FATEK

M Series

Programmable Controller

Function/Function Block User Manual

NEXT Level *SOLUTION*

The contents of the manual will be revised as the version changes, and this version may not be the final version. Please go to http://www.fatek.com technical support area to download the latest version of the manual.

FATEK AUTOMATION CORP

INDEX

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 product, 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.

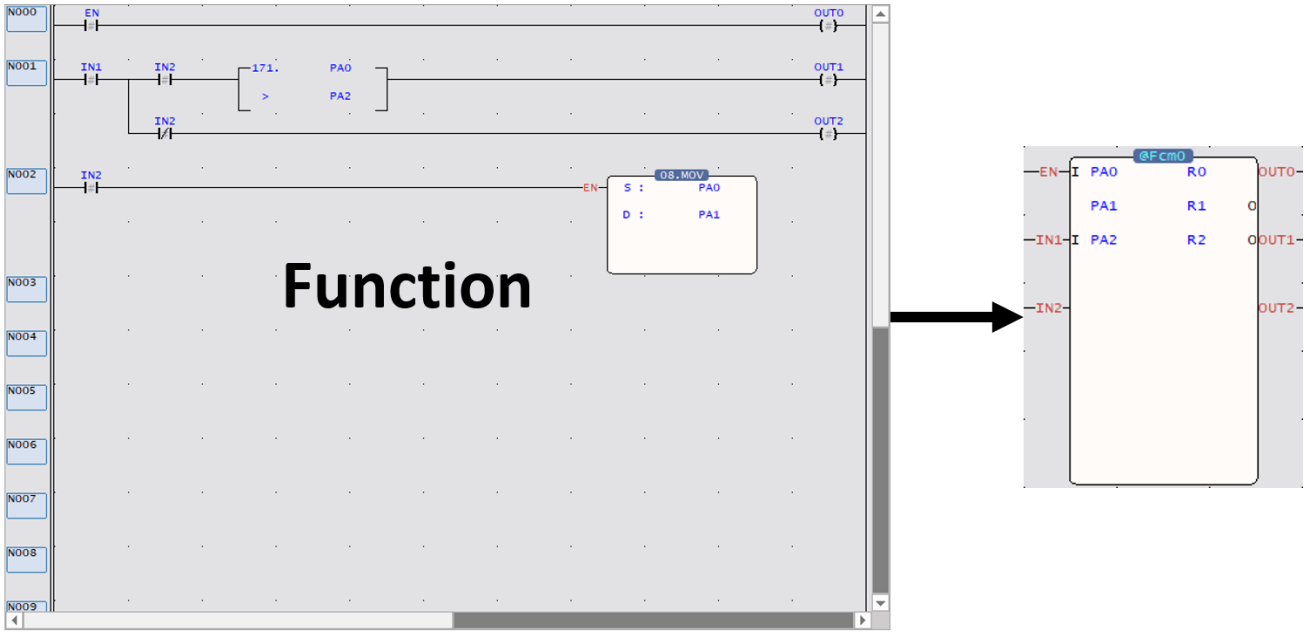
Change of specifications
The product specifications and accessories may be subject to change along with the technical improvement or other reasons. In the event that the published specifications or performance need to be changed or where significant structural change is required, FATEK will change the 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.

1

Advantage of Using Function/Function Block

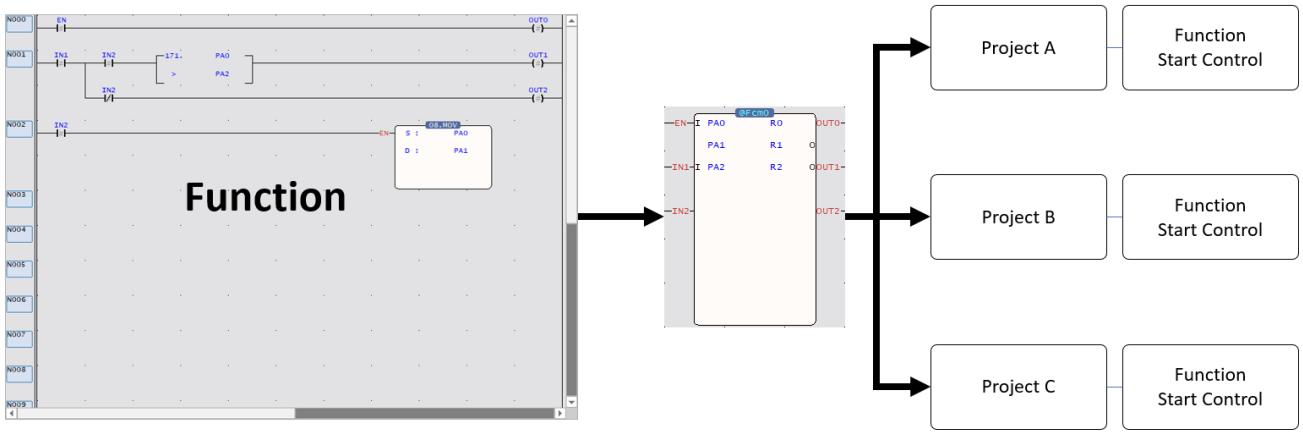
1-1 Easy To Read

Function/function blocks (using FCM) can wrap complex programs into Function instruction. PLC user only needs to understand the input and output actions of Function instruction, and don't need to delve into the program content of Function instruction.



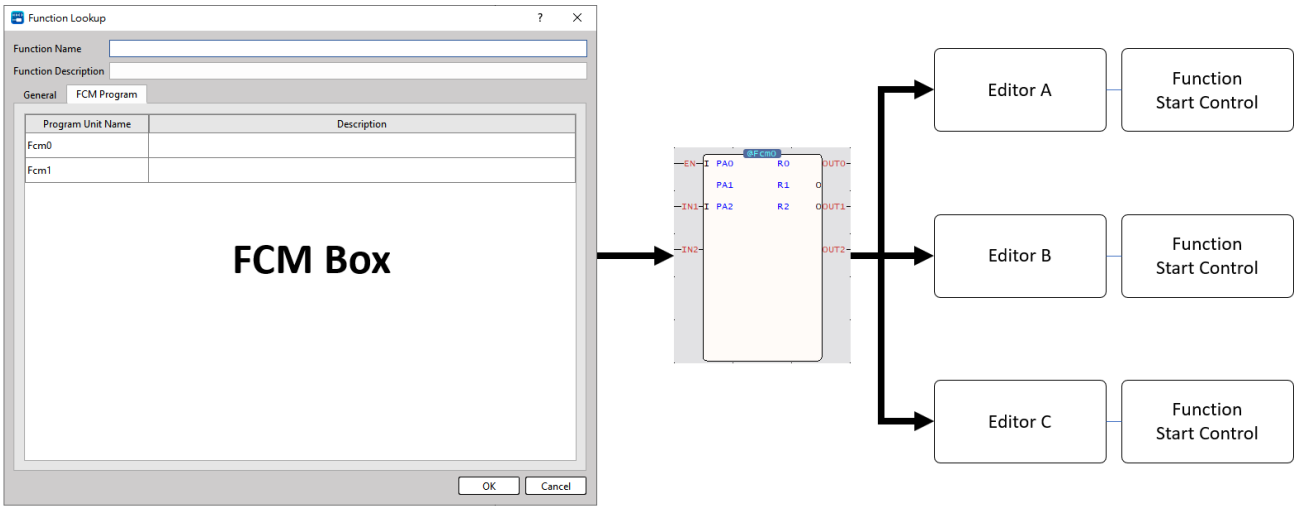
1-2 Multiple Use

Function/function blocks (using FCM) can wrap frequently used programs into functional instructions. Multiple use of Function instructions does not require renumbering the internal register of Function/Function Block. If the user accumulates a large number of Function/Function Block (using FCM) examples, they can be reused for different projects, speeding up the user's programming of the project.



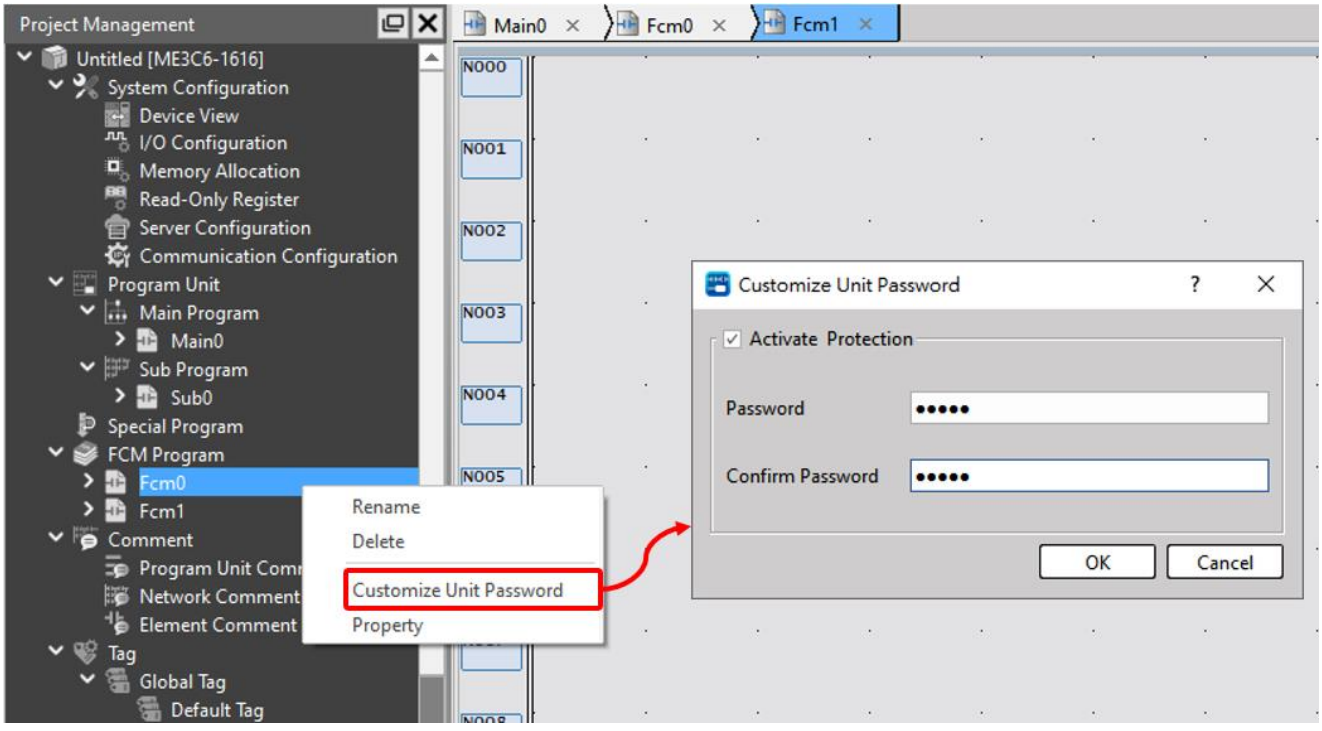
1-3 Improve Programming Quality

As mentioned earlier, Function/Function Block (using FCM) can be reused, Sharing the FCM program can reduce differences in program stability caused by different developers.



1-4 Program Protection

Setting a【Customize Unit Password】 to protect Function/Function Block (using FCM) can effectively protect Function/Function Block instructions from being arbitrarily viewed and edited by other users when sharing Function/Function Block instructions, , thus protecting the intellectual property of the program designer.



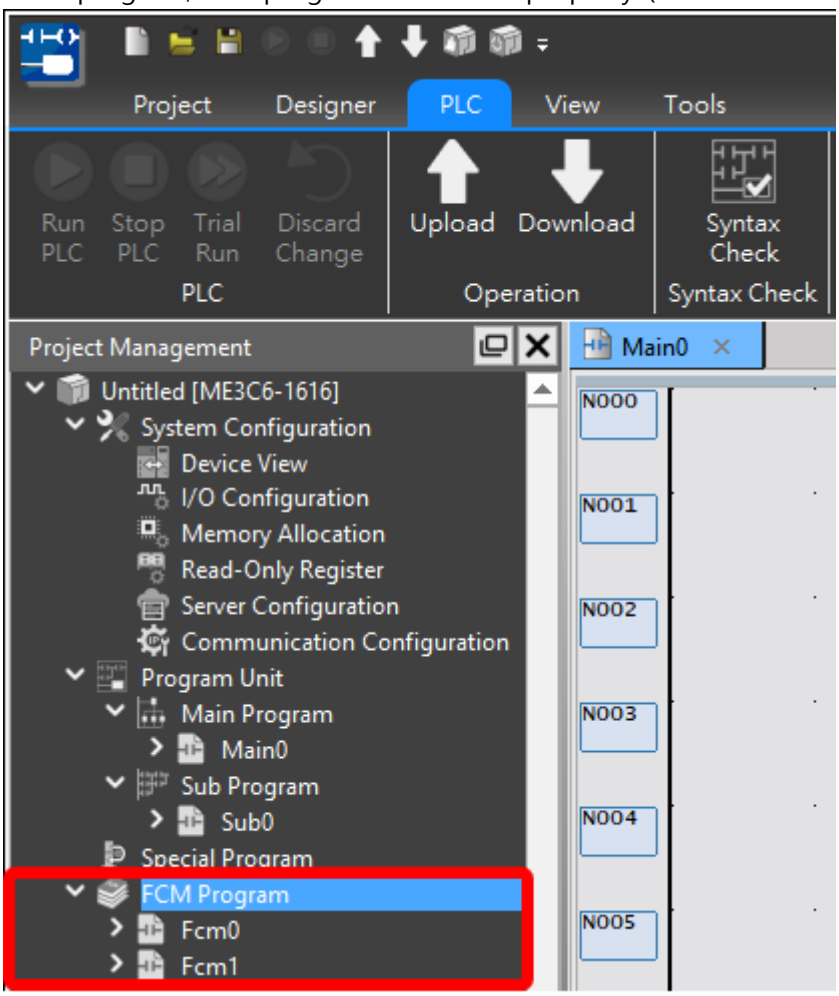
2

FCM Program

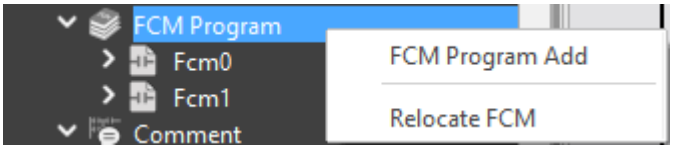
In this chapter, I will introduce how to use FCM programs in the Main program.

2-1 FCM Box

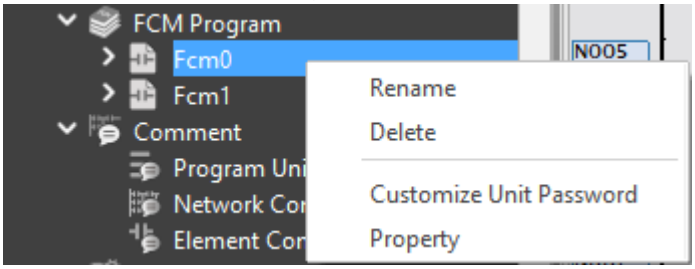
New FCM Program saved in 【Project Management】>【Program Unit】>【FCM Program】.Like Main program, FCM Program can relocate, rename and customize password. FCM program is different with other program, FCM program need to set property. (Please refer to Chapter 3-4 for FCM Property)



Use the right mouse button to click 【FCM Program】 to 【Add FCM Program】 or 【Relocate FCM】.



Use the right mouse button to click 【FCM Program Unit】to 【Rename】,【Delete】,【Customize Unit Password】,set【Property】.

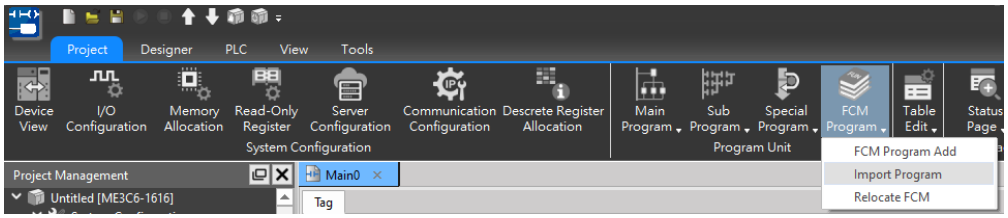


2-2 Import / Export FCM Program

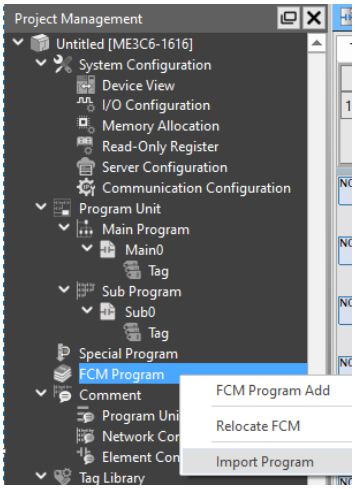
Through UperLogic, user can import or export FCM programs and reuse them in different projects.

- **Import:**

Click the function bar 【Project】 to 【Program Unit】to 【FCM Program】 to 【Import Program】 with the mouse;

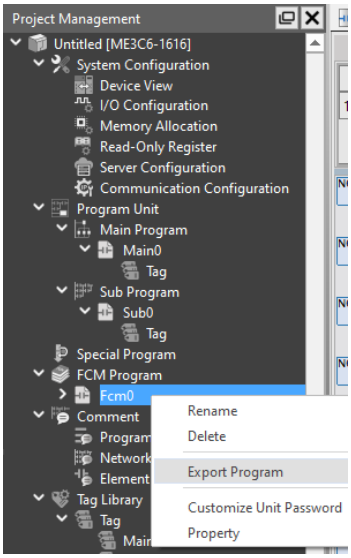


or right click 【FCM Program】in the project management window and select【Import Program】.



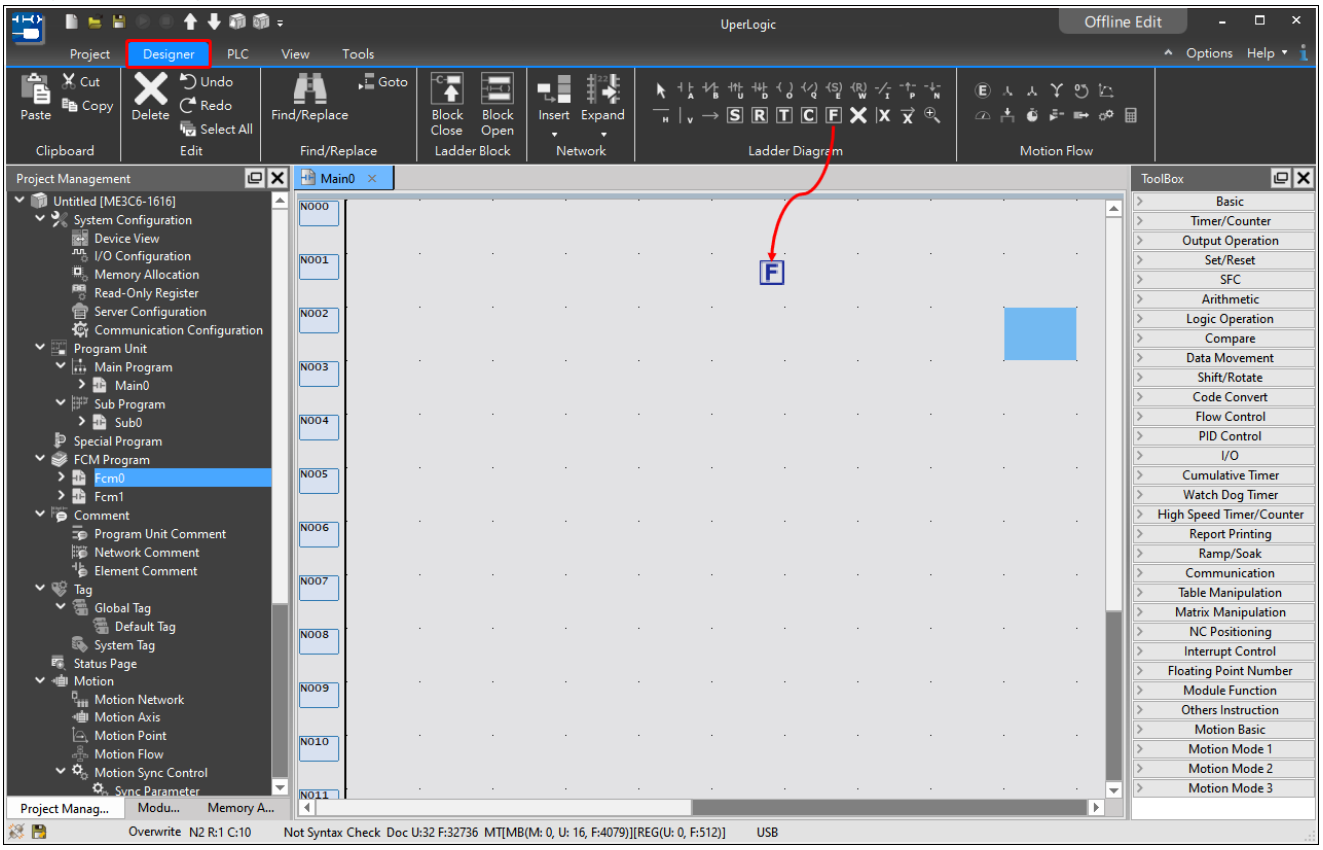
- **Export:**

Right click the program to be exported in project management window and select 【Export Program】.

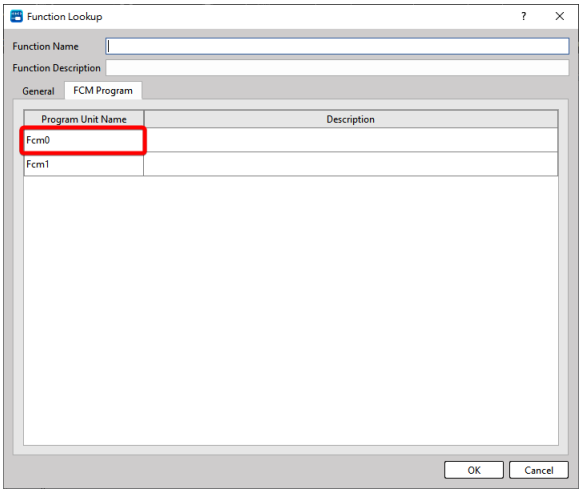


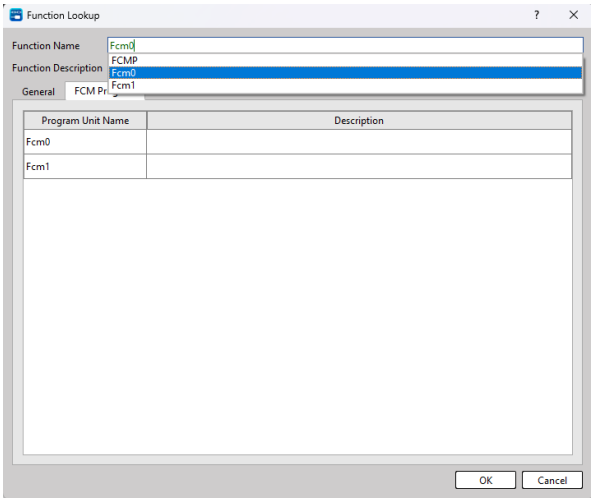
2-3 Using Function/Function Block Method

Step 1. Use the【Designer】>【Ladder Diagram】>Button【F】to add FCM program unit into the Main program editing screen, or use the hotkey "F" to call【Function】libraries.

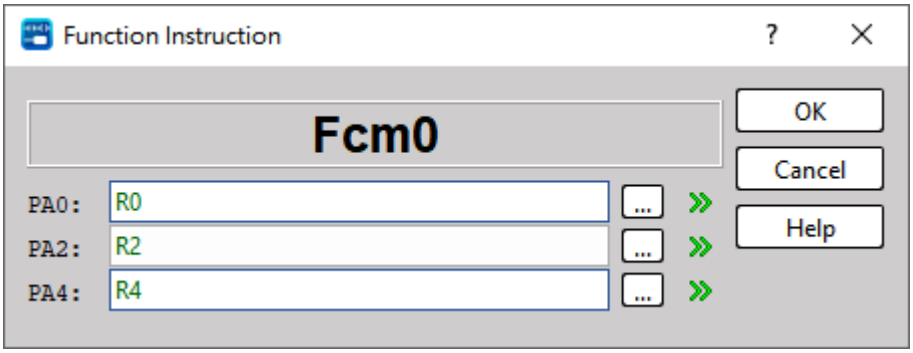


Step 2. pop up【Function Lookup】 screen > Switch to FCM program page > Select the FCM program to add to the Main program (There are 2 FCM programs in the picture below), or directly enter the FCM program name in the Function Name field.

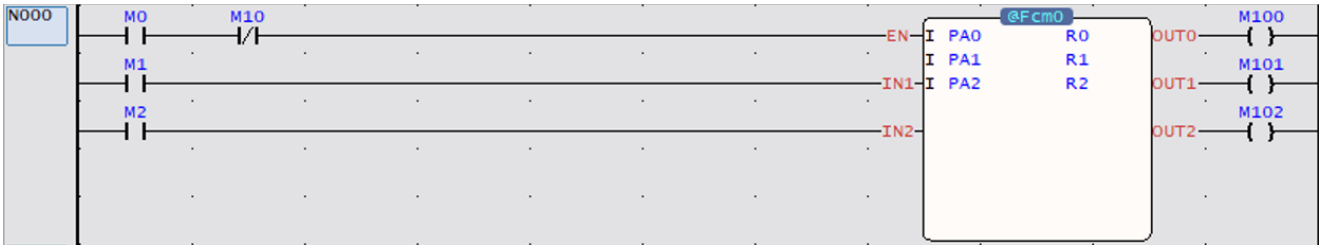




Step 3. After selecting the FCM program, UperLogic pops up the FCM function instruction setting window. Just like the general Function instruction, fill in the register or constant in the parameters.



Step 3. The FCM program is same as the general Function instruction, using objects such as A contact and B contact to control input pins and monitor output pins.



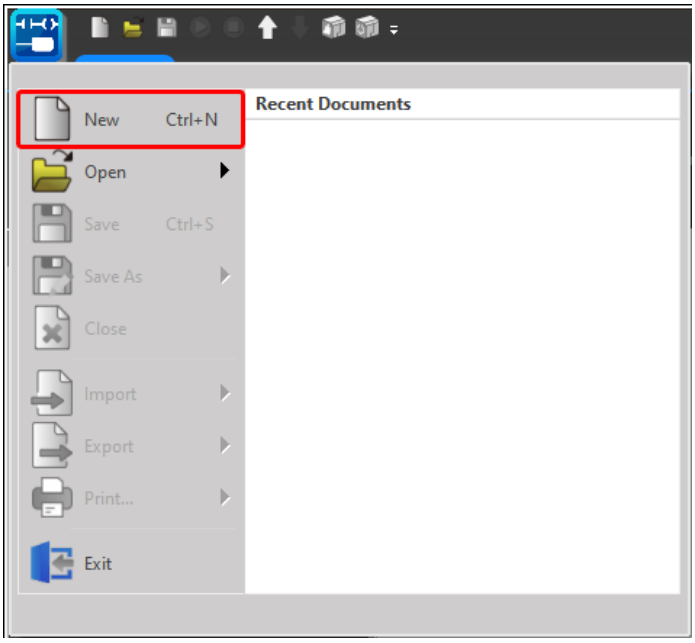
3

Function/Function Block Program

In this chapter, it will introduce how to create a FCM program in UperLoigc, set FCM parameters and properties.

3-1 Create a new Project

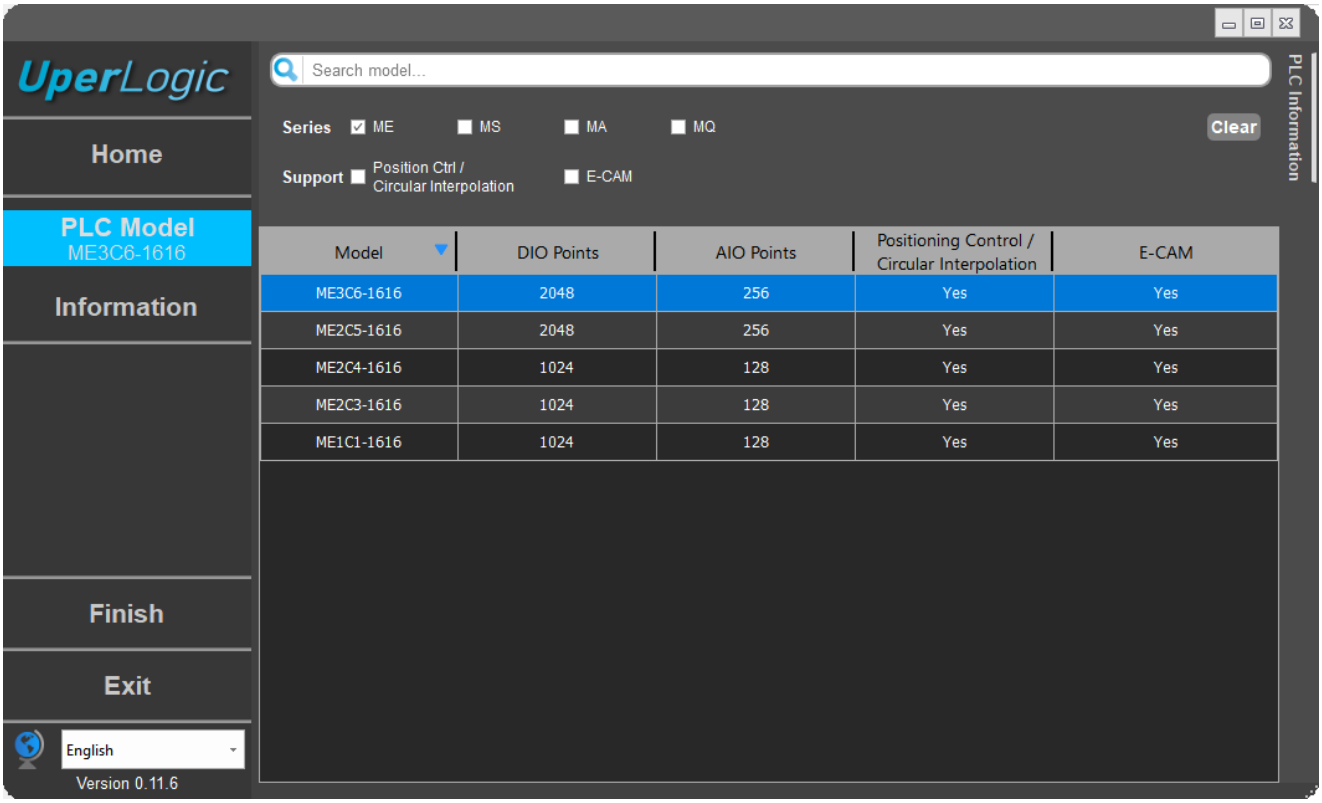
Step 1. Select the【Menu】in the upper left corner of UperLogic. >【New】



Step 2. The PLC module selection window pops up

① Select product Series (Select the ME series in the picture below)

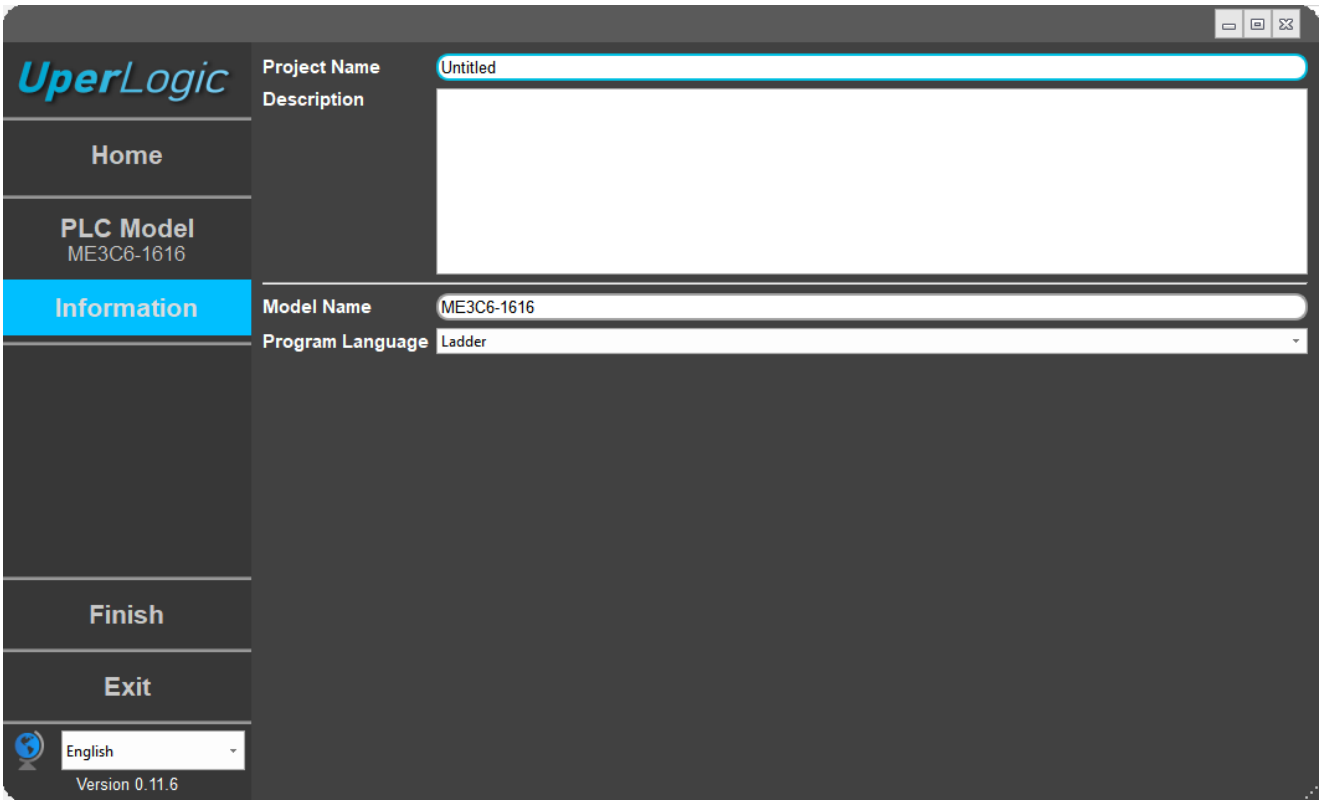
② Select Model (Select ME3C6-1616 in the picture below)


UperLogic
Home
PLC Model
ME3C6-1616
Information
Finish
Exit
English
Version 0.11.6
Search model...
Series
ME
MS
MA
MQ
Clear
Support
Position Ctrl / Circular Interpolation
E-CAM
PLC Information
Model
DIO Points
AIO Points
Positioning Control / Circular Interpolation
E-CAM
ME3C6-1616 2048 256 Yes Yes
ME2C5-1616 2048 256 Yes Yes
ME2C4-1616 1024 128 Yes Yes
ME2C3-1616 1024 128 Yes Yes
ME1C1-1616 1024 128 Yes Yes

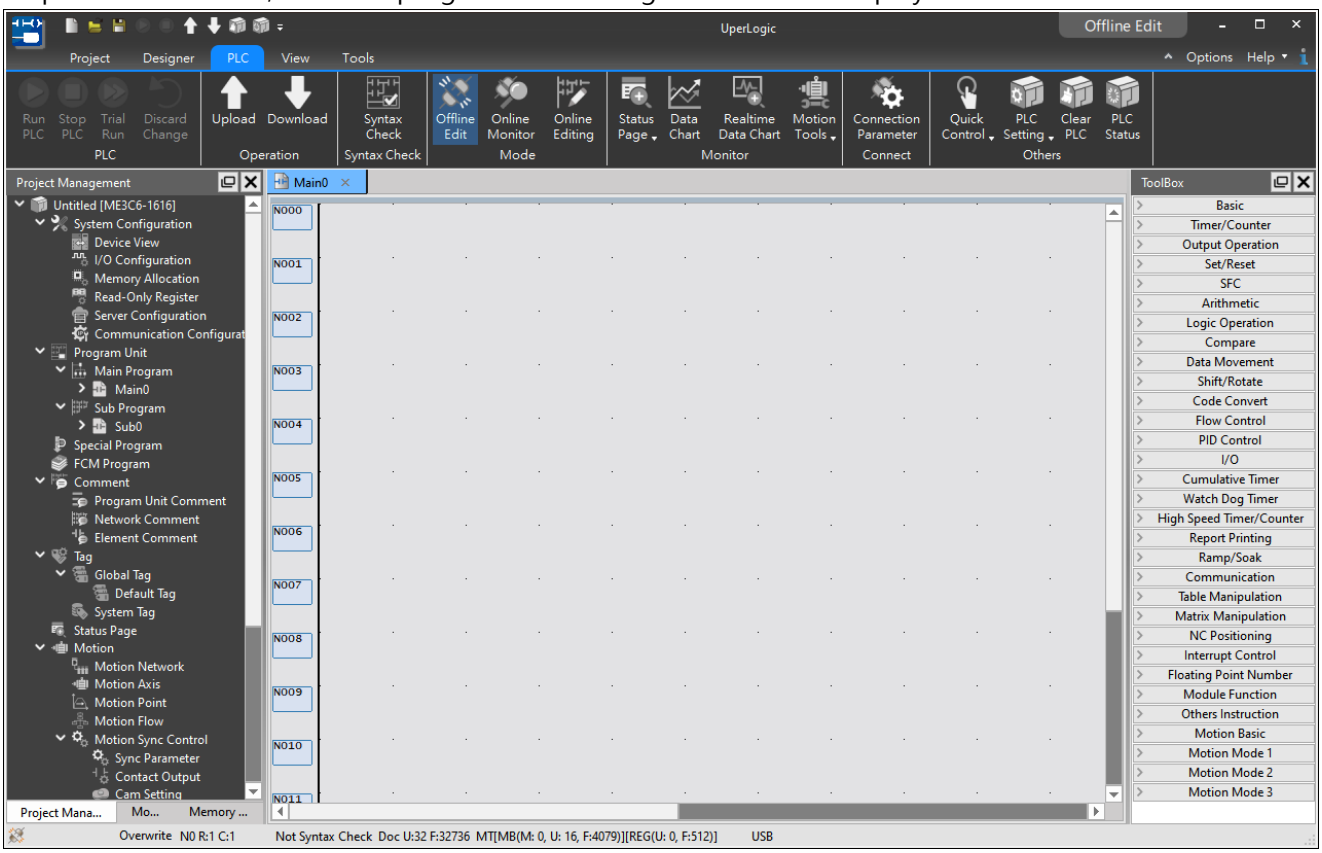
Step 3. Set project information

③ Select programming language (Select the ladder in the picture below)

① Finish

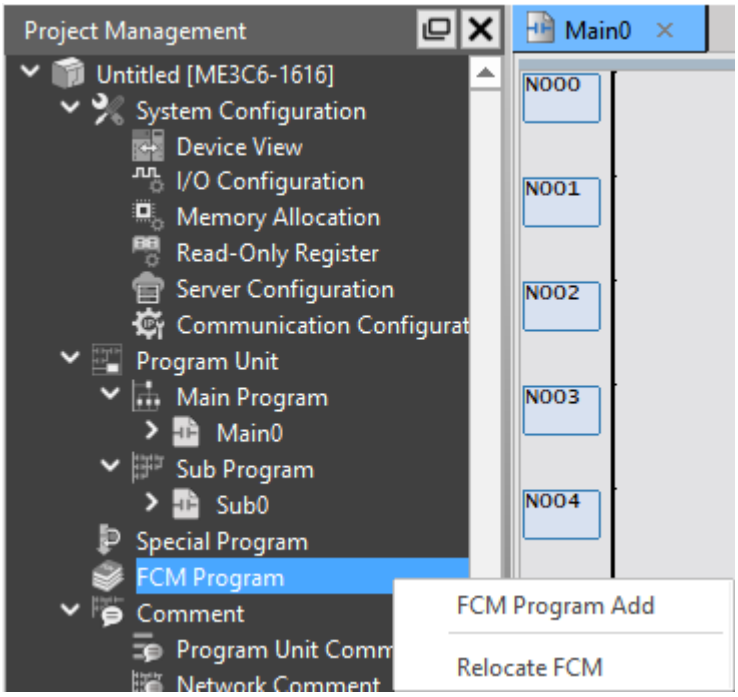


Step 4. After finish, the main program unit editing screen will be displayed.

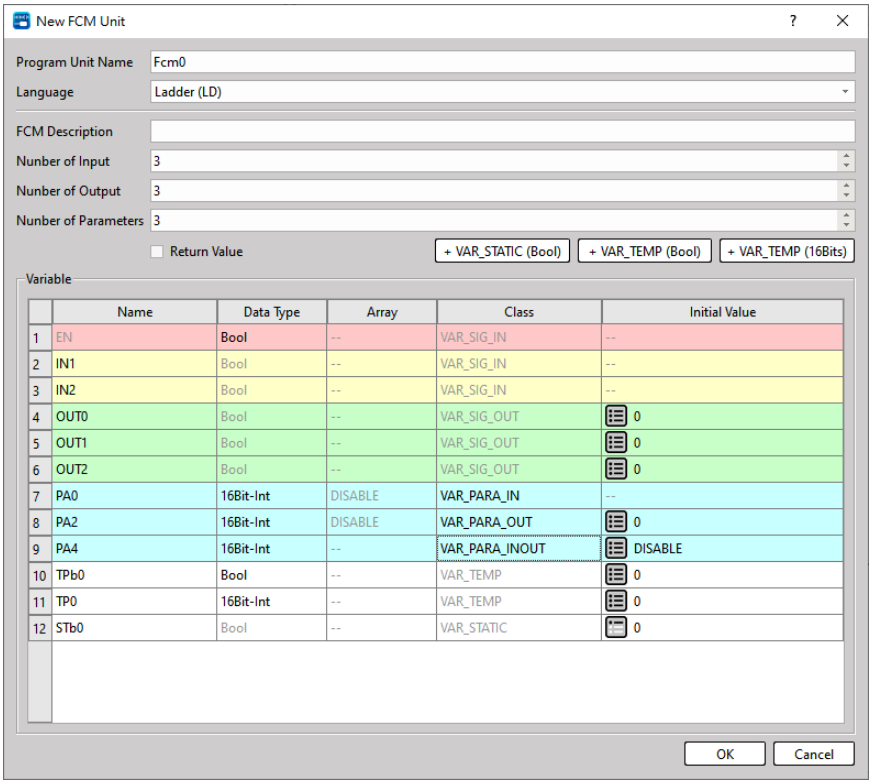


3-2 Add A New FCM Program

Step 1. Select【Project Management】in the Uperlogic>【Program Unit】>【FCM Program】>Right click>【FCM Program Add】



Step 2. Pops up【FCM Property】setting windows > Set【FCM Property】> After the setting is completed, press [OK] to complete the new FCM program. (Please refer to Chapter 3-4 for FCM properties.)



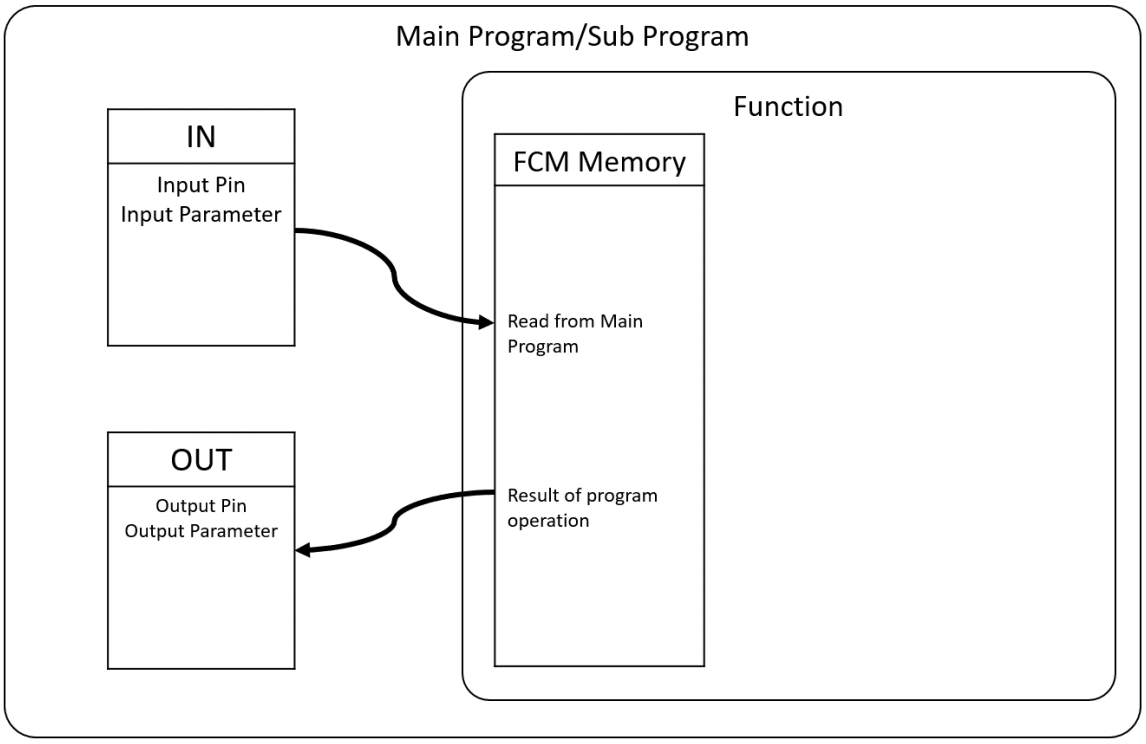
3-3 FCM Variable

FCM has independent internal relays and internal registers. Internal variables can only be used in FCM program. Some internal relays and internal registers can exchange data with external program unit registers. These are local variables, only valid within the specific FCM. Each time the FCM is executed, it has its own independent variable space, ensuring that the function block can be reused without affecting other logic.

[Note] **When encapsulating the FCM function block, it is important to pay attention to the use of global register types, such as R/D/M. Avoid misuse and accessing duplicate register locations in other programs during calls, as this may cause logic malfunctions. To fully encapsulate the FCM, only the following types of variables must be used within the FCM program.**

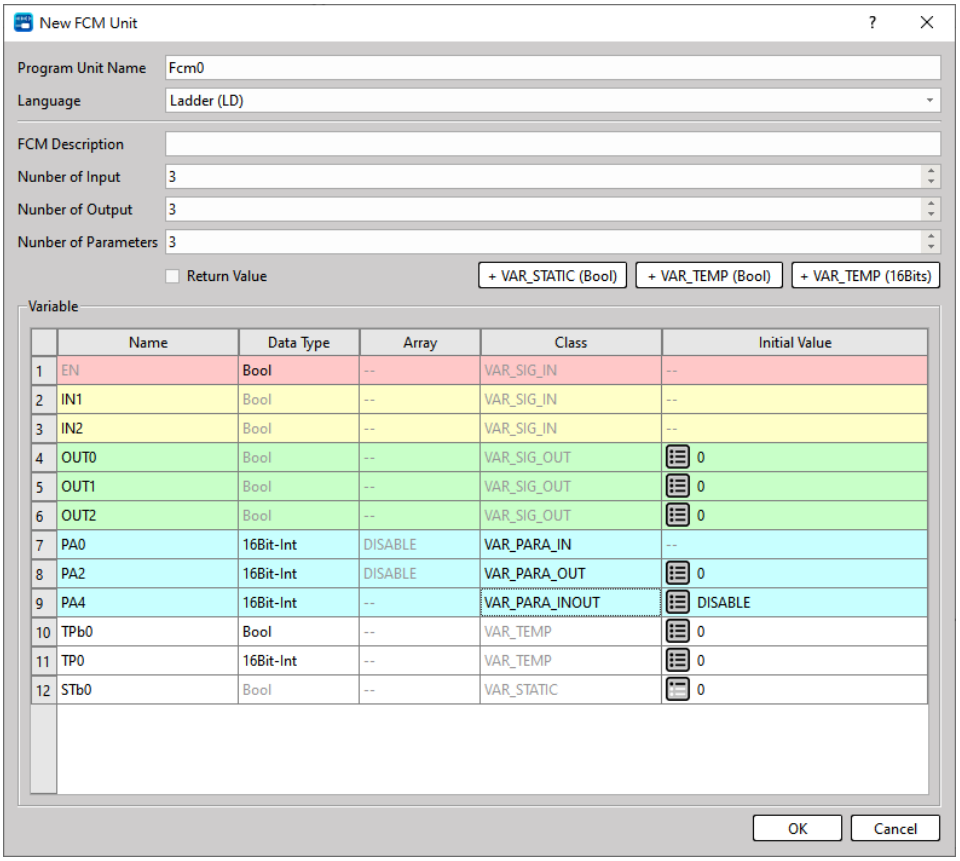
FCM Variable:

ITEM		
Bit	VAR_SIG_IN	Input Pin
	VAR_SIG_OUT	Output Pin
	VAR_TEMP	Internal Variable Temp Bit
	VAR_STATIC	Internal Variable Static Bit
WORD	VAR_PARA_IN	Input Parameter
	VAR_PARA_OUT	Output Parameter
	VAR_PARA_INOUT	Input /Output Parameter / internal Variable Static Word
	VAR_TEMP	internal Variable Temp Word


Main Program/Sub Program
Function
IN
Input Pin
Input Parameter
FCM Memory
Read from Main Program
OUT
Output Pin
Output Parameter
Result of program operation

3-4 FCM Property

FCM Property



- Program Unit Name : Function/Function Block Program Name。(English and Number Only)
- Language: Editing Language of Function/Function Block Program, e.g., LD(Ladder),ST, etc.。
- Number of Input : Number of Function/Function Block Program Input Pins。
- Number of output: Number of Function/Function Block Program Output Pins。
- Number of Parameter : Number of Function/Function Block Program Parameters。
- +VAR_STATIC: Add Internal Static Variable
- +VAR_TEMP: Add Internal Temp Variable
- Variable : Set the Name, Data Type, Array, Class, Initial Value of the variable.

3-4-1 Variable Name

All variables can be modified to have names that are a mixture of English and numbers, except for EN, which is used as the Function instruction trigger pin, so it has a fixed name. Use different background colors to distinguish different types of variables.

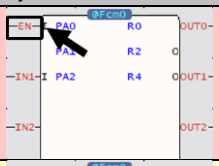
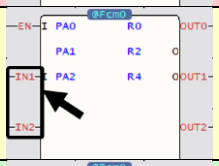
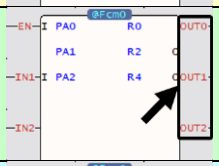
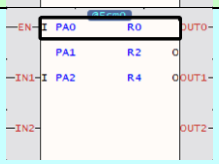
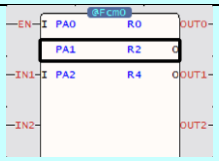
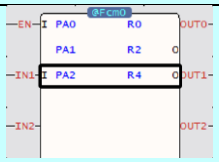
Variable Name

Variable	Length	Color of Background
EN	20 words by English letters and numbers	Red
Input	20 words by English letters and numbers	Yellow
Output	20 words by English letters and numbers	Green
Parameter	20 words by English letters and numbers	Blue
STATIC Variable	20 words by English letters and numbers	White
TEMP Variable	20 words by English letters and numbers	White

3-4-2 Variable Class

These are local variables, only valid within the specific FCM. Each time the function block is executed, it has its own independent variable space, ensuring that the FCM can be reused without affecting other logic.

Variable Class

Variable	Class	Description	Symbol
EN	VAR_SIG_IN	The pin that triggers FCM execution in the program unit. When FCM is executed, it receives the bool status from the program unit.	
Input	VAR_SIG_IN	When FCM is executed, it receives the bool status from the program unit.	
Output	VAR_SIG_OUT	Output the Bool status of the FCM operation result to the program unit	
Parameter	VAR_PARA_IN	When FCM is executed, it receives the value from the program unit.	 * Mark【 I 】means IN type
	VAR_PARA_OUT	Output the value of the FCM operation result to the program unit	 *Mark【O】means OUT type
	VAR_PARA_INOUT	When FCM is executed, it first receives the value from the program unit. When the FCM operation is completed, Output the value of the FCM operation result to the program unit	 * Mark【 I 】means IN type *Mark【O】means OUT type
STATIC Variable	VAR_STATIC	The life cycle of VAR_STATIC lasts from the moment the FCM is called, which means the result from the last call will remain and be available in the next call.	
TEMP Variable	VAR_TEMP	The life cycle of VAR_TEMP ends at the time the FCM returns.	

[Note] When encapsulating the FCM function block, it is important to pay attention to the use of global register types, such as R/D/M. Avoid misuse and operations that access duplicate register locations in other programs during calls, as this may cause logic malfunctions. To fully encapsulate the FCM, only the aforementioned types of variables must be used within the FCM program.

STATIC BIT:

- The life cycle of VAR_STATIC lasts from the moment the FCM is called, which means the result from the last call will remain and be available in the next call.
- Static bit is only initialized once at the first call of the FCM

VAR_STATIC BIT : VAR_STATIC (Bool)

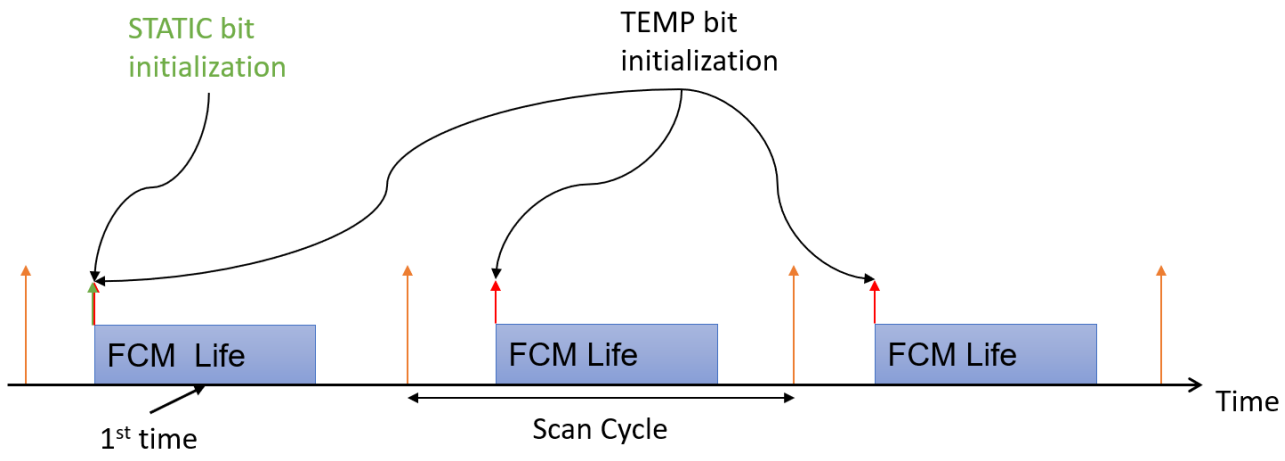
VAR_ STATIC WORD : VAR_PARA_INOUT

TEMP BIT:

- The life cycle of VAR_TEMP ends at the time the FCM returns.
- Temp bit is re-initialized at each cycle

VAR_TEMP BIT : VAR_TEMP (Bool)

VAR_TEMP WORD : VAR_TEMP (16Bit-Int, 16Bit-Uint, 32Bit-Int, 32Bit-Uint, Float)



STATIC BIT & TEMP BIT initialization timing diagram

3-4-3 Variable Data Type

Variable Data Type

Variable	Data Type	Description
EN	Bool	High Level Trigger
	Bool, Pulse Rising	Rising Edge Trigger
Input	Bool	
Output	Bool	
Parameter	16Bit-Int, 16Bit-Uint, 32Bit-Int, 32Bit-Uint, Float	
	Pointer	Class must be VAR_PARA_INOUT
STATIC Variable	Bool	
TEMP Variable	Bool, 16Bit-Int, 16Bit-Uint, 32Bit-Int, 32Bit-Uint, Float	

Trigger type of EN

High Level Trigger **(Bool)**: Which runs a particular FCM when EN is ON.

Rising Edge Trigger **(Bool, Pulse Rising):** Which runs a particular FCM at the moment when EN changes the status from OFF to ON.

Edge trigger function is more efficient than Level trigger function because Edge trigger function is only executed at the rising edge of EN signal. This can significantly reduce system loading.

Example: In chapters 3-5, If the user wants to calculate the water level depth, the rising edge only needs to trigger the execution of the Container function once. If high level trigger, the Container function will be executed once in each scan cycle.

Pointer parameter vs Regular parameter

【Pointer parameter】 VAR_PARA_INOUT (Pointer)

1. The memory **address** of the parameter is passed into the FCM. (Call by Reference)
2. Pointer parameter doesn't need to have an initial value as FCM always reads value from the memory address.
3. the FCM program changes the value of parameter directly.

*Only when the parameter is used as VAR_PARA_INOUT can select the【Pointer parameter】.

【Regular parameter】 VAR_PARA_INOUT (16Bit-Int, 16Bit-Uint, 32Bit-Int, 32Bit-Uint, Float)

1. Only the value of parameter is copied to the temp memory of FCM. (Call by Value)
2. Regular parameter can be initialized either with an initial value or the value passed from main program
3. The FCM program changes the value passed from the parameter doesn’ t affect the value of parameter. (Ex. If R0 is regular parameter, R0 value is not changed during the FCM life.)

【Multi-cycle instruction】

Multi-cycle instructions run in the background and don't end with scan cycle.
INOUT type parameter will then get unexpected value from unmanaged memory. So multi-cycle instructions must use 【Pointer parameter】to run.

*Please refer to Appendix 1 Multi-cycle Instruction

3-4-4 Variable Array

Array serves as a way to pass a list of registers into and out of the FCM memory.

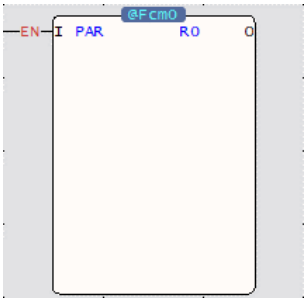
Array

Variable	Class	Array	Description
Number of Parameter	VAR_PARA_INOUT	PAR [0...199]	Maximum length is 200.

*Array INOUT parameter behavior is the same as Pointer INOUT parameter (Please refer to 3-4-3 Variable Data Type)

Example:

Set a VAR_PARA_INOUT parameter and set 5 lengths of Array. Fill in R0 in the FCM instruction PAR.



	Name	Data Type	Array	Class	Initial Value
1	EN	Bool	--	VAR_SIG_IN	--
2	PAR0	16Bit-Int	PAR0[0..4]	VAR_PARA_INOUT	DISABLE

When EN triggers the FCM Function Instruction, pass ARRAY into the FCM Function Instruction

Main Unit		Function Instruction
R0	→	PAR[0]
R1	→	PAR[1]
R2	→	PAR[2]
R3	→	PAR[3]
R4	→	PAR[4]

When the FCM Function Instruction operation is completed, ARRAY is passed out of the FCM Function Instruction.

Main Unit		Function Instruction
R0	←	PAR[0]
R1	←	PAR[1]
R2	←	PAR[2]
R3	←	PAR[3]
R4	←	PAR[4]

3-4-5 Variable Initial Value

Setting the initial value of FCM variables

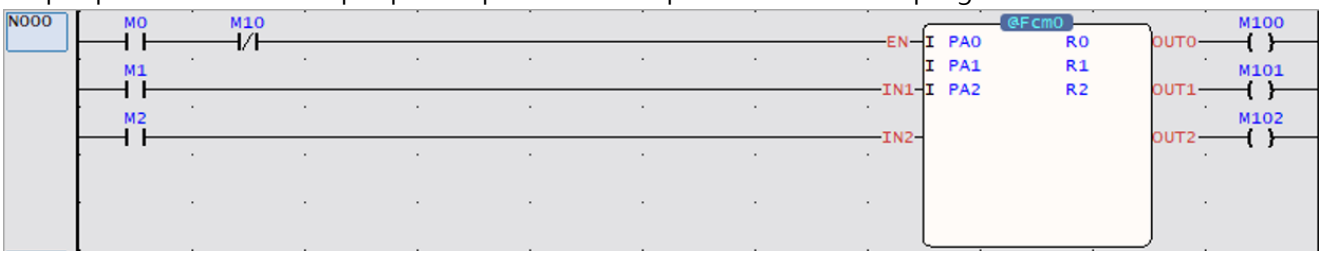
- IN parameters don’ t need to have an initial value as the main program always pass a value to IN parameter (Register or constant)
- OUT parameters has to be initialized with an initial value
- INOUT parameters can be initialized either with an initial value or with a value passed from main program
- Static variables are only initialized with initial values when FCM is executed for the first time.

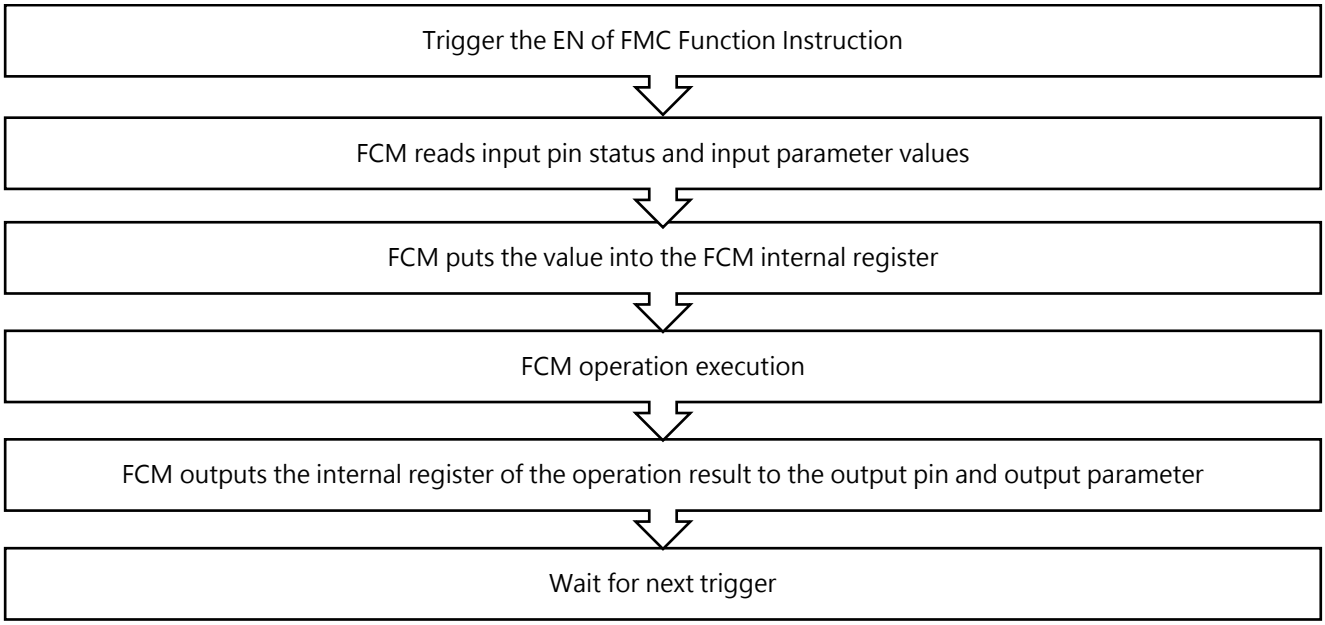
Variable Initial Value

Variable	Class	symbol	Value
EN	VAR_SIG_IN	--	
Input	VAR_SIG_IN	--	
Output	VAR_SIG_OUT	Initialized every time when FCM is executed	Determine the data type of the initial value based on the data type of the variable
Parameter	VAR_PARA_IN	--	
	VAR_PARA_OUT	Initialized every time when FCM is executed	Determine the data type of the initial value based on the data type of the variable
	VAR_PARA_INOUT	Initialized every time when FCM is executed	Determine the data type of the initial value based on the data type of the variable or turn off the initialization function and initialize it with the value passed by the main program unit. *Fill in the blank value to indicate [DISABLE], without initialization.
STATIC Variable	VAR_STATIC	Only initialized the first time when FCM is executed	Determine the data type of the initial value based on the data type of the variable
TEMP Variable	VAR_TEMP	Initialized every time when FCM is executed	Determine the data type of the initial value based on the data type of the variable

3-5 Function/Function Block Instruction (Using FCM)

F The FCM instruction is the same as the general Function instruction. Trigger the EN pin by different conditions to execute the FCM instruction. The FCM instruction will execute the ladder program of the FCM based on the input pin status and input parameter value, and then put the operation results into the output parameters and output pins to provide subsequent control to the program unit.

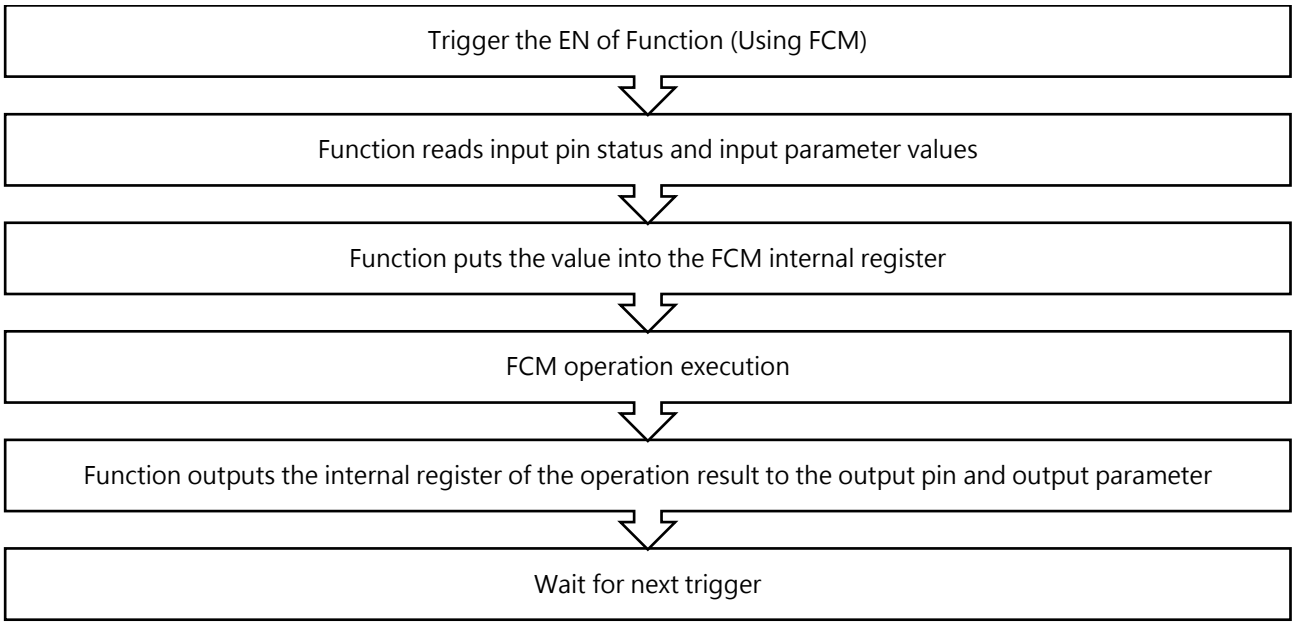




3-5-1 Function Block vs Function

Function

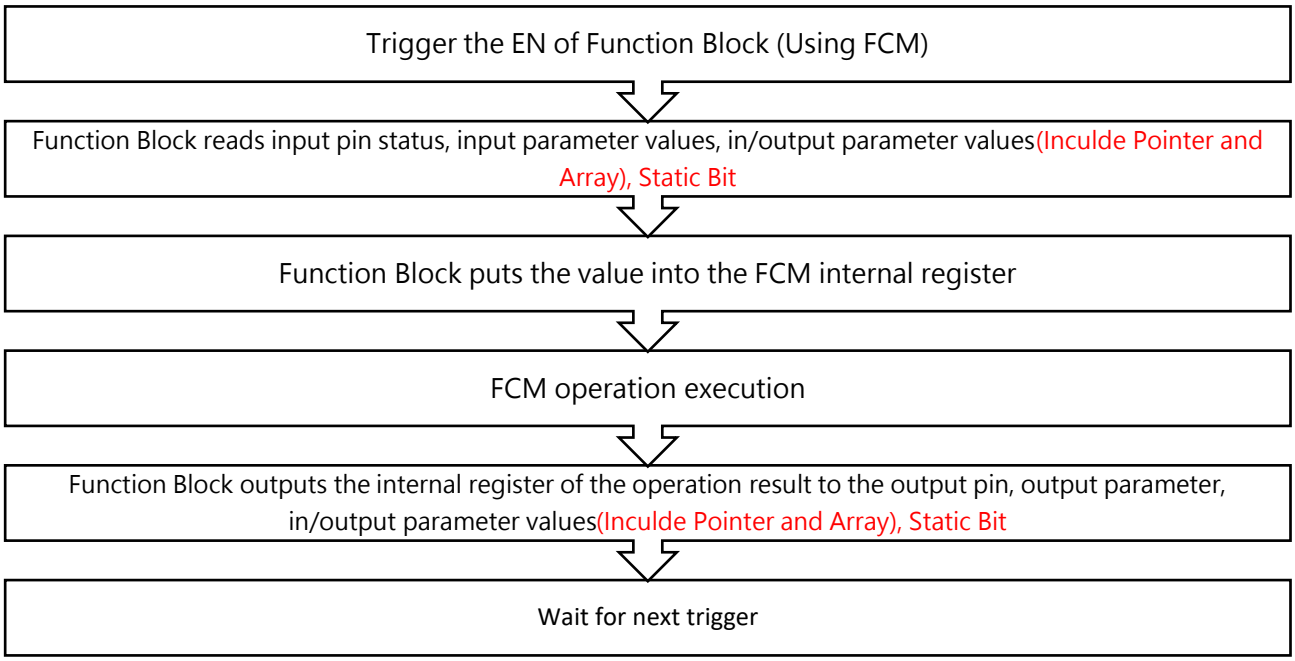
Functions are code blocks or subroutines without dedicated memory. Each starts from the initial value.



Function Block

Function block are code blocks that store their values permanently in instance code blocks, so they remain available after the block has been executed. The next execution can continue with the previous results.

* Please refer to Chapter 3-4-2



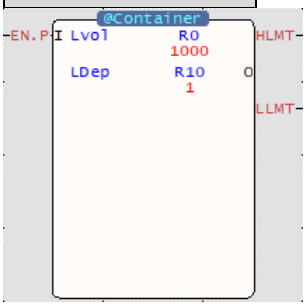
3-6 Function Program Sample

3-6-1 Water Container

Application

A cube-like water container with fixed inside length and width. Make a function which be able to calculate the depth of water according to the water volume parameter. If the volume is over 6000ml, high limit alarm is ON. If the volume is less than 1000ml, low limit alarm is ON.

Function Instruction

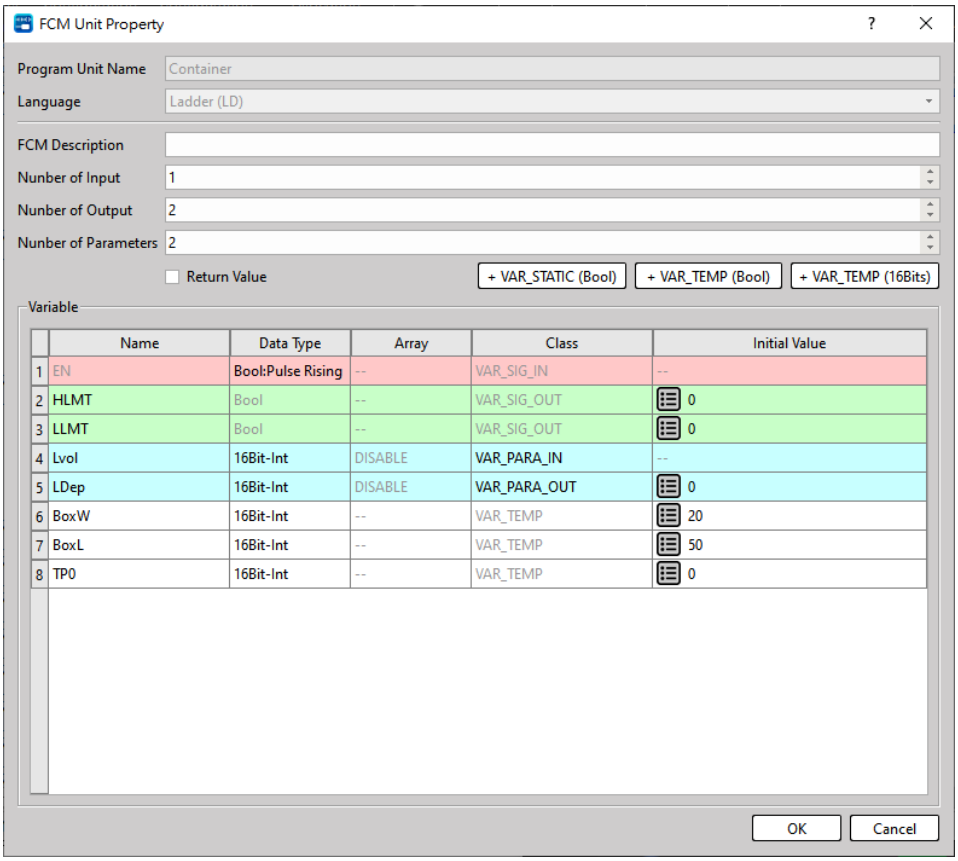


Input/Output Pins and Parameters

Parameter	Name	Description
Input Pins	EN.P	Rising edge trigger
Output Pins	HLMT	High limit alert
	LLMT	Low limit alarm
Input Parameters	Lvol	Water Level volume
Output Parameters	LDep	Water Level depth

Function Property

1 number of input (input pin), 2 numbers of output (output pin), 2 numbers of parameter, 3 VAR_TEMPs.

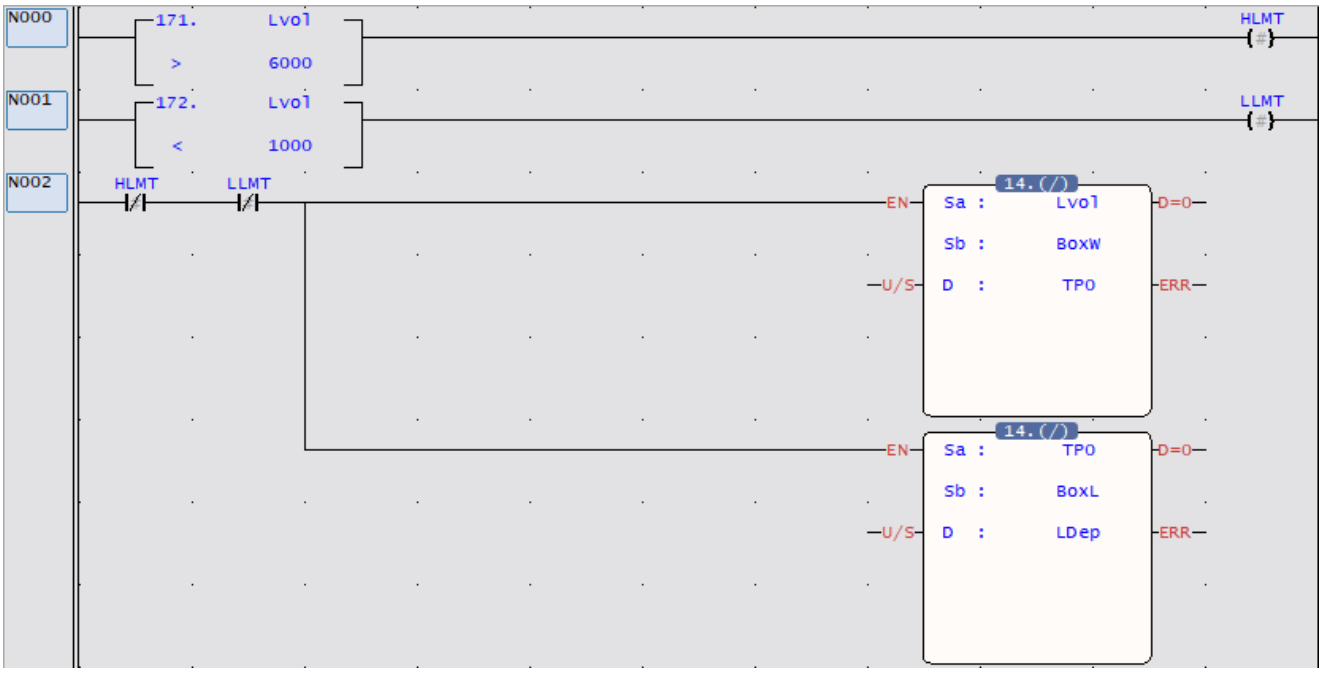


Varialbe

Name	Data Type	Array	Class	Initial Value	Description
EN.P	Bool; Pulse Rising	--	VAR_SIG_IN	--	Rising edge trigger
HLMT	Bool	--	VAR_SIG_OUT	Initialized to 0 every time	High limit alert
LLMT	Bool	--	VAR_SIG_OUT	Initialized to 0 every time	Low limit alarm
Lvol	16Bit-Int	DISABLE	VAR_PARA_IN	--	Water Level volume
LDep	16Bit-Int	DISABLE	VAR_PARA_OUT	Initialized to 0 every time	Water Level depth
BoxW	16Bit-Int	--	VAR_TEMP	Initialized to 20 every time	The width of the box is fixed at 20 units
BoxL	16Bit-Int	--	VAR_TEMP	Initialized to 50 every time	The length of the box is fixed at 50 units
TP0	16Bit-Int	--	VAR_TEMP	Initialized to 0 every time	For calculation

Function Internal Program

【Container】



N000 Use Fun171 to compare the input Lvol (water level volume) and trigger HLMT (output upper limit alarm) when it exceeds 6000 units.

N001 Use Fun172 to compare the input Lvol (water level volume) and trigger the LLMT (output lower limit alarm) when it is lower than 1000 units.

N002 If the input Lvol (water level volume) does not exceed HLMT (high limit alarm) and is lower than LLMT (lower limit alarm), divide Lvol (water level volume) by the width and length of the container to calculate LDep (water level depth), and output it to the outside.

Function Internal Program in ST Language

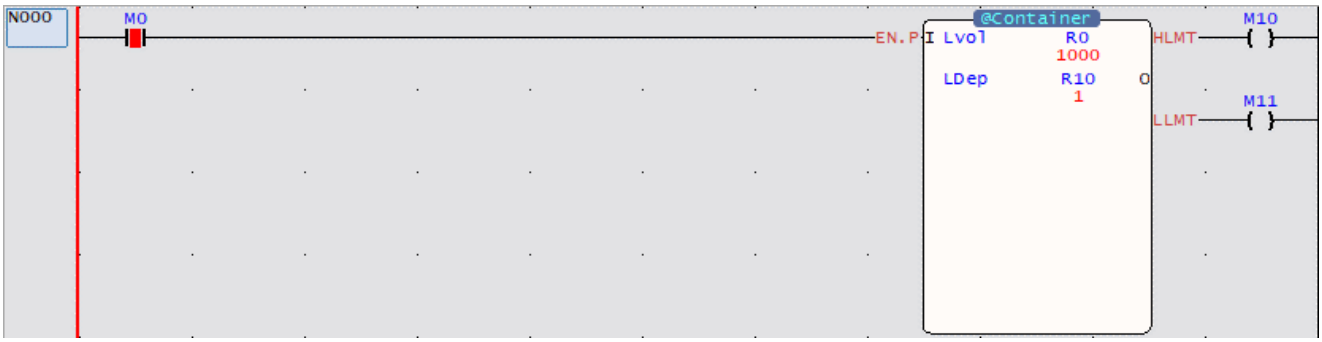
```
IF Lvol > 6000 THEN
    HLMT := TRUE;
END_IF

IF Lvol < 1000 THEN
    LLMT := TRUE;
END_IF

IF NOT HLMT AND NOT LLMT THEN
    TP0:= Lvol / BoxW;
    LDep:= TP0 / BoxL;
END_IF
```

behavioral description

【Container】



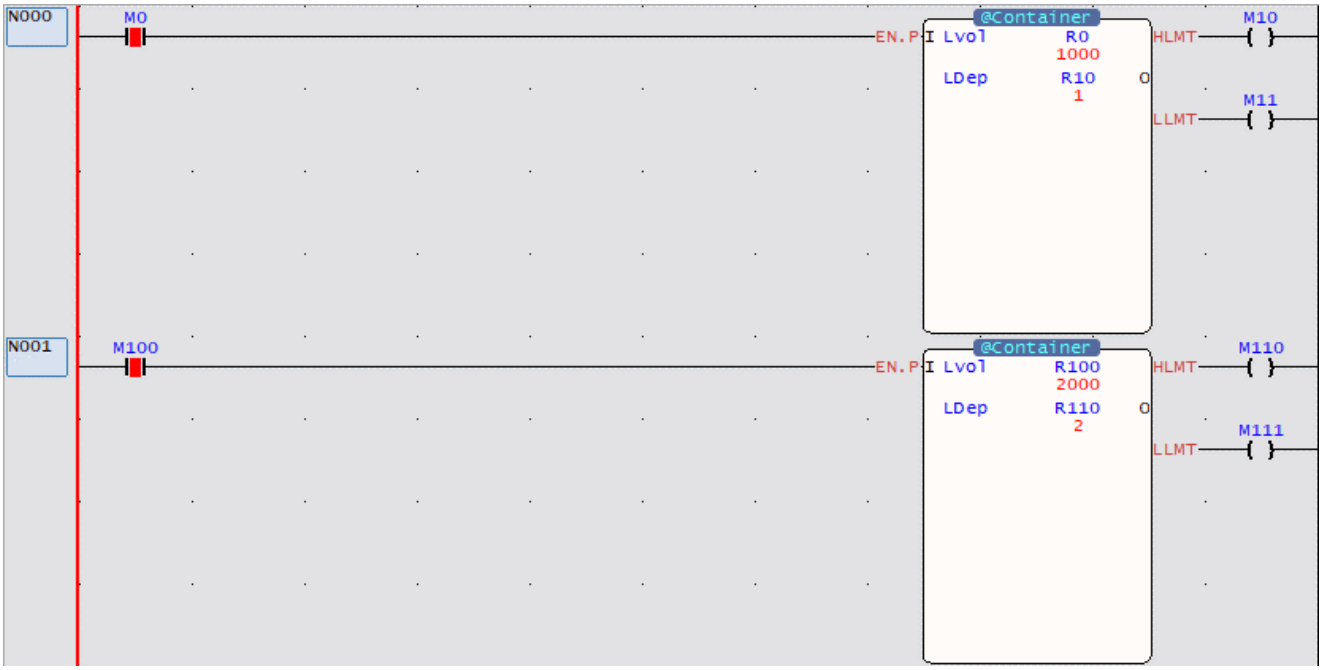
Step 1. Enter Lvol (water level volume) in R0

Step 2. Trigger EN.P of the Function instruction with M0 (Rising edge trigger)

Step 3. R10 will display the calculated LDep (water level depth) or High and low limit alarm (HLMT/LLMT) output alarm

Multiple use

Multiple use of Function instructions in the main program



N000 Input 1000 units of water volume and output 1 unit of water depth.

N001 Input 2000 units of water volume and output 2 units of water depth.

The result of 2 FCMs not interfering with each other.

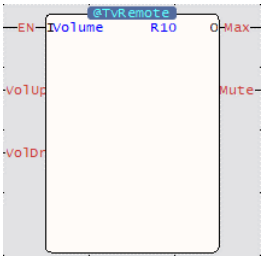
3-7 Function Block Program Sample

3-7-1 TV Remote

Application

A TV remote controls the sound volume of TV and can be adjusted based on its previous volume. The volume value ranges from 0~10.

Function Block Instruction

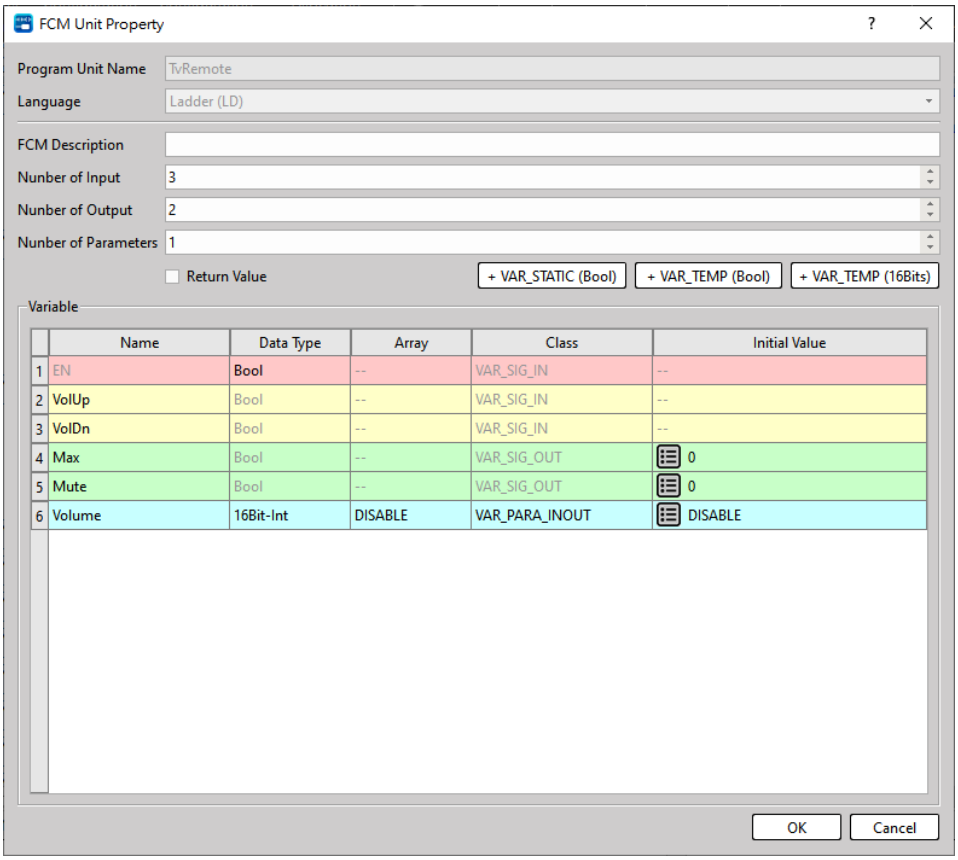


Input/Output Pins and Parameters

Parameter	Name	Description
Input Pins	EN	High Level Trigger
	VolUp	Volume Up
	VolDn	Volume Down
Output Pins	Max	Volume is Max
	Mute	Volume is Mute
Input/Output Parameters	Volume	Volume

Function Block Property

3 numbers of input (input pins), 2 numbers of output (output pins), and 1 number of parameter.



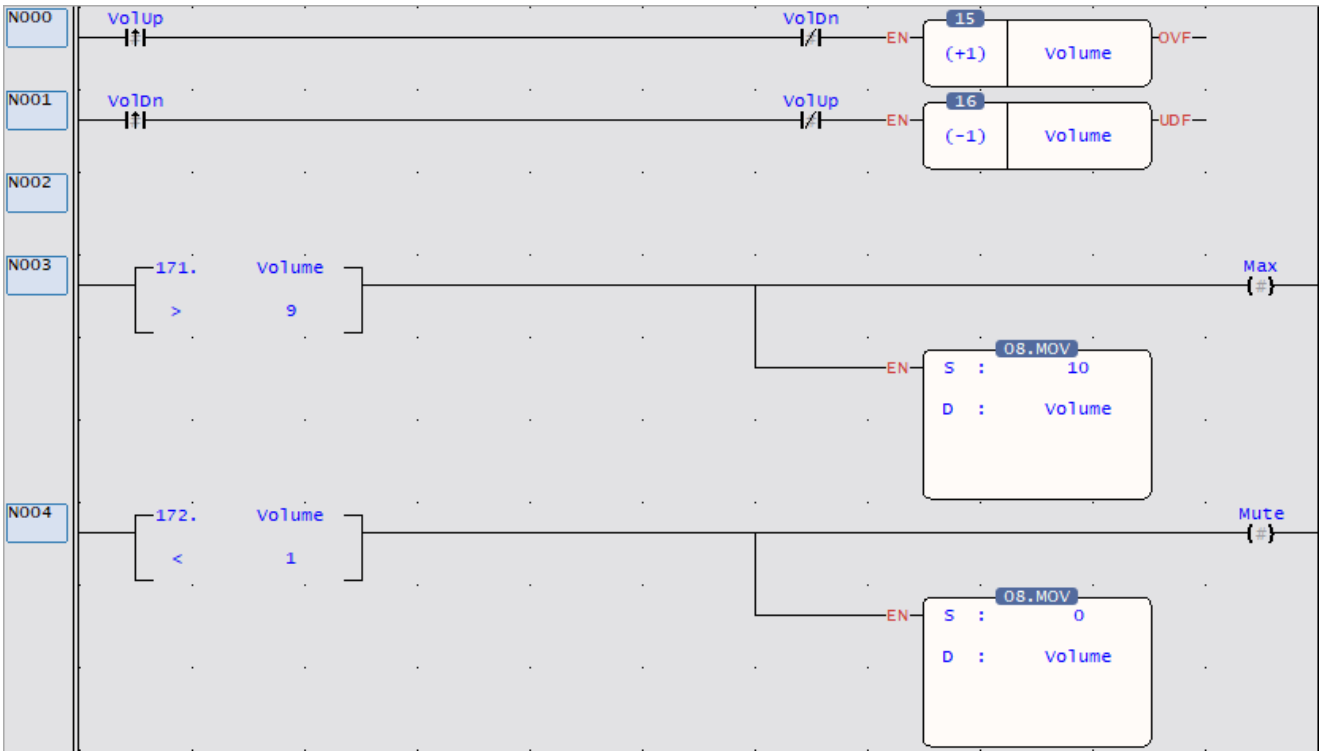
Variable

Name	Data Type	Array	Class	Initial Value	Description
EN	Bool	--	VAR_SIG_IN	--	High Level Trigger
VolUp	Bool	--	VAR_SIG_IN	--	Volume Up
VolDn	Bool	--	VAR_SIG_IN	--	Volume Down
Max	Bool	--	VAR_SIG_OUT	Initialized to 0 every time	Volume is Max
Mute	Bool	--	VAR_SIG_OUT	Initialized to 0 every time	Volume is Mute
Volume	16Bit-Int	DISABLE	VAR_PARA_INOUT	DISABLE	Volume

*The volume here uses VAR_PARA_INOUT to remember the last volume. Input the last volume from the external register to the Function Block when executing the Function Block. At the end of the Function Block, the volume is output to the external register for use when the Function Block is executed next time.

Function Block Internal Program

【TV Remote】



N000 When VolUp is triggered at rising-edge, we use Fun15 to increase volume.

N001 When VolDn is triggered at rising-edge, we use Fun16 to decrease volume.
The above operations work shall work exclusively in this case.

N002 With the use of Fun171, the volume will remain at 10 and set Max as ON at the same time when over 9

N003 With the use of Fun172, the volume will remain at 0 and set Mute as ON at the same time when under 1.

Function Block Internal Program in ST Language

```
IF R_TRIG( S:=VolUp ) THEN
    IF NOT VolDn THEN
        Volume:= Volume+1;
    END_IF
END_IF

IF R_TRIG( S:=VolDn ) THEN
    IF NOT VolUp THEN
        Volume:= Volume-1;
    END_IF
END_IF

if Volume > 9 THEN
    Max:=TRUE;
    Volume:=10;
ELSEIF Volume < 1 THEN
    Mute:=FALSE;
    Volume:=0;
END_IF
```

Action Description

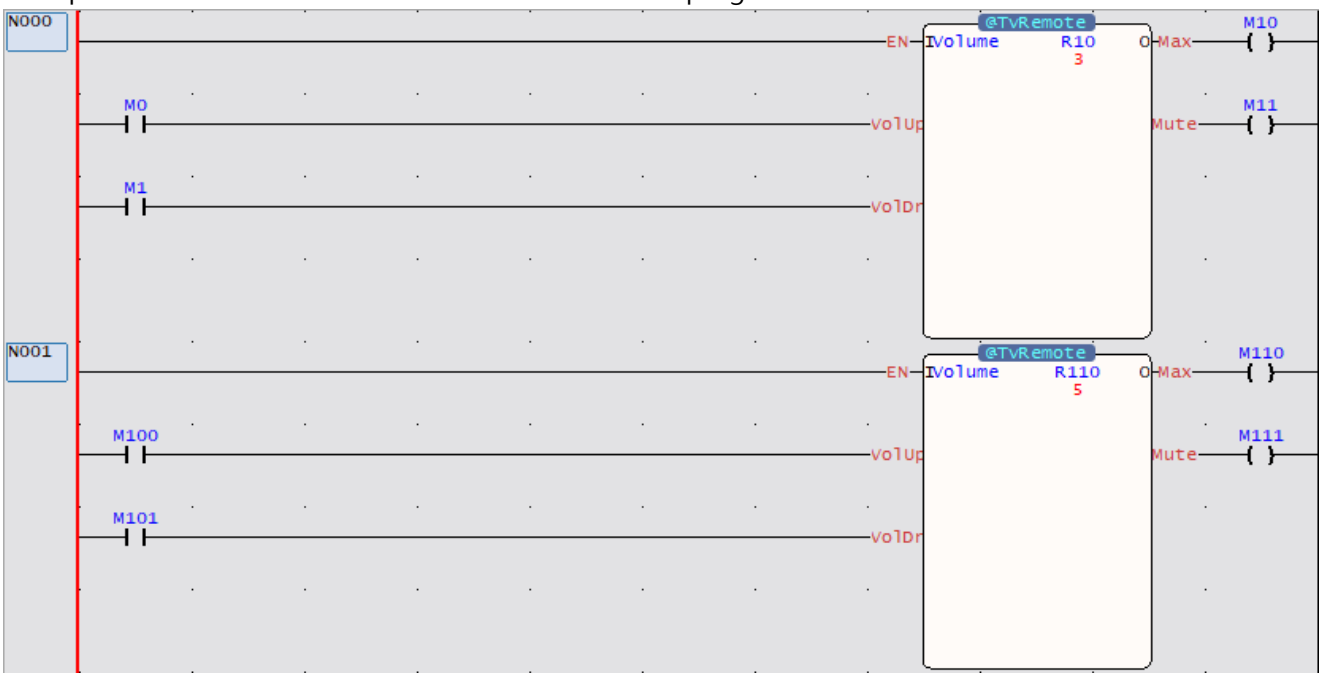
Step 1. Trigger VolUp (volume up) or VolDn (volume down) to increase/decrease the volume.

Step 2. The Volume up to 10 and can't continue to increase.

Step 3. The Volume down to 0 and can't continue to decrease.

Multiple use

Multiple use of Function Block instructions in the main program



N000 Control the R10 of volume

N001 Control the R110 of volume

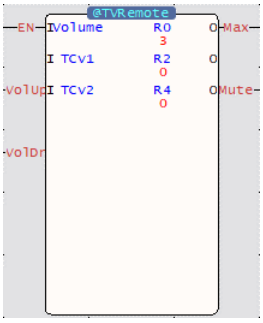
The result of 2 FCMs not interfering with each other.

3-7-2 TV Remote Timer

Application

The TV remote example can only be increase/decrease volume one unit per time. The real TV remote allows user to increase volume faster as the button pressed longer. Volume value range from 0~500.

Function Block Instruction

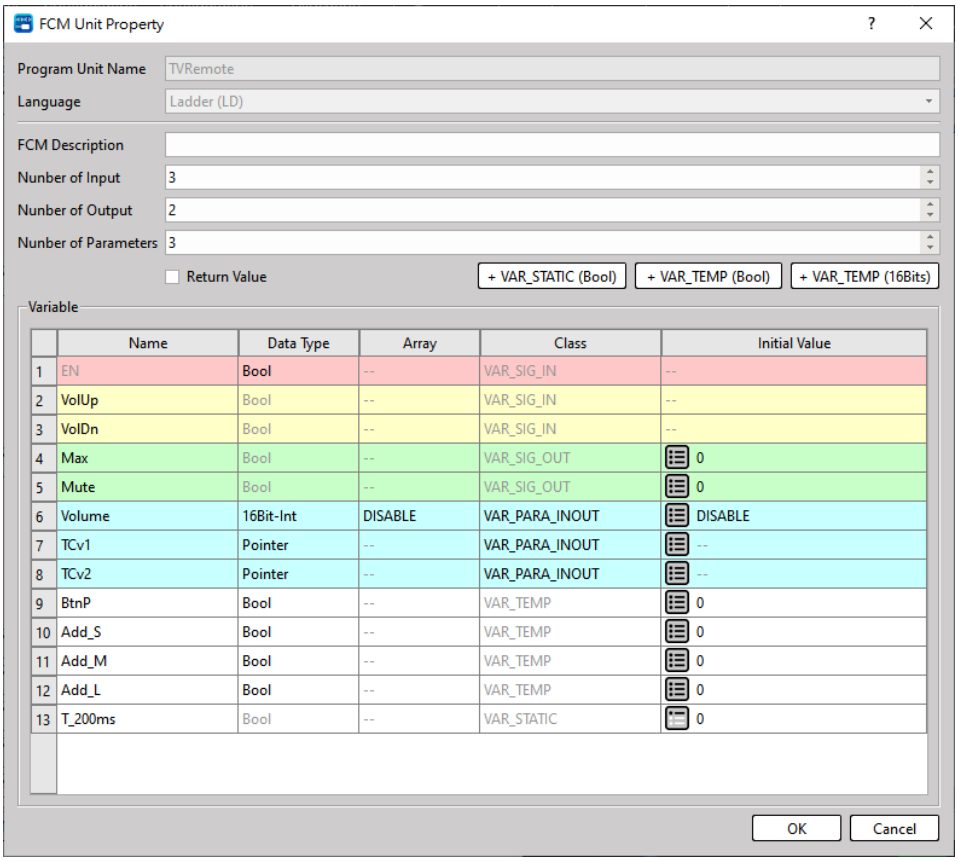


Input/Output Pins and Parameters

Parameter	Name	Description
Input Pins	EN	High Level Trigger
	VolUp	Volume Up
	VolDn	Volume Down
Output Pins	Max	Volume is Max
	Mute	Volume is Mute
Input/Output Parameters	Volume	Volume
	TCv1	For Fun87 Timer
	TCv2	For Fun87 Timer

Function Property

3 numbers of input (input pins), 2 numbers of output (output pins), and 3 numbers of parameter.



	Name	Data Type	Array	Class	Initial Value
1	EN	Bool	--	VAR_SIG_IN	--
2	VolUp	Bool	--	VAR_SIG_IN	--
3	VolDn	Bool	--	VAR_SIG_IN	--
4	Max	Bool	--	VAR_SIG_OUT	0
5	Mute	Bool	--	VAR_SIG_OUT	0
6	Volume	16Bit-Int	DISABLE	VAR_PARA_INOUT	DISABLE
7	TCv1	Pointer	--	VAR_PARA_INOUT	--
8	TCv2	Pointer	--	VAR_PARA_INOUT	--
9	BtnP	Bool	--	VAR_TEMP	0
10	Add_S	Bool	--	VAR_TEMP	0
11	Add_M	Bool	--	VAR_TEMP	0
12	Add_L	Bool	--	VAR_TEMP	0
13	T_200ms	Bool	--	VAR_STATIC	0

Variable

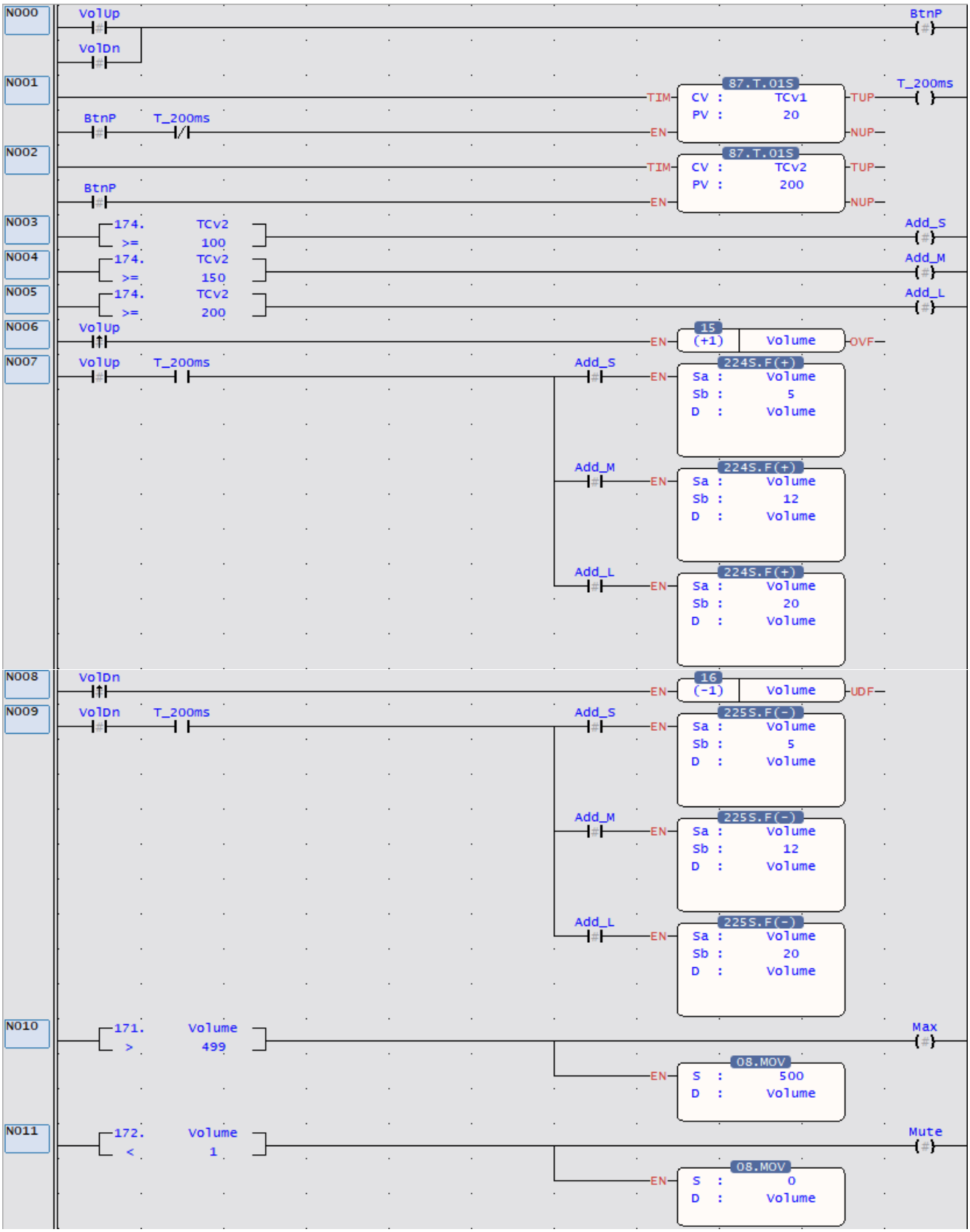
Name	Data Type	Array	Class	Initial Value	Description
EN	Bool	--	VAR_SIG_IN	--	High Level Trigger
VolUp	Bool	--	VAR_SIG_IN	--	Volume Up
VolDn	Bool	--	VAR_SIG_IN	--	Volume Down
Max	Bool	--	VAR_SIG_OUT	Initialized to 0 every time	Volume is Max
Mute	Bool	--	VAR_SIG_OUT	Initialized to 0 every time	Volume is Mute
Volume	16Bit-Int	DISABLE	VAR_PARA_INOUT	DISABLE	Volume
TCv1	Pointer	--	VAR_PARA_INOUT	--	For Fun87 Timer CV
TCv2	Pointer	--	VAR_PARA_INOUT	--	For Fun87 Timer CV
BtnP	Bool	--	VAR_TEMP	Initialized to 0 every time	
Add_S	Bool	--	VAR_TEMP	Initialized to 0 every time	
Add_M	Bool	--	VAR_TEMP	Initialized to 0 every time	
Add_L	Bool	--	VAR_TEMP	Initialized to 0 every time	
T_200ms	Bool	--	VAR_STATIC	Initialized to 0 on the first time	For Fun87 Timer out

*The volume here uses VAR_PARA_INOUT to remember the last volume. Input the last volume from the external register to the Function Block when executing the Function Block. At the end of the Function Block, the volume is output to the external register for use when the Function Block is executed next time.

*The example uses the multi-cycle instruction Fun87. In order to allow TIMER's timing and output status to span multiple scan times in the Function Block, Fun87's CV uses Pointer(VAR_PARA_INOUT) · Fun87's out pin uses VAR_STATIC BIT.

Function Internal Program

【TV Remote_Timer】



N000 Connect VolUp and VolDn in parallel into a BtnP signal
N001 In VolUp or VolDn is ON, Fun87 is used to generate a 200ms pulse.
N002 In VolUp or VolDnON, use Fun87 timing.
N003 Fun87_TCv2 outputs Add_S when the timer reaches 1 second
N004 F Fun87_TCv2 outputs Add_M when the timer reaches 1.5 seconds
N005 Fun87_TCv2 outputs Add_L when the timer reaches 2 seconds
N006 VolUp ON. Rising edge increases volume.
N007 Keep pressing VolUp for more than 1 second, and use 200ms pulses to increase differently according to Fun87_TCv2.
N008 VolDn ON, Rising edge lowers the volume.
N009 Keep pressing VolDn for more than 1 second, and use 200ms pulses to increase differently according to Fun87_TCv2.
N0010 With the use of Fun171, the volume will remain at 500 and set Max as ON at the same time when over 499.
N0011 With the use of Fun172, the volume will remain at 0 and set Mute as ON at the same time when under 1.

Function Internal Program in ST Language

```
BtnP:=VolUp or VolDn;

ACTimer_10MS( TIM:= true, EN:=BtnP AND NOT T_200ms , CV:=TCv1 , PV:=20     , TUP=>T_200ms ,
NUP=>InZone);
ACTimer_10MS( TIM:= true, EN:=BtnP , CV:=TCv2 , PV:=200    , TUP=>InZone    , NUP=>InZone);

IF TCv2>=200 Then
      Add_L:=true;
elseIF TCv2>=150 Then
      Add_M:=true;
elseIF TCv2>=100 Then
      Add_S:=true;
END_IF

IF R_TRIG( S:=VolUp ) THEN
    Volume:= Volume+1;
END_IF

IF VolUp and T_200ms THEN
            IF Add_S THEN
            Volume:= Volume+5;
      ELSEIF Add_M THEN
```

```
            Volume:= Volume+12;
        ELSEIF Add_L THEN
            Volume:= Volume+20;
        END_IF
END_IF

IF R_TRIG( S:=VolDn ) THEN
        Volume:= Volume-1;
END_IF

IF VolDn and T_200ms THEN
            IF Add_S THEN
            Volume:= Volume-5;
        ELSEIF Add_M THEN
            Volume:= Volume-12;
        ELSEIF Add_L THEN
            Volume:= Volume-20;
        END_IF
END_IF

if Volume > 499 THEN
        Max:=TRUE;
        Volume:=500;
ELSEIF Volume < 1 THEN
        Mute:=TRUE;
        Volume:=0;
END_IF
```

動作說明

Step 1. Trigger VolUp (volume up) or VolDn (volume down) to increase/decrease the volume.

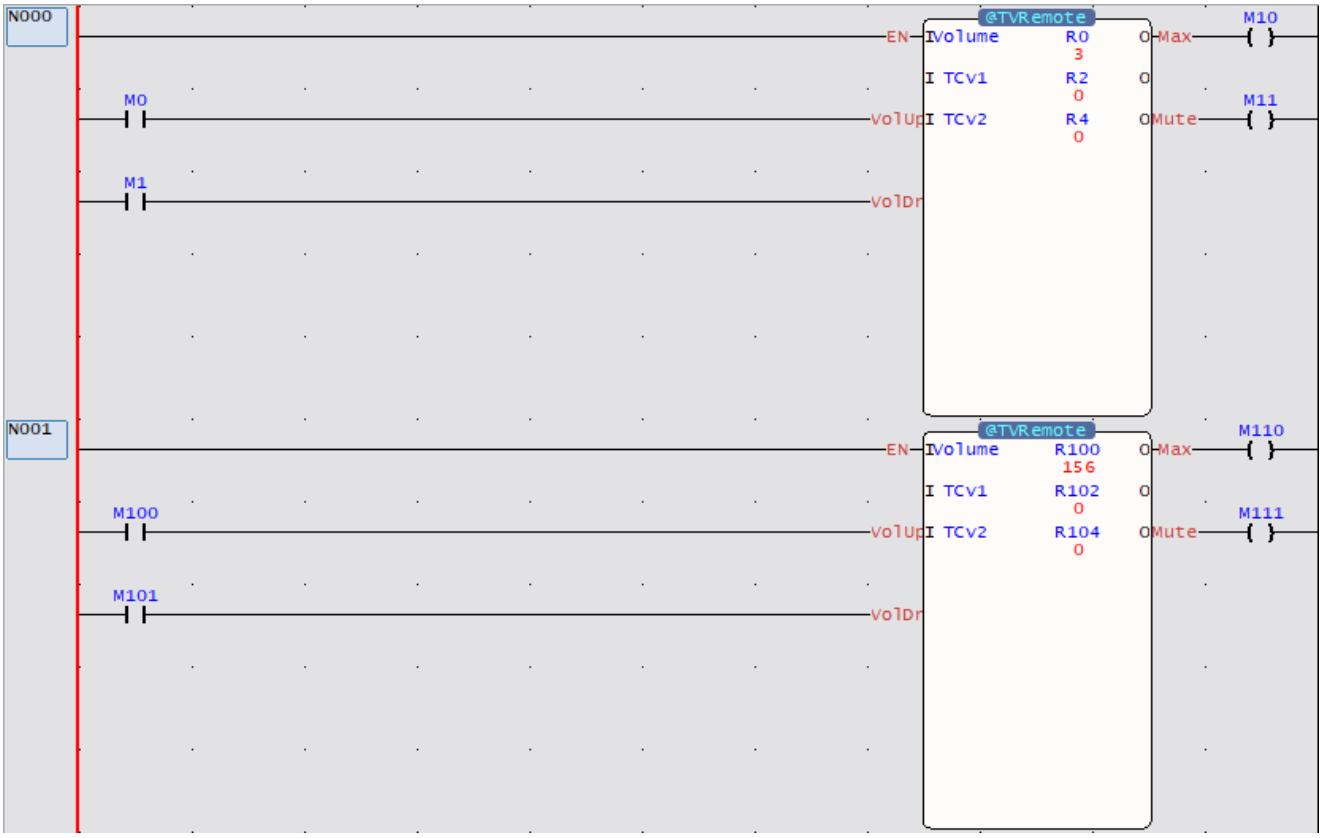
Step 2. Press to increase or decrease the volume by 1 level within 1 second. Press for 1 to 1.5 seconds to increase or decrease the volume by 5 levels every 0.2 seconds. Press for 1.5 to 2 seconds to increase or decrease the volume by 12 levels every 0.2 seconds. Press for more than 2 seconds to increase or decrease the volume every 0.2 seconds. Reduce the volume by 20 bars.

Step 3. The Volume up to 500 and can't continue to increase.

Step 4. The Volume down to 0 and can't continue to decrease.

Multiple use

Multiple use of Function Block instructions in the main program



N002 Control the R0 of volume

N003 Control the R100 of volume

The result of 2 FCMs not interfering with each other.

【Appendix 1】 Multi-cycle Instruction

Multi-cycle Instruction

Function Number	Function Name	Function Number	Function Name
Fun33	LCNV	Fun176	MFFlowStart
Fun34	MLC	Fun177	MFSysStop
Fun38	PID2	Fun178	MFHome
Fun87	T.01S	Fun179	MFPointMov
Fun88	T.1S	Fun180	MFJog
Fun89	T1S	Fun181	MFChgTbPrm
Fun98	RAMP2	Fun183	MFFlowResume
Fun99	TPCTL2	Fun184	MFFlowHalt
Fun115	DBUF	Fun185	MFSysRstAlm
Fun137	ICA	Fun186	MFFlowStop
Fun138	ICF	Fun187	MFSysInit
Fun139	HSPWM	Fun188	MFSysRCPR
Fun140	HSPSO	Fun189	MFSysRCPW
Fun141	MPARA	Fun191	MFSysCAMR
Fun142	PSOFF	Fun192	MFSysCAMW
Fun143	PSCNV	Fun193	MFGearMPG
Fun144	HSPWM2	Fun194	MFVelCtl
Fun148	MPG	Fun195	MFTorqCtl
Fun150	MBUS	Fun196	MFSysCAMGen
Fun151	CLINK	Fun197	MFAxMov
Fun152	NCR	Fun198	MFMapTbPrm
Fun156	CMCTL	Fun235	MFSysSetVirt
Fun160	RWFR	Fun236	MFSDOR
Fun161	WRMP	Fun237	MFSDOW
Fun162	RDMP	Fun238	MFAxLMov
		Fun239	MFAxCirMov
		Fun240	MFPathMov

Amendment record

Version	Modification date	Description
V1.0	2024/12/20	Draft