

OMC System Software
High-performanceHMI
VFSTModule
User Manual

IM41S61-E

Notices
<ul style="list-style-type: none"> ● The reproduction, transmission or use of this document or its contents is not permitted without express written authority. ● Information and specifications in this document are subject to change without notice. ● While information in this document is well edited and checked, mistake or omission may exist. Please don't hesitate to contact SUPCON if you have any question about this document. ● Please contact SUPCON via email "SMS@supcon.com" if you have any question.

Trademarks
<p>Trademarks or marks SUPCON, SPlant, Webfield, ESP-iSYS, MultiF, InScan, SupField are all registered, registering or using by Zhejiang SUPCON Technology Co., Ltd., which owns the properties of all trademarks or marks above. Without the written authority from Zhejiang SUPCON Technology Co., Ltd, no individual or company shall use any trademarks or marks above. We reserve the right to take legal action for any individual or company using trademarks or marks above illegally.</p>






Symbol Definition	
	WARNING: Indicates information that a potentially hazardous situation which, if not avoided, could result in serious injury or death.
	RISK OF ELECTRICAL SHOCK: Indicates information that Potential shock hazard where HAZARDOUS LIVE voltages greater than 30V RMS, 42.4V peak, or 60V DC may be accessible.
	ESD HAZARD: Indicates information that Danger of an electro-static discharge to which equipment may be sensitive. Observe precautions for handling electrostatic sensitive devices
	ATTENTION: Identifies information that requires special consideration.
	TIP: Identifies advice or hints for the user.

Table of Contents

Section 1 Overview	1
1.1 Fundamental Functions	1
1.2 Techniques Specification	1
1.3 Introductions to Interface	2
Section 2 Introductions to Operation	4
2.1 New Custom Function Block	4
2.2 Set Custom Function Block	5
2.3 Use Custom Function Block	5
2.4 Add Parameter or Variable	6
2.4.1 Parameter and Variable's Classification	6
2.4.2 Add and Configure Input Parameter	7
2.4.3 Add reference variables	9
2.4.4 Add alias variable	10
2.4.5 Add Built-in function block	11
2.4.6 Check Added Items	13
2.5 Set Alarm and Rule	15
2.5.1 Self-defined Alarm	15
2.5.2 Set Alarm	16
2.5.3 Set Rule	18
2.6 Other Operations	19
2.6.1 Delete Parameter or Variable	19
2.6.2 Parameter Dragging Function	20
2.6.3 Import and Export the Custom Function Block Program	20
2.6.4 Arrange Memory	22
2.7 Compile	22
2.7.1 Compile in Custom Function Block Software	22
2.7.2 Compile in Configuration Management Software	23
2.8 Excerpt Custom Function Block	23
Section 3 Global Function Block Logic Editing	25
Section 4 ST Language Programming	26
4.1 Programming foundation	26
4.1.1 Data Type	26
4.1.2 Function and Keyword	26

4.1.3 Operator	27
4.1.4 Constant and Annotation	27
4.1.5 Syntax Rule	28
4.1.6 Timer	32
4.1.7 Array Parameters	33
4.1.8 Global Array.....	34
4.2 Auxiliary Function	38
4.2.1 Bookmark	38
4.2.2 Programming Assistant.....	40
4.2.3 Color settings	41
4.2.4 Find	41
4.2.5 Replace	42
4.3 Notes for Application.....	42
4.4 Example	43
4.4.1 Example Description	43
4.4.2 Operation Steps:.....	43
Section 5 SFC language programming.....	48
5.1 Programming Foundation	48
5.1.1 Elements of SFC Program.....	48
5.1.2 Program Structure	48
5.1.3 Programming Rule	49
5.2 SFC Program Running Rule.....	51
5.2.1 State Transition Graph.....	51
5.2.2 Program Running Role	52
5.3 Add/Delete Sequence Step and Branch	53
5.3.1 Add Sequence Step.....	53
5.3.2 Add Selection Branch	53
5.3.3 Add Parallel Branch.....	54
5.3.4 Extend Selection Branch	55
5.3.5 Extend Parallel Branch.....	56
5.3.6 Jump	57
5.3.7 Delete Step or Branch	59
5.4 SFC Function Block External Default Pins	59
5.4.1 Description	59
5.4.2 Linking Mode	60
5.5 SFC Function Block Debug Panel	60

5.5.1 Mode	61
5.5.2 Command and State.....	61
5.5.3 Step Color and State	62
5.5.4 Parameter View and Download	62
5.6 . Example	62
5.6.1 Example Description	62
5.6.2 Operation Steps.....	63
Section 6 Appendix	70
6.1 ST Language Programming Software	70
6.1.1 ST Language Programming Software Interface command list	70
6.1.2 Function and Function List	71
6.2 SFC Language Programming Software	81
6.2.1 SFC Language Programming Software Interface Commands List	81
6.2.2 SFC Language Programming Software Shortcut Menu	81
Section 7 Revision.....	83

VFSTModule User Manual

Section 1 Overview

1.1 Fundamental Functions

Custom function block programming software (VFSTModule.exe) is for users to create custom function block according to requirements of control techniques so that the logic can be reused in FBD programming software. Custom function block is equal to subprogram, including input, output, inner variable and inner program logic defined by users.

Custom function block can be achieved by ST and SFC language, and global function block can be achieved by ST, SFC and FBD languages.

The global function block is almost the same with the custom function block in compiling. This manual mainly introduces the custom function block programming software, and the difference of global function block programming software will be pointed out.

1.2 Techniques Specification

- The name of custom function block must begin with letter and can only contain English letters, numbers and underline. It can not exceed 16 characters. Names can't be repeated among custom function blocks or system function blocks. It cannot be modified after it is named. Descriptions cannot exceed 64 characters and can be modified; It is noted that the referenced parameters support to start with numbers.
- The name of input/output parameter can not exceed 8 characters. The name of inner parameter can not exceed 15 characters. The name of function block variable can not exceed 24 characters, and must begin with letter and only contains English letters, numbers and underline. The name can't be repeated among every parameter.
- The total amount of input parameter, output parameter, inner parameter, configuration parameter and supervision parameter is 512. The single parameter number is limited by the parameters total amount. The maximum amount of temporary variable and alias variable is 64 and 128 respectively. There is no limit for the amount of reference variable. Function block variable is limited to the 8K memory region for one custom function block.
- The maximum amount of custom function block is 250 within a single control station.

1.3 Introductions to Interface

The interface of VFSTModule (the following figure shows the interface of ST language programming software, and the interface of SFC and FBD language programming software is similar) is shown in Figure 1-1.

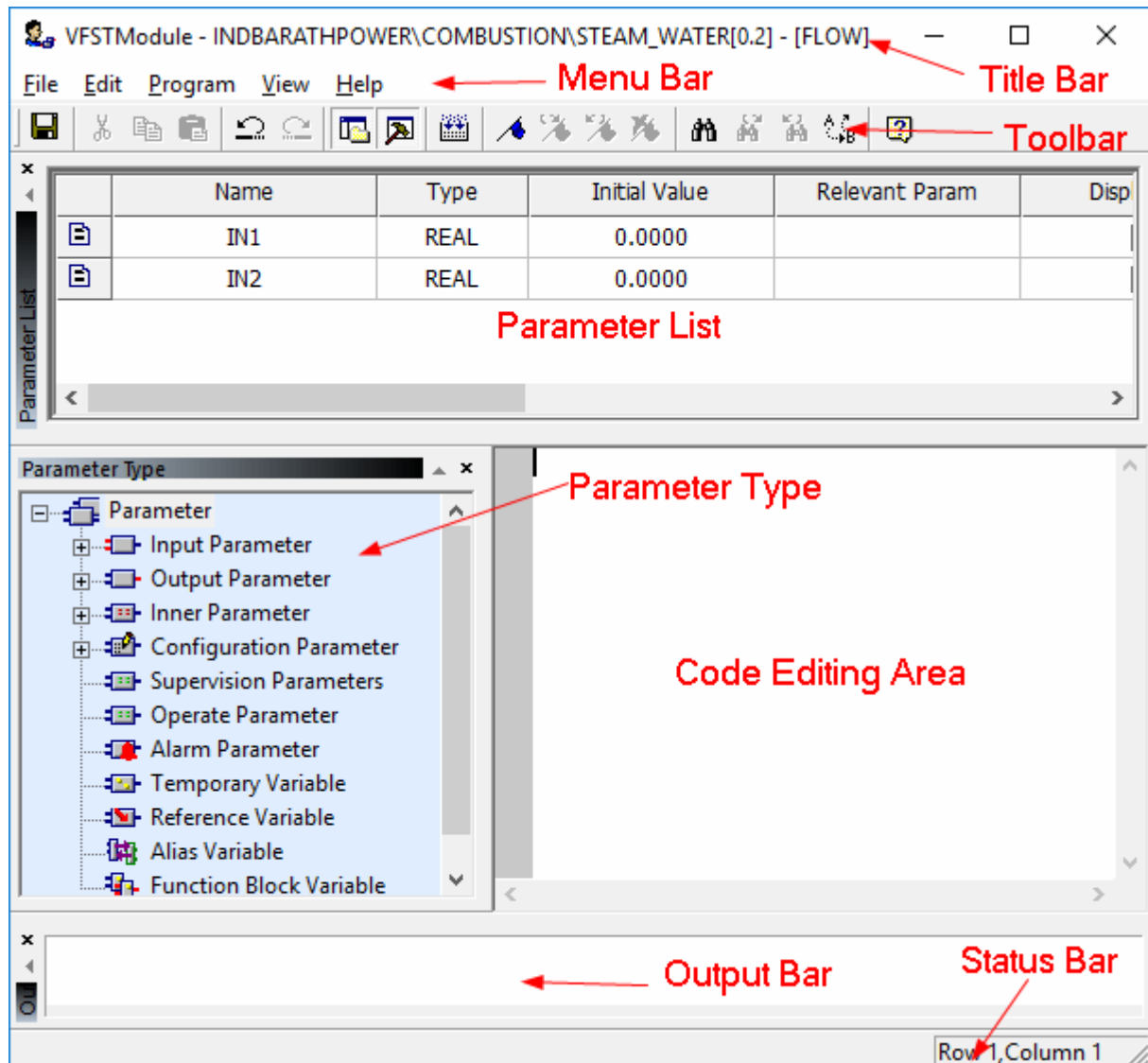


Figure 1-1 Interface of custom function block programming software (ST)

- Title Bar
Display software title, current project name, control domain name, controller name and address, and the name of current custom function block.
- Menu Bar
Including menus of file, edit, program, view and help, and each menu contains several submenus.
- Toolbar
List some frequently used menu items in the form of icons, making it convenient for users to operate.

- Code Editing Area
Edit codes of the custom function block.
- Parameter Type
Located on the left of main interface and display information of all parameters of current function block.
- Parameter List
Located on the top of interface and display information of parameter selected in the window parameter type.
- Output Bar
Located at the bottom of interface and mainly display information of program compiling.
- Status Bar
Display current operation information and some prompt information. Introductions to Menu Bar.

Section 2 Introductions to Operation

The procedure of add, edit and excerpt function block is shown as follows.

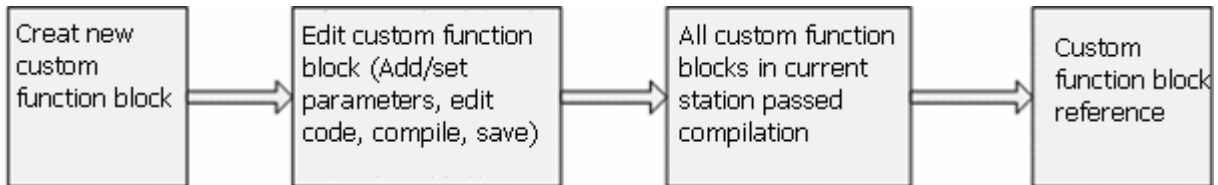


Figure 2-1 Edit flow of custom function block



Tip:

The functions and corresponding figures in this section is based on ST language programming software, which is similar with those based on SFC language programming software.

2.1 New Custom Function Block

Select the node of custom function block under control station in configuration management software, and select “New” in the right-click menu, as shown in Figure 2-2.

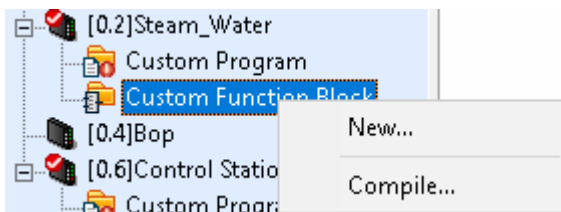


Figure 2-2 New custom function block

Dialog box of new custom function block will pop up, as shown in

Figure 2-3.

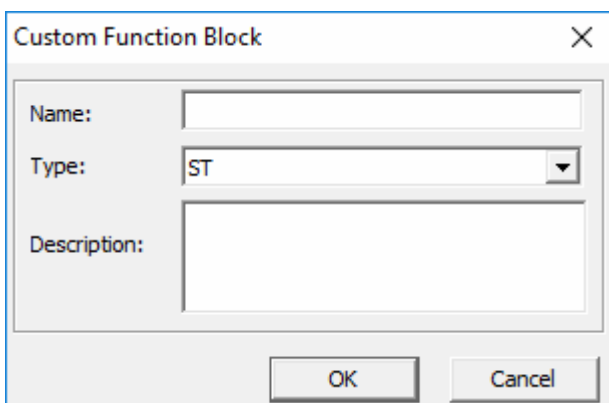


Figure 2-3 Dialog box of Custom Function Block

Select ST language or SFC language. Input name and descriptions (both must conform to the naming rule) of custom function block, and click “OK” to create a new custom function block. Users can select FBD language for global function block.

2.2 Set Custom Function Block

Users can set properties and password of custom function block after a new one is created, please refer to *VFExplorer User Manual* for specific setting method.

2.3 Use Custom Function Block

Select “Custom Function Block” node under control station in the work area of configuration management software, and choose custom function block program to be edited on program list on the right, double click the name of the function block or right click it and select “Edit” in right-click menu to open the custom function block to be edited.

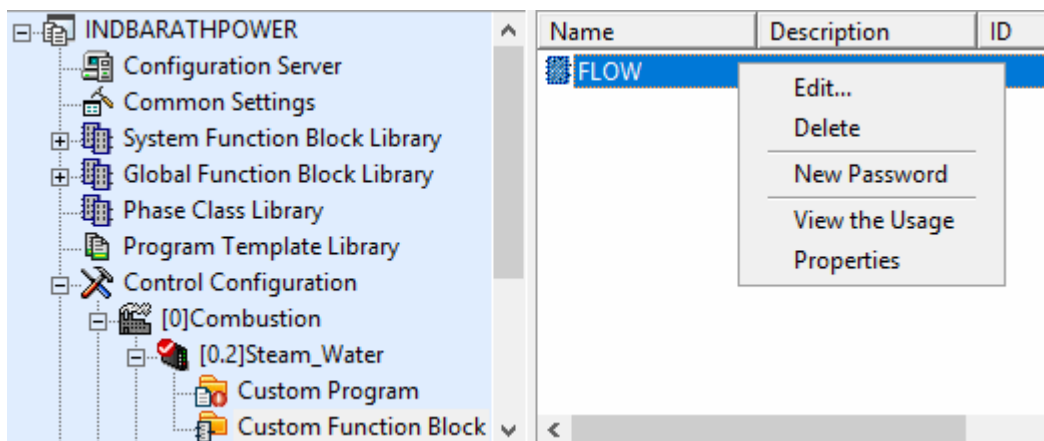


Figure 2-4 Start up custom function block software

2.4 Add Parameter or Variable



Tip:

- Reference variables and function block variables are unavailable when SFC language is used.
- Parameters or variables of global function block should not include field of BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, UNDEFINE, NULL, PRI, SUP or ACTAPRI_. "PRI" is used as the field of tag's alarm privilege, "SUP" is used as the field of tag's alarm suppression, "PRI" and "SUP" will be generated as "HHHPRI" and "HHHSUP" after set alarm. "ACTAPRI_" is used as the field of tag's highest activate alarm.
- The decimal digits of parameters of custom function block (including global function block) are all 4.
- VFSTModule supports locking parameters by the command of "View > Lock Parameter Bar". After locked parameter bar, parameters can not be modified until unlocked.
- The description of parameters and variables in custom function block (including global function blocks) should not contain line breaks.

2.4.1 Parameter and Variable's Classification

When editing function blocks, many types of parameters will be used. Different parameters have different functions, support different data types. Functions and data types each parameters support in each user function block are shown in the table below.

Parameter Type	Function Illustration	Other Illustration
Input parameter	It serves as the external interface, reads in external tag data and transfers the operation results to external tags.	Support BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL*, STRING, and array in the data type shown above (apart from STRING)
Output parameter	It works as the external interface of function blocks. Reference the external tag data and transfer the calculation results to the external tags.	Support BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL*, STRING, and array in the data type shown above (apart from STRING)
Built-in parameter	It works in specific memory region and is used only in the custom function block program.	Support BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL*, STRING, and array in the data type shown above (apart from STRING)
Configuration parameter	It can set default value in advance. It is excerpted in custom function block program, and is visible in FBD program.	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL*, EUNIT, STRING, and array in the data type shown above (apart from EUNIT and STRING)
Supervision parameter	Custom supervision parameters, except FLAG, can set default values in advance and are excerpted in custom function block program. They can be set in supervision. There is no FLAG parameter for custom function block.	Support BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL*, STRING, and array in the data type shown above (apart from STRING)

Parameter Type	Function Illustration	Other Illustration
Operational parameter	The parameter can be set in advance and invoked in the custom program, and it can be modified when operating.	Support BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL*, STRING, and array in the data type shown above (apart from STRING)
Alarm parameter	There is only one alarm parameter ENALM in global function block, which is created synchronously with custom function block and cannot be modified.	-
Temporary variable	It does not work in specific memory region, and the memory region is distributed again in every operation period.	Support BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL*, STRING.
Reference variable	It is able to directly excerpt specific external tag, refer the tag or read in the values of the tag, and it can be used when the custom function block use as instance. There is no reference variable for global function block. Users can use reference variable to be compatible with the configuration of earlier version software. Alias is recommended.	Global function block doesn't support referencing variables. ST custom function block doesn't support referencing global function block (doesn't support referencing tags with arrays included).
Alias variable	In the case of that user needs several custom function blocks with same internal logic and difference reference parameters (tags) that not serving as input/output pins, these parameters can be used as alias variables. In FBD programming software, corresponding actual tags are excerpted to the alias variables achieving repeatedly usage of the custom function blocks. Alias variable is more flexible than reference variables.	Support BOOL, SINT, USINT, INT, UINT, DINT, UDINT, REAL, STRING, and array in the data type shown above (apart from STRING)
Function block variable	It can reference FBD function block for custom function block program. The user function block with alias variable can not be referenced.	Doesn't support function block variables as arrays. ST custom function block supports custom function block type included with arrays. ST global function block doesn't support function block types included with arrays.

*: Only the FCU811-S control station can create custom function blocks with LREAL-type parameters.

2.4.2 Add and Configure Input Parameter



Tip:

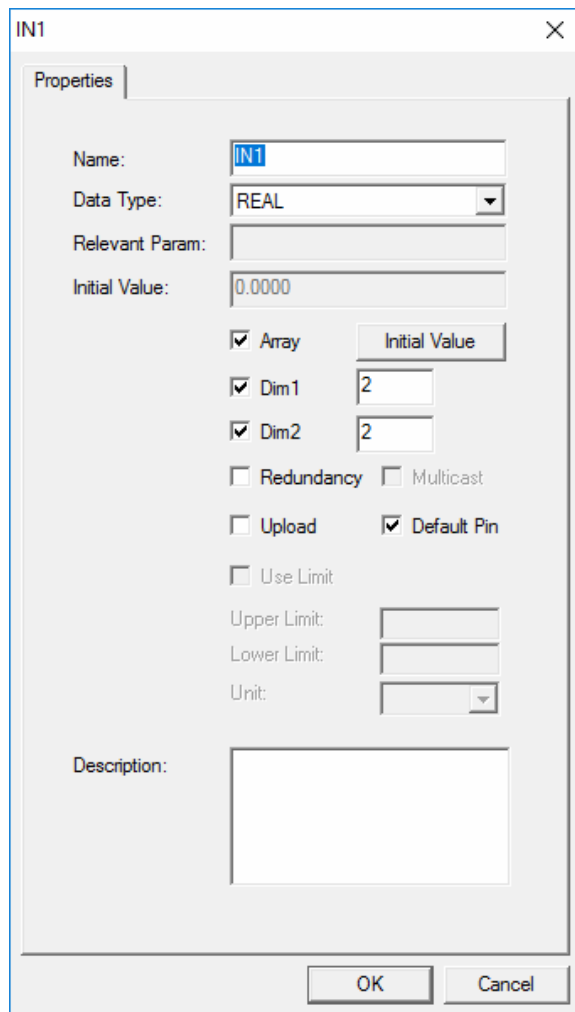
Ways to add output parameter, inner parameter, configuration parameter, supervision parameter, operate parameter and temporary variable are like the way to add input parameter.

1. Select "Input Parameter" in "Parameter Type Window", and click "Add" in the right-click menu. The newly added parameters are in the "Input Parameter".
2. Configure parameter property

Double click the added parameters, "Property" dialog box pops up. Configure the property according to the table below.

Configuration item		Configuration instructions
Name		The parameter name used to configure the parameter, the parameter name will be displayed when the function block is applied.
Type of data		To set the data type of the parameter, see the section " parameters and variables of ."
Related parameters		When the parameter type is built-in parameter, configuration parameter, monitoring parameter, operation parameter, alarm parameter, the variable associated with the parameter can be configured.
Initial value		Used to configure the initial value of the parameter. When the parameter is an array type, you need to specify the initial value of the internal value of the array through the button "Initial Value" on the right side of "Array".
Array	Checkbox	Check "array" parameter specifies the number of array type, dimensions and the initial value at this time can be arranged in an array.
	Initial value	<ul style="list-style-type: none"> Configuring the dimension, click "Initial Value" to configure the initial value for each array element in input parameters, output parameters, built-in parameters, configuration parameters, monitoring parameters, and operating parameters. The data type of the array parameter will set the initial value of the array parameters before modification to 0 or OFF. Alias parameters don't support initial value settings included with arrays.
	Dimension 1	When the parameter is a one-dimensional array, dimension 1 needs to be configured , and dimension 2 is not checked . The value of dimension 1 or dimension 2 are both within [2, 32].
	Dimension 2	
Redundancy		After checking this parameter, the parameters will be synchronized from the main controller to the redundant controller periodically. On the contrary, it is not synchronized. It should be noted that the STRING parameter does not support redundancy.
Multicast		Used to specify whether the parameter supports multicast. Only multicast parameters can be used as related parameters for other parameters. It should be noted that the STRING parameter does not support multicast.
Upload		After checking this parameter, after applying the function block in the user program, the parameter supports uploading when executing "Save Function Block Real-time Value". It should be noted that STRING type parameters do not support uploading.
Pin display by default		After the parameter is checked, the pin corresponding to the parameter will be displayed when the function block is applied in the user program.
Enable range		This configuration is valid only when the parameter is the operating parameter of the Phase function block. After the Enable range is checked, the upper range limit, lower range limit and unit can be configured.
Description		The description information used to configure the parameter.

Repeat step 1) and step 2) to complete the addition of all input parameters.

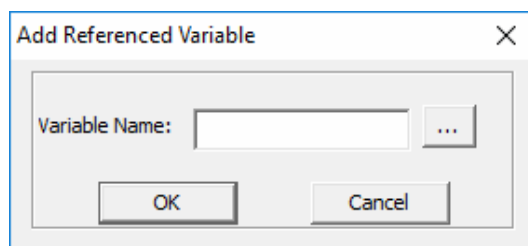


The image shows a dialog box titled "IN1" with a close button (X) in the top right corner. The dialog has a "Properties" tab. Inside, there are several input fields and checkboxes. The "Name" field contains "IN1". The "Data Type" is set to "REAL". The "Relevant Param" field is empty. The "Initial Value" is "0.0000". There are checkboxes for "Array" (checked), "Dim1" (checked), "Dim2" (checked), "Redundancy" (unchecked), "Multicast" (unchecked), "Upload" (unchecked), "Default Pin" (checked), and "Use Limit" (unchecked). Next to the "Array" checkbox is a button labeled "Initial Value". Below the checkboxes are input fields for "Upper Limit", "Lower Limit", and "Unit". At the bottom is a large "Description" text area. At the very bottom are "OK" and "Cancel" buttons.

Figure 2-5 The newly added input parameter property configuration dialog box


2.4.3 Add reference variables

Select the corresponding reference variable, alias variable or function block variable which needs to add, select "Add" in right-click menu the corresponding dialog box will pop up:



The image shows a dialog box titled "Add Referenced Variable" with a close button (X) in the top right corner. It has a "Variable Name" input field followed by a browse button (three dots). At the bottom are "OK" and "Cancel" buttons.

Figure 2-6 Add reference variable

Click browse button  on the right, dialog box for tag selection will pop up, and choose the required tag.

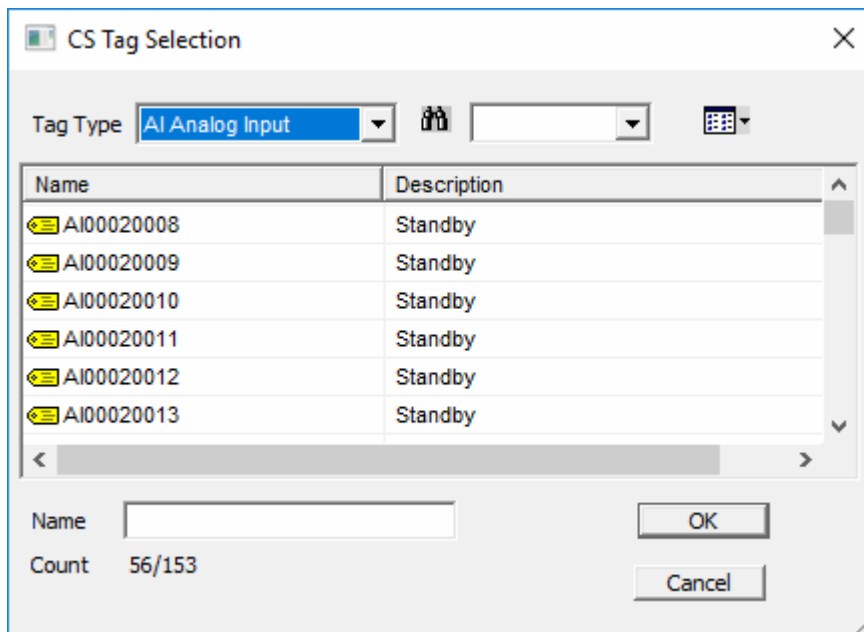


Figure 2-7 Select tag

2.4.4 Add alias variable

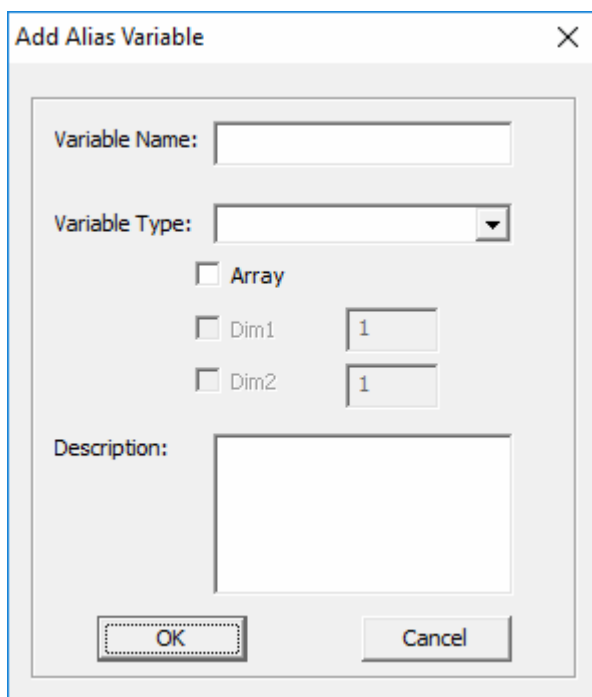


Figure 2-8 Add alias variable

Input variable name and description, select variable type, after click “OK”, a new alias variable will be added.

**Attention:**

- If a custom FBD modifies the variable by adding alias variables (i.e., there were no alias variables before, but there are after the modification) or removes alias variables (i.e., there were alias variables before, but there aren't after the modification), the FBD needs to be deleted and re-added to ensure its proper functioning.
- When alias variable is array, it only supports configuring array dimension rather than configuring the initial value of array.
- When alias variable references STRING tags, this variable cannot reference this parameter when the internal logic inside the function block is edited.

2.4.5 Add Built-in function block

1. In the parameter type window, select “Function block variable”, select “Add function block variable” and then the dialog box pops up.

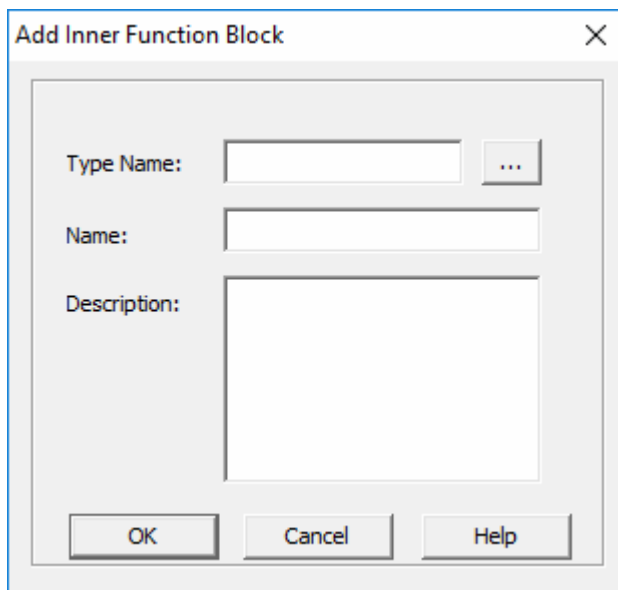



Figure 2-9 Add function block variable

2. Click browse button  behind “Type Name”, the dialog box of "Function Block Selection" will pop up, which displays all types of function block, as shown in Figure 2-10, and choose one function block from them.

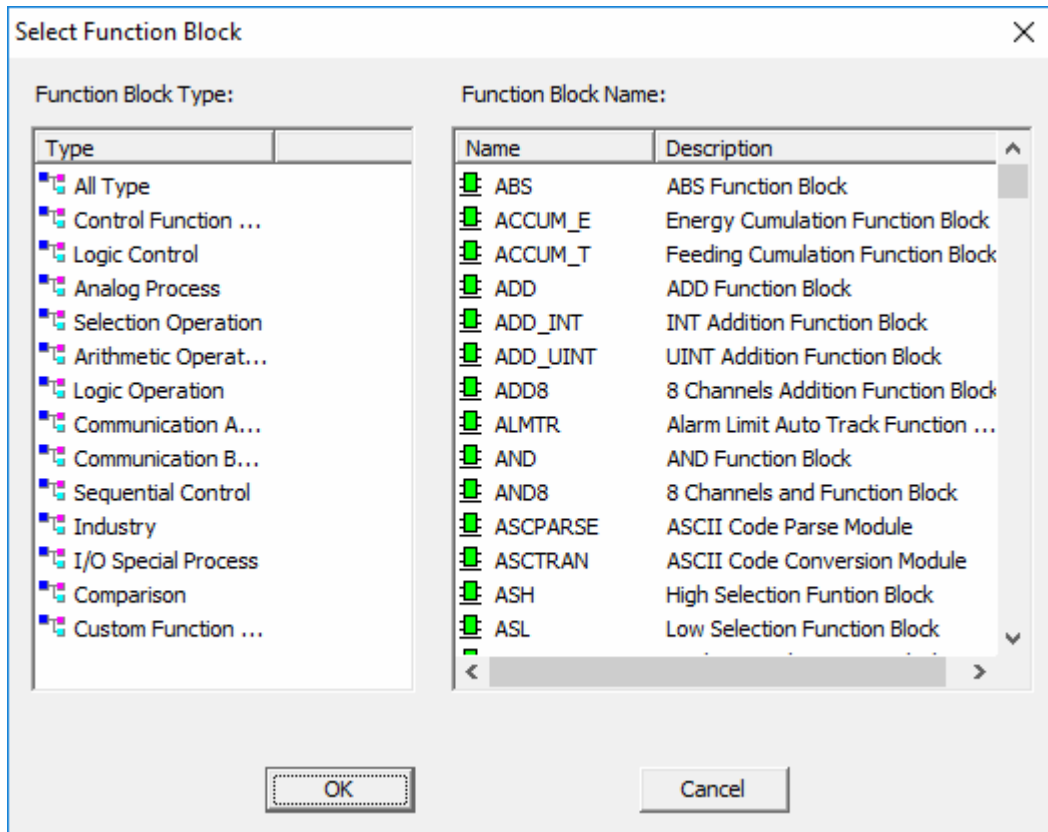


Figure 2-10 Add a function block variable

- Input name and description in dialog box shown in Figure 2-9, after click "OK", a new function block variable will be added.
- After adding function block variables, select function block variables in the parameter navigation tree and select "Function block parameter settings" in the right-click menu, and then the dialog box pops up.

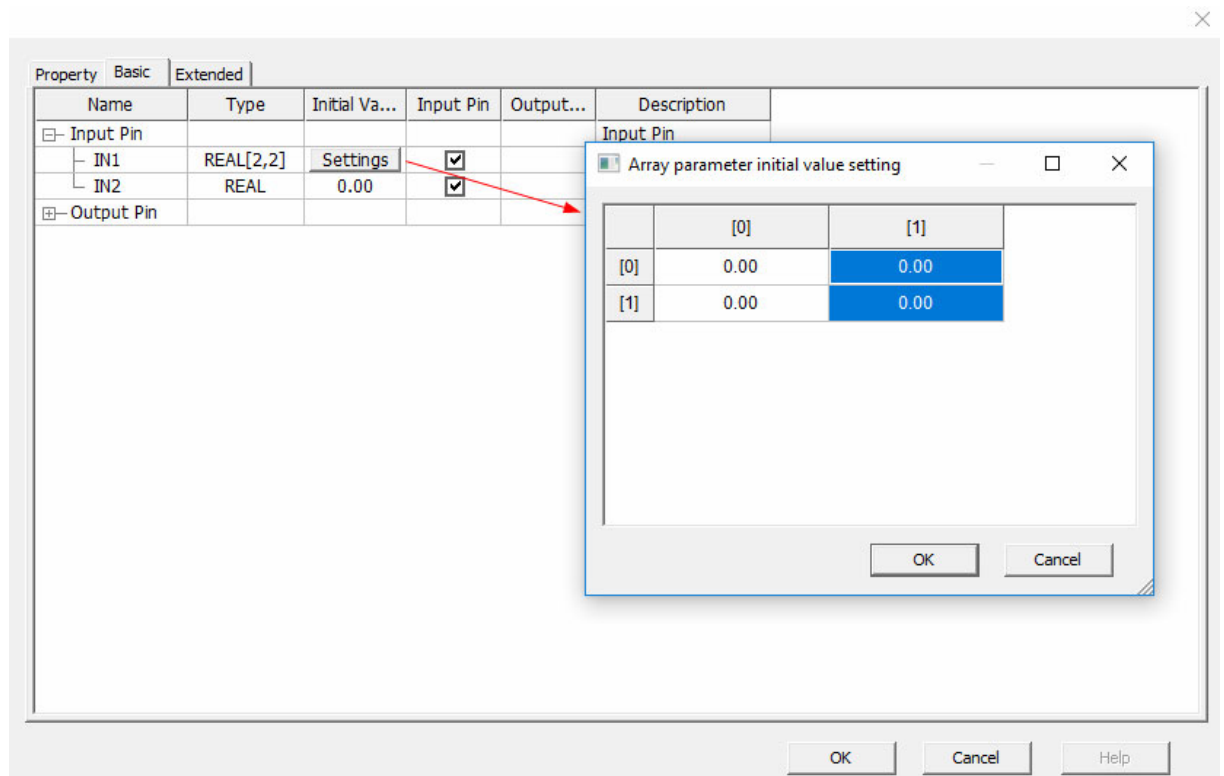


Figure 2-11 Function block variables' initial value example

5. Configure the initial values of function block parameters.

- If the parameter type is normal, input initial values in "Initial value" column.
- If the parameter type is array, click "Set" button in the "Initial value" column and users can specify initial values of the parameters of the function blocks in the pop-up dialog box.



Attention:

When a function block variable refers to a function block containing **STRING** parameters, the function block variable cannot be referenced when editing the internal logic of the function block.

2.4.6 Check Added Items

After parameters/variables of each type are added for custom function block, they are displayed as in Figure 2-12:

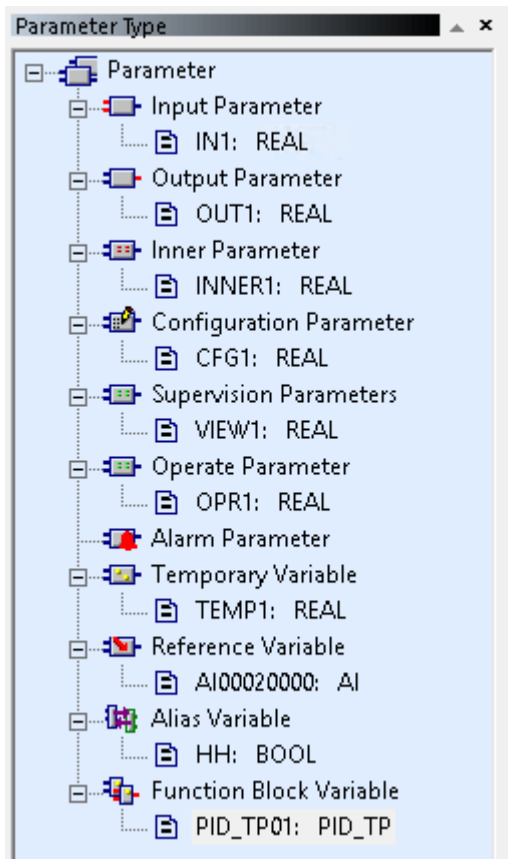


Figure 2-12 After adding parameter of each type

- Names of newly added input parameter, output parameter, inner parameter, configuration parameter, supervision parameter, operate parameter and temporary variable will be automatically created. Names are created in the form of “type+ number” according to different types. For example, the first added input parameter is named IN1, and the second is IN2, if the name has been used, then the number will plus 1 until finding out a unique name. Other parameters are named by users and names cannot be repeated.
- Names of new input parameter, output parameter, inner parameter, configuration parameter, supervision parameter, operate parameter and temporary variable are set by users and can not repeat. The data type is REAL by default (Please refer to 4.1.1 for instructions of data type).
- Alarm parameter can not be added for custom function block. After parameters/variables of each type are added for global function block, they are displayed as follow:

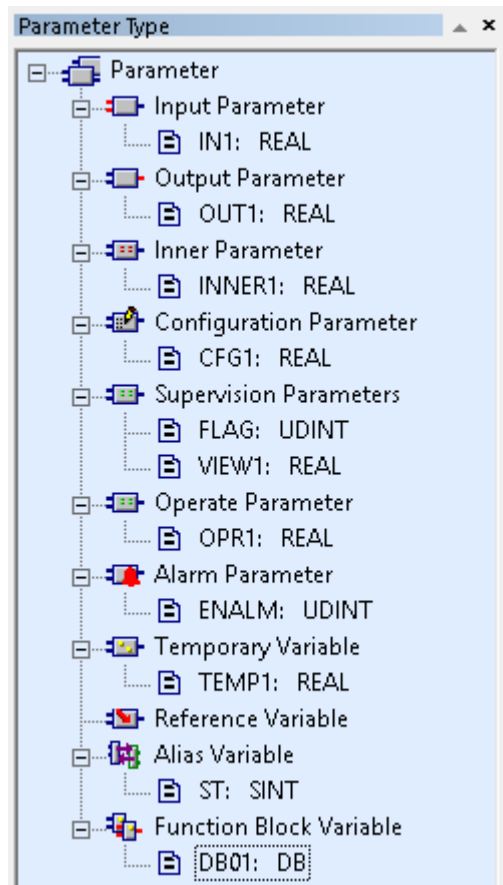


Figure 2-13 After adding parameter of each type

- The default name of supervision parameter created by alarm setting for global function block is FLAG, and its type is UDINT. It can not be modified. The default name of alarm parameter created by alarm setting for global function block is ENALM, and its type is UDINT. Both parameters can not be modified.



Tips:

The name of parameter and variable must start with letter and consist of English letters, numbers and underline. Name cannot be repeated among parameters.

Reference variables and function block variables are unavailable when SFC language is used.

2.5 Set Alarm and Rule

Custom function block dose not support the setting alarm and rule function, while the global function block supports it.

2.5.1 Self-defined Alarm

Click **Program/ Self-defined Alarm** to pop up the dialog below, in which can add self-defined alarm type. Maximum of custom alarms in single global function block is 20.

Custom Alarm ✕

No.	Name	Description	Level
0	A	Case1	Low
1			Log
2			Low
3			Medium
4			High
5			Urgency
6			Safe Related
7			
8			
9			
10			
11			
12			
13			
14			
15			

OK Cancel

Figure 2-14 Self-defined Alarm

Name, description and alarm priority can be set in the figure above. Name of new alarm cannot repeat with system alarm or existed custom alarm in this global function block.

After setting the self-defined alarm type, user can view and select the self-defined alarm in alarm parameter settings. Its features are same with system alarm.

Please publish and compile after creating or modifying the self-defined alarm to validate it.



Tips:

New self-defined alarm is in level “Low” in default.

2.5.2 Set Alarm

Users can use the alarm parameter in supervision by setting alarm.

1. Select the “ENALM” in “Parameter”, right-click and select “Alarm Settings” to open the “Alarm Settings” dialog. Right-click “Parameter” or select the “FLAG” in the “Supervision Parameters” of global function block can also perform the alarm settings.
2. Select the alarm name in the drop-down menu of “Name” in the “Alarm Settings” dialog, including the custom alarm, and the “Description”, “Type” will be shown automatically. The alarm cannot repeat in the alarm settings dialog, which means the same character strings are not allowed for names.

3. Select the alarm type in “Use Alarm Enable”. The selected parameter will be shown in the alarm parameter settings of properties settings when invoking the global function block.
4. Select the parameters of alarm in drop-down menu of “Related Parameter”.

Note: parameters in the drop-down menu are multicast.

5. Repeat step 1 and step 2 to finish the settings of all alarm parameters.
6. Click “OK” to complete the operation.

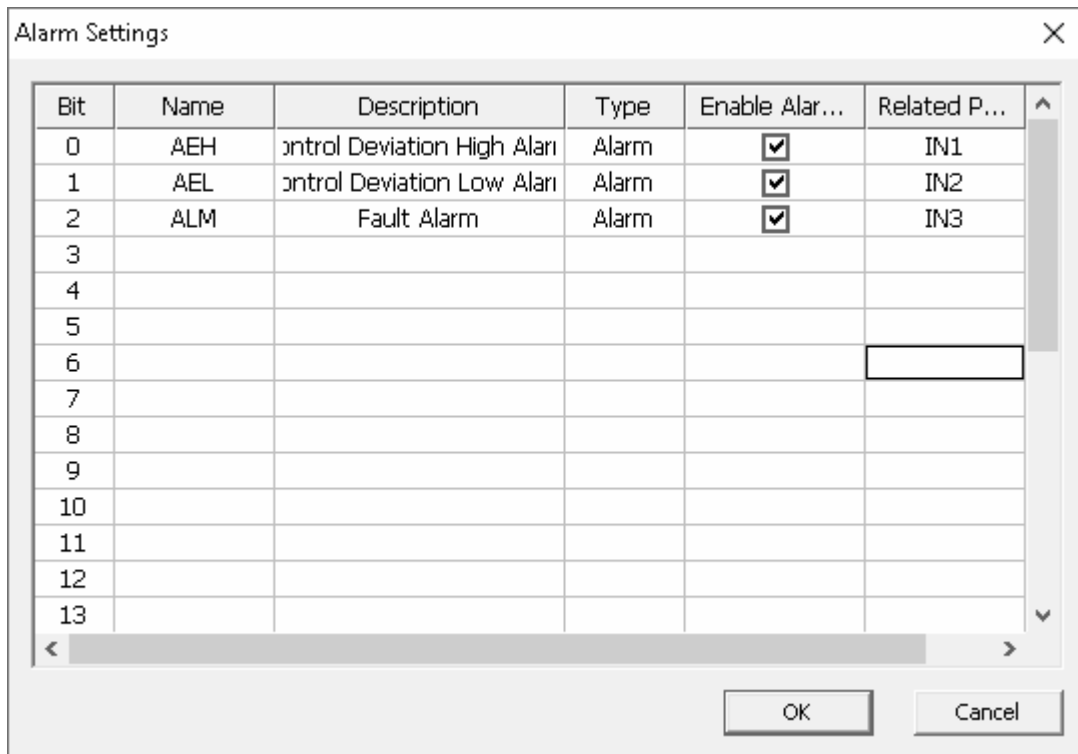


Figure 2-15 Alarm Settings Dialog

In the figure above, the 0~2 bits of FLAG refer to the three states of function block, AEH, AEL and ALM, 3~31 bits have no sense. If users name it as G_A when invoking the global function block, the state information AEH/AEL/ALM will be generated when some values of the 0~2 bits of FLAG are 1 in using the global function block. Open the G_A properties settings dialog, AEH and ALM will be shown in the alarm parameter setting (ENALM). If AEH and ALM are set as alarm enable (select the “Show Alarm Enable”), the AEH/ALM alarm will be generated when the values of 0~2 bits of FLAG are 1.

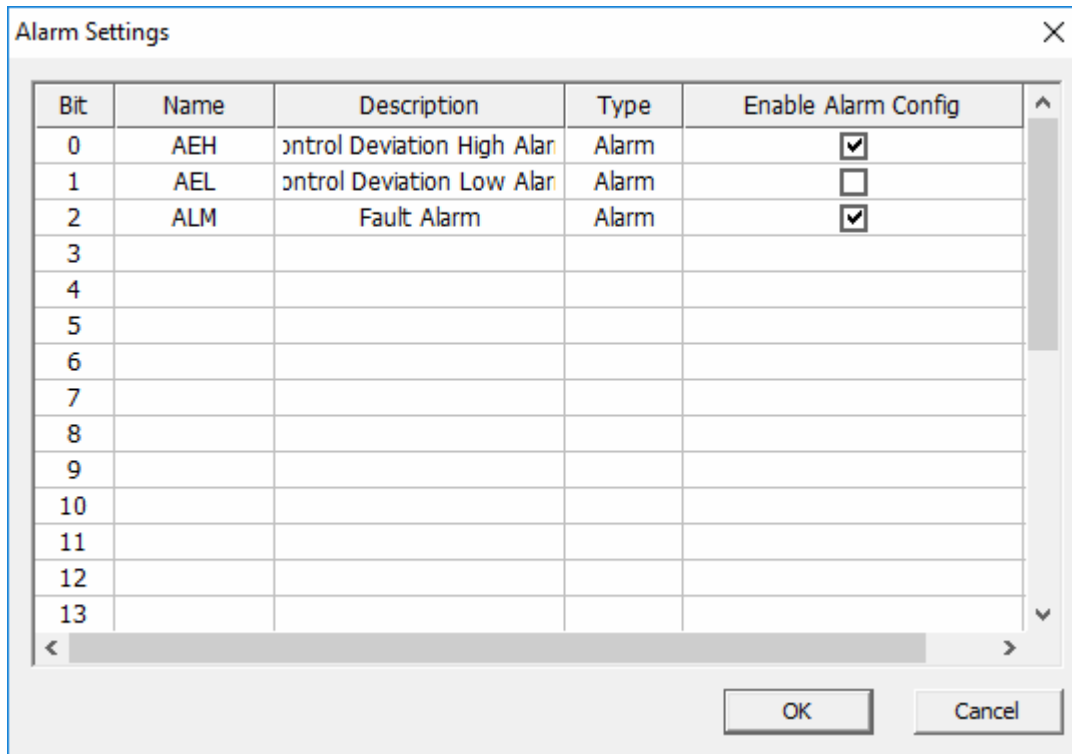


Figure 2-16 Alarm Settings Example

In the figure above, AEH and ALM will be shown in the alarm parameter settings when invoking global function block. If AEH and ALM are set as alarm enable and generate alarm, the 0 bit and 2 bit show the values of associated parameters individually.

2.5.3 Set Rule

Parameter rules configuration is used to define the binding relationship among the internal parameters in the global function block. Rules can be set for multicast parameters. The detailed steps are shown below.

1. Select "Parameter", right-click and select "Parameter Rule Settings" to open the "Parameter Rule Settings" dialog.
2. Select the parameter name in the pull-down menu. Parameter names can not repeat, or a prompt will pop up.
3. Select parameters for "Maximum Parameter", "Minimum Parameter" and "Unit Parameter". Maximum parameter and minimum parameter must be configuration parameters. The initial values can be input in the textbox according to the actual needs. Unit parameter can only be configured in configuration parameter. Select EUNIT for the configuration parameter to be set.
4. Repeat step 2 and step 3 to complete the configuration of all parameter rules.
5. Click "OK" to complete the operation.

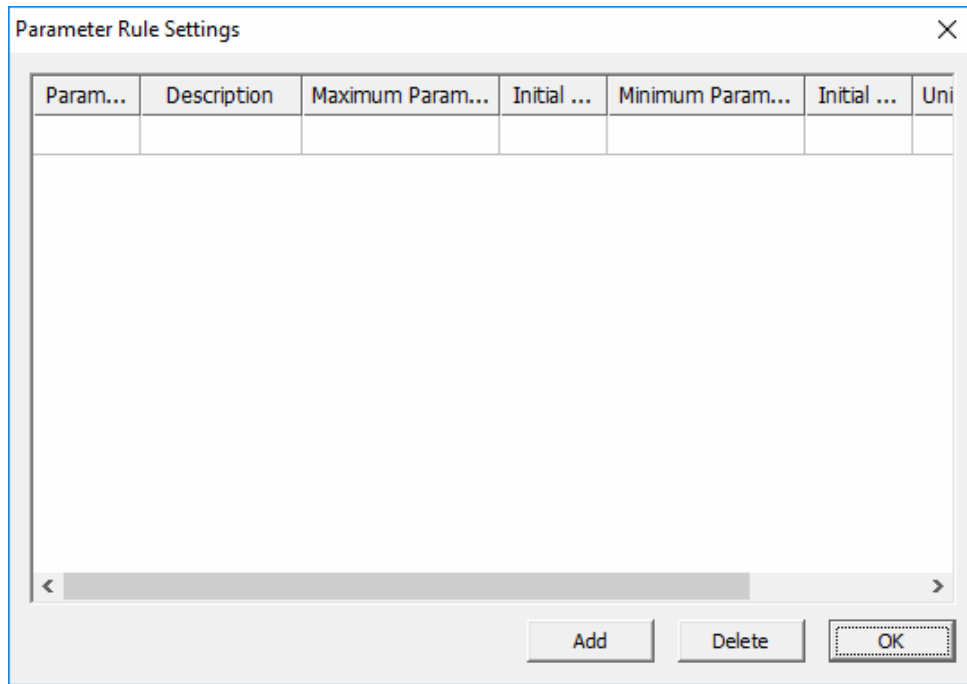


Figure 2-17 Parameter Rule Settings Dialog

2.6 Other Operations

2.6.1 Delete Parameter or Variable

Select parameter to be deleted in parameter type and choose “delete” in the right-click menu, as shown in Figure 2-18.

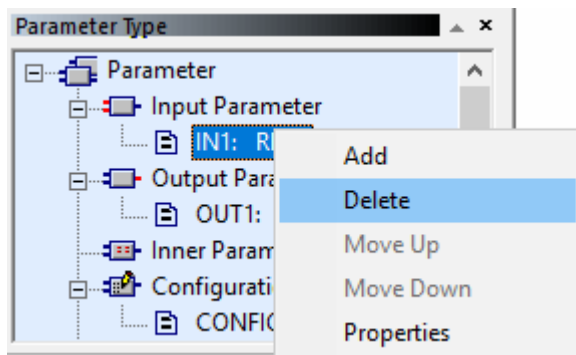


Figure 2-18 Delete parameter

Then the delete confirmation box will pop up, and select “OK” to delete the parameter.

In addition, users can also delete it by Delete key after selecting parameter in the window of parameter type.

Other types of variable can be deleted in the same way.

**Tip:**

When the function block has been excerpted by user program, parameters can't be added or deleted except temporary variable and reference variable, but the inner logic can be modified.

2.6.2 Parameter Dragging Function

In the parameter type window, you can realize the order adjustment among parameters of the same type by dragging them as well as the type switching among parameters of different types.

- Dragging function to parameters of different types

Input parameters, output parameters, built-in parameters, configuration parameters, operation parameters and temporary variables can exchange each other's type by the dragging function.

- Dragging function to parameters of the same types

Input parameters, output parameters, built-in parameters, configuration parameters, monitor parameters, operation parameters, temporary variables, reference variables, alias variables and parameters inside the function block variables can exchange each other's order by the dragging function. After dragging them, you can check the newest order in the "Parameter List".

Select the parameter needed, press the left mouse button and drag it to the target node, and confirmation dialog box pops up, click "OK" to complete the dragging operation.

2.6.3 Import and Export the Custom Function Block Program

The software supports importing and exporting the custom function block program, which is convenient for users to reuse and backup the program.

Users can export the custom function block program in the format of STF. or import the backup program.

Export

Click "File" and select "Export" to pop up the following dialog.

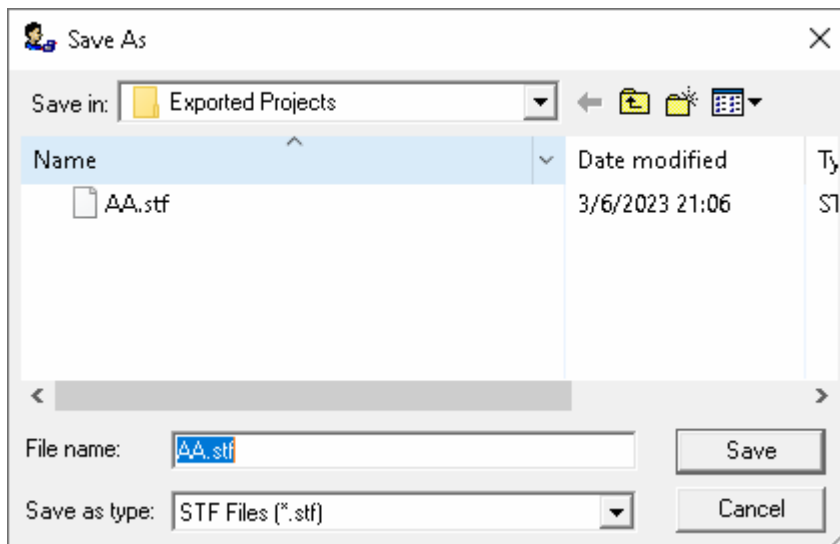


Figure 2-19 Export Dialog

Select the directory and enter the file name. Click “Save” to complete the operation.

Import

Click “File” and select “Import” to open the following dialog.

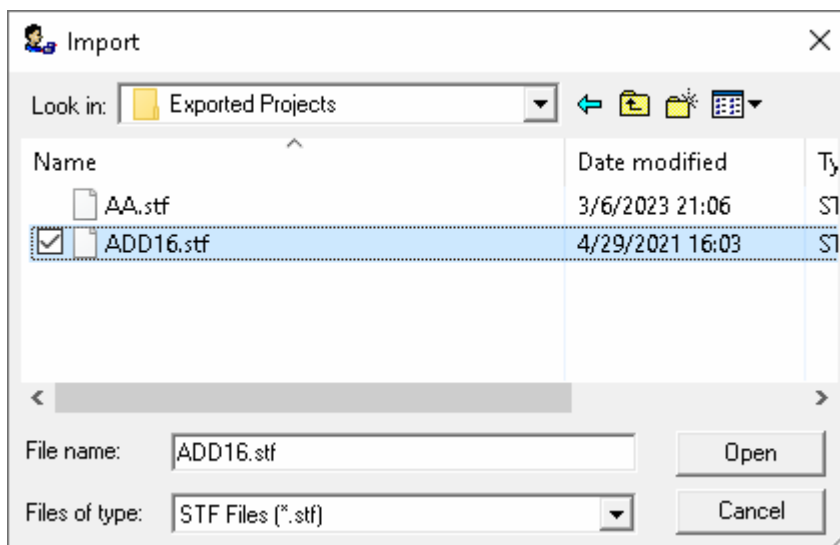


Figure 2-20 Import Dialog

Select an STF file, click “Open” and complete the operation.



Tips:

- The importing cannot be performed if the custom function block has been referenced to the custom program.
- The suffix of custom function block file programmed by SFC language is `sfcl`.

-
- The suffix of custom function block file programmed by ST language is stf.
-

2.6.4 Arrange Memory

There is space in memory table of global function block parameter because of that the last parameter (or function block variable) is in the rear of memory. Rearrange function counts the memory occupied space of single global function block for the memory address of parameter (or function block variable).

1. Click **Program/ Memory Arrangement** to pop up the dialog below.

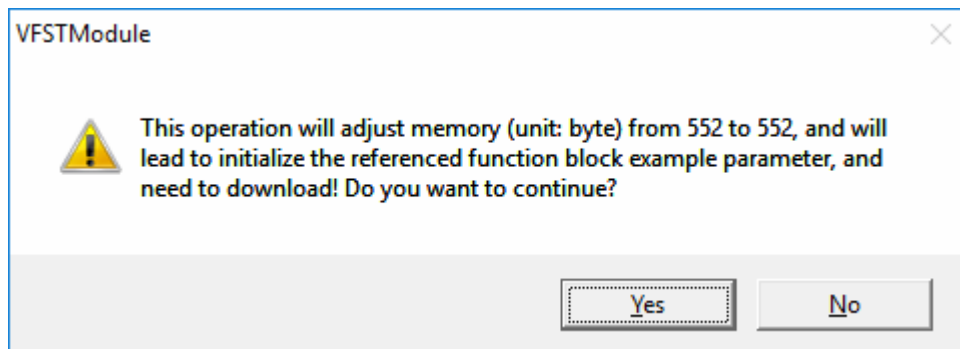



Figure 2-21 Arrange memory

2. Click **Yes** to arrange memory automatically. Click **Save** or **Compile** to complete the operation.

2.7 Compile

The compile of user function block can be divided into two types, one is compiling in custom function block software to check the syntax; the another is compiling in configuration management software, producing code file that can be operated by controller.

2.7.1 Compile in Custom Function Block Software

Click "Compile" icon  on the toolbar to compile current custom function block, and the syntax check of ST language is provided to prompt error locations. Compile information will be displayed in output bar.

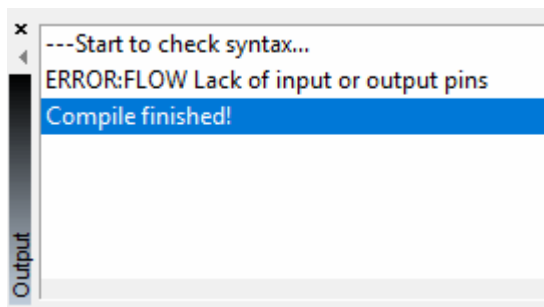


Figure 2-22 Compile information of single custom function block

The cursor can locate to location of the error code after double click error prompts line on output bar.

2.7.2 Compile in Configuration Management Software

The second method is to compile all custom function blocks within a single station in configuration management software, and check syntax of all function blocks and produce corresponding operation file. Select the “Custom Function Block” node of a control station and choose “Compile” in the right-click menu:

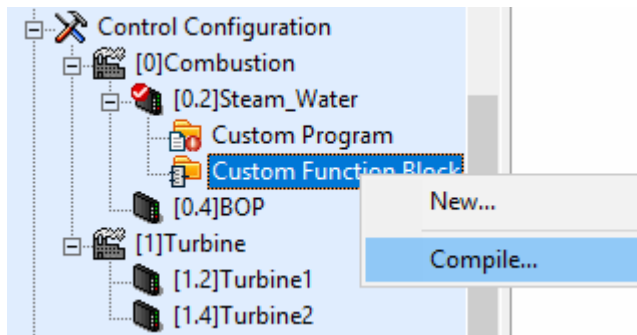


Figure 2-23 Compile all the function blocks in single station

All custom function blocks in the control station can be compiled, and compile information will be displayed in the output window of configuration management software.



Tips:

Custom function block can be excerpted in FBD program only when passing the compiling mentioned above.

2.8 Excerpt Custom Function Block

When all custom function blocks in a single station have passed compile, users can click pull-down button and select “Custom” in the window of function block library in FBD programming software interface. If the compile is not passed, item of “Custom” will not exist.

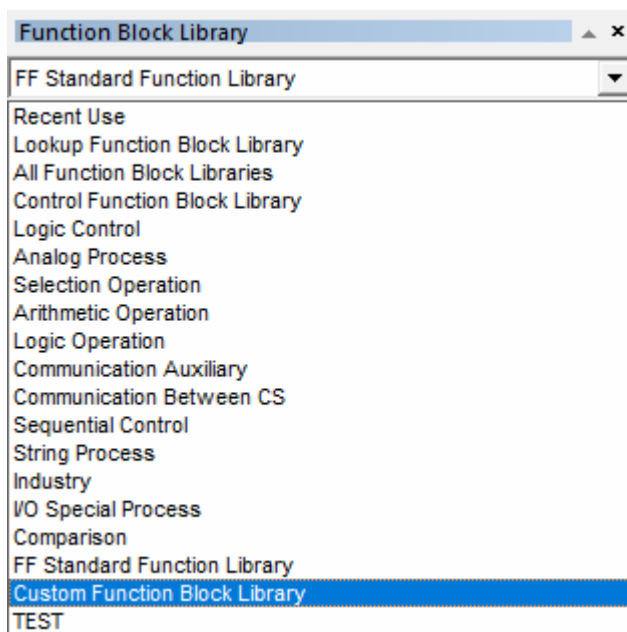


Figure 2-24 custom function block library in FBD programming software

All the custom function blocks will be listed below after selecting “Custom”, users can select and excerpt them according to actual requirements.

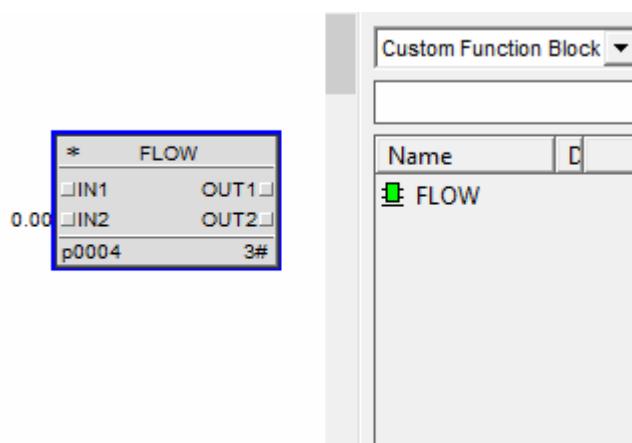


Figure 2-25 Invoke custom function block

Section 3 Global Function Block Logic Editing

The software can perform the editing for logic of global function block.

- Operations for parameter and variable of global function block refer to 2.4 Add Parameter or Variable.
- Compiling for the global function block refer to the 2.7Compile.

Users can use the global function block in any control station, and perform the operations of creating, properties settings (such as panel and symbol) by VFExplorer. Details refer to *Config Explorer User Manual 3.2 Global Function Block Configuration*.

Global function block can be imported to global function block folder, details refer to *Explorer User Manual 3.2.5 Other Operations of Global Function Block Folder*.

Global function block can be exported in the format of .ex, details refer to *Explorer User Manual 3.2.6 Other Operations of Global Function Block*.

Section 4 ST Language Programming

The code programming environment of ST language is provided, which conforms to the rule of ST language, and the colored display of system function, variable name, keyword, annotation and automatic complement (lowercase to capital letter) are available, system also provide bookmark function, find/ replace function in code editing area, etc.

4.1 Programming foundation

4.1.1 Data Type

8 basic data types are provided for custom function block.

Table 4-1 Data type

Type	Bit	Scale
BOOL	8	ON, OFF
SINT	8	-128 ~ 127
USINT	8	0~ 255
INT	16	-32768 ~ 32767
UINT	16	0~65535
DINT	32	-2147483648~2147483647
UDINT	32	0~4294967295
REAL	32	Floating point number
LREAL	64	Double-precision floating-point number
STRING	512	-

Set data type of each parameter according to various requirements; please refer to introduction of data type settings in "Data Type".



Tips:

- LREAL is only available for controller FCU811-S in custom function block.
- EUNIT is not data type configuration, and only configuration parameter can be set.

4.1.2 Function and Keyword

ST language supports system functions provided by system function block library (the definition statement of function is provided by Function.xml under installation directory), the functions are displayed with different color and all automatic complements are capital case. Please refer to appendix for system functions.

The syntax keyword defined by ST language is shown below. Keyword is displayed in blue by default.

Table 4-2 Keyword

Keyword	Function
WHILE;DO;END_WHILE	make WHILE sentence
FOR;TO;BY; DO; END_FOR	make FOR sentence
IF;THEN;ELSEIF;END_IF	make IF sentence
ELSE	make IF or CASE sentence
REPEAT;UNTIL;END_REPEAT	make REPEAT sentence
CASE;OF;END_CASE	make CASE sentence
EXIT	Exit sentence
AND;NOT ;MOD;OR;XOR	Operator
ON;OFF;TRUE;FALSE	BOOL type value
RETURN	Keyword of ST language, which is usually not used in user function block

4.1.3 Operator

Operators provided by ST language and their priority are shown below.

Table 4-3 Operator

Serial No.	Operator Name	Operator	Priority
1	Bracket operator	()	1
2	Invoke Function operator	SIN()	2
3	Logic not operator	NOT, ~	3
4	Multiply operator	*	5
5	Mod operator	MOD	5
6	Divide operator	/	5
7	add operator	+	6
8	subtract operator	-	6
9	greater-than operator	>	7
10	greater-than or equal-to operator	>=	7
11	less-than operator	<	7
12	less-than or equal-to operator	<=	7
13	not-equal-to operator	<>	8
14	equal operator	=	8
15	AND operator	AND, &	9
16	XOR operator	XOR	10
17	OR operator	OR	11

4.1.4 Constant and Annotation

Constant

Constants in ST language can be denoted in several ways.

Table 4-4 Constant denotation

Serial No.	Type	Denotation (examples)
1	Bool type	ON, OFF ,TRUE,FALSE
2	Floating-point type	1.3 or 1.3E5
3	Integer type	10 or 8#10 (octal 10) or 16#EF(hexadecimal EF)

Annotation

Annotation is denoted by (**), content of annotation is displayed in color (green by default), nesting is not allowed in annotation, and it supports annotations with several lines and brackets.

```
(*Annotation starts here,  
IF INNER1>INNER2 THEN  
OUT2:=ON;  
ELSE  
OUT2:=OFF;  
END_IF;  
Ends here*)
```

4.1.5 Syntax Rule

Assignment Statement

Assign the value of expression on the right of “:=” to variable on the left.

Example: OUT1 := IN1;

Function: assign IN1 to OUT1.

Selection Statement

- IF statement

IF statement defines that a group of statements will be executed only when corresponding logic expression is TRUE. If the logic expression is FALSE, these statements won't be executed or another group of statements set in ELSE (ELSEIF) will be executed.

■ Syntax

```
IF condition expression1 THEN  
... (*statement serial 1*)  
ELSEIF condition expression 2 THEN  
... (*statement serial 2*)  
ELSEIF condition expression 3 THEN  
... (*statement serial 3*)  
ELSE  
... (*statement serial 4*)  
END_IF  
... (*statement serial 5*)
```

■ Explanation

Execution steps:

Calculate condition expression1 follow IF, if the value is ON, then implement corresponding statement serial1, and then program will jump from IF block to statement serial5; if the value is OFF, jump to step 2;

If the program has ELSEIF (several ELSEIF branches are available), it will calculate condition expression2 follow ELSEIF, if the value is ON, then implement corresponding statement serial2, and then program will jump from IF block to statement serial5; if the value is OFF, jump to step 3;

Calculate condition expression2 follow ELSEIF, if the value is ON, then implement corresponding statement serial3, and then program will jump from IF block to statement serial5; if the value is OFF, program will implement statement serial4 follow ELSE, and then jump from IF block to statement serial5.

■ Example

Find the maximum value among A, B, C which are of different value, and assign the maximum value to x:

...

IF A > B THEN

x := A;

ELSE

x := B;

END_IF;

IF x < C THEN

x := C;

END_IF;

...

● CASE statement

CASE statement defines the variable values with integer type and several statement groups when the variable value is different. When variable value is equal to certain value, corresponding statement will be executed, and when there's no matched value, the program will execute statement group in ELSE (ELSE branch is defined in CASE statement).

■ Syntax

CASE Integer variable OF

1 :

*... (*sentence serial1*)*

```

2 :
...  (*statement serial 2*)
ELSE
...  (*sentence serial 3*)
END_CASE;
...  (*statement serial 4*)

```

■ Explanation

Execute corresponding statement serial according to the value of integer variable, when no corresponding value is found, the program will execute the statement serial follow ELSE.

■ Example

```

...
CASE IN1 OF
1: OUT1 := 1;
2: OUT1 := 2;
ELSE OUT1 := 3 ;
END_CASE;
...

```

● Loop Statements

Loop statements includes FOR statement, WHILE statement and REPEAT statement. They all have end condition and a group of statements, and when the end condition is not TRUE, the group of statement will be executed in loop. Loop will end when the program executes EXIT statement. When there is nesting in loop, EXIT statement will only means to exit the loop of layer in which EXIT lies. Function of “wait for synchronization” is not allowed in loop statement.

● FOR statement

■ Syntax

```

FOR I := 1 TO 100 BY 2 DO
    DOSOMETHING();
END_FOR;

```

■ Explanation

In FOR statement, I is control variable, 1 is initial value, 100 is end value, 2 is step value. Control variable, initial value, end value and step value in FOR statement must be integer type. Step value is 1 by default.

First assign an initial value to control variable, and then judge the end condition, and when the value of control variable is greater than end value, the corresponding statement serial won't be executed at all.

When the value of control variable is less than or equal to end value, corresponding sentence serial will be executed, and the control variable will accumulate by step value. And then, the judgment of end condition will continue, and once it is achieved, the loop will end, or operations above will be executed round and round.

- WHILE statement

- Syntax

```
WHILE condition DO  
    ...    (*statement group*)  
END_WHILE;
```

- Explanation

First judge the condition, and when it is FALSE, corresponding statements won't be executed at all. When it is TRUE, the statement serial will be executed.

After executing statement serial, the judgment of condition will continue, and when it is TRUE, the loop will continue, or it will end.

- REPEAT statement

- Syntax

```
REPEAT  
    ...    (*statement group*)  
UNTIL end condition  
END_REPEAT;
```

- Explanation

First execute the statement group once, and then judge the end condition, therefore, corresponding statement group will be executed at least once. When end condition is TRUE, the loop will end, or it will continue to circulate.

EXIT and EMPTY

- EXIT statement

EXIT statement is used to end loop statement before the end condition becomes TRUE. (end the FOR, WHILE, REPEAT)

If EXIT statement is in nesting loops, the inner-most circuit will exit (when EXIT is located here), then the first statement after loop will be executed. (execute the END_FOR, END_WHILE or END_REPEAT).

Example: if the value of FLAG is 0, SUM will be 15 after execution. If the value of FLAG is 1, SUM will be 6.

```
...
SUM := 0;
FOR I := 1 TO 3 DO
  FOR J := 1 TO 2 DO
    IF FLAG = 1 THEN
      EXIT;
    END_IF;
    SUM := SUM + J;
  END_FOR;
  SUM := SUM + I;
END_FOR;
...
```

- EMPTY statement

Empty statement is produced by semicolon “;”.

Example:

```
SUM := SUM + 1;
; (*empty statement*)
; (*empty statement*)
SUM1 = SUM + SUM2;
```

4.1.6 Timer

Second timer TIMERS and 100ms timer TIMERMS are available for FCU711-S; Second timer TIMERS, 100ms timer TIMERMS and minute timer TIMERM are provided for FCU712-S and FCU713-S.

Table 4-5 Timer type and relevant information

Type	Expression	Range of n	Data type	Time Range
Second timer	TIMERS[n]	0~511	UINT	0~65535, unit is seconds.
100 ms timer	TIMERMS[n]	0~127	UINT	0~65535, unit is 100 milliseconds.
Minute timer	TIMERM[n]	0~127	UINT	0~65535, unit is minutes.

Example:

```
Assign IN with ON, 20s later, OUT will be ON;
IF IN = ON AND TIMERS[1] >= 20 THEN
  OUT := ON ;
END_IF ;
```

**Tip:**

Timers with the same number can't be excerpted by user function block programs of different serial numbers. For example, user function block programs A and B can't excerpt timer TIMERS[1] simultaneously.

4.1.7 Array Parameters

ST user function blocks and ST global function blocks support array parameters. When the logic of ST user function block and ST global function block needs to refer to array parameters, the form of "parameter name [array member subscript 1, array member subscript 2]" is required, and the subscript of array member supports digital constant And variables.

- Array parameters

```
ARTINT[3,3] := 1.0;
```

```
OUT2 := ARTINT[3,2];
```

Among them, the subscript of the array starts with 0 and supports integer constants, integer variables or expressions. For example, ARTINT[X, X+Y], where X and Y are integers.

- Function block variables support function block types that contain array parameters:

```
TAG1.ARTINT[3,3] := 1.0;
```

```
OUT2 := TAG1.ARTINT[3,2];
```

As shown in the figure below, the monitoring parameter "L" of a user function block is a one-dimensional array parameter with a dimension of 32. When quoted in the program code, the form of "L[I]" can be used.

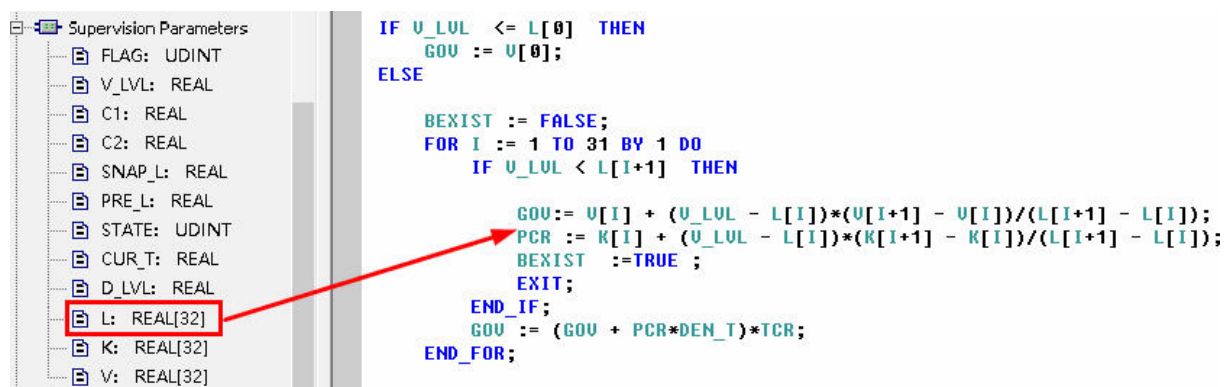


Figure 4-1 Array parameter's application examples

4.1.8 Global Array

3 Types of global array are supported. Every type of array contains most 32 rows*128 columns

Array Definition

The array supported by High-performanceHMI are described in the following table.

Table 4-6 Global Array Supported by High-performanceHMI

Array Mark	Array Type	Element Type
ARRREAL[X,Y]	32 rows*128 columns (X=0~31, Y=0 ~ 127)	REAL
ARRBOOL[X,Y]	32 rows*128 columns (X=0~31, Y=0 ~ 127)	BOOL
ARRUINT[X,Y]	32 rows*128 columns (X=0~31, Y=0 ~ 127)	UINT
ARRBOOLEX[X,Y]	32 rows*32columns (X=0~31, Y=0 ~ 31)	BOOL
ARRREALEX[X,Y]	32 rows*32columns (X=0~31, Y=0 ~ 31)	REAL



Attention:

Do not use global arrays over range, and do not set value repeatedly.

Example

- Assign value to every element of the array, output the value of the element in the 20th row, 50th column of the array.
 - When the following conditions are met, output ON, otherwise output OFF.
 - When the value of the element in 10th row, 10th column of REAL type array is equal to 50.0;
 - When the value of the element in 10th row, 10th column of UINT type array is equal to 50;
 - When the value of the element in 10th row, 10th column of BOOL type array is equal to ON;
- New a custom function block program, named as ARR, and enter the interface of the custom function block program
 - Add parameters and variables

Add input parameters:

 - Name: IN_UINT; Type: UINT
 - Name: IN_BOOL; Type: BOOL
 - Name: IN_REAL; Type: REAL

Add output parameters:

 - Name: OUT; Type: BOOL
 - Name: OUT_UINT; Type: UINT

- Name: OUT_BOOL; Type: BOOL

- Name: OUT_REAL; Type: REAL

Add temporary variables:

- Name: X; Type: INT

- Name: Y; Type: INT

- Code edit

Input code as follows in the programming interface:

```
FOR X: =0 TO 31 BY 1 DO
```

```
  FOR Y: =0 TO 127 BY 1 DO
```

```
    ARRREAL [X,Y] := IN_REAL;
```

```
    ARRBOOL [X,Y] := IN_BOOL;
```

```
    ARRUINT [X,Y] := IN_UINT;
```

```
  END_FOR;
```

```
END_FOR;
```

```
OUT_BOOL: = ARRBOOL [19,49];
```

```
OUT_UINT: = ARRUINT [19,49];
```

```
OUT_REAL: = ARRREAL [19,49];
```

```
IF ARRREAL [9, 9] = 50.0 AND ARRBOOL [9, 9] = ON AND ARRUINT [9,9]=50
```

```
THEN
```

```
  OUT: = ON;
```

```
ELSE
```

```
  OUT: = OFF;
```

```
END_IF;
```

Configuration interface is shown as follows.

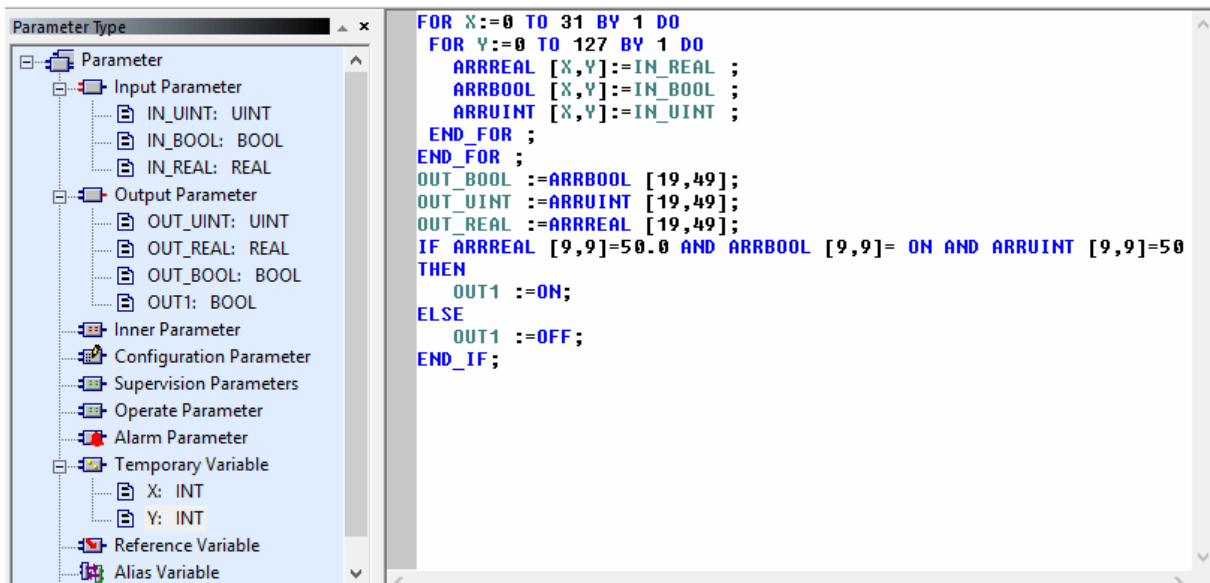


Figure 4-2 Custom function block Completes

Compile

After syntax check passed successfully, exit from VFSTModule, and compile the block in VFExplorer.

Function block reference

Add a custom program of FBD type, and enter the interface of FBD custom program. Then select ARR function block in custom function block library, connect the input/output pins to the tags, and compile, shown as Figure 4-3.

Suppose that the following tags have been in the tag table:

Name: ARR_UINT; Type: custom integer tag (UINT)

Name: ARR_REAL; Type: custom analog tag

Name: ARR_BOOL; Type: custom digital tag

Name: ARR_OUT; Type: custom digital tag

Name: ARR_OUT_UINT; Type: custom integer tag (UINT)

Name: ARR_OUT_REAL; Type: custom analog tag

Name: ARR_OUT_BOOL; Type: custom digital tag

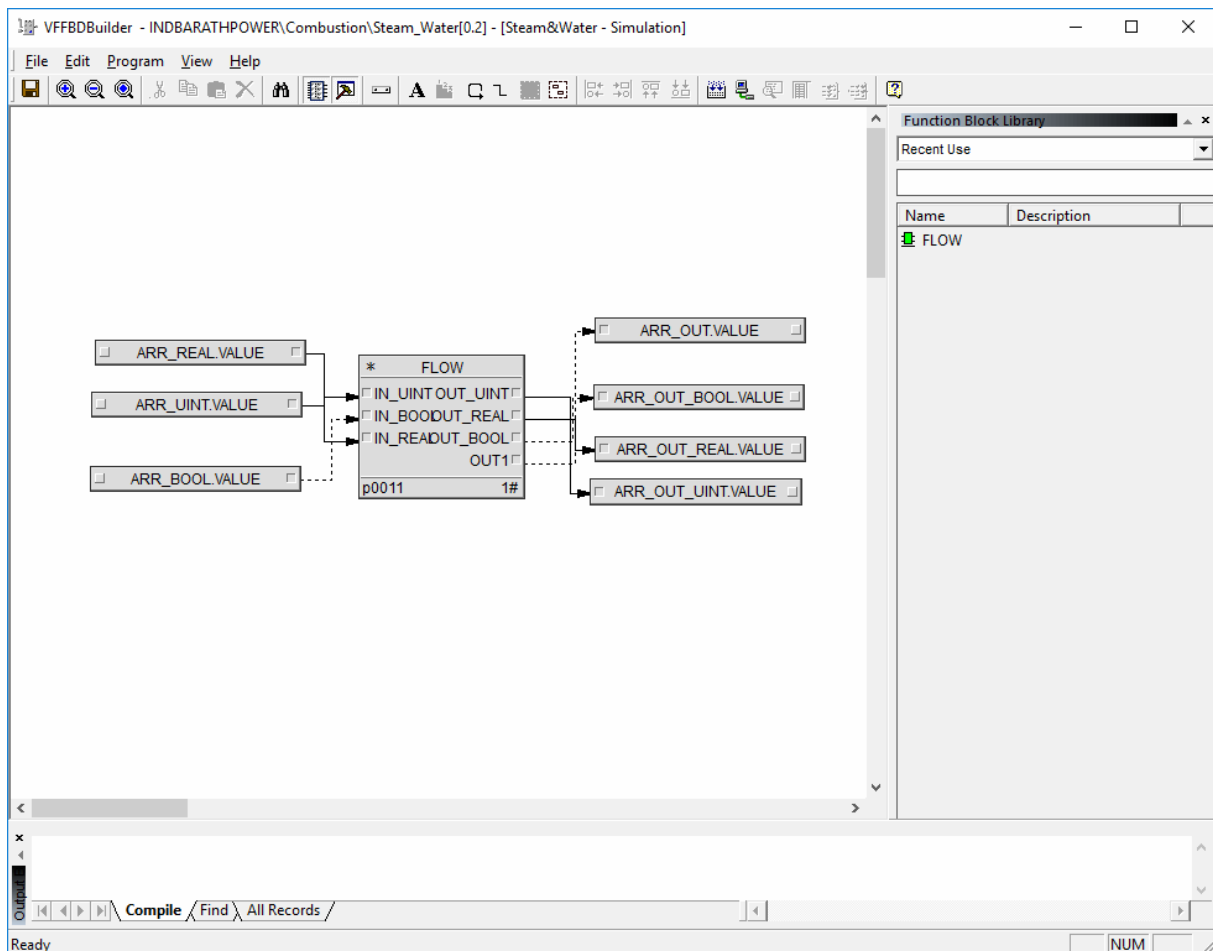


Figure 4-3 Custom function block reference

Download

Execute download operation in VFExplorer.

Debug

After successfully downloading, enter the interface as shown in Figure 4-3, to execute the function block debug. Click the tag and assign 50, ON, 50.0 to the input respectively. The output is shown in Figure 4-4.

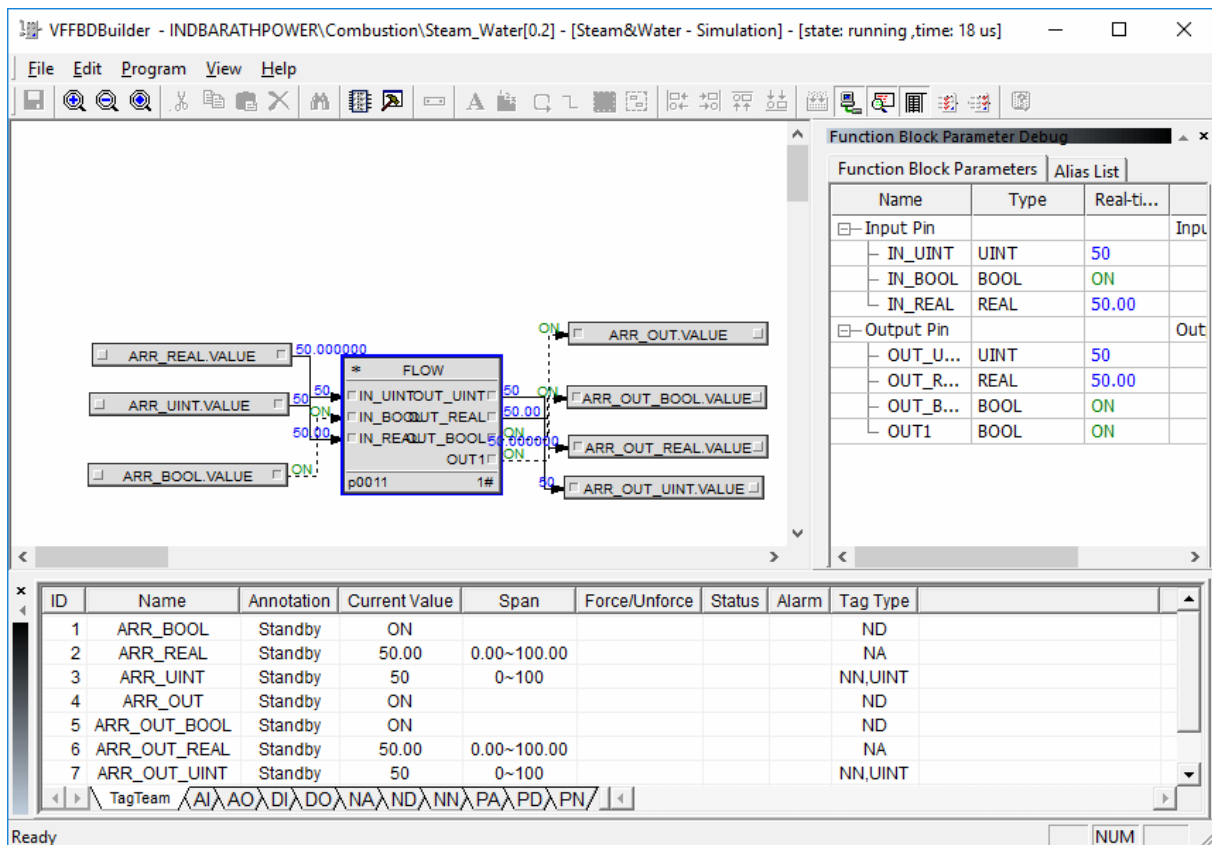


Figure 4-4 Function block debug

**Tip:**

Details for program debug refer to *FBD Programming Software User Manual 6.5 Online Debug*.

4.2 Auxiliary Function

4.2.1 Bookmark

If the logic of function block is very complicated and the code is very long, bookmark could help users to mark the specific location of the code, making it easy to find the specific row.

After setting bookmark in specific row, user could find the row with the function *Next Bookmark/Previous Bookmark* in edit menu or toolbar. Bookmark includes simple bookmark and shortcut bookmark.

Simple bookmark

Simple bookmark is used to mark the specific row in the code, and be found with the function *Next Bookmark/Previous Bookmark*. The operation step is as follows.

Select the row that needs to mark, click the *Set Bookmark* button  in toolbar, or select menu

Edit/ Bookmark

The row set with a bookmark has a blue symbol before it, shown as follows.


```

FOR X:=0 TO 31 BY 1 DO
  FOR Y:=0 TO 127 BY 1 DO
    ARRREAL [X,Y]:=IN_REAL ;
    ARRBOOL [X,Y]:=IN_BOOL ;
    ARRUINT [X,Y]:=IN_UINT ;
  END_FOR ;
END_FOR ;
OUT_BOOL :=ARRBOOL [19,49];
OUT_UINT :=ARRUINT [19,49];
OUT_REAL :=ARRREAL [19,49];
IF ARRREAL [9,9]=50.0 AND ARRBOOL [9,9]= ON AND ARRUINT [9,9]=50
THEN
  OUT1 :=ON;
ELSE
  OUT1 :=OFF;
END_IF;

```

Figure 4-5 Simple bookmark display

The bookmark could be found with the function *Next Bookmark/Previous Bookmark* in the edit menu or toolbar.

Select the row that be marked, click the *Set Bookmark* button , or select menu **Edit/Bookmark**, the bookmark would be cancelled.

Shortcut bookmark

Shortcut bookmark is also used to mark the specific location of code. It could be marked with number, which differs from simple bookmark. Setting steps is as follows.

Move the cursor to the row that needs to mark, press *Ctrl + Number*, and the bookmark would be set. For example, set a bookmark with NO. 2 before certain row: move cursor to the row and press *Ctrl+2* (press *Ctrl+2* again to cancel the shortcut bookmark), the shortcut bookmark would be set, shown as follows.

```

FOR X:=0 TO 31 BY 1 DO
  FOR Y:=0 TO 127 BY 1 DO
    ARRREAL [X,Y]:=IN_REAL ;
    ARRBOOL [X,Y]:=IN_BOOL ;
    ARRUINT [X,Y]:=IN_UINT ;
  END_FOR ;
2|END_FOR ;
OUT_BOOL :=ARRBOOL [19,49];
OUT_UINT :=ARRUINT [19,49];
OUT_REAL :=ARRREAL [19,49];
IF ARREAL [9,9]=50.0 AND ARRBOOL [9,9]= ON AND ARRUINT [9,9]=50
THEN
  OUT1 :=ON;
ELSE
  OUT1 :=OFF;
END_IF;

```

Figure 4-6 Setting and display of shortcut bookmark

After the shortcut bookmark is set, press *Alt +Number* to jump to the corresponding row marked with bookmark.



Tip:

These functions Clear All Bookmarks, Next Bookmark and Previous Bookmark in toolbar are only available to simple bookmark, not to shortcut bookmark. The simple bookmarks and shortcut bookmarks those have been set in the code can be saved, and would be seen next time starting the software.

4.2.2 Programming Assistant

Programming assistant helps user to write the logic code of custom function block easily. The special information of functions, variables, keywords, etc. could be found in programming assistant.

For example, input letter “I” in the code edit area, and programming assistant will pop up. Then the cursor would be located in the item beginning with “I”, shown as follows.

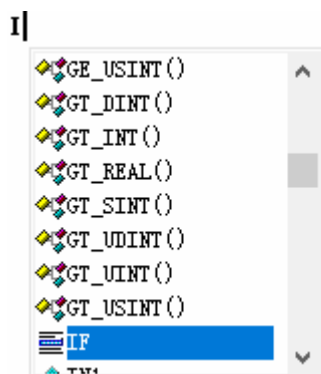


Figure 4-7 Programming Assistant

Users could select the item they need in programming assistant.

4.2.3 Color settings

Users are provided to set the color of key word, function, parameter and note for the code convenient reading. Select menu command **Edit/Color Settings**, and the Color Settings dialog box pops up, shown as follows.

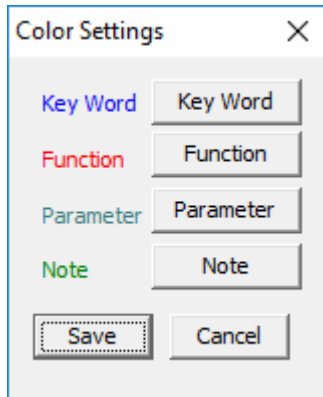



Figure 4-8 Color Settings

Click the corresponding button of key word, function, parameter, note, and the color window will pop up. Select the color, then click “OK”. After all items settings completion, click “Save” to save settings.

4.2.4 Find

Select menu command **Edit/Find** or click the button  in toolbar, the dialog box shown as follows pops up.

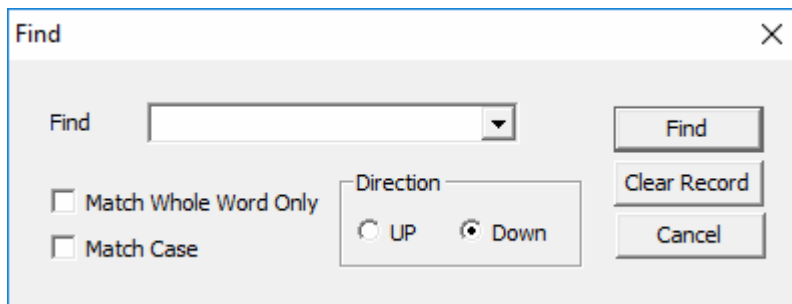




Figure 4-9 Find dialog box

Input the key word to be found, set finding conditions, and click button “Find”.


Finding conditions include:

- Match Whole Word Only: The information to find and input key word is matched completely.
- Match Case: Match the case of the letters.
- Direction: Find the information upward or backward from the place the cursor located.

The system saves the key word having been set. Click the pull-down button beside the edit box of key word, and select the previous key word having been set. Users also could click “Clear Record” to clear all the key words saved by the system.

After finding the first information matched with key word, users are able to click the buttons *Find Next*  and *Find Previous*  to find the other matched information.

4.2.5 Replace

Select menu command **Edit/Replace** or click the button  in toolbar, the dialog box shown as follows pops up.

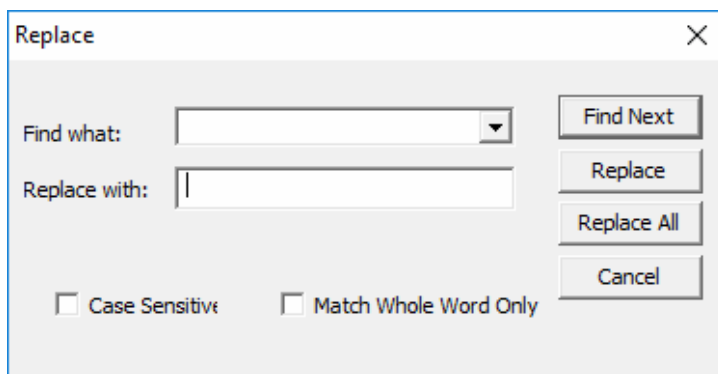


Figure 4-10 Replace dialog box

Users could replace the key words one by one after finding them.

Support replacing all. After input the key word and content to replace, click “Replace All” to replace all the key words in one time.

4.3 Notes for Application

- **Type match:** Parameter types on both sides of assignment and comparison statements must match each other, automatic type switch is not provided by compiler, and users can switch type by function such as `SINT_TO_DINT ()` provided by system.
- **Condition statement:** all condition statements can't add semicolon in the end. Special attention should be paid to UNTIL statement: semicolon cannot be added to the end of condition statement.
- **FOR statement:** the control variable, initial value, condition judgment value and step value of FOR statement must be integers. Step value can't be 0 which may cause infinite loop.
- **Math operations like exponent:** users can use system function block function such as `POW()` to implement operations.
- Custom function block should have at least one input pin, one output pin, and codes are necessary.

- While modifying the applied custom ST program to add new alias variable, the original application of the custom ST program will be invalid. Delete the invalid application, and add the modified ST program again and connect.

4.4 Example

4.4.1 Example Description

Build a new selection function block named SELECT and input parameters: IN1 and IN2 (REAL type); output parameter OUT1 (REAL type) and OUT2 (BOOL type), inner parameter OP (BOOL type), reference variable ABC_0 (ND type), alias variable IN (BOOL type), function block variable TP1 (TP function block).

Function:

If inner parameter OP=ON, then OUT1=IN1, else OUT1=IN2.

If alias parameter IN=ON, then OUT2 will be ON and become OFF 20s later, else OUT2=OFF.

4.4.2 Operation Steps:

Build New custom function block

Select node “custom function block” in configuration tree of configuration management software, and select “New” command in its right-click menu:

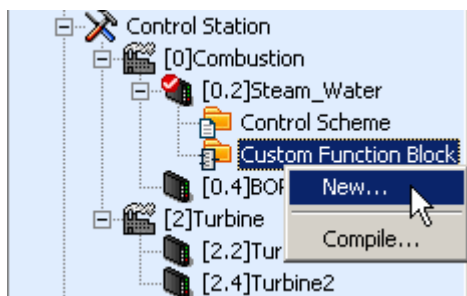


Figure 4-11 New

Dialog box of ‘Custom Function Block’ will pop up, as shown in Figure 4-12.

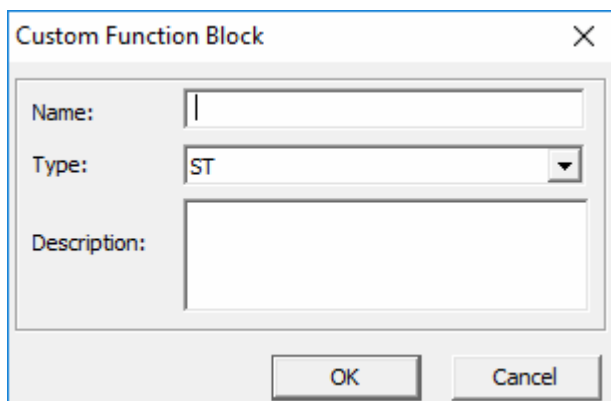


Figure 4-12 Window of “New Custom Function Block”

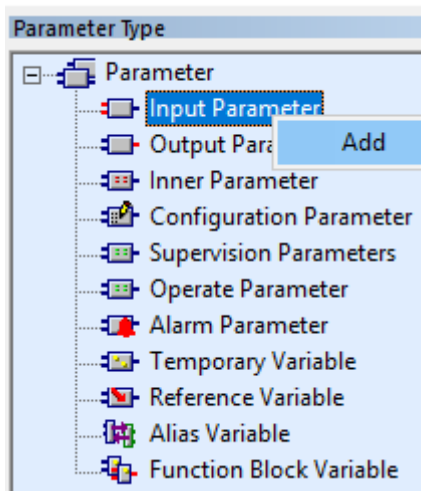
Select type of ST. Input name and descriptions, and then click “OK”.

Start up editing

In the window of configuration property list in configuration management software, double click function block program newly built to start up interface of custom function block software for editing.

Add parameter

Select node “Input Parameter” in the window of parameter type, and select “Add” in its right-click menu.

**Figure 4-13 Add Parameter**

Add two input parameters IN1 (REAL type) and IN2 (REAL type), two output parameters OUT1 (REAL type) and OUT2 (BOOL type), one inner parameter INNER1, one reference parameter Aabc_0 (ND type), one alias variable IN (BOOL type), one function block variable TP1 (TP function block).

Parameter property setting

Double-click INNER1 node in the window of parameter type to pop up window of parameter properties settings.

The screenshot shows a dialog box titled 'INNER1' with a 'Properties' tab. The 'Name' field contains 'OP'. The 'Data Type' dropdown menu is open, showing 'BOOL' selected. The 'Initial Value' field contains '0.0000'. Below this, there are several checkboxes: 'Array' (unchecked), 'Dim1' (unchecked), 'Dim2' (unchecked), 'Redundancy' (checked), 'Multicast' (unchecked), 'Upload' (unchecked), 'Default Pin' (unchecked), and 'Use Limit' (unchecked). To the right of 'Array' is an 'Initial Value' button. Below the checkboxes are three empty text boxes for 'Upper Limit', 'Lower Limit', and 'Unit'. At the bottom is a large empty text box for 'Description'. At the very bottom are 'OK' and 'Cancel' buttons.

Figure 4-14 Parameter property setting

Change name to OP, change type to BOOL, and click “OK”.

Add code and compile

Add codes shown below in editing area:

```

IF SOE_ND0002000.VALUE = ON THEN
    OP := ON ;
END_IF ;
IF OP = ON THEN
    OUT1 :=IN1;
ELSE
    OUT1 :=IN2;

```

```

END_IF;
    IF IN = ON THEN
        OUT2 := ON ;
    END_IF ;
    IF IN = OFF THEN
        OUT2 := OFF ;
    END_IF ;
    (*use TP to operate 20s timing to OUT2*)
    TP1.SET := IN ;
    TP1.DT := 20.0 ;
    TP1();
    OUT2 := TP1.OUT;

```

After it is saved and compile is passed, compile all custom function blocks of the station in configuration management software.

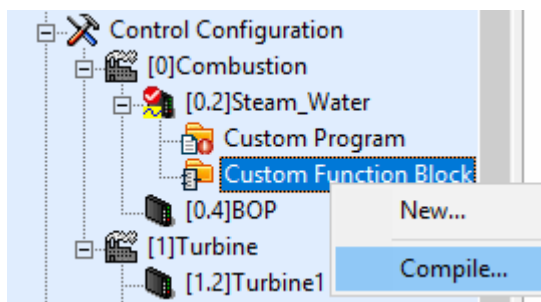


Figure 4-15 Compile of all custom function blocks in the station

Excerpt Function block

After all custom function block passed compile, the newly-built function block SELECT can be excerpted by FBD program.

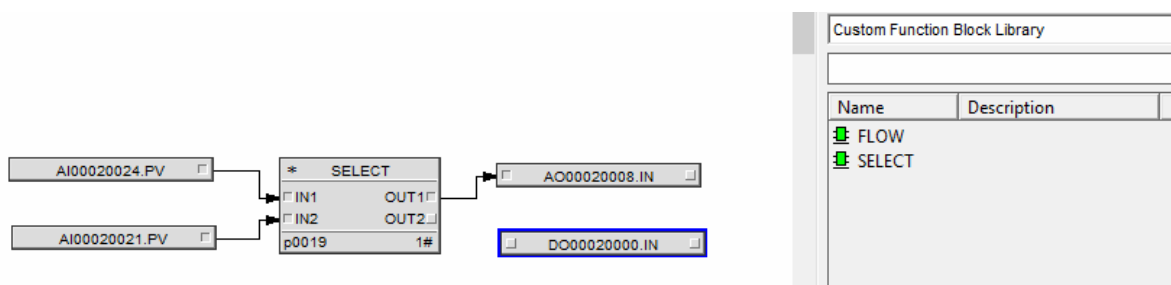


Figure 4-16 Excerpt custom function block

Reference Tag

Real tag for alias variable can be excerpted in property setting/ alias list of function block.

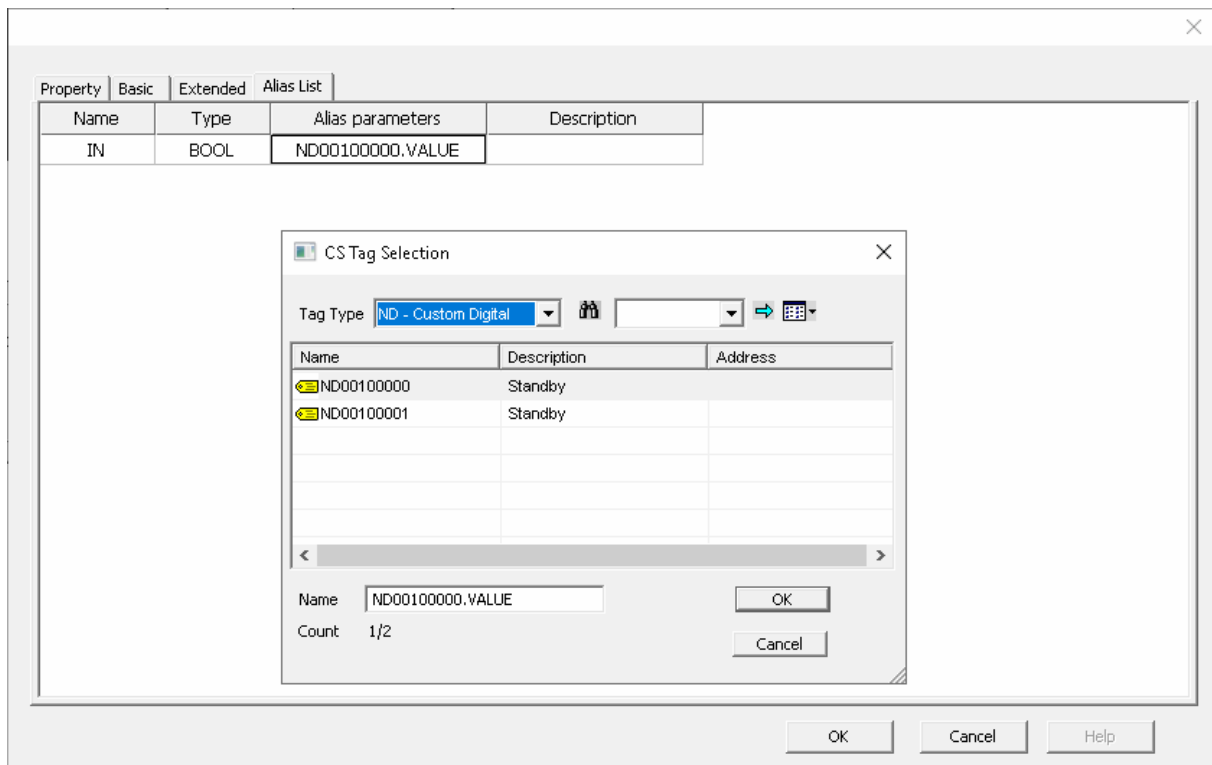


Figure 4-17 Alias tag associated with real tag

After FBD program is saved and passed compile, users can debug custom function block of the program.



Tips:

- Only the code logic of referenced custom function block can be modified (there if no limit for referenced global function block).
- If custom function block excerpted contains alias parameter, it must refer to real tag, or it won't pass the compile.

Section 5 SFC language programming

5.1 Programming Foundation

5.1.1 Elements of SFC Program

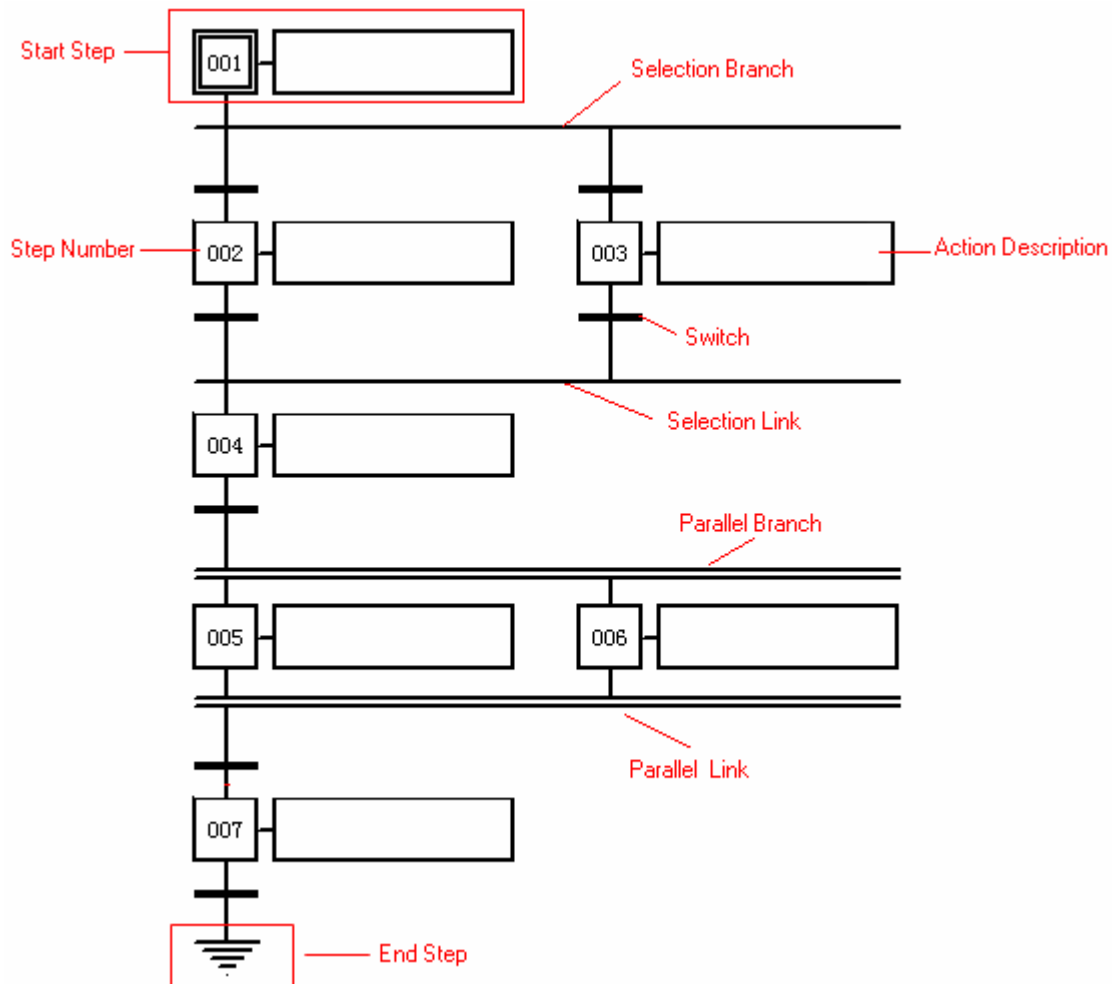
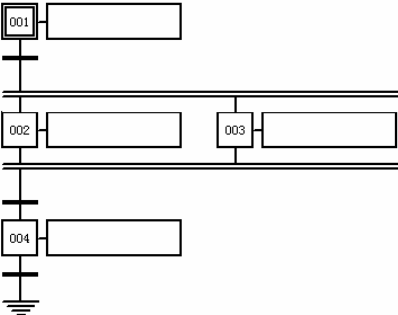
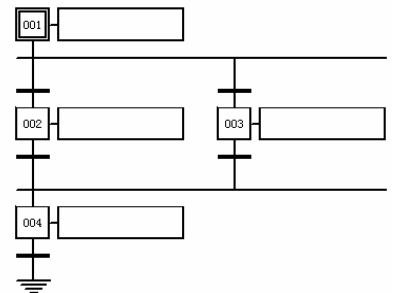
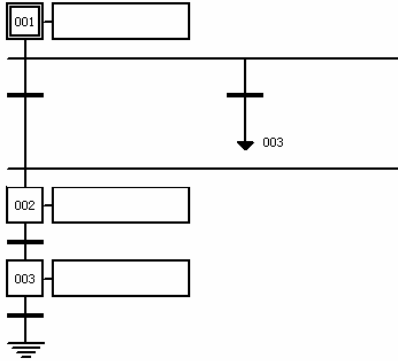
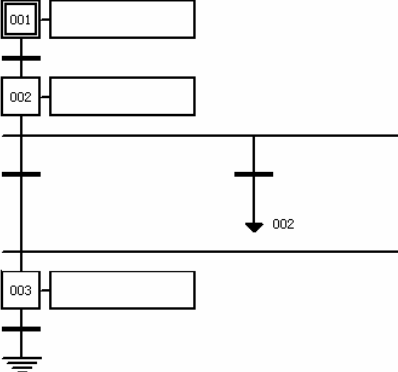


Figure 5-1 SFC Program

5.1.2 Program Structure

Table 5-1 SFC Program Structure

Program Structure	Structure Name
	Sequence Structure

Program Structure	Structure Name
	Parallel Structure
	Selection Structure
	Jump Structure
	Circle Structure

5.1.3 Programming Rule

- Supporting to sequence, parallel, selection, circle jump structure
- Rules of adding sequence step, branch and jump are shown as follows.

Table 5-2 Role of adding and deleting sequence step, branch, switch

Operation	Operation Conditions(Be Satisfied At the Same Time)	Operation Results
-----------	-----------------------------------------------------	-------------------

Operation	Operation Conditions(Be Satisfied At the Same Time)	Operation Results
Add Sequence Step	A. The current sequence step number does not reach the upper limit.	1) A switch and an sequence step would be added when selecting a sequence step. 2) An sequence step and a switch would be added when selecting a switch.
	B. The selected element is sequence step (including start step) or switch.	
Add Selection Branch	The selected element is sequence step (including start step).	A selection branch, selection link, an sequence step and a switch would be added.
Add Parallel Branch	A. The selected element is Switch.	A parallel branch, 2 sequence steps, a parallel link and a switch would be added.
	B. The amount of sequence steps of current program does not reach the upper limit. 2 or more steps are able to be added into the program.	
Delete Sequence Step	A. The selected element is sequence Step (except for start step and end step).	1) The selected sequence step and a switch would be deleted. 2) If the selected sequence step is in a parallel branch and there are not another 2 steps in the branch, the parallel branch would be deleted when the selected step is deleted.
	B. The element before the selected sequence step is not selection link.	
Delete Switch	A. The element before the selected sequence step is selection link.	If there remains a last branch after the selected switch being deleted, the selection branch would be deleted.
	B. The element after the selected sequence step is selection branch.	
Set jump	A. The selected element is switch.	Item <i>Conditional Jump</i> in <i>Switch Property</i> dialog box is available, and the Jump Objective could be pointed to the steps already in the program. (The step outside the parallel branch should not jump to the step inside the branch. And the step inside the parallel branch could only jump to the step inside the branch.
	B. The element after the switch is selection link	
	C. The selected branch is not the 1 st branch (the left branch).	

- The maximum number of sequence steps is 128. And each selection node and parallel node most supports 16 branches.
- Every branch could contain sub-branch, and there is no limit for the nesting. And the maximum number of steps should not over 128.
- The program logic in the step is programming in ST language.
- The expression of switch condition could be programmed in ST language. Arithmetic and Boolean calculation are supported by the system.
- State programming is supported, including RUNNING, PAUSING, RESUMING, STOPPING .

Tip:


Conversion condition sentence of SFC user function block cannot be edited by temporary variable.

5.2 SFC Program Running Rule

5.2.1 State Transition Graph

The State Transition Mechanism is shown as follows.

Function Block State Transition Graph

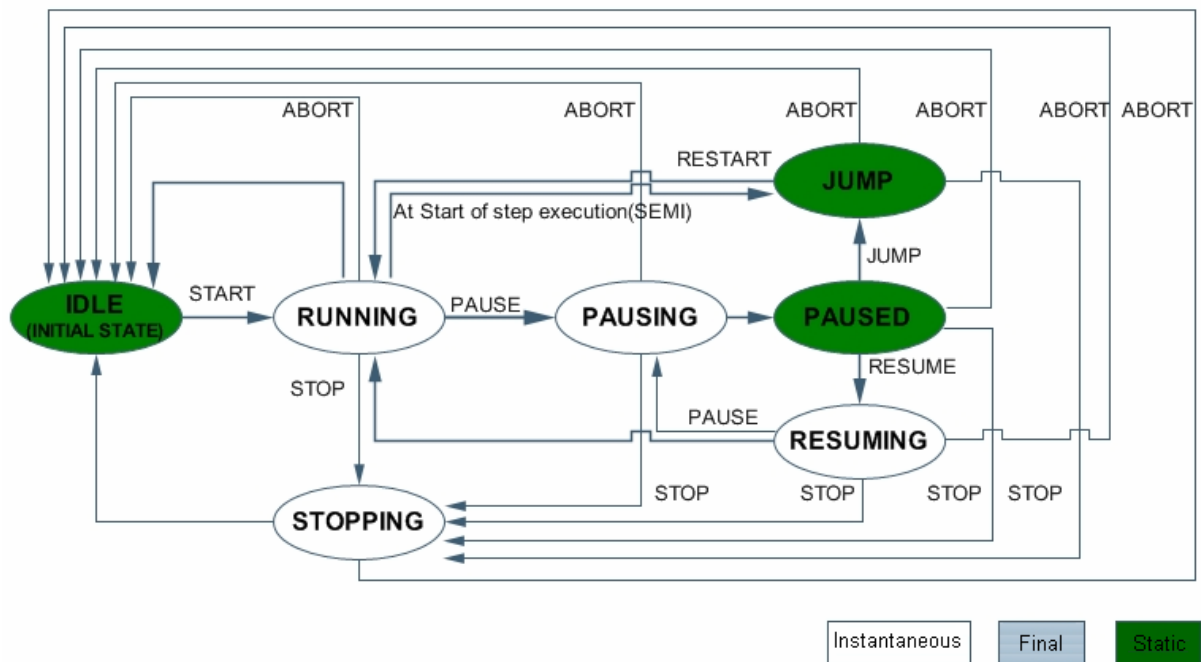


Figure 5-2 Function block state transition graph

4 state programming are corresponding to 4 instantaneous states.

State description

Table 5-3 State list

Name	Type	Description
IDLE	Static	Idlesse, Initial State
RUNNING	Instantaneous	Working State, Main Logical
PAUSING	Instantaneous	Pause
PAUSED	Static	Pause in Some State
JUMP	Static	Jump State
RESUMING	Instantaneous	Resume to Working State
STOPPING	Instantaneous	End

State transition within single layer

Table 5-4 Description to State transition within single layer

Current State	Command	State In Result	Annotation
---------------	---------	-----------------	------------

Current State	Command	State In Result	Annotation
IDLE	START	RUNNING	
RUNNING		IDLE	
	PAUSE	PAUSING	
	STOP	STOPPING	
	ABORT	IDLE	
	(SEMI)	JUMP	In Semi-automation mode, enter <i>jump state</i> after a sequence step is executed.
PAUSING		PAUSED	Execute In Sequence
	ABORT	IDLE	
	STOP	STOPPING	
PAUSED	JUMP	JUMP	
	RESUME	RESUMING	
	STOP	STOPPING	
	ABORT	IDLE	
JUMP	RESTART	RUNNING	
	STOP	STOPPING	
	ABORT	IDLE	
RESUMING		RUNNING	Execute In Sequence
	PAUSE	PAUSING	
	STOP	STOPPING	
	ABORT	IDLE	
STOPPING		IDLE	
	ABORT	IDLE	


5.2.2 Program Running Role

- Two modes are supported by SFC function block, respectively automation and Semi-automation mode. In automation mode, the program would be executed according to the program logic. In Semi-automation mode, the SFC program would transfer to waiting state after every step, and need to select *reset* command to continue the program. Semi-automation mode is used for single-step debug.
- Sequential execution. The program executes from the start step, and exit in the end step.
- The current activated step would keep being executed in every scan period, until the switch condition has been satisfied. Then the current activated step would be inactivated, and the next step would be activated. If *Execute Once* is selected (set in Set Step Property), the current step would be scanned in the first period after the function block being loaded.
- In selection branch, only the leftmost branch that satisfies the switch condition would be executed.
- When the switch condition before the parallel structure is satisfied, all the parallel branches within a parallel structure would start to run. And the parallel structure stops running at the time that all branches finishing their running and the switch condition being satisfied.

- When the switch condition before the jump is satisfied, program logic would jump to the target step, and run from that step.
- If the switch condition is *True*, program logic would go to next step. Otherwise, the switch should be scanned in every control period.

5.3 Add/Delete Sequence Step and Branch

5.3.1 Add Sequence Step

Select the node to add (step node or switch node). As shown in below, to add a step between T1 and S2, select T1, and click  in toolbar.

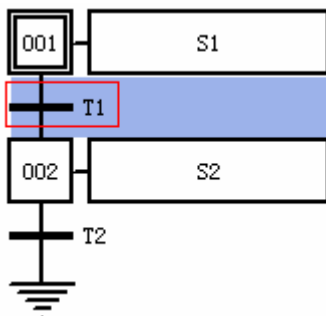


Figure 5-3 Add a sequence step

After an sequence step being added, the sequence number of the following steps would changed automatically, shown as follows.

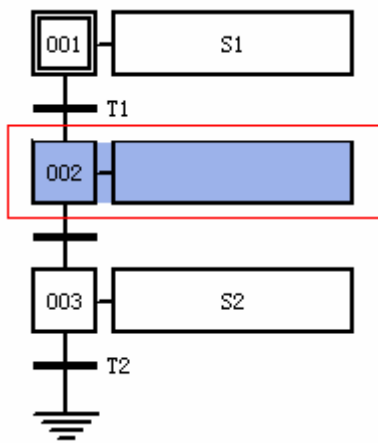



Figure 5-4 A sequence step is added

5.3.2 Add Selection Branch

Select the node needed to add branch. As shown in below, to add a branch after S1, select S1, and click  in toolbar.

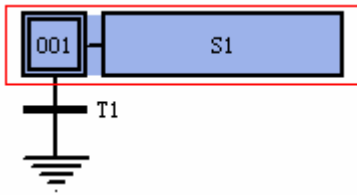


Figure 5-5 Before adding branch

Then a selection branch and a step node would be added, shown as follows.

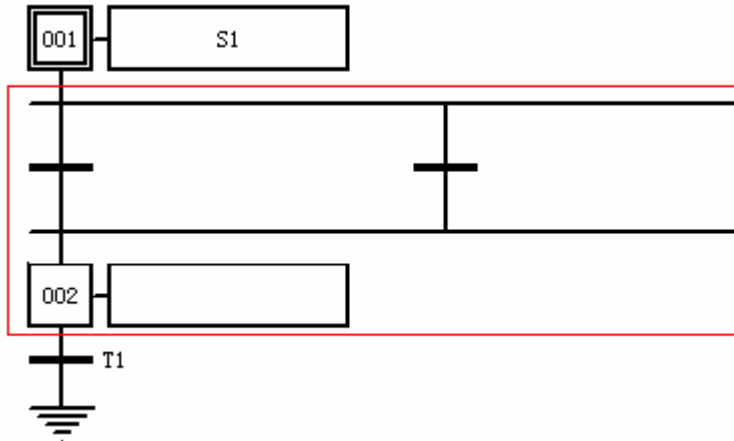



Figure 5-6 After adding branch

5.3.3 Add Parallel Branch

Select a switch node, and click  in the toolbar, shown as follows.

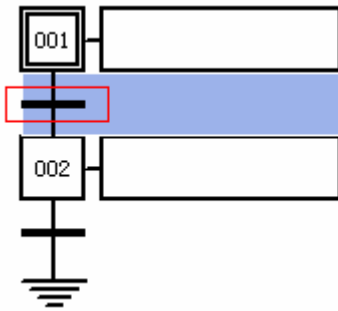


Figure 5-7 Add parallel branch

Then a parallel branch would be added, shown as follows.

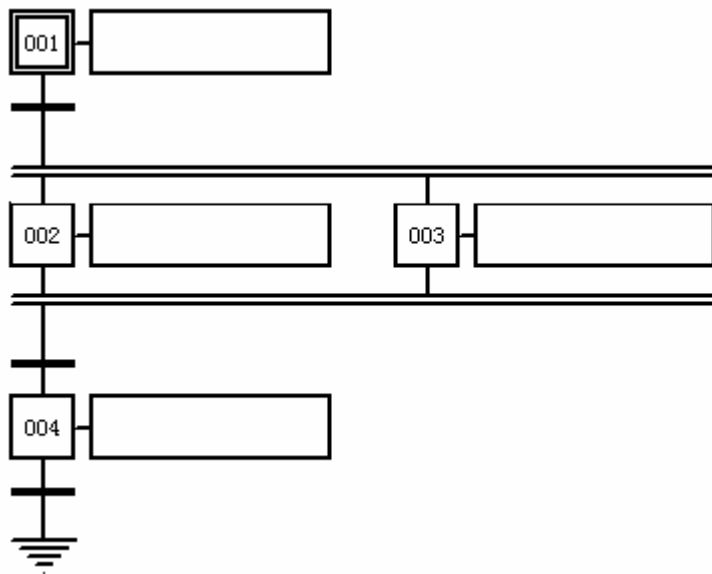



Figure 5-8 After a parallel branch is added

5.3.4 Extend Selection Branch

Select the branch node to extend, as shown Figure 5-9, and click  in toolbar.

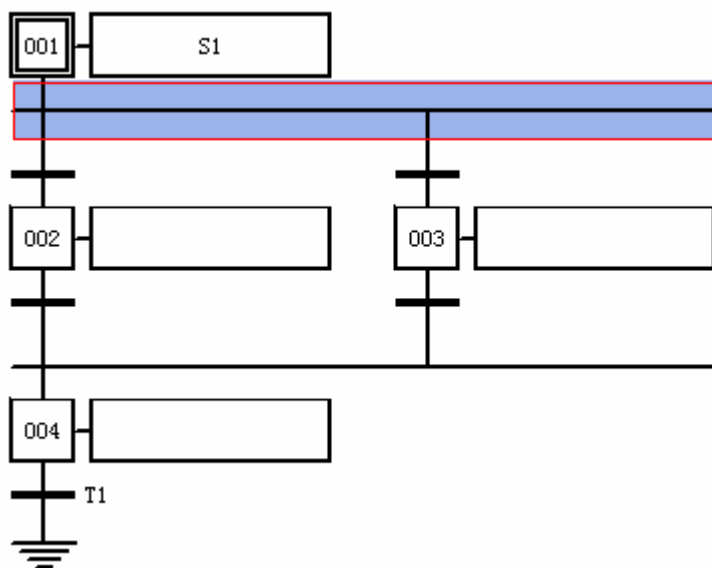


Figure 5-9 Extend selection branch

Then the branch would extend, shown as follows.

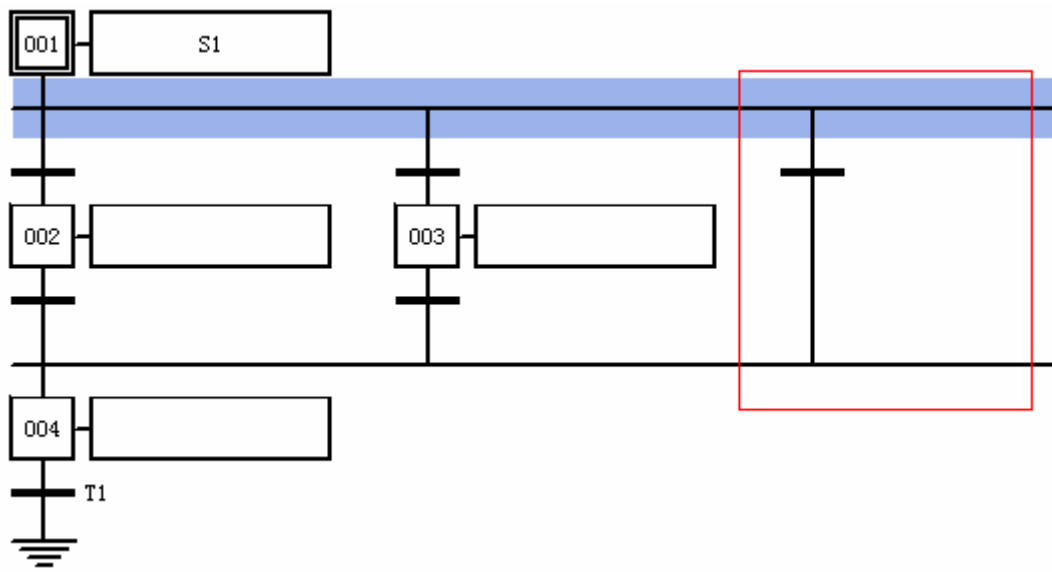



Figure 5-10 After the branch extending

5.3.5 Extend Parallel Branch

Select the branch node to extend, as shown Figure 5-11, and click  in toolbar.

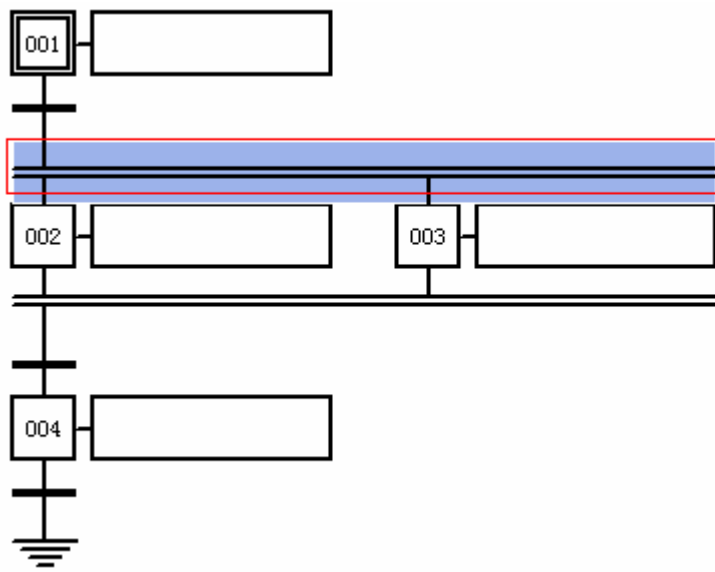


Figure 5-11 Before the parallel branch extending

Then the parallel branch extend would extend, shown as follows.

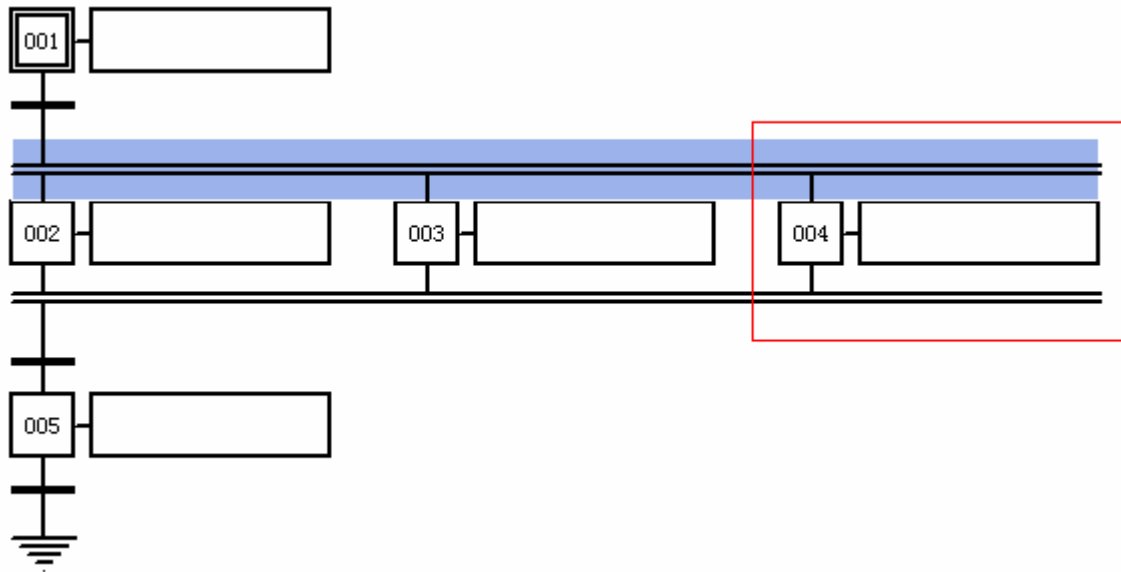


Figure 5-12 After the parallel branch extending

5.3.6 Jump

Select the switch node followed with *Selection Link*, double-click the switch, and the *Switch Property* dialog box pops up, shown as follows.



Tip:

Temporary variable, operate parameter and supervision parameter can be used for the switch conditions comparison.

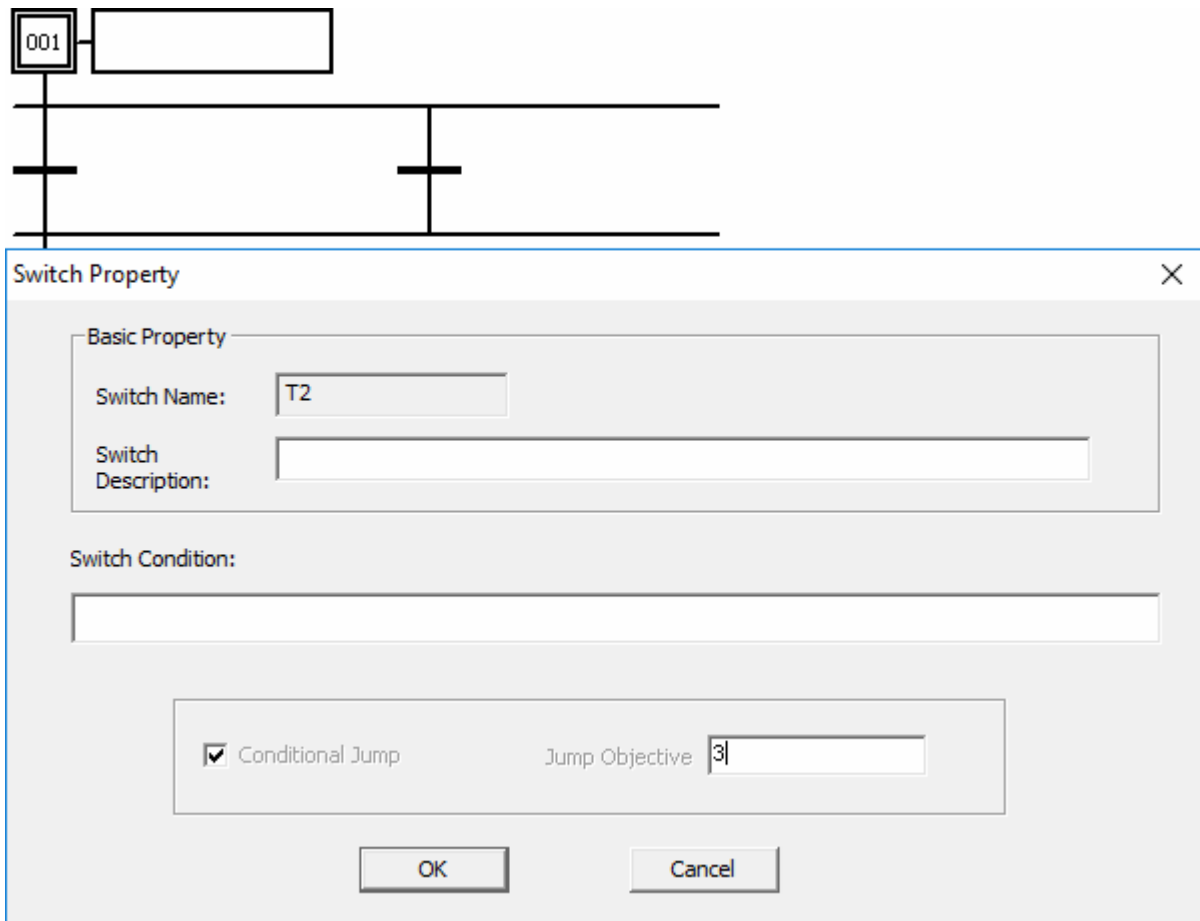


Figure 5-13 Switch property

Select *Conditional Jump* and set *Jump Objective* to form a *Jump*, as shown in Figure 5-14.

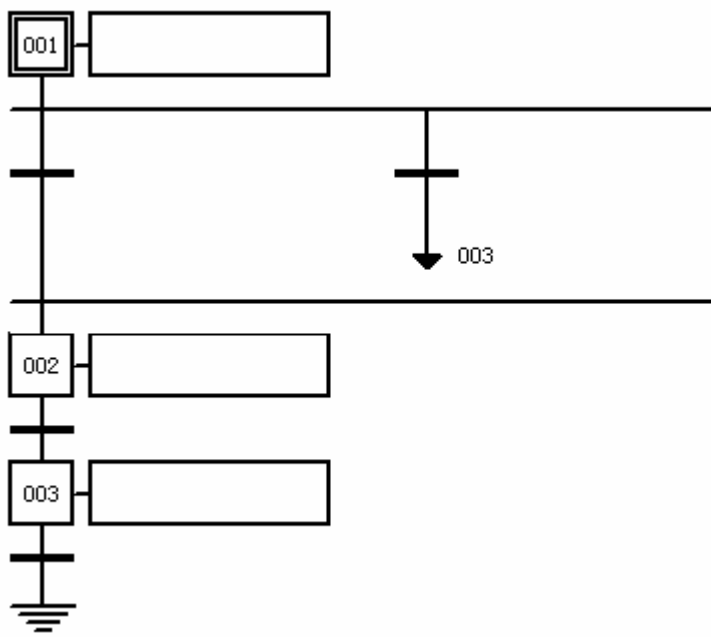


Figure 5-14 A Jump has been added

5.3.7 Delete Step or Branch

Refer to the delete role description of Programming Rule in 0.

5.4 SFC Function Block External Default Pins

5.4.1 Description

SFC function block contains properties such as modes, commands, states. The properties don't need to be defined in edit interface, and would be displayed after the function block being called in FBD program, shown as follows.

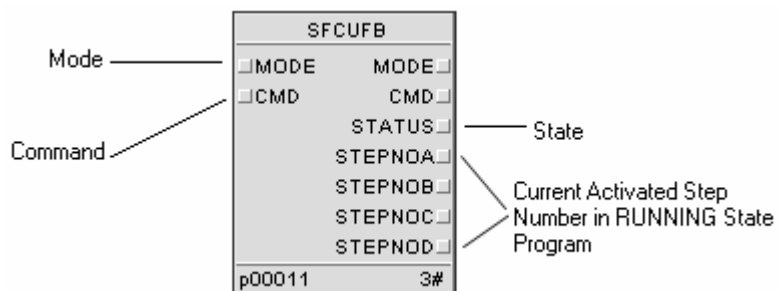


Figure 5-15 SFC function block

Table 5-5 SFC function block external default pins description

Name	Data Type	Initial Value	Pin Type	Description
MODE MODE_	USINT	0	Input, Output	Function Block Execution Mode Automatic: 0 Manual: 1
CMD CMD_	USINT	0	Input, Output	Operating the execution process by modifying commands. START: 1 PAUSE: 2 RESTART: 3 RESUME: 4 STOP: 5 ABORT: 6 RESET: 7
STATUS	USINT	0	Output	Function Block Current State IDLE: 0 COMPLETE: 1 RUNNING: 2 PAUSING: 3 PAUSED: 4 JUMP: 5 RESUMING: 6 STOPPING: 8

Name	Data Type	Initial Value	Pin Type	Description
STEPNOA STEPNOB STEPNOC STEPNOD	UDINT	0	Output	<p>The activated step number of RUNNING state program. 4 UINT variables, totally 128 bits, representing the step state of RUNNING state program.</p> <p>Inactivated: 0 Activated: 1</p> <p>STEPNOA represent 1st ~ 32nd steps from low bit to high bit</p> <p>STEPNOB represent 33rd ~ 64th steps from low bit to high bit 歩</p> <p>STEPNOA represent 65th ~ 96th steps from low bit to high bit</p> <p>STEPNOA represent 97th ~ 128th steps from low bit to high bit</p>

5.4.2 Linking Mode

Figure 5-16 Link of mode, command, state Figure 5-16 Connecting tag MODE_SFCUFB1 to input pin MODE, and output pin MODE to tag MODE_SFCUFB, achieves the input/output of function block SFCUFB mode from/to tag MODE_SFCUFB1. And the module mode could be modified in supervision interface if the tag is excerpted in *graph*.

The linking mode and function of tag CMD_SFCUFB1 is similar to MODE_SFCUFB1.

SFCUFB would keep receiving command value until the command value is canceled, and it would affect the function block debug and program logic execution. The linking mode shown in Figure 5-16 solves this problem in an efficient way. After function block receives and processes the command coming from pin CMD, the command value would be reset, and assign to output pin CMD. So tag CMD_SFCUFB1 would be assigned, and reset its command value automatically.

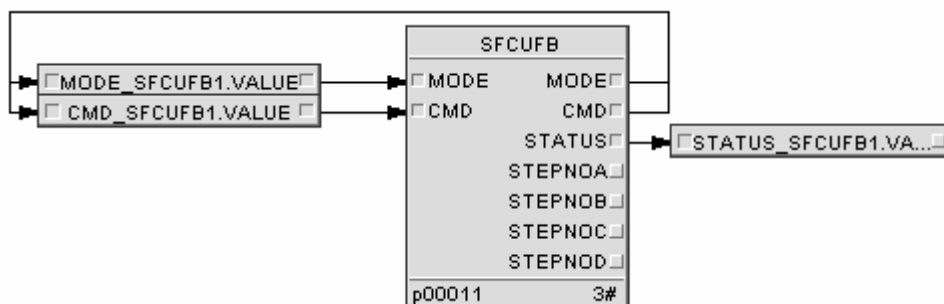


Figure 5-16 Link of mode, command, state

5.5 SFC Function Block Debug Panel

Users could call the edited SFC function block in FBD program layout. And online debug would be executed after compiling and downloading. Right-click the SFC function block and select debug logic in online mode to open the debug interface.

5.5.1 Mode

Automatic mode and semi-automatic mode are supported. The description is shown as follows.

Table 5-6 Mode description

Mode	Description
Automatic(AUTO)	Default. Execute next step automatically when switch condition is satisfied
semi-automatic(SEMI)	Enter JUMP state when switch condition is satisfied. And continue the execution after receiving reset command.

5.5.2 Command and State

SFC function block contains 7 commands: START, PAUSE, RESTART, STOP, RESUME, JUMP, ABORT.

Table 5-7 SFC function block debug command list

Command	Description
START	Enter running state
PAUSE	Pause in RUNNING or RESTARTING state, and enter PAUSING state. Click RUSUME to continue
RESTART	Click RESTART after JUMP, and the program would run from the current step
RESET	Reserved
STOP	Stop the execution of current module, and enter STOPPING process, and then enter IDLE state.
RESUME	Continue running from the stop spot
JUMP	Jump to specified step from current step(Select the specified step, click <i>JUMP</i> button)
ABORT	For stopping the abnormal state. Stop RUNNING, PAUSING, RESTARTING, STOPPING or PAUSED state, and enter IDLE state.



Tip:

If RESUMING, STOPPING, PAUSING states are edited when the program editing, the corresponding program would be executed after clicking RESUME, STOP, PAUSE buttons.

The command button would be enabled or disabled along with the state changing, shown as follows.

Table 5-8 Command button state list

Command State	START	STOP	PAUSE	RESUME	RESTART	ABORT	JUMP
JUMP		Enable			Enable	Enable	Enable
RUNNING		Enable	Enable			Enable	
STOPPING						Enable	
PAUSING		Enable				Enable	

Command State	START	STOP	PAUSE	RESUME	RESTART	ABORT	JUMP
PAUSED		Enable		Enable		Enable	Enable
IDLE	Enable						
RESUMING		Enable	Enable			Enable	

5.5.3 Step Color and State

Table 5-9 Step color and state list

The start step is IDLE state	RUNNING	RUNING FINISHING or STOP	Other
Black twinkling	Green	Blue	Black

5.5.4 Parameter View and Download

Users can view on parameters, and set the input/output parameter values without connecting to practical tags, for the program debug.

Parameter Types available for downloading: input parameter, output parameter, alias parameter.

Parameter Types unavailable for downloading: configuration parameter.

5.6 . Example

In engineering practical applications, some control function is sequence logic control, using SFC to realize would be more concise and objective.

5.6.1 Example Description

Some product is created by the mixture consist of stuff A and B. The production steps are as show as follows.

- Preparation: The temperature in inside the reactor (TI-101) is lower than 20 Celsius degree. liquid level (LI-101) is lower than 1%.
- Fill in stuff A: Turn on LV-101. When the liquid level reaches 20%, turn off the valve.
- Fill in stuff B: Turn on LV-102. When the liquid level reaches 70%, turn off the valve.
- Leave still: lasting 10 minutes.
- Heat up: Turn on TV-101, heat up to 70 Celsius degree
- Cool down: Cooling for 10 minutes.
- Discharge: Turn on FV-101, when the liquid level decreases to 1%. The reaction process is finished.

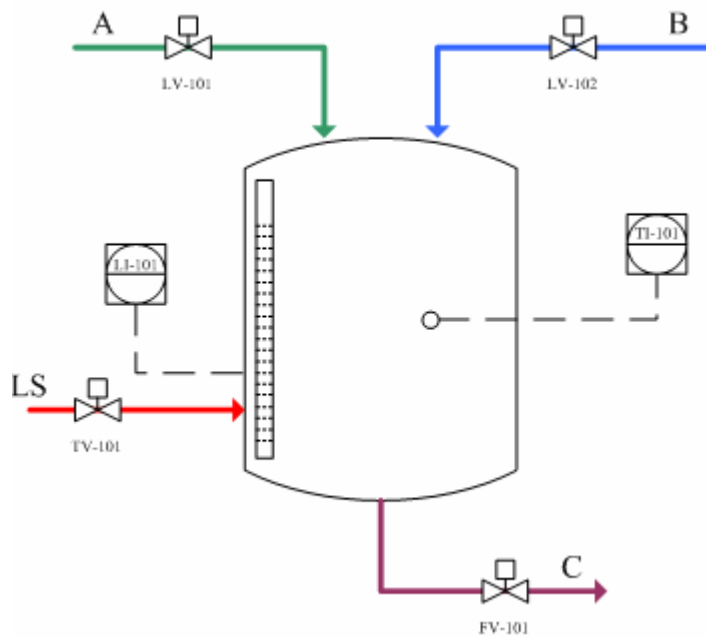


Figure 5-17 Schematics

5.6.2 Operation Steps

Operation Step:

New

In VFExplorer, select *Custom Function Block* node in configuration tree, and then select *New* in the right-clicked menu.

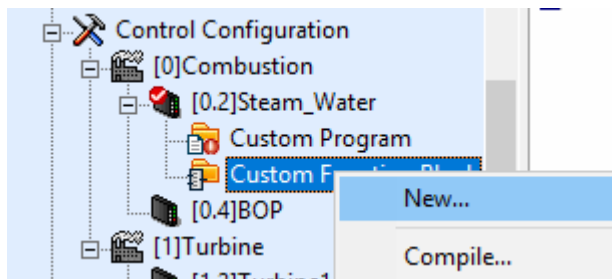


Figure 5-18 New

A new dialog box pops up. Select SFC type, and input the Name and Description, shown as follows.

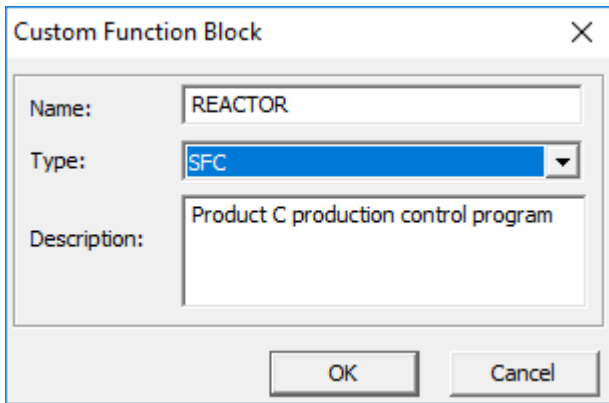


Figure 5-19 New dialog box

Edit

Double-click the created custom function block in the configuration property sub-window of VFExplorer, to start up VFSTModule.

Add parameter and set parameter property

Add parameters in the *Parameter Type* sub-window of VFSTModule, and set parameter property, shown as follows.

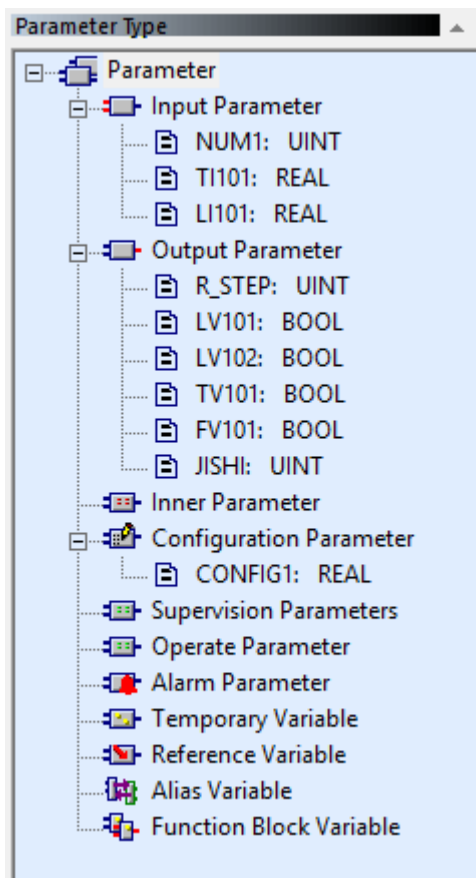


Figure 5-20 Add parameters

The initial values of NUM1, R_STEP, JISHI, CONFIG1 is 0. And the initial values of BOOL type

parameters are OFF.

Set step and switch property

Add 7 sequence steps and set the properties. The SFC program is shown in Figure 5-21.

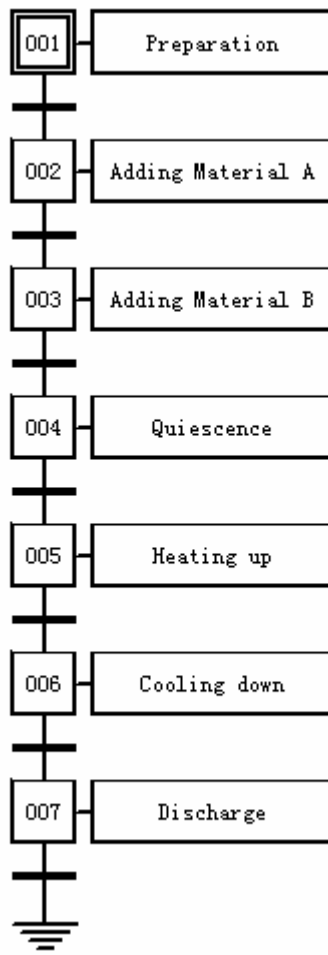


Figure 5-21 SFC program

- Preparation

- ST code:

```

IF TI101 < 20.0 AND LI101 < 1.0 THEN
    R_STEP := 2;
ELSE
    R_STEP := 1;
END_IF;
  
```

- Switch condition:

```

R_STEP = 2
  
```

1) Adding Material A

■ ST code:

```
IF LI101 >= 20.0 THEN  
    LV101 := OFF;  
    R_STEP := 3;  
ELSE  
    LV101 := ON;  
    R_STEP := 2 ;  
END_IF;
```

■ Switch condition:

```
R_STEP = 3
```

● Adding Material B

■ ST code:

```
IF LI101 >= 70.0 THEN  
    LV102 := OFF ;  
    R_STEP := 4 ;  
    TIMERS[NUM1] := 0 ;(*Initiating the timer*)  
ELSE  
    LV102 := ON;  
    R_STEP := 3 ;  
END_IF;
```

■ Switch condition:

```
R_STEP =4
```

● Quiescence

■ ST code:

```
IF TIMERS[NUM1] >= 600 THEN  
    R_STEP := 5 ;
```

ELSE

R_STEP := 4 ;

END_IF;

JISHI := TIMERS[NUM1]; (*The count value of timer can be obtained by viewing the variable JISHI in the panel parameter list*)

- Switch condition:

R_STEP = 5

- Heating up

- ST code:

IF T101 >= 70.0 THEN

TV101 := OFF;

R_STEP := 6 ;

TIMERS[NUM1] := 0;

ELSE

TV101 := ON ;

R_STEP := 5;

END_IF;

- Switch condition:

R_STEP = 6

2) Cooling down

- ST code:

IF TIMERS[NUM1] >= 600 THEN

R_STEP := 7 ;

ELSE

R_STEP := 6 ;

END_IF;

JISHI := TIMERS[NUM1];

- Switch condition:

R_STEP = 7

- Discharge

- ST code:

IF LI101 <= 1.0 THEN

FV101 := OFF ;

R_STEP := 0 ;

ELSE

FV101 := ON;

R_STEP := 7;

END_IF;

- Switch condition:

R_STEP = 0

Compile

Save and compile the program in VFSTModule, and then compile all the custom function block in the station in VFExplorer.

Reference function block

The new created custom function block REACTOR could be called in the FBD program after being compiled in VFExplorer, as shown in Figure 5-22.

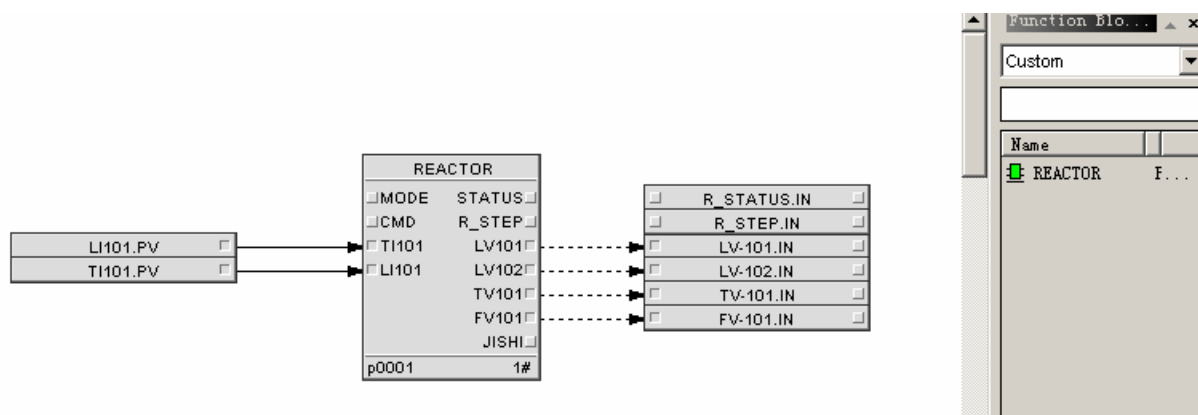


Figure 5-22 Custom function block calling

Table 5-10 Parameter and corresponding tags

Parameter	Tag
TI101	TI-101
LI101	LI-101
LV101	LV-101

Parameter	Tag
LV102	LV-102
TV101	TV-101
FV101	FV-101
STATUS	R_STATUS
R_STEP	R_STEP

After compiling FBD program, the debug process of custom function block could be executed.



Tip:

The modifiable parts of the custom function block having been called are code and temporary parameters.

The alias parameters in custom function block should refer to certain tag. Otherwise, the custom function block would not compile successfully.

Function block debug

Double-click REACTOR in the *Online* mode, and the debug panel pops up. Refer to 2.3 for the debug details.

Function Block Debug



















Right-click REACTOR online to open a debug panel. Debug the function block.

Section 6 Appendix

6.1 ST Language Programming Software

6.1.1 ST Language Programming Software Interface command list

Table 6-1 Menu and function list

Menu	Sub-Menu	Toolbar Icon	Function
File	Save(Ctrl+S)		Save Active Documents
	Import		Import Existed STF File
	Export		Save the current custom block program as STF file
	Print(Ctrl+P)		Print the current custom block program
	Print Preview		Display entire pages
	Print property		Change the printer and printing options
	Exit		Exit
Edit	Undo(Ctrl+Z)		Undo
	Redo(Ctrl+Y)		Redo
	Cut(Ctrl+X)		Cut
	Copy(Ctrl+C)		Copy
	Paste(Ctrl+V)		Paste
	Delete(Ctrl+D)		Delete
	Select All(Ctrl+A)		Select the entire document
	Find(Ctrl+F)		Find the specific text
	Find Next(F3)		Find Next
	Find Previous(Shift+F3)		Find Previous
	Replace(Ctrl+H)		Replace
	Bookmark(Ctrl+F2)		Set bookmark
	Clear All Bookmarks		Clear All Bookmarks
	Next Bookmarks		Next Bookmark
	Previous Bookmarks		Previous Bookmark
	Color Settings		Color Settings
Program	Compile(F7)		Compile
View	Statue Bar		Hide/Show the Statue Bar
	Output Bar		Hide/Show the Output Bar
	Parameter List		Hide/Show the Parameter List
Help	Online Help (F1)		Show the online help of ST language programming.
	About		Display the program information, version and copyright.

Right-click different objects in the view will show different menus, as shown below.

Table 6-2 Shortcut menu and function list

Selected Node	Menu Item	Function
Parameter Type Node	Add	Add Parameter of Selected Type
Parameter Node	Add	Add Parameter of Selected Type
	Delete	Delete Selected Parameter
	Move Upward	Move Upward the Selected Parameter
	Move Downward	Move Downward the Selected Parameter
	Object Properties	View and Set the Parameter Properties
	Modify Alias Variable	View and Modify the Alias Parameter Properties
	Modify Built-in Functional Module	View and Modify the Function Block Parameter Properties
	Function block Parameter Settings	View and Set the Function Block Properties
Code Edit Area	Undo	Undo
	Redo	Redo
	Cut	Cut the selected text
	Copy	Copy the selected text
	Paste	Paste the selected text
	Delete	Delete the selected text
	Set Bookmark	Set Bookmark
	Clear All Bookmarks	Clear All Bookmarks
	Add As Reference Variable	Add selected text as reference variable

6.1.2 Function and Function List



Tip:

Details and example of ST function please refer to the ST function help file in software.

Arithmetic Operation, Type Switch Function

- AVE_REAL()

Arithmetic calculation function, takes average from 2 REAL data.

- Return Value

REAL

- Example:

OUT1:=AVE_REAL (IN1,IN2);

- Description

IN1 and IN2 should be both REAL input parameters, OUT1 is REAL output function.

- Approximate Function

AVE_SINT AVE_INT AVE_DINT AVE_UINT AVE_UDINT AVE_USINT

- MOD_SINT()

Arithmetic calculation function, takes model from 2 SINT data.

■ Return Value

SINT

■ Example

```
OUT1:=MOD_SINT(IN1,IN2);
```

■ Description

IN1 and IN2 should be both SINT input parameters, OUT1 is SINT output function.

■ Approximate Function

MOD_INT MOD_DINT MOD_UINT MOD_UDINT MOD_USINT

● REAL_TO_DINT ()

Type switch function, switch the input LREAL (32-bit float) to DINT (32-bit long integer).

■ Return Value

REAL

■ Example

```
IF(REAL_TO_DINT (50000.001)<(50000.001+0.00001)
```

```
AND
```

```
REAL_TO_DINT (50000.001)>(50000.001-0.00001) )
```

```
THEN
```

```
CASENUM :=CASENUM+1.0;
```

```
ELSE
```

```
    ERR:=ON;
```

```
    ERR_CFG :=15;
```

```
END_IF ;
```

■ Description

CASENUM is REAL output parameter, ERR is BOOL output parameter, ERR_CFG is DINT output parameter.

■ Approximate Function

REAL_TO_DINT DINT_TO_REAL REAL_TO_UDINT UDINT_TO_REAL

REAL_TO_INT INT_TO_REAL REAL_TO_UINT UINT_TO_REAL

REAL_TO_SINT SINT_TO_REAL REAL_TO_USINT USINT_TO_REAL

DINT_TO_UDINT UDINT_TO_DINT DINT_TO_INT INT_TO_DINT

DINT_TO_UINT UINT_TO_DINT DINT_TO_SINT SINT_TO_DINT

DINT_TO_USINT USINT_TO_DINT UDINT_TO_INT INT_TO_UDINT

UDINT_TO_UINT UINT_TO_UDINT UDINT_TO_SINT SINT_TO_UDINT

UDINT_TO_USINT USINT_TO_UDINT INT_TO_UINT UINT_TO_INT

INT_TO_SINT SINT_TO_INT INT_TO_USINT USINT_TO_INT UINT_TO_SINT
 SINT_TO_UINT UINT_TO_USINT USINT_TO_UINT USINT_TO_SINT
 SINT_TO_USINT

Trigonometric Function

- ABS_ST()

Trigonometric function, takes absolute value of input data.

- Return Value

REAL

- Example

```
IF(ABS_ST(IN1) =100.0)
THEN
OUT1:=MOD_SINT(IN1,IN2);
END_IF;
```

- Description

IN1 can be positive or negative, but should be REAL value.

- Approximate Function

SQRT_ST: extraction of square root for input data.

EXP_ST: take index of input data.

LN_ST: execute natural logarithm for input data.

LG_ST: execute logarithm for input data with base 10.

- LG2_ST()

Execute logarithm with base IN1, and antilogarithm IN2.

- Return Value

REAL

- Example

```
IF(LG2_ST(IN1,IN2)>3.0 OR LG2_ST(IN1,IN2)<2.9)
THEN
OUT1:=ON;
END_IF;
```

- Description

IN1 and IN2 are REAL data, CASENUM is REAL output parameter, ERR is BOOL output parameter, ERR_CFG is DINT output parameter.

- Approximate Function

POW_ST: execute power function for input data.

- SIGN_ST()

Sign function, output is 0 when input is 0. Output is -1 when input is less than 0. Output is 1 when input is larger than 1.

■ Return Value

REAL

■ Example

SIGN:=SIGN_ST(-2.0) ;

■ Description

SIGN is REAL output parameter. -2.0 can be parameter of any REAL parameter.

● SIN_ST()

Sine function of trigonometric function.

■ Return Value

REAL

■ Example

SIN:=SIN_ST(IN1) ;

■ Description

IN1 is REAL input parameter, SIN is REAL output parameter.

■ Approximate Function

ASIN_ST: arcsine function

SINH_ST: hyperbolic sine function

ASINH_ST: arc hyperbolic sine function

COS_ST: cosine function

ACOS_ST: arc cosine function

COSH_ST: hyperbolic cosine function

ACOSH_ST: arc hyperbolic cosine function

TAN_ST: tangent function

ATAN_ST: arc tangent function

ATAN2_ST: (IN1/IN2) arc tangent function

TANH_ST: hyperbolic tangent function

ATANH_ST: arc hyperbolic tangent function

COT_ST: cotangent function

ACOT_ST: arc cotangent function

COTH_ST: hyperbolic cotangent function

ACOTH_ST: arc hyperbolic cotangent function

SEC_ST: secant function

CSC_ST: cosecant function

- ATAN2_ST()

Trigonometric function, arc tangent function of (IN1/IN2), different usage with SIN_ST, so it will be introduced alone here.

- Return Value

REAL

- Example

OUT1:=ATAN2_ST(IN1,IN2);

- Description

IN1 and IN2 are REAL input parameter, OUT1 is REAL output parameter.

Logical Operation, Shift Operation Function

- AND_BOOL()

Logical operation function, execute logical and operation for 2 BOOL data.

- Return Value

BOOL

- Example

OUT1:=AND_BOOL(IN1,IN2);

- Description

IN and IN2 are BOOL input parameter, OUT1 is BOOL output parameter.

- Approximate Function

OR_BOOL: execute logical or operation for 2 BOOL data.

NOT_BOOL: execute inverse operation for input BOOL data.

XOR_BOOL: execute logical exclusive or operation for 2 BOOL data.

WIPEOUT_BOOL: execute not and operation for first BOOL input and second BOOL input.

- AND_SINT()

Logical calculation function, execute bit and operation for 2 SINT data.

- Return Value

SINT

- Example

OUT1:=AND_SINT(IN1,IN2);

- Description

IN and IN2 are all SINT input parameters, OUT1 is SINT output parameter.

- Approximate Function

AND_USINT AND_INT AND_UINT AND_DINT AND_UDINT

- OR_SINT()

Logical operation function, execute bit or operation for 2 SINT data.

- Return Value

SINT

- Example

```
OUT1:=OR_SINT(IN1,IN2);
```

- Description

IN1 and IN2 are both SINT input parameter, OUT1 is SINT output parameter.

- Approximate Function

OR_USINT OR_INT OR_UINT OR_DINT OR_UDINT

- NOT_SINT()

Logical operation function, execute inverse operation for SINT data.

- Return Value

SINT

- Example

```
OUT1:=NOT_SINT(IN1);
```

- Description

IN1 is SINT input parameter, OUT1 is SINT output parameter.

- Approximate Function

NOT_USINT NOT_INT NOT_UINT NOT_DINT NOT_UDINT

- XOR_SINT()

Logical operation function, execute exclusive or operation for 2 SINT data.

- Return Value

SINT

- Example

```
OUT1:=XOR_SINT(IN1,IN2);
```

- Description

IN1 and IN2 are both SINT input parameter, OUT1 is SINT output parameter.

- Approximate Function

XOR_USINT XOR_INT XOR_UINT XOR_DINT XOR_UDINT

- ROL_SINT()

Shift operation function, execute circle left shift for IN2 for SINT data input IN1.

- Return Value

SINT

- Example

OUT1:=ROL_SINT(IN1,IN2);

■ Description

IN1 is SINT input parameter, IN2 is USINT input parameter, OUT1 is SINT output parameter.

■ Approximate Function

ROL_USINT ROL_INT ROL_UINT ROL_DINT ROL_UDINT

ROR_SINT: execute circle right shit for IN2 for SINT data input IN1.

ROR_USINT ROR_INT ROR_UINT ROR_DINT ROR_UDINT

● SHL_SINT()

Shift operation function, execute logical/ arithmetic left shit for IN2 for SINT data input IN1.

■ Return Value

SINT

■ Example

OUT1:=SHL_SINT(IN1,IN2);

■ Description

IN1 is SINT input parameter, IN2 is USINT input parameter, OUT1 is SINT output parameter.

■ Approximate Function

SHL_USINT SHL_INT SHL_UINT SHL_DINT SHL_UDINT

SHR_SINT: execute logical/ arithmetic right shit for IN2 for SINT data input IN1.

SHR_USINT SHR_INT SHR_UINT SHR_DINT SHR_UDINT

● LSR_SINT()

Shift operation function, execute right shit for IN2 for SINT data input IN1.

■ Return Value

SINT

■ Example

OUT1:=LSR_SINT(IN1,IN2);

■ Description

IN1 is SINT input parameter, IN2 is USINT input parameter, OUT1 is SINT output parameter.

■ Approximate Function

LSR_USINT LSR_INT LSR_UINT LSR_DINT LSR_UDINT

Take Bit Operation Function

● GBIT_SINT()

Take bit operation function, take IN2 from SINT data input IN1 的 IN2, return ON if it is 1, otherwise return OFF.

■ Return Value

BOOL

■ Example

OUT1:=GBIT_SINT(IN1,IN2);

■ Description

IN1 is SINT input parameter, IN2 is USINT input parameter, OUT1 is BOOL output parameter.

■ Approximate Function

GBIT_USINT GBIT_INT GBIT_UINT GBIT_DINT GBIT_UDINT

Set Bit Operation Function

- SBIT_SINT()

Set bit operation function, set value for a bit of SINT data, return set bit later value, shift exceeding limit is invalid.

■ Return Value

SINT

■ Example

OUT1:=SBIT_SINT(IN1,OFFSET,VAL);

■ Description

Write value VAL for OFFSET shift position of IN1, IN1 is SINT parameter, OFFSET is USINT shift value, VAL is BOOL parameter.

■ Approximate Function

Shift value and write value type of approximate function are same with SBIT_SINT.

SBIT_USINT SBIT_INT SBIT_UINT SBIT_DINT SBIT_UDINT

Comparison, Limit, Maximum, Minimum, Select Function

- EQ_BOOL()

Comparison function, execute equal comparison for 2 BOOL data.

■ Return Value

BOOL

■ Example

IF (EQ_BOOL(IN1,ON)) THEN

OUT1:=SBIT_SINT(INNER2,INNER1,VAL);

END_IF ;

■ Description

2 data compared should be in same type.

■ Approximate Function

Return vale of approximate function is BOOL.

EQ_SINT EQ_USINT EQ_INT EQ_UINT EQ_DINT EQ_UDINT EQ_REAL

LT_SINT: execute less than comparison for 2 SINT data.

LT_USINT LT_INT LT_UINT LT_DINT LT_UDINT LT_REAL

GT_SINT: execute larger than comparison for 2 SINT data.

GT_USINT GT_INT GT_UINT GT_DINT GT_UDINT GT_REAL

LE_SINT: execute less than or equal to comparison for 2 SINT data.

LE_USINT LE_INT LE_UINT LE_DINT LE_UDINT LE_REAL

GE_SINT: execute larger than or equal to comparison for 2 SINT data.

GE_USINT GE_INT GE_UINT GE_DINT GE_UDINT GE_REAL

NE_BOOL: execute not equal comparison for 2 SINT data.

NE_SINT NE_USINT NE_INT NE_UINT NE_DINT NE_UDINT NE_REAL

● EQX_REAL()

Comparison function, execute extended comparison for 2 REAL data.

■ Return Value

BOOL

■ Example

IF (EQX_REAL(5.0,4.9999,-0.0002)) THEN

OUT1:=ON;

END_IF ;

■ Description

EQX_REAL(IN1, IN2, DB), IN1, IN2 and DB are all REAL data, which can compare

IN1and IN2. If the difference of the 2 values is less than DB, the 2 values are considered as equal.

■ Approximate Function

NEX_REAL: execute not equal extended comparison for 2 REAL data.

● LIM_SINT()

Limit function, limit the SINT data.

■ Return Value

SINT

■ Example

IF (LIM_SINT(10,8,9) = 9) THEN

OUT1:=ON;

END_IF ;

■ Description

LIM_SINT (MAX, IN, MIN), MAX, IN and MIN are all SINT data. When MAX<MIN, return 0; when IN>MAX, return MAX; when IN<MIN, return MIN; when MAX>IN> MIN, return IN.

■ Approximate Function

LIM_USINT LIM_INT LIM_UINT LIM_DINT LIM_UDINT LIM_REAL

● MAX_SINT()

Select function, take maximum of 2 SINT data.

■ Return Value

SINT

■ Example

OUT11:=MAX_SINT(-21,-20);

■ Description

MAX_SINT(IN1, IN2), IN1, IN2 are all SINT data, OUT11 is SINT output parameter.

■ Approximate Function

MAX_USINT MAX_INT MAX_UINT MAX_DINT MAX_UDINT MAX_REAL

MIN_SINT: take minimum of 2 SINT data.

MIN_USINT MIN_INT MIN_UINT MIN_DINT MIN_UDINT MIN_REAL

● SEL_BOOL()

Select function, select function of BOOL data.

■ Return Value

ON or OFF

■ Example

IF (SEL_BOOL(ON,ON,OFF) = OFF) THEN

OUT1:=ON;

END_IF ;

■ Description

SEL_BOOL(SWA,IN1,IN2), if SWA=ON, return IN2; otherwise return IN1. IN1, IN2 are all BOOL data. If it is function SEL_INT, IN1 and IN2 are both INT data.

■ Approximate Function









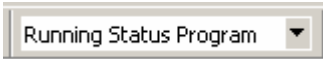
SEL_SINT SEL_USINT SEL_INT SEL_UINT SEL_DINT SEL_UDINT

SEL_REAL

6.2 SFC Language Programming Software

6.2.1 SFC Language Programming Software Interface Commands List

Table 6-3 Menu and function list

Menu	Sub-Menu	Toolbar Icon	Function
File	Save(Ctrl+S)		Save Active Documents
	Import		Import Existed sfc File
	Export		Save the current custom block program as STF file
	Print(Ctrl+P)		Print the current custom block program
	Print Preview		Display entire pages
	Print Settings		Print settings
	Exit		Exit
Edit	Cancel (Ctrl+Z)		Undo
	Restore (Ctrl+Y)		Redo
	Increase Sequence Steps		
	Increase Branch		
	Extend Branch		
	Delete (Delete)		Delete the select content
	Select All(Ctrl+A)		Select the entire document
	Copy SFC (Ctrl+C)		Copy the select content
	Paste SFC (Ctrl+V)		Paste the select content
Program	Compile		Compile
View	Zoom In		Zoom in
	Zoom Out		Zoom out
	Standard View		Restore to standard view
	Status Bar		Hide/Show the Statue Bar
	Output Bar		Hide/Show the Output Bar
	Parameter List		Hide/Show the Parameter List
Help	About		Display the software version and copyright
			Status switch

6.2.2 SFC Language Programming Software Shortcut Menu

Right-click different objects in the view will show different menus, as shown below.

Table 6-4 Shortcut menu and function list

Selected Node	Menu Item	Function
Parameter Type Node	Add	Add Parameter of Selected Type
Parameter Node	Add	Add Parameter of Selected Type
	Delete	Delete Selected Parameter
	Move Upward (Disable for Alias Variable)	Move Upward the Selected Parameter

Selected Node	Menu Item	Function
	Move Downward (Disable for Alias Variable)	Move Downward the Selected Parameter
	Object Properties (Disable for Alias Variable)	View and Set the Parameter Properties
	Modify Alias Variable (only for alias variable)	View and Modify the Alias Parameter Properties
	Modify Built-in Functional Module	View and Modify the Function Block Parameter Properties
	Function block Parameter Settings	View and Set the Function Block Properties
Code Edit Area	Undo	Undo
	Redo	Redo
	Cut	Cut the selected text
	Copy	Copy the selected text
	Paste	Paste the selected text
	Delete	Delete the selected text
	Set Bookmark	Set Bookmark
	Clear All Bookmarks	Clear All Bookmarks
	Add As Reference Variable	Add selected text as reference variable

Section 7 Revision

Table 7-1 Retrofit list of the version

Document Version	Applicable software version	Remarks
V1.0 (20230301)	OMC High-performanceHMI V4.70.00.00	First release
V1.1 (20230830)	OMC High-performanceHMI V5.10.00.00-M	Updated screenshots.