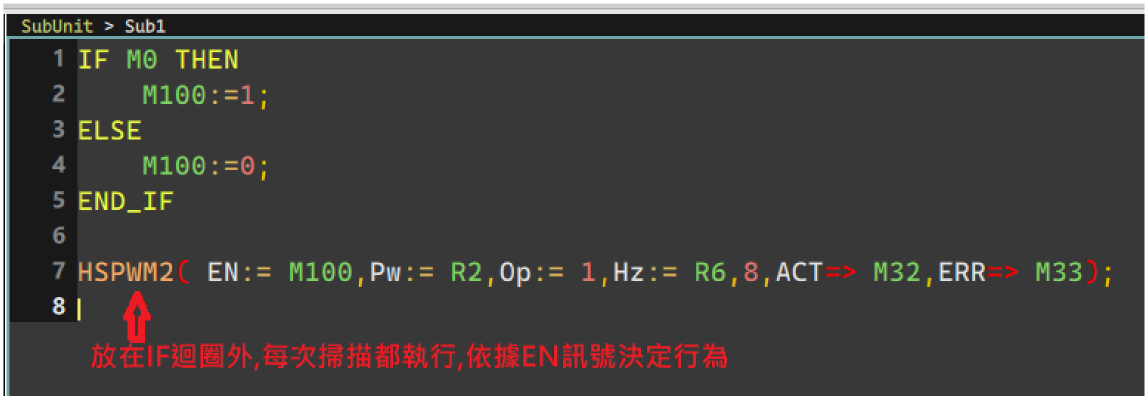
Omit the return parameter: (由於回傳數值及代表結果，則可以直接帶入一些四則運算或是條件是判斷)

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

3-9-6 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

```
IF M0 THEN
    M100:=1;
ELSE
    M100:=0;
END_IF

HSPWM2( EN:= M100,Pw:= R2,Op:= 1,Hz:= R6,8,ACT=> M32,ERR=> M33);
```

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
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	

3-9-7 Enable/Disable Interrupt and Special Instructions

Please refer to FUN 145, 146

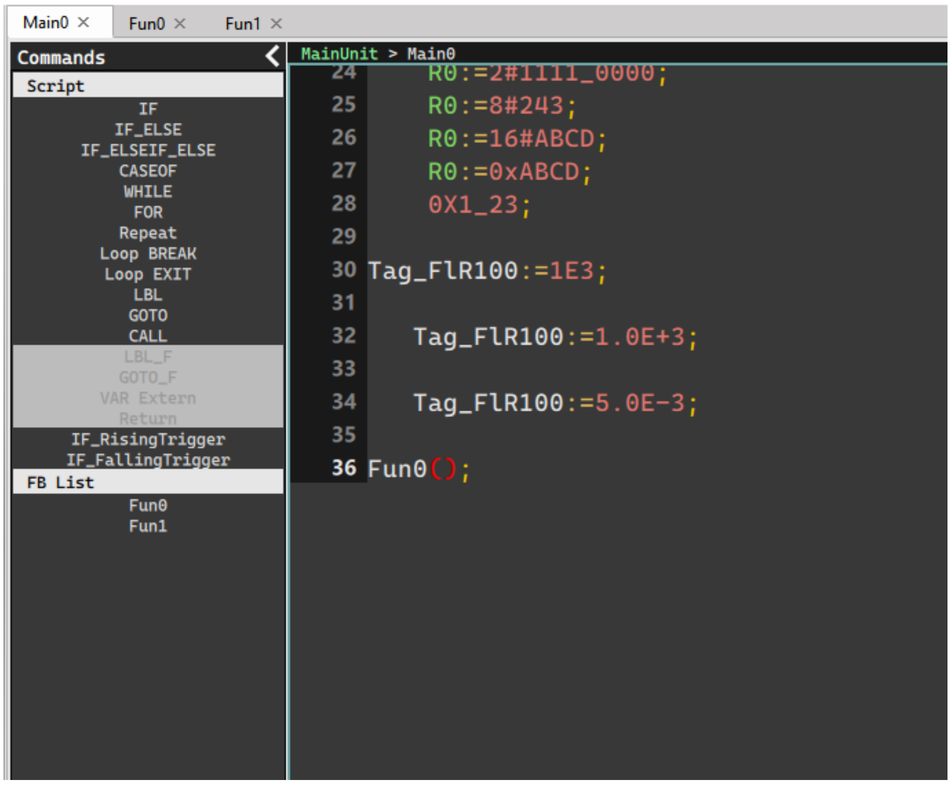
```
INTEnable( LB:= (* Interrupt number (Range 1~48) *));
INTDisable( LB:= (* Interrupt number (Range 1~48) *));
```

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.

3-10 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.



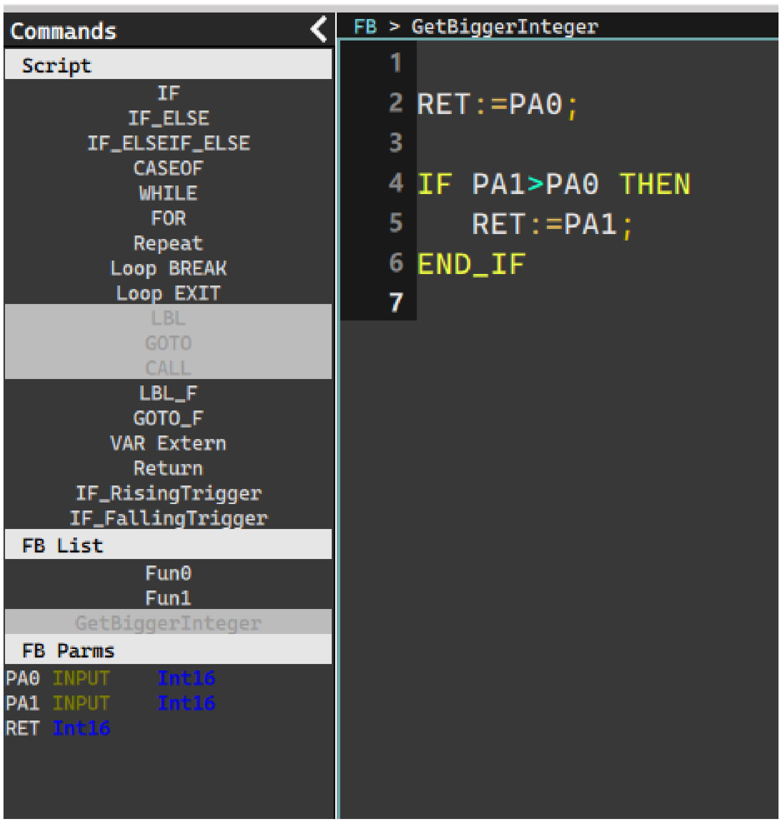
```
    R0:=2#1111_0000;
    R0:=8#243;
    R0:=16#ABCD;
    R0:=0xABCD;
    0X1_23;

Tag_FlR100:=1E3;

   Tag_FlR100:=1.0E+3;

   Tag_FlR100:=5.0E-3;

Fun0();
```

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

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

EX:

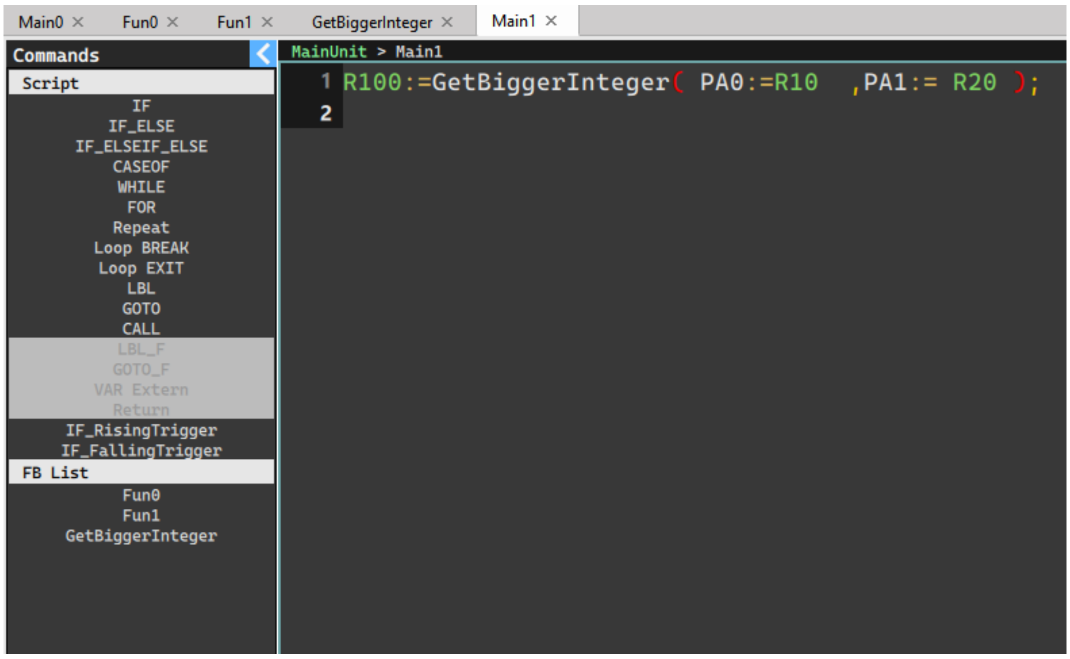
FCM:



```
PA0 INPUT    Int16
PA1 INPUT    Int16
RET Int16
```

```

RET:=PA0;

IF PA1>PA0 THEN
    RET:=PA1;
END_IF

```

Calling side:



```
R100:=GetBiggerInteger( PA0:=R10   ,PA1:= R20 );

```

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

4

Establishing ST Program in Uperlogic

This chapter explains the precautions for the basic system configuration and the contents related to peripheral devices.

4-1 ST Editing Envorinment in Uperlogic

You can add M-series modules to the right of M-series PLC CPU modules to match different applications. Available modules include Digital I/O Module, Analog I/O Module, Temperature Module, Network Module, Load Cell Module, etc.

4-2 Editing ST Statement

Description of the precautions for system configuration.

4-3 Insert API and FBD

Use generally available standard (above CAT5 shielding level) RJ-45 (Ethernet) connectors 將 M 系列 CPU 模組 EtherCAT 埠與支援 E

4-4 Labeling Function

If the 上位裝置 or other systems want to connect with the M-series PLC, users can choose any port of USB, RS-485 or EtherNET to connect with it. It can also be connected with M-series PLC through UperLogic. For detailed connection and setting methods, please refer to the M-series PLC software Interface User Manual.

5

ST Program Examples

"UperLogic" is the name of logic editing and testing application software for FATEK M-series PLC, which can be used for PLC logic editing, network setting, servo control, temperature control and other functions.

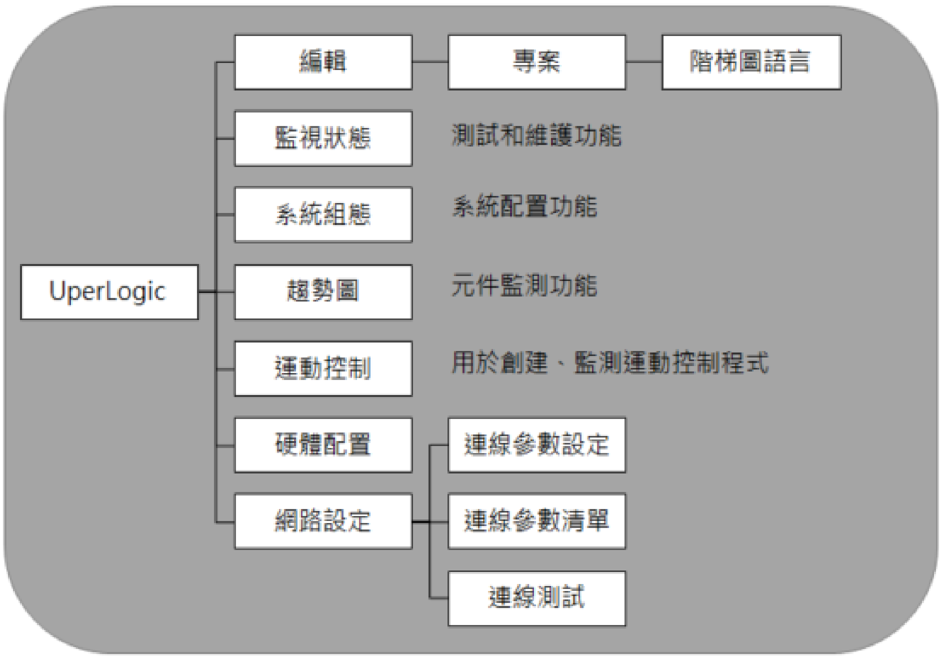


Image of UperLogic functions

5-1 Example Description

Through the TYPE-C USB connector or RJ-45 (Ethernet) connector, the M-series PLC module can be directly connected to the computer (recommended operating system: Windows 7 or above) with the UperLogic editing software installed.

5-2 Hardware Planning

5-3 Program Planning

5-4 Building Example Program

6

還需要加進手冊的內容