Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

**DANGER**

indicates that death or severe personal injury will result if proper precautions are not taken.

**WARNING**

indicates that death or severe personal injury may result if proper precautions are not taken.

**CAUTION**

indicates that minor personal injury can result if proper precautions are not taken.

**NOTICE**

indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by personnel qualified for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

**WARNING**

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.
# Table of contents

## 1 Basic principles

1.1 Aim ....................................................................................................................... 7

1.2 Block-specific definitions .................................................................................... 7

## 2 Configuring function diagrams based on IEC

2.1 Overview of Engineering based on IEC ............................................................... 9

2.2 Using the function diagram .................................................................................. 9

2.2.1 Creating a function diagram ............................................................................ 9

2.2.2 Structure of an IEC function diagram ............................................................. 11

2.2.3 Basic functions of an IEC function diagram .................................................. 12

2.2.4 Pasting objects via the clipboard .................................................................... 13

2.2.5 Navigating between half-open partner connections ........................................ 13

2.3 Placing a template in the function diagram ....................................................... 14

2.4 Using the signal ................................................................................................... 15

2.4.1 Signal-specific definitions .............................................................................. 15

2.4.2 Overview of the signal types .......................................................................... 16

2.4.3 Checking the signal type ................................................................................ 17

2.4.4 Placing the signal ............................................................................................ 17

2.4.5 Display of the signal ...................................................................................... 18

2.4.6 Controlling the display of the cross-reference ............................................... 21

2.4.7 Navigating in the function diagram with signals ........................................... 22

2.4.8 Using the signal - cross-reference list ............................................................. 22

2.5 Locking a connector by means of a constant signal value ................................. 23

2.6 Using the function block ..................................................................................... 24

2.6.1 Definitions for function blocks ...................................................................... 24

2.6.2 Display of the function block ........................................................................ 26

2.6.3 Placing the function block ............................................................................. 27

2.6.4 Placing the macro block ................................................................................. 27

2.6.5 Controlling the connector display .................................................................. 28

2.6.6 Using the function block ................................................................................ 29

2.7 Editing and updating macro blocks ................................................................. 30

2.7.1 Update macroblocks ...................................................................................... 30

2.7.2 Outputting attributes in the symbol of the macro block .................................. 30

2.7.3 Editing and checking the detail drawing ....................................................... 31

2.7.4 Editing inconsistency ...................................................................................... 32

2.8 Bulk refreshing of macro blocks ....................................................................... 34

2.8.1 Opening the "Update macroblocks" tab ......................................................... 34

2.8.2 Initializing and updating the "Update macroblocks" tab ................................... 34

2.8.3 Preliminary work: Checking use and properties ........................................... 35

2.8.4 Updating macro blocks in bulk ....................................................................... 36

2.9 Printing the function diagram with internal circuits ....................................... 37
# Table of contents

## 2.10 Connecting blocks
- Connection types for block connectors................................................................. 38
- Connecting block connectors by means of a reference............................................ 38
- Connecting block connectors by means of an intermediate variable...................... 40
- Representation of connections in the IEC function diagram..................................... 40
- Using connections.................................................................................................... 41
- Negating a connection.............................................................................................. 42

## 2.11 Parameterizing blocks....................................................................................... 43

## 2.12 Creating and numbering step sequences.......................................................... 44
- Overview of step sequences..................................................................................... 44
- Creating step sequences (SFCs)................................................................................. 46
- Closing a half-open connection of loops and jumps................................................ 48
- Adding new branches to parallel branches and alternative branches..................... 50
- Moving parts of the step chain up or down............................................................. 50
- Moving or copying step chain objects with the mouse.......................................... 51
- Automatic numbering of steps, transitions or actions............................................. 52

## 2.13 Placing and using the comment block................................................................. 53

## 2.14 Interfaces............................................................................................................ 54

## 3 Code generation based on IEC 61131.................................................................. 55
- Overview of code generation..................................................................................... 55
- Overview of code objects........................................................................................ 56
- Entering code classes............................................................................................... 56
- Definition of "Function diagram"............................................................................. 56
- Definition of "Configuration".................................................................................. 57
- Definition of "Resource"......................................................................................... 59
- Definition of "Task"................................................................................................. 59
- Definition of "Program organization unit" (POU)...................................................... 59
- Signals (Variables)................................................................................................. 60
- Overview of the COMOS symbol tables................................................................. 60
- Creating a symbol table.......................................................................................... 61
- Basic principle of the partial update of symbol tables............................................. 61
- Specific entries in the symbol table........................................................................ 62

## 3.3 Describing and exporting codes.......................................................................... 62
- Preparing code generation for a block...................................................................... 62
- Exporting the block................................................................................................. 62
- Preparing code generation for a program............................................................... 63
- Exporting the program............................................................................................. 63

## 4 Administration......................................................................................................... 65
- Managing function diagrams based on IEC............................................................... 65
- Overview of report templates.................................................................................. 65
- Creating a user-defined template for a function diagram........................................ 65
- Setting the connector type combinations............................................................... 66
- Reference: Variables in the signal symbol script.................................................... 67
- Reference: Variables in the options script of the function diagrams....................... 68
- Adjusting block base objects.................................................................................. 69
- Overview of block base objects.............................................................................. 69
- Creating a user-defined block.................................................................................. 69
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.6.3</td>
<td>Specifications for a connector interlock</td>
<td>70</td>
</tr>
<tr>
<td>4.1.6.4</td>
<td>Creating user-defined macro blocks</td>
<td>70</td>
</tr>
<tr>
<td>4.1.6.5</td>
<td>Creating user-defined dynamic blocks</td>
<td>71</td>
</tr>
<tr>
<td>4.1.6.6</td>
<td>Creating a template for a macro block</td>
<td>71</td>
</tr>
<tr>
<td>4.1.6.7</td>
<td>Creating a base object for the comment block</td>
<td>72</td>
</tr>
<tr>
<td>4.1.7</td>
<td>Checking and adjusting standard tables</td>
<td>73</td>
</tr>
<tr>
<td>4.1.7.1</td>
<td>Overview of the standard tables for function diagrams</td>
<td>73</td>
</tr>
<tr>
<td>4.1.7.2</td>
<td>Opening standard tables</td>
<td>74</td>
</tr>
<tr>
<td>4.1.7.3</td>
<td>Combinations permitted by data hierarchy</td>
<td>74</td>
</tr>
<tr>
<td>4.1.7.4</td>
<td>Adjusting standard tables</td>
<td>75</td>
</tr>
<tr>
<td>4.2</td>
<td>Managing code generation based on IEC 61131</td>
<td>76</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Reference: Base objects for code generation</td>
<td>76</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Creating text templates for a placeholder text</td>
<td>77</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Reference: Symbolism in an own function group</td>
<td>78</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Overview: Assigned function group</td>
<td>79</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Integration of variants/control structures</td>
<td>79</td>
</tr>
<tr>
<td>4.2.5.1</td>
<td>Control structure #Exist-#Default-#EndExist</td>
<td>79</td>
</tr>
<tr>
<td>4.2.5.2</td>
<td>Control structure #If-#Else-#EndIf</td>
<td>80</td>
</tr>
<tr>
<td>4.2.5.3</td>
<td>Control structure #ForEach-#Next</td>
<td>81</td>
</tr>
<tr>
<td>4.2.5.4</td>
<td>Control structure #include</td>
<td>82</td>
</tr>
</tbody>
</table>
1 Basic principles

1.1 Aim

The "Logical" product is intended for the administration and use of function diagrams. The functions and controls for a sequential control system can be planned using the function diagrams. In this case, a function or control is mapped in the function diagram both graphically and logically. The display is independent from the technical realization.

In addition, interfaces for code creation are prepared.

In COMOS, function diagram templates are prepared based on the standard IEC 61131-3.

1.2 Block-specific definitions

Definition of block

In COMOS, blocks are all objects that can be placed in a function diagram. The following blocks are available to you:

- Standard functions, standard function blocks and extended function blocks
- Signals
- Step sequence objects

Definition of block type

A block type is a generally valid, reusable function block. A block type is independent of the configured PLC, because its algorithm does not process PLC address areas. A block type communicates with the process using the input and output parameters.

Each block type has a type definition, which contains the following:

- Type name
  The type name describes the block type function.
  Example: The "ADD_R" block type adds the values of the input parameters and transfers the result using the output parameter.

- Input parameters
  Input parameters apply the values that the block type requires in order to manage its task.

- Algorithm
  An algorithm is a sequence of instructions for the processing of a task.

- Output parameters
  Output parameters transfer the result to the process.
Definition of block instance

When you place a block type in a function diagram, a block instance is generated. You can create as many block instances as required from one block type. Each block instance gets a unique name within a function diagram.

See also

Overview of step sequences (Page 44)
2.1 Overview of Engineering based on IEC

Engineering with the help of a function diagram includes the following basic steps:

- Creating a function diagram
  See also chapter Creating a function diagram (Page 9).

- Placing a template in the function diagram
  See also chapter Placing a template in the function diagram (Page 14).

- Placing additional signals in the function diagram input and output area
  See also chapter Placing the signal (Page 17).

- Placing additional blocks in the function diagram
  See also chapter Placing the function block (Page 27).

- Connecting blocks and signals
  See also chapter Connecting blocks (Page 38).

- Creating step chains
  See also chapter Creating and numbering step sequences (Page 44).

The sequence of the steps is useful, but not technically mandatory.

2.2 Using the function diagram

2.2.1 Creating a function diagram

Requirement

- The "Units" tab is open in the Navigator.
- A position with measuring functions and signals has been created.
  You can find more information on this topic in the "EI&C Operation" manual, keyword "Creating and placing positions".

Procedure

1. From the "Position" or "Measurement function" context menu, select "New > FF.00?? Function diagram".

Result

The function diagram is created in the Navigator below the position or measurement function.
See also

Managing function diagrams based on IEC (Page 65)
2.2.2 Structure of an IEC function diagram

The following illustration shows the structure of a function diagram based on the IEC standard:

① Toolbar
The toolbar contains blocks which are prepared in the function diagram template.

② Standard toolstrip

③ Header line
The header line displays the legend for the input and output signals.

④ Area for input signals/left slot
Input signals transfer a value from the process to the function diagram for processing.

⑤ Area for output signals/right slot
Output signals transfer a value processed by the function diagram back to the process.

⑥ Working area
You place and connect blocks in the working area.

⑦ Symbol bar
The symbol bar contains blocks which are compiled from the base data.

⑧ Drawing header
The top area of the drawing header displays the legend for the input and output signals. These specifications are identical to ②. The lower area shows selected attributes of the function diagram.
2.2 Using the function diagram

See also

Basic functions of an IEC function diagram (Page 12)

2.2.3 Basic functions of an IEC function diagram

Requirement

- A function diagram has been created and is open. See chapter Creating a function diagram (Page 9).

Basic functions

In COMOS, a function diagram is of the "interactive report" type. This means that it provides all editing options of an interactive report.

You can find additional information on this topic in the "COMOS Platform Operation" manual, keyword "Working with interactive reports".

Document-defined symbol bar

Frequently-used blocks are deposited in a document-defined symbol bar. The document-defined symbol bar is displayed to the right of the function diagram.

Collision check

Objects and groups cannot be placed on the function diagram if they overlap each other.

The following drawing elements are not included in the collision check:

- Lines
- Circles
- Free texts

Fixed symbol sizes

Blocks on function diagrams have a fixed size.

Sample data and library in the database

The base objects for blocks are located in the base objects at the following location:

"@30 > M40 > A50 > A10 > A20 > A10 > A10 IEC 61131-3"

See also

Structure of an IEC function diagram (Page 11)
2.2.4 Pasting objects via the clipboard

Purpose of the "Paste" context menu

The "Paste" context menu is available in interactive reports. This menu accesses the COMOS clipboard. Using the "Cut" or "Copy" context menu, objects are copied to the clipboard and then placed on any interactive report with the "Paste" command.

General use of the "Paste" context menu

You can find additional information on this topic in the "COMOS Platform Operation" manual, keyword "Copying and cutting in interactive reports".

Special operation for the "EI&C" product

- Drawing types for electrical engineering (M41_P2, ...) and fluidics (M43_P1): The "Paste" context menu places the engineering object parallel to the original.
- Exception: If the object is located next to the document and you copy across documents, the copy is not located parallel to the original but next to the new document.

See also

Moving or copying step chain objects with the mouse (Page 51)

2.2.5 Navigating between half-open partner connections

Requirement

- A function diagram has a half-open connection. It is connected with another half-open connection. The partner connection can lie on the same page of the diagram, on a different page or on a different function diagram.
- The function diagram is open.
### Procedure

1. Select the half-open connection.

2. You have the following options:
   - Navigate to the partner connection with the `<Page down>` key.
     Navigate back with the `<Page up>` key.
   - Select the "Connection > Navigation to partner connection" command from the context menu.
   - Click the "Navigate to successor" button on the toolbar.
     Navigate back with the "Navigate to predecessor" button.

   Requirement: In the options script of the report template the option `EnablePaging` has the value `True`. You can find additional information on this topic in the "EI&C Administration" manual, keyword "EnablePaging".

### Result

- The half-open partner connection is selected and zoomed in.
  If necessary, the page or diagram is changed to this purpose.

### Note

**Second diagram closed**

If the second diagram was closed, it is loaded into the tab of the diagram already opened.

- For connections across diagrams: The corresponding diagram is selected in the Navigator.

### 2.3 Placing a template in the function diagram

#### Introduction

With the aid of templates for function diagrams, you can perform various standardized measurement and control tasks. In general, a template contains several already interconnected blocks. SIEMENS provides example templates for function diagrams based on the IEC standard.

#### Requirement

- A function diagram has been created and is open. See chapter Creating a function diagram (Page 9).
- Templates for function diagrams have been created in the base data. See chapter Creating a template for a macro block (Page 71).
Procedure

1. Click in the empty area of the report.
2. Select the "Place template" command in the context menu. The "Select Document as Template" window opens.
3. Select the template and click "OK".
4. Click on the desired position of the report.

Alternative method

You can also copy a function diagram into the properties of a "Position". To do this, open the "Properties" of the "Position" and select the "New > Copy in function diagram" command in the context menu on the "Tasks" tab.

The function diagram is created below the "Position".

2.4 Using the signal

2.4.1 Signal-specific definitions

The COMOS object "Signal"

Signals represent values from the process. In COMOS, a signal requires at least the following attributes:

- "Signal data" tab: "Signal designation"
- "System data" tab: "Signal type"
  The signal type also determines whether the signal is placed in the left slot or the right slot.
  - "Input" signal type
    Transfers a value from the process to the function diagram for processing.
    Example: a sensor passes a measurement signal to the process controller. You place the input signal in the left area of a function diagram.
  - "Output" signal type
    The function diagram transfers a value back to the process.
    Example: a control block passes a control signal to the process. You place the output signal in the right area of a function diagram.
    See also chapter Overview of the signal types (Page 16).
- "System data" tab, "Signal data type"
  The signal data type transfers its setting to the connector subtypes. The connector subtype controls the type test when connecting to function diagrams. Data type processing has the following constraints:
  - If an array connector is involved, you must change the subtype for all connectors.
  - If the connector or one of the array connectors is already connected, you cannot make any changes.
Definition of "Slot"

Slots are predefined placement areas for a signal on the left or right edge of the function diagram. Each slot represents an input or output of the function diagram.

In the default setting, a signal must be placed in a slot. The number of slots therefore limits the number of placeable signals.

Definition of "source/sink pair"

For a source/sink pair, a signal is entered in two signal objects. The two signal objects are placed on two different function diagrams. Once the signal object is marked as an output signal, with the second placement, the signal object is designated as an input signal.

The two signal objects together are the source/sink pair.

See also

Placing the signal (Page 17)

2.4.2 Overview of the signal types

Hardware

This signal type represents measurement signals and actuating signals from the process. In the context of a unit, a signal is applied on an I/O card, where it is processed.

In the COMOS standard model, these signals are available in the "A20 Signal Engineering" folder and were created within the framework of the IC engineering. You can find more information on this topic in the "EI&C Operation" manual, keyword "Creating and designing functions".

- A Hardware Input (left slot)
- B Hardware Output (right slot)

Software

This signal type represents a communication signal from a higher or a parallel level. In the context of a unit, a communication signal is transported via a data bus.

- C Software Input (left slot)
- D Software Output (right slot)
Variable

This signal type represents a controller information block. In the context of a unit, a variable signal is used to store states in the controller.

- E Variable Input (left slot)
- F Variable Output (right slot)

See also

Placing the signal (Page 17)

2.4.3 Checking the signal type

Requirement

- A position with measuring functions and signals has been created.
  You can find more information on this topic in the "EI&C Operation" manual, keyword "Creating and placing positions".

Procedure

1. Open the properties of the signal.
2. Select the "Attributes > System data" tab.
3. Check the "Signal type" field.

See also

Placing the signal (Page 17)

2.4.4 Placing the signal

Requirement

- A position with functions and signals has been created.
  You can find more information on this topic in the "EI&C Operation" manual, keyword "Creating and placing positions".

- A function diagram has been created and is open.
  See chapter Creating a function diagram (Page 9).
Procedure

1. Select the signal in the Navigator.
2. Drag&drop the signal onto the function diagram:
   - To place an "Input" type signal, drag&drop it into a free slot on the left.
   - To place an "Output" type signal, drag&drop it into a free slot on the right.
   - To place a signal that is not an "Input" or "Output" type, drag&drop it into a free slot either on the left or right.

Using output signals as an input signal

An output signal placed on a function diagram can serve as an input signal for a different function diagram. For this purpose, place the output signal on the second function diagram in a free left-hand slot.

The output signal receives a cross-reference to its second placement. See also chapter Display of the signal (Page 18).

Automatic check for "Variable Output" and "Variable Input"

If the database is configured in such a way that the signals in the Navigator are located under a function diagram, and you drag a signal of the type "Variable" from there to a function diagram, you can only place the signal as follows:

- "Variable Output":
  - Initial placement: In right-hand slots
  - Subsequent placement: In left-hand slots
- "Variable Input":
  - Initial placement: In left-hand slots
  - Subsequent placement: In right-hand slots

See also

- Structure of an IEC function diagram (Page 11)
- Negating a connection (Page 42)
- Signal-specific definitions (Page 15)
- Using the signal - cross-reference list (Page 22)

2.4.5 Display of the signal

Text variables in the signal symbol

A signal can have up to six text variables.
The illustration below shows the position and meaning of the text variables using the example of an output signal:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>① Device label</td>
<td>② Description</td>
<td>③ Reference</td>
<td>④ Label or name of the signal</td>
<td>⑤ Location/channel</td>
</tr>
</tbody>
</table>
Display of multiplacements (dynamic reference lists)

Output signals that are placed in a different function diagram as an input signal have a cross-reference. The cross-reference has two representation variants:

- Representation with reference arrow:

![Image of signal with reference arrow]

- Representation with reference texts:

The labels of the reports on which the signals are placed in the input area are displayed under the signal symbol:

![Image of signal with reference texts]
You can change between the variants. See also chapter Controlling the display of the cross-reference (Page 21).

2.4.6 Controlling the display of the cross-reference

Default in the database

Representation with cross-reference arrow

Requirement

- A signal is placed on a function diagram A in the output area, on a function diagram B in the input area.
  See chapter Placing the signal (Page 17).
- Function diagram A is open.

Changing the display on the function diagram

1. In the function diagram toolbar, activate or deactivate the option "Show cross-references".
2. Click the "Apply" button on the toolbar:

3. Click "Reevaluate document":

Result

The display of the cross-references is updated on the function diagram:

- Option deactivated:
  - The cross-reference text is displayed for signals in whose properties the "Show cross-references" option is activated.
  - The cross-reference arrow is displayed for the other signals.
- Option activated:
  The cross-reference texts are displayed.

The setting is stored in the properties of the function diagram: "System data" tab, "Show cross-references" attribute

It is used when the function diagram is opened next.

Changing the display at the signal

1. Open the properties of the signal.
2. Select the "System data" tab.
3. Activate the “Show references” option.
4. Confirm your entries.

Result: If the signal has a cross-reference, the cross-reference text is always displayed. The report setting does not apply for this signal.

See also
- Navigating in the function diagram with signals (Page 22)
- Display of the signal (Page 18)

2.4.7 Navigating in the function diagram with signals

Introduction
Each signal knows all of its uses, its template, and its position in the Navigator.

Requirement
- A function diagram has been created and is open. See chapter Creating a function diagram (Page 9).
- A signal is placed on the function diagram. See chapter Placing the signal (Page 17).

Procedure
1. To navigate in the function diagram with the signal, select the "Navigate > Documents > <Usage>" command in the context menu of the signal. The <Usage> entry has the following format:<Group sheet number> <Document name> <Document description> Slot <Number>. You can find additional information on this topic in the "COMOS Platform Operation" manual, keyword "Navigating".

2.4.8 Using the signal - cross-reference list

Requirement
- A function diagram has been created and is open. See chapter Creating a function diagram (Page 9).
- A signal is placed on the function diagram. See chapter Placing the signal (Page 17).
2.5 Locking a connector by means of a constant signal value

Introduction
If you enter a constant signal value at the unconnected connector, the connector is locked. You can no longer connect the connector.

Requirement
- The blocks have the classification “AV”. See chapter Specifications for a connector interlock (Page 70).
- You use a macro block for the connector interlock. See chapter Placing the macro block (Page 27).

Entering a constant signal value via the text field
1. At the block, search for a connector which is equipped with an input box with a hash sign.
2. Hover the mouse over the hash sign until a green frame appears:

   ![Green frame](image)

4. Overwrite the hash sign with the signal value.
5. Confirm with <Return>.
You can no longer connect the connector.
Entering a constant signal value via the "Connector visibility" dialog

1. Select the block.
2. Select the "Options > Connector visibility" command in the context menu of the block.
3. Replace the ""#"" sign with the constant signal value.

Deleting a constant signal value

1. At the block, search for a connector which is equipped with a constant signal value.
2. Hover the mouse over the input box for the constant signal value until a green frame is displayed.
4. Delete the constant signal value.
5. Confirm with <Return>.
   The hash sign appears in the input box.
You can connect the connector again.

Connecting a connector with input option for constant signal value

Provided that no constant signal value is set, the connector can be connected.

1. At the block, search for a connector which is equipped with an input box with a hash sign.
2. Connect the connector.
   The input box for the constant signal value no longer appears.
If you break the connection, the input box for the constant signal value is displayed again.

Controlling connector visibility

In the "Connector visibility" dialog, the interlocked connectors are displayed.
See also chapter Controlling the connector display (Page 28).

2.6 Using the function block

2.6.1 Definitions for function blocks

Note
Generic term "Function block"

In this manual, the term "Function block" is used as a generic term for standard functions, standard function blocks and extended function blocks. If a specific type is meant, this is explicitly mentioned.
Standard function

Standard functions are defined by the IEC standard and are supported by all manufacturers. They represent subtasks that cannot be divided any further. They process the values transferred from the input signals and return the calculated return values to the output signals.

Examples:
- AND-gate
- OR gate

Standard function block

Standard function blocks are defined by the IEC standard and are supported by all manufacturers. They extend standard functions and represent a self-contained subtask of a sequential control system. They process the values transferred from the input signals and return the calculated return values to the output signals.

Example: Forward counter

Extended function block

Extended function blocks are based on the IEC standard. They represent a function or a position. They process the values transferred from the input signals and return the calculated return values to the output signals.

Example: A control block for a valve.

Special cases of the extended function blocks are dynamic blocks and macro blocks.

Dynamic block

The following applies to dynamic blocks:
- The block connectors can be hidden and displayed again in the function diagram.
- The distance of the connectors can be controlled via the properties.

See also chapter Controlling the connector display (Page 28).

Macro block

Macro blocks are used in hierarchical function engineering. A macro block encapsulates a complex function logic - the inner circuitry. If you place a macro block on a higher-level function diagram, the inner circuitry is available. The representation of the inner circuitry is highly simplified in this case. You have the option of viewing the inner circuitry in detail.

See also chapter Editing and updating macro blocks (Page 30) and chapter Editing and checking the detail drawing (Page 31).

As with dynamic blocks, the following also applies to macro blocks:
- The block connectors can be hidden and displayed again in the function diagram.
- The distance of the connectors can be controlled via the properties.
2.6 Using the function block

2.6.2 Display of the function block

Display of a function block

A function block is displayed with a block header and the block interface:

- **Block header**
  The information displayed on the block header depends on the block used. For standard functions, the block header may be missing entirely or may contain a small symbol, such as a block for an AND-gate.

- **Block interface**
  The block connectors are displayed in the block interface. Each block connector has a name, as well as a "type" and a "subtype". The "subtype" defines the connector data type, for example, "BOOL".
  - On the left: block inputs
  - On the right: block outputs

Height

The height of the function block depends on the number of connectors.

For standard functions, you can adjust the height via drag&drop. Additional connectors are automatically generated at the input end during this process. Example: AND-gate

Visibility and distance of the connectors

With dynamic blocks and macro blocks, you can set for each connector whether the connector is visible in the symbol. See also chapter Controlling the connector display (Page 28).

Inner circuitry of macro blocks

The lower-level function diagram represents the complex internal structure, or inner circuitry, of the macro block. For the inner circuitry, a greatly simplified display is generated that you can connect and move as usual. See also chapter Editing and updating macro blocks (Page 30).
2.6.3 Placing the function block

Requirement

- A function diagram has been created and is open.
  See chapter Creating a function diagram (Page 9).

Procedure

1. To insert a block from the symbol bar, click the symbol, then click on the desired position in the function diagram.
2. To insert a block from the Navigator, drag & drop the block onto the function diagram.

Result

The block is inserted as a block instance in the function diagram. If you have inserted the block from the symbol bar or the base objects, the block will be created in the Navigator as an object below the function diagram.

See also

Placing the macro block (Page 27)

2.6.4 Placing the macro block

Requirement

- A function diagram has been created and is open.
  See chapter Creating a function diagram (Page 9).
- A template for a macro block is prepared.
  See chapter Creating a template for a macro block (Page 71).

Placing the macro block

1. Select the macro block in the "Units" tab in the Navigator. The macro block is located below the function.
2. Alternatively, create a new macro block with the help of the example objects:
   - Select the "Base objects" tab in the Navigator.
   - Search for the macro block base object.
   - Drag & drop the base object onto the "Units" tab.
   - Select the macro block in the "Units" tab in the Navigator.
3. Drag & drop the macro block onto the function diagram.
Checking the placed macro block

Each macro block has a reference to a subordinate function diagram. To check the reference to the function diagram, follow these steps:

1. Open the properties of the placed macro block.
2. Select the "Attributes > System data" tab.
3. Check the "Macro block" field.

Macro block base object in the database

You can find the base object of the IEC macro block at the following location:

"@30 > M40 > A50 > A10 > A20 > A10 > A10 > A40 > A10 Macro block"

Similar functions

The structure and the use of FUP assemblies are comparable with the following functions:

- Templates in the "Platform" product.
  You can find more information on this topic in the "COMOS Platform Administration" manual, keyword "Administration of templates".
- Assemblies in the "EI&C" product.
  You can find more information on this topic in the "EI&C Operation" manual, keyword "Placing and using assemblies".

See also

Editing and updating macro blocks (Page 30)

2.6.5 Controlling the connector display

You can control the connector display for dynamic blocks and macro blocks.

Requirement

- A function diagram has been created and is open. See chapter Creating a function diagram (Page 9).
- A macro block or dynamic block is placed. See chapter Placing the function block (Page 27).
Defining connector visibility

1. Select the macro block or dynamic block on the function diagram.
2. Select the "Options > Connector visibility" command in the context menu.
3. Select the connectors you want to be visible in the window.
   The symbol graphic is accordingly displayed as expanded or reduced in the function diagram.

Note
If you connected the connectors in the current function diagram, you can no longer hide the connectors.

Defining the connector distance

To define the distance between the connectors and thereby the size of the macro block or dynamic block, follow these steps:

1. Select the block in the function diagram.
2. Select the "Properties" command in the context menu.
3. Select the "Attributes > System data" tab.
4. To select the relative distance, follow these steps:
   – Only with macro blocks: Select the "Current grid size" entry from the "Connector distance mode" list.
   – Select the desired entry from the "Connector distance" list.
     The minimum distance between connectors is four grid points.
5. To select the absolute distance, follow these steps:
   – Select the "Simple" entry from the "Connector distance" list.
   – Only with macro blocks: Select the desired entry from the "Connector distance mode" list.
     For a 4 grid, the "4 coordinate points" setting corresponds to grid distance 1.

2.6.6 Using the function block

Requirement

- A function diagram has been created and is open. See chapter Creating a function diagram (Page 9).
- At least one block is placed on the function diagram. See chapter Placing a function block (Page 27).
Copying the block

1. Select the block on the report.
2. Select the "Copy" command in the context menu.
3. Select the "Paste" command from the context menu.
   In the case of a connected block, its connections are not copied along with it.

Deleting the block

1. Select the block on the report.
2. Select "Delete" from the context menu.
   When you delete a connected block, the connections are labeled as inconsistent. In this case, delete the inconsistent connections or connect them to another block connector.

2.7 Editing and updating macro blocks

2.7.1 Update macroblocks

Requirement
- A function diagram has been created and is open.
  See chapter Creating a function diagram (Page 9).
- A macro block is placed.
  See chapter Placing the macro block (Page 27).

Procedure
To update a macro block, select it on the function diagram and then select the command "Options > Refresh macroblock" in the context menu.

To update multiple macro blocks, use the plugin "Update macroblocks". See also chapter Bulk refreshing of macro blocks (Page 34).

2.7.2 Outputting attributes in the symbol of the macro block

Requirement
- A function diagram has been created and is open.
  See chapter Creating a function diagram (Page 9).
- A macro block is placed.
  See chapter Placing the macro block (Page 27).
Procedure

1. Create the "USERPARAM" tab in the macro block base object.
2. Create attributes with the prefix "ADDITIONALP" in the "USERPARAM" tab.

Result

The attributes are entered at the bottom of the macro block in a box.

2.7.3 Editing and checking the detail drawing

Requirement

- A function diagram has been created and is open.
  See chapter Creating a function diagram (Page 9).
- A macro block is placed.
  See chapter Placing the dynamic macro block (Page 27).

Displaying detail drawing

1. Select the desired macro block on the function diagram.
2. In the context menu, select the "Detail drawing > Show" command.
   The function diagram with the detail drawing is opened write-protected.

Changing the detail drawing

1. Select the macro block.
2. In the context menu, select the "Detail drawing > Change" command.
3. Select a project from the list.
   If you use a separate variant of a macro block, choose the storage project used.
4. In the Navigator, search for the macro block.
   Example: "Current project > Templates > M42 Function engineering > ..."
5. Select the macro block.
6. Confirm the prompt "Should the macro block be updated?" with "Yes":
   The update of an individual block follows the same rules as the bulk update.
   See also chapter Updating macro blocks in bulk (Page 36).
   Alternatively, you can update the macro block with the command "Options> Refresh macroblock".
2.7 Editing and updating macro blocks

**Editing the detail drawing (generating a copy)**

To generate a variant of the macro block detail drawing, follow these steps:

1. Select the macro block.
2. In the context menu, select the command "Detail drawing > Edit (create new)".
3. Select a tab from the list.
4. Select one of the two options:
   - **Project**
     The project currently open.
   - **Base project**
     COMOS switches to the base project with this option. If you do not have the required rights, COMOS cancels the editing.
5. Select a storage location in the Navigator.
   The object structure created in the template branch of the base project is displayed.
   If you selected the "Project" option, your template is not stored in the base project. Instead, the tree structure is reproduced in the current project and your template is created in the location you have selected.
   The copied function diagram, complete with the macro block detail drawing, is opened. You can change and save the detail drawing.

See also

Editing inconsistency (Page 32)

**2.7.4 Editing inconsistency**

**Requirement**

- A function diagram has been created and is open.
  See chapter Creating a function diagram (Page 9).
- A macro block is placed.
  See chapter Placing the macro block (Page 27).

**Causes of inconsistencies**

An inconsistent macro block might have the following causes:

- The prompt "Should the macro block be updated?" was confirmed with "No".
- The macro block template used was changed outside the current function diagram.

An inconsistent macro block is displayed in red in the function diagram.
Analyzing inconsistency

If a macro block is displayed in red, you can check the inconsistency as follows:

1. Select the macro block.
2. Select the "Inconsistency > Analyze" command in the context menu.
   The context menu "Inconsistency" is only available if there are actually inconsistent objects on the report.

Correcting inconsistency automatically

1. Select the macro block.
2. In the context menu, select the "Detail drawing > Change" command.
3. Confirm the prompt "Should the macro block be updated?" with "Yes".

Correcting inconsistency manually

1. Select the block on the function diagram.
2. Select the "Navigate > Object" command in the context menu.
3. Delete the macro block object connectors in the Navigator.
4. Delete the macro block placed in the function diagram.
   The connections are retained but marked as inconsistent with red.
5. Place the macro block back on the function diagram.
6. Connect the macro block to the function diagram.
   The connections are marked as consistent again.
7. Click the "Reevaluate document" button on the toolbar to update the function diagram.

**Note**

This process may result in already configured connections being lost or rewired.

See also

Bulk refreshing of macro blocks (Page 34)
2.8 Bulk refreshing of macro blocks

2.8.1 Opening the "Update macroblocks" tab

1. Select the "Plugins > Automation > Update macroblocks" command in the menu bar. The "Update macroblocks" tab opens.

See also

Initializing and updating the "Update macroblocks" tab (Page 34)

Preliminary work: Checking use and properties (Page 35)

2.8.2 Initializing and updating the "Update macroblocks" tab

Requirement

- Several macro blocks are used on function diagrams in unit planning. See chapter Placing the dynamic macro block (Page 27).
- The "Update macroblocks" tab is open. See chapter Opening the "Update macroblocks" tab (Page 34).
Procedure

1. Drag&drop a folder or object from the Navigator into the "Start object" field. In the COMOS standard model, use a folder from the "Units" tab.

2. Drag&drop a macro block from the Navigator into the "Macro block" field. In the COMOS standard model, use a macro block from the "Units" tab. COMOS searches the template for the macro block and then lists all macro blocks with the same template. The previous template is displayed in the "Template" field.

Updating lists

To react to modified data, follow these steps:

1. Click the "Refresh" button next to the "Start object" field.

See also

Preliminary work: Checking use and properties (Page 35)

2.8.3 Preliminary work: Checking use and properties

Requirement

- Several macro blocks are used on function diagrams in unit planning. See chapter Placing the macro block (Page 27).
- The "Update macroblocks" tab is open. See chapter Opening the "Update macroblocks" tab (Page 34).

Checking the use of macro blocks

1. Set the start fields. See also chapter Initializing and updating the "Update macroblocks" tab (Page 34).

2. Check the use in the "Used in document" column.

3. Check the hierarchical context of the macro blocks:
   - Select the required entry from the list.
   - Select the "Navigate > Macro block" command from the context menu.

Checking entries in the control fields in advance

1. Select the "Navigate" command in the context menu of a control field.
Checking macro blocks in advance

1. Select the required entry from the list.
2. Select the "Properties" command in the context menu.
3. Check the macro block properties.

See also

Updating macro blocks in bulk (Page 36)

2.8.4 Updating macro blocks in bulk

Requirement

- Several macro blocks are used on function diagrams in unit planning.
  See chapter Placing the dynamic macro block (Page 27).
- The "Update macroblocks" tab is open.
  See chapter Opening the "Update macroblocks" tab (Page 34).
Configuring the "Update macroblocks" tab

1. In the "New template" field, enter the template from which the macro blocks are to be derived.
   In the COMOS standard model, use a template from the base project. You can find this template in the "Base objects" tab in the following node: "@Template Templates".

2. Select the respective macro blocks from the list using one of the following options:
   - Selecting/clearing the checkbox
   - "Select all macroblocks" command in the context menu of the list
   - "Select selected macroblocks" command in the context menu of the list
   - "Deselect all macroblocks" command in the context menu of the list
   The context menu of the list is only visible when at least one macro block is listed.

3. Activate the "Refresh placed macroblocks" option.
   - Option is activated.
   During the macro block update, all connections that no longer occur by name and type in the new macro block template are deleted.
   - Option is deactivated.
   Old connections are retained if they were connected.

Note

Reconstruction of connections after an update

Connections are maintained when a macro block template is changed if the following requirements are met:

- In the new macro block, there is a connector with the same name and type.
- The connector with the same name and type is near the previous connector.

If both conditions are met, the connection is made again. If required, the connection is kinked in order to reach the new connection point.

If not all connections can be made, the macro block is inconsistent and displayed in red.

Update macroblocks

1. Click on the "Start" button.

2.9 Printing the function diagram with internal circuits

Requirement

- At least one macro block is used in a function diagram.
  See chapter Placing the dynamic macro block (Page 27).
Procedure

1. To activate the "Print internal circuits of macro blocks" project option, follow these steps:
   - Open the project properties.
   - Select the "Automation > El&C" category.
   - Enable the "Print internal circuits of macro blocks" option in the "Settings" control group.

2. To print the document using the Print Manager, select the "Documents > Print" command in the menu bar.
   The detail drawing is ignored if you select the "Print" function in the function diagram.

2.10 Connecting blocks

2.10.1 Connection types for block connectors

Connection rules

One block output is always connected to one block input. When connecting, if you move a block with the mouse, the possible target locations are highlighted.

The following rule applies for connecting blocks:
- The data types of the block inputs and outputs must be compatible.

Graphical interruptions for connections

In order to connect blocks with interrupted lines, you have the following options:
- Connecting block connectors by means of a reference
  A reference is generated automatically when you connect block connectors on different pages in a function diagram. A reference is not an object. References are used for the following tasks:
  - For reasons of clarity, the connection between two blocks is interrupted on a sheet.
  - A connection across sheets is required for documentation purposes only.
- Connecting block connectors by means of an intermediate variable
  The two block connectors on different function diagrams are now connected by means of the "Connector" signal type. In the Navigator, the "Connector" is displayed as a separate object below each function diagram. The intermediate variable is an object. Intermediate variables are used for the following tasks:
  - The connection is acquired in object queries.

See also

Connecting block connectors by means of a reference (Page 39)
Connecting block connectors by means of an intermediate variable (Page 40)
2.10.2 Connecting block connectors by means of a reference

Requirement

- A function diagram has been created and is open. See chapter Creating a function diagram (Page 9).
- The blocks have been created. See chapter Placing a function block (Page 27).

Procedure

1. Click the "Connection" button on the toolbar.
2. Click the block output, then click on an available position in the function diagram.
3. In order to leave the connection open, right-click on the report. An inconsistent connection is displayed in the function diagram.
4. Select the inconsistent connection.
5. Select the "Connection > Memorize [block output label]" command in the context menu of the inconsistent connection. "Memorize" means that the outgoing connection from the block connector is saved. The stored connection is maintained until you "memorize" (save) another connection. This allows you, for example, to connect one block output to several block inputs.
6. Also create a half-open, inconsistent connection at the target block input.
7. Select the inconsistent connection.
8. Select the "Connection > Connect with [block output label]" command in the context menu of the inconsistent connection.

Disconnecting a memorized connection

1. Select the connection.
2. Select the "Connection > Disconnect <Block connector label>" command in the context menu.

The connection between the blocks is disconnected. The connection is displayed as "inconsistent" on both block connectors. Either delete the inconsistent connections or connect them to other block connectors.

Additional information

You can find more information on this topic in the "EI&C Operation" manual, keyword "Open connections".

See also

Connection types for block connectors (Page 38)
2.10.3 Connecting block connectors by means of an intermediate variable

Requirement

- A function diagram has been created and is open. See chapter Creating a function diagram (Page 9).
- The function diagram contains blocks. See chapter Placing the function block (Page 27).

Procedure

1. Create a signal of the "Connector" signal type. To do so, select the New > A80 Signals > A90 Signal types > K Connector" command in the Navigator in the context menu of the "Signal Engineering" folder.
2. Open the properties of the signal.
3. Enter a meaningful description.
4. Drag&drop the signal onto the function diagram whose block output value you want to continue on the other function diagram. The block output value is used as a signal source.
5. Connect the signal to the block output.
6. Drag&drop the signal from the Navigator onto the function diagram in which you want to connect the signal to a block input. You can place the signal on several function diagrams and simultaneously connect it with different block inputs (signal targets) in this way.
7. Connect the signal to the block input or block inputs.

See also

Connection types for block connectors (Page 38)

2.10.4 Representation of connections in the IEC function diagram

Display of the line

Connections between blocks are displayed by a line. The line color defines the status of the connection:

<table>
<thead>
<tr>
<th>Line color</th>
<th>Connection status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>The connection is in compliance with the connection rules and is consistent.</td>
</tr>
<tr>
<td>Red</td>
<td>The connection is in violation of the connection rules or is inconsistent.</td>
</tr>
</tbody>
</table>

Depending on the data type of the connected block inputs and block outputs, the line width and the line type vary.
Display of graphical interruptions for a connection

Interrupted connections and connections across sheets are enabled via automatically generated references in accordance with the following syntax:

- \[\text{<Name of the function diagram>}.<\text{sheet number}>.<\text{slot number}>\]

The following illustration shows the reference of a connection across sheets:

Representation of negated connections

See chapter **Negating a connection** (Page 42).

See also

**Using connections** (Page 41)

2.10.5 **Using connections**

Checking and editing inconsistent connections

Inconsistent connections are not stored in the database and displayed as "not connected".

1. Select the inconsistent connection.
2. Select the "Inconsistency > Analyze" command in the context menu.
3. Correct or delete the inconsistent connection.

Tracking a signal via connections

1. Select the connection or the reference. The connection is highlighted in color.
2. In the context menu under "Navigate", select the desired target location of the connection, e.g. block connectors or base objects.
Manually routing connections

The connections automatically search for the best route to the function diagram. When you move an object, the connection is automatically re-routed. You can also route the connections manually.

1. Select a connection or several connections.
2. Select the “Options > Reroute” command in the context menu.
   During rerouting, the dynamic connector points are re-set depending on the position of the objects on the function diagram.
   Dynamic connection points are connectors that can occur at any position of a connection line or component.

See also

Negating a connection (Page 42)

2.10.6 Negating a connection

Requirement

- A function diagram has been created and is open. See chapter Creating a function diagram (Page 9).
- A connection has been created at a block on the function diagram. See chapter Connection types for block connectors (Page 38).

Negating a connection

1. Select the connection.
2. In the context menu of the connection select the "Settings > Signal Is Negated" command.
The following negation displays are possible:

- **Two blocks are consistently connected with one another**
  The negation symbol is displayed at the block input. The value at the block input is negated. The negation remains in place even if you move or copy the negated connection.
  The following illustration shows a negated connection between two connected blocks:

- **Several components are connected with a block input**
  This situation arises when you work with dynamic connectors in a function diagram. A connection line is thereby connected to another connection line.
  In this case, the context menu "Settings > Signal is negated" is available for each of the connection lines. If each of the connection lines is negated, the negation symbol is displayed not only at the block input, but also at the block outputs.
  The following figure shows negated connections that run in a common block input:

- **Several connections, some of which are half-open, are connected with a block input**
  If half-open connections are dynamically connected, the negation symbol is shown at the dynamic connection point.

### 2.11 Parameterizing blocks

**Introduction**

For documentation purposes, you can specify block inputs in greater detail or, in the case of control blocks, enter range limits for setpoint checking.
2.12 Creating and numbering step sequences

2.12.1 Overview of step sequences

Step chain planning

You plan step chains in function diagrams. The function diagrams may contain solely step chains or combine step chains with standard functions.

Overview of the step chain objects

Step chains consist of the following objects:

- An initial step chain
  The initial step chain consists of the input step, transition and final step.
- Further steps
- Transitions
• Actions
• Branches:
  – Parallel branches
  – Alternative branches
  – Jumps
  – Loops
In addition, you connect the step chain with the following objects:
• Transitions with signals
  The signals represent the switching conditions.
• Steps with actions
• Optional: With standard functions and constants

Example

The following illustration shows a step chain with switching conditions (left) and actions (right). The values of several signals are processed with a standard function. The result of the standard function is returned to the transition as a switching condition:

Base objects for step chains in the database

The objects are located under the following node: "@30 > M40 > A50 > A10 > A20 > A10 > A10 > A30 Step chain"
2.12 Creating and numbering step sequences

See also

Creating step sequences (SFCs) (Page 46)

2.12.2 Creating step sequences (SFCs)

Requirement

- A function diagram has been created and is open. See chapter Creating a function diagram (Page 9).
- The required signals have been created. Signals are located in the Navigator under their function, in the "A20 Signal Engineering" folder.
- Action blocks are located under the required signals.

Note

As a rule, signals and action blocks are created during IC engineering. You can find additional information on this topic in the "EI&C Operation" manual, keyword "Creating devices, channel requests and signals".

If signals or action blocks are missing, first create them in the Navigator by using the context menu "New".

Placement rules

- Place the signals in the input area of the function diagram.
- Place the steps and branches in the central area of the function diagram.
- You place steps and branches on the connections between transitions and steps.
- You place action blocks on the right of their step.
• You can place further steps or branches in parallel branches and alternative branches.
• Steps and transitions have to alternate in step chains. If you insert a step chain object into a step chain, several objects are inserted automatically:

<table>
<thead>
<tr>
<th>Placed step chain object</th>
<th>Inserted objects</th>
</tr>
</thead>
</table>
| Step                          | • Step
|                               | • Transition                                          |
| Parallel branch               | • Two branches with one step each
|                               | • Transition after or before the parallel branch      |
| Alternative branch            | • Two branches with one transition each               |
|                               | • Step after or before the alternative branch         |
| Jump or loop before a step    | • Jump step or loop step                              |
|                               | • Transition in the branch                            |
|                               | • Transition after the jump or the loop               |
| Jump or loop after a step     | • The step becomes a jump step or loop step.          |
|                               | • Transition in the branch                            |

Connection rules

• In order to connect symbols, you use the "Connection" tool from the function diagram toolbar.
• Connect the right output of a step with the input of the appropriate action.
• Connect the output of a signal with the left input of its transition.

Procedure

1. Place the initial step chain via the symbol bar.
   The following objects and placed and connected automatically:
   - Input step
   - Transition
   - Final step
2. Place the signal suitable for the transition from the Navigator.
3. Connect the signal and transition.
4. Drag&drop the action block appropriate for the step from the Navigator on the right next to the input step.
   An action is created and placed on the function diagram in the Navigator.
5. Connect the step and action.
6. Proceed as follows to add individual steps to the step chain:
   – Place a step on a connecting line via the symbol bar.
   – Place the action block appropriate for the step and the signal appropriate for the transition as described in steps 2 to 5. Connect the objects as described above.

7. Proceed as follows to add branches to the step chain:
   – Place an alternative branch, parallel branch, loop or a jump on a connecting line via the symbol bar.
   – Place the appropriate action block and the appropriate signal as described in steps 2 to 5. Connect the objects as described above.
   – In the case of jumps and loops a half-open connection is inserted as the return connection. The connecting line of loops is dashed. As long as the connection is half-open, it is displayed as inconsistent. Close the connection. See also chapter Closing a half-open connection of loops and jumps (Page 48).

See also
   Automated numbering of steps, transitions or actions (Page 52)
   Adding new branches to parallel branches and alternative branches (Page 50)

2.12.3 Closing a half-open connection of loops and jumps

Requirement

- A function diagram is open.
  See also chapter Creating a function diagram (Page 9).
- A step chain with loop or jump is placed on the function diagram.
  See also chapter Creating step sequences (SFCs) (Page 46).

Procedure

1. Select the half-open connection of the loop or the jump.
2. Select the "Connection > Memorize" command from the context menu. The unconnected connector is memorized.
3. Draw a further half-open connection. Starting point: The connecting line before the step at which the loop or the jump is to end.
   Permitted steps:
   – For loops: Steps before the loop step
   – For jumps: Steps before and after the jump step
4. Select the half-open connection.
5. Select the "Connection > Connect with <Name of the memorized connector>" command from the context menu.

Result

- The connectors are connected in the database.
- Symbols for connections across sheets are displayed at the ends of the half-open connection lines.

Example of a loop:

![Diagram of a loop]

See also

Navigating between half-open partner connections (Page 13)
2.12.4 Adding new branches to parallel branches and alternative branches

Requirement

- A function diagram is open. See also chapter Creating a function diagram (Page 9).
- A step chain with a parallel branch or an alternative branch is placed on the function diagram. See also chapter Creating step sequences (SFCs) (Page 46).

Procedure

1. Click a step chain object in the toolbar.
2. Move the mouse to the horizontal line that marks the beginning of the branch. The mouse pointer now has a vertical form.
   Example of an alternative branch:
   
   ![Alternative Branch Diagram]
   
   3. Release the mouse button.

Result

A further branch is added to the branch. The step chain object is inserted into the branch.

2.12.5 Moving parts of the step chain up or down

Introduction

Initial situation:
- You need space to insert new objects into a step chain.
- You have deleted objects from a step chain. You want to represent the remaining objects in compact form.

Solution: Starting from a start symbol you move the following parts of the step chain up or down:
- The start symbol
- All the symbols lying under it
- The symbols lying at the same height as the start symbol
2.12 Creating and numbering step sequences

Requirement

- A function diagram is open. See also chapter Creating a function diagram (Page 9).
- A step chain is placed on the function diagram. See also chapter Creating step sequences (SFCs) (Page 46).

Procedure

1. Select the start symbol.
2. In the context menu of the symbol select the command "Options > Move step chain objects down" or "... > Move step chain objects up".
3. In the window "Move step chain objects up" or "Move step chain objects down" enter the number of grid points by which the symbols are moved.

   **Note**
   **Changing the grid size**

   You can change the grid size via the report toolbar.

4. Confirm your input.

Result

- The symbols are moved vertically.
- Moving can cause connections across pages to arise or disappear.
- Moving up: The symbols are moved at the maximum to below the unmoved part of the step chain. This ensures that symbols do not overlap.
- Input step and signals are not moved.

2.12.6 Moving or copying step chain objects with the mouse

Requirement

- A function diagram is open.
- A step chain is placed on the function diagram.
**Procedure**

1. Ensure that sufficient space is available where you want to insert objects in the step chain. If necessary, create space. See also chapter [Moving parts of the step chain up or down](Page 50).

2. Select the symbols of the step chain objects which you want to move or copy. Select the selection so that the step chain logic is not violated during cutting or pasting. Example: Always select a step together with a transition.

3. Proceed as follows:
   - To move objects drag&drop the symbols to a different position in the step chain.
   - To copy objects keep the <C> key pressed and drag&drop the symbols to a different position in the step chain.

   The connecting line of the step chain changes to yellow if pasting is allowed. Alternatively you can also copy, cut and paste step chain objects by using the context menu.

**Result**

- The symbols are inserted on the connecting line and connected accordingly in the database. Pasting can cause connections across pages to arise or disappear.
- New objects are created during copying.
- During moving the old connecting line is closed and the unmoved step chain objects are reconnected correspondingly.

**2.12.7 Automatic numbering of steps, transitions or actions**

**Requirement**

- A function diagram has been created and is open. See chapter [Creating a function diagram](Page 9).
- Step sequence elements are placed. See chapter [Creating step sequences (SFCs)](Page 46).

**Procedure**

1. Select at least two step sequence elements of a type.
2. Select the "Options > Automatic consecutive numbering" command in the context menu.
3. In the "<Type name> Start number" field, enter a number as the starting point for the numbering.
2.13 Placing and using the comment block

Introduction

A comment block can be available at a signal or extended function block.

Requirement

- A function diagram has been created and is open.
  See chapter Creating a function diagram (Page 9).
- The function diagram contains a signal or an extended function block.
  See chapter Placing a signal (Page 17) and Placing the function block (Page 27).
- The base object of the comment block is prepared.
  See chapter Creating a base object for the comment block (Page 72).

Creating a connected comment block

1. Select the "New > A70 Comment block" command from the context menu of the signal or block.
   The column header sticks to the mouse pointer.
2. Click on the desired position of the report.
   The comment block created is automatically connected to the signal or function block.

Creating a free comment block

1. Drag&drop the following object from the base data onto the function diagram: "@10 > A60 > A20 > A90 > A10 Comment block"
   An icon and a text block are displayed.

Editing the comment text in a comment block

1. Single-click twice (do not double-click) on the text area of the text block.
2. Enter a comment.
   To create a break in the text block, use the <Return> key.
3. Click on the report outside of the text block.
   The entry is completed.

Moving the comment block

1. Select the text block.
   The icon cannot be selected and shifts together with the text block.
2. Moving the comment block.
Enlarging the comment block

1. Single-click twice (do not double-click) on the edge of the text block.
2. Change the size of the text block using the grab points.

Exchanging the icon (pictogram)

1. Select the text block.
2. Select the "Properties" command in the context menu.
3. Select a path to another icon in the "Pictogram" field.

Comment block base object

The base object is in the following node:
"@10 > A60 > A20 > A90 > A10 Comment block"

2.14 Interfaces

- Siemens: SIMATIC S7
  You can find additional information on this topic in the "Integrated Engineering" manual.

- FD-based code generation
  See also chapter Code generation based on IEC 61131 (Page 55).
3.1 Overview of code generation

Introduction

The function diagram-based code generation produces control codes based on IEC 61131. Two types of code generation are supported:

- Placeholder technique
- Function diagram technique

You can combine these two techniques.

Overview of the placeholder technique

Text-based code templates are generated using the placeholder technique. For the placeholder technique code blocks are created as text templates in the according target language. Project specific variables are displayed via placeholders which in concrete application cases are replaced with project specific content. In order to do so, the system allows you to automatically generate the block code and connector structure by means of placeholder evaluation.

Overview of the function diagram technique

Graphical planning is used to create a system-neutral and language-neutral code on a function diagram. The neutrally generated code is then exported to the specific target system. The following target systems are supported:

- IEC 61131-3
- SIMATIC "Instruction List (IL)"
- "Structured Text (ST)"

See also

Overview of code objects (Page 56)
3.2 Overview of code objects

3.2.1 Entering code classes

Code classes are defined in order to enable the use of COMOS control code generation in any database structures.

The following code classes are available to you:

- Configuration
- Function
- Hardware
- Function block in library
- Function block in library (specified parameters)
- Language
- Program Organization Unit (POU)
- Resource
- Task
- Block with text replacement
- Variable
- Version

Procedure

1. Open the properties of a block.
2. Select the "System data" tab.
3. Select a code class in the "CO class" attribute.
4. Confirm your input.

Result

Individual objects are identified by class, independent of the COMOS classes.

See also

Definition of “Function diagram” (Page 56)

3.2.2 Definition of “Function diagram”

The function diagram is located underneath the POU. On this chart, the entire program for the function is described as a network by means of the graphical connection of individual elements.
For placed elements, we differentiate between three different groups:

- **Group 1**
  Contains the standard functions for IEC 61131. For these functions, the textual components of the selectable target systems for generation are stored in COMOS. These are standard IEC modules that are marked as relevant for generation by the “CO class” attribute.

- **Group 2**
  Contains user-defined library modules. For these modules you have to state how to generate a call from the source code for the target system. The content of the module or code is not evaluated. It is required that it is available in the target system. A corresponding base object with connectors has to be created for these objects in order to enable use in the engineering project.

- **Group 3**
  Contains user-defined modules that have to be generated from a placeholder text. While placing this POU, the connector structure is calculated and created from the relevant placeholders in order to enable the interconnecting, just as for the other module types.

See also

- Definition of “Program organization unit” (POU) (Page 59)
- Definition of “Configuration” (Page 57)

### 3.2.3 Definition of “Configuration”

A configuration consists of at least one resource based on IEC 61131. The resource contains \(<1...n>\) independent tasks or programs.

For each configuration, you can create a document that describes the hierarchical structure of the tasks and programs as well as the call sequence for single blocks of a resource.

The creation of the contained POUss via this projection can be started on the configuration level. For complete generation, you select the target system and target language.
The COMOS configuration consists of a document that can interconnect resources, tasks and program organization units.

A call, or the according order of the text blocks is generated via the connection information of the complete program course, depending on the target system.
3.2.4 Definition of "Resource"

A resource is a controlling CPU in which a program should run. A resource belongs to a configuration.

You assign a symbol table object and at least one task to the resource.

3.2.5 Definition of "Task"

A task is called cyclically. A task contains a list of programs to be executed, which are assigned to it in a specific order.

There are two ways to make this assignment:

- Via a link object
- Graphically via the connection information of the objects

3.2.6 Definition of "Program organization unit" (POU)

The higher-level object for a program or text block is referred to as a program organization unit (POU).

Each POU corresponds to a block that has to be generated in the target system or a text block. This POU can contain several networks, logics/routines and function diagrams. The name of the POU object has to be identical with the name of the program module or the routine in the target PLC. The module number can automatically be awarded via a higher-level object in the project level.

The POU is an object from which the code generation can be initiated. At this level, you select the target language and the target system. Code generation for individual blocks is triggered at this level.

The following code stocks are stored in text format in the POU:

- Result of the last generation run
- Imported source code

A POU enables delta handling between different code states.

The POU covers the controlling and management of the subordinate described logics. A POU can be placed on the configuration document and is therefore decisive for the definition of block or routine sequences.
3.2.7 Signals (Variables)

All necessary variables are created as COMOS signals. The existing information on the signal object is used for the identification of the signal type.

Signals you implement as a channel receive a channel request. The channel request contains the relevant data for the controller draft.

The channel requests have the following attributes:

- "Y00A00505 Address"
  Hardware address
  The hardware address is optional.

- "Y00A00565 Symbolic address"
  Symbolic address
  The symbolic address has to be unique throughout the program and project.

- "Y00A00517 Data type"

- "Y00A00011 Comment"
  The comment does not have to be set.

- "Y00A00566 Symbol name"
  The symbol name is used for the replacement logic as a local variable name.

These attributes are necessary for the creation and management in the symbol table and are used for text replacement. The data type of the signal is passed on to the subordinate object.

Example for attribute values in the "Attributes > System data" tab:

<table>
<thead>
<tr>
<th>Address</th>
<th>999</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symb. address</td>
<td>F001.A+</td>
</tr>
<tr>
<td>Comment</td>
<td>Alarm High</td>
</tr>
<tr>
<td>Symbol name</td>
<td>A+</td>
</tr>
<tr>
<td>Data type</td>
<td>Boolean, Bit</td>
</tr>
</tbody>
</table>

3.2.8 Overview of the COMOS symbol tables

Introduction

Symbol tables are generated from an engineering project in COMOS for the code generation from placeholder templates as well as for the export to a target system. These are created and managed independently of the future target system in a defined format.

Control systems offer the possibility to symbolically address the used variables. These are deposited in form of tables and can then be used like direct hardware addresses. The data set for a variable contains the symbolic address as well as the real hardware address as well as the data type and a comment line.
In addition to simple data types, you can also assign control internal data structures. Function blocks and data blocks are part of these data structures.

**See also**

[Creating a symbol table](#) (Page 61)

### 3.2.8.1 Creating a symbol table

Symbolic addresses are automatically assigned in the process of the engineering project by the unit structure. The real hardware address is assigned by means of implementation. The symbol table is created and updated.

**Procedure**

1. Select the symbol table object that is assigned to the respective controller CPU (resource).
2. Select the "Symbol table > Refresh symbol table" command in the context menu.

**Result**

All unit parts and function units that have a reference to the symbol table of the assigned CPU are scanned. As a result, all relevant entries for the control program are created in the symbol table and saved in XML format.

**See also**

[Basic principle of the partial update of symbol tables](#) (Page 61)

### 3.2.8.2 Basic principle of the partial update of symbol tables

In addition to the complete new generation of the symbol table there is also the option to start a partial update for individual function groups.

- In the function group context menu, update commands are available.
- Refreshing a function group based symbol table deletes all entries in the symbol table that are part of the function group.
- After the entries in the symbol table were deleted they are created again for the complete table and added in its refreshed form.

For a version comparison, the last status generated in COMOS as well as the last status exported to the controller and imported from there are stored as an XML string. You can perform a text-based status comparison.

**See also**

[Specific entries in the symbol table](#) (Page 62)
### 3.2.8.3 Specific entries in the symbol table

Other than the symbol table entries which are relevant for the controller, additional entries are managed in the COMOS environment. They are needed to process module templates for text replacement and for partial updates. The symbolic address used in the control system can be freely defined by the user and calculated on the attribute with the help of a script from the unit structure. Additionally a grouping information (function group name or subunit description) is managed, as well as a definite local group part symbol (local symbol).

The entries for the function group and the local symbol are saved in a separate column in order to enable direct access to the entries via this identification. This offers the option to apply a search strategy for the replacement texts of the program templates.

A result example is displayed in the following:

<table>
<thead>
<tr>
<th>Local name</th>
<th>Symbol</th>
<th>Address</th>
<th>Data type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A+</td>
<td>F001,A+</td>
<td></td>
<td>BOOL</td>
<td>Alarm High</td>
</tr>
<tr>
<td>A−</td>
<td>F001,A−</td>
<td></td>
<td>BOOL</td>
<td>Alarm Low</td>
</tr>
<tr>
<td>AV+</td>
<td>F001,AV+</td>
<td></td>
<td>WORD</td>
<td>Alarm High</td>
</tr>
<tr>
<td>AV−</td>
<td>F001,AV−</td>
<td></td>
<td>WORD</td>
<td>Alarm Low</td>
</tr>
<tr>
<td>FB110</td>
<td>FB110</td>
<td>FB110</td>
<td>FB110</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>F001,I</td>
<td></td>
<td>WORD</td>
<td>Flow</td>
</tr>
</tbody>
</table>

### 3.3 Describing and exporting codes

#### 3.3.1 Preparing code generation for a block

**Introduction**

After completing the preliminary administrative work, you can assign software functions to the unit functions at the unit end with the help of a function diagram.

**Procedure**

1. Create a new position below the folder “50 I&C (instrumentation & control)”.
2. Select the “New > POE Program organization unit” command in the context menu of the position.
3. The program organization unit can subsequently be described by a function diagram, a placeholder block or by a combination of a function diagram and a placeholder block:
   - To select a function diagram, select the "New > FF.001 function diagram IEC, A3" command in the program organization unit context menu.
   - To select a placeholder block, select the "New > A10 Auto-symbol placeholder module" command in the program organization unit context menu.
4. Open the function diagram you previously created.

5. To interconnect the placeholder block, drag&drop the placeholder block onto the function diagram.
   You can trigger the code generation.

---

**Note**

Interconnect the placeholder block only to internal signals. You can connect the placeholder block upstream or downstream on a separate function block diagram.

---

See also

- Exporting the block (Page 63)

### 3.3.2 Exporting the block

**Requirement**

- A POU is created.  
  See chapter Preparing code generation for a block (Page 62).
- A function diagram and a placeholder block have been created under the POU.  
  See chapter Preparing code generation for a block (Page 62).
- The placeholder block has been placed in the function diagram and has been processed.  
  See chapter Preparing code generation for a block (Page 62).

**Procedure**

1. Select the program organization unit.
2. Select the "Export IEC 61131-3" command in the context menu.
   With this procedure, only a single block can be described in terms of software. In order to describe a complete control program in terms of software, you assign a calling structure, "CFB Caller FB" and a "CPU-assigned PLC" to the new position.

See also

- Preparing code generation for a program (Page 63)

### 3.3.3 Preparing code generation for a program

**Introduction**

In order to describe a complete control program in terms of software, you assign a "CFB Caller FB" calling structure and an "Assigned station" to the new position.
Procedure

1. If the "Tasks > SW interface" tab already has a task for "Caller FB" and "Assigned station", assign the targets using drag&drop.
   If the corresponding tasks do not exist, select the "New > Assigned station" or "New > Caller FB" command in the context menu and then assign the targets.

2. Open the configuration diagram.

3. Using drag&drop, drag the program organization unit to the configuration diagram.

4. Using the connection tool, connect the program organization unit to the existing program organization units.

5. Once you have completed the interconnection, you can generate the code for the complete control program.

See also

Exiting the program (Page 64)

3.3.4 Exporting the program

Requirement

- A configuration and function diagram have been created for the configuration.
  See chapter Creating a function diagram (Page 9).
- POUs have been created, edited and interconnected on the function diagram of the configuration.
  See chapter Preparing code generation for a block (Page 62).
- The engineering task for the software interface has been edited.
  See chapter Preparing code generation for a program (Page 63).

Procedure

1. Select the "Export IEC 61131-3" command in the context menu of the "KFG1 configuration" object.
4.1 Managing function diagrams based on IEC

4.1.1 Overview of report templates

The following table shows the prepared report templates for function diagrams. The report templates are located in the document tree of the base project.

<table>
<thead>
<tr>
<th>Template</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A30.F.FF.FF_A10.FF_A10</td>
<td>Function diagram</td>
<td>Based on IEC, with toolbar with standard IEC function blocks and step chain elements</td>
</tr>
</tbody>
</table>

See also

[Creating a user-defined template for a function diagram](Page 65)

4.1.2 Creating a user-defined template for a function diagram

Introduction

To create a user-defined template for a function diagram, use the standard functionality of the interactive reports.

Procedure

To enter metadata for the function diagram, follow these steps:

1. Open the function diagram properties.
2. Select the "Attributes > Header" tab.
   This tab shows the objects from which values are used for the metadata.
3. Enter a description for the function diagram.
4. Click "OK".
   You can find more information on this topic in the "COMOS Platform Administration" manual, keyword "Administration of report templates".
4.1.3 Setting the connector type combinations

Interaction with the "Contact point type options (signals)" standard table

The "Allowed connection type combinations" dialog is an alternative interface for editing the following standard table: "@10 > M00 > A10 > Y10M00N00007 > Y10M00N00007S Contact point type options (signals)"

All changes in the "Allowed connection type combinations" dialog are applied in the "Contact point type options (signals)" standard table and vice versa.

You cannot open the "Allowed connection type combinations" dialog and the standard table at the same time because the interfaces are not updated immediately. This means that outdated information could be displayed in the respective other dialog.

Certain connection type combinations are fixed.

See also chapter Combinations permitted by data hierarchy (Page 74).

Opening the "Allowed connection type combinations" dialog

1. Change to the base project.
2. Select the "Administrator > Base data > Allowed connection type combinations" command in the menu bar.

Display of the table

The table occupies the space available in the working area.

If the size of the working area is changed, the table is enlarged or reduced accordingly.

Defining individual connection type combinations

1. Click on a white box.
   All check boxes on the diagonals are already selected and cannot be cleared.
   You cannot connect an input to an input or an output to an output. This is why you cannot select these combinations.

Defining multiple similar connection type combinations

1. Click on a white box and hold down the mouse button.
2. Holding the button down, drag the mouse over the white box.

Result

When the table is saved, the allowed type combinations are written to the "Contact point type options (signals)" system table. A sequential entry is generated to this purpose in the "Name" column.
4.1.4 Reference: Variables in the signal symbol script

Variables

The following table shows the variables used to configure the signals in the options script of a function diagram template. All variables are of the "DOUBLE" data type:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SignalLeft_X</td>
<td>Specifies the distance between a signal and the left edge of the input area in the report. Unit: Millimeters</td>
<td>Starting from a zoom factor of 125% or greater, the distance between two grid points is 1 mm.</td>
</tr>
<tr>
<td>SignalRight_X</td>
<td>Specifies the distance between a signal and the right edge of the output area in the report. Unit: Millimeters</td>
<td>Starting from a zoom factor of 125% or greater, the distance between two grid points is 1 mm. This distance is calculated starting from the left edge. To perform this calculation, subtract the required distance from the width of the report and assign the difference to the variable.</td>
</tr>
<tr>
<td>SignalTop_Y</td>
<td>Specifies the distance between the first signal and the top edge of the report. Unit: Millimeters</td>
<td>The signals are placed in the input and output area of the report starting from the distance at the top.</td>
</tr>
<tr>
<td>SignalSlot_Count</td>
<td>Specifies the number of entries in the input and output area of the report. Unit: none</td>
<td>If the number of entries multiplied by the height of the entries is greater than the height of the report, the entries exceeding the height are not displayed in the report. ( &lt;\text{number of entries}&gt; \times &lt;\text{height of entry}&gt; \leq &lt;\text{height of report}&gt; )</td>
</tr>
<tr>
<td>SignalSlot_Height</td>
<td>Specifies the height of an entry in the input and output area of the report. Unit: Millimeters</td>
<td>If the number of entries multiplied by the height of the entries is greater than the height of the report, the entries exceeding the height are not displayed in the report. ( &lt;\text{number of entries}&gt; \times &lt;\text{height of entry}&gt; \leq &lt;\text{height of report}&gt; )</td>
</tr>
<tr>
<td>SignalWidth</td>
<td>Specifies the width of a signal in the report. Unit: Millimeters</td>
<td>The variable value is only evaluated for symbols without a graphical display. The sum of the variable values of &quot;SignalWidth1&quot; and &quot;SignalWidthGraphic&quot; must not exceed the variable value of &quot;SignalWidth&quot;.</td>
</tr>
<tr>
<td>SignalWidth1</td>
<td>Specifies the width of the front part of a signal in the report. Unit: Millimeters</td>
<td>The variable value is only evaluated for symbols without a graphical display. The sum of the variable values of &quot;SignalWidth1&quot; and &quot;SignalWidthGraphic&quot; must not exceed the variable value of &quot;SignalWidth&quot;.</td>
</tr>
<tr>
<td>SignalWidthGraphic</td>
<td>Specifies the width of the back part of a signal in the report. Unit: Millimeters</td>
<td>The variable value is only evaluated for symbols without a graphical display. The sum of the variable values of &quot;SignalWidth1&quot; and &quot;SignalWidthGraphic&quot; must not exceed the variable value of &quot;SignalWidth&quot;.</td>
</tr>
</tbody>
</table>
Comparison of SignalWidth and SignalWidth1

See also

Reference: Variables in the options script of the function diagrams (Page 68)

4.1.5 Reference: Variables in the options script of the function diagrams

The following table shows the variables used to configure the function diagram template in the options script. All variables are of the "STRING" data type:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Specifies the report template type.</td>
<td>For a function diagram, assign the &quot;FUP&quot; value to this variable. This creates placed base objects below the document (unlike with ET reports). &quot;FUP&quot;-type documents always have multiple pages.</td>
</tr>
<tr>
<td>EnablePaging</td>
<td>Activates the buttons &quot;Navigate to predecessor&quot; and &quot;Navigate to successor&quot; in the toolbar.</td>
<td>You can find additional information on this topic in the &quot;EI&amp;C Administration&quot; manual, keyword &quot;EnablePaging&quot;.</td>
</tr>
<tr>
<td>Symboltype</td>
<td>Specifies the standard you use for configuration in the function diagram.</td>
<td>Assign the following value to the variable: IEC: &quot;M42_P1&quot;</td>
</tr>
<tr>
<td>UseSlot</td>
<td>Controls the use of slots.</td>
<td>To prevent placed signals from overlapping, activate the &quot;UseSlot&quot; option in the &quot;System data&quot; tab in the signal properties. When the &quot;UseSlot&quot; option is activated, the signals are always positioned to the left or right edge of the function diagram.</td>
</tr>
<tr>
<td>PageModeY = 1</td>
<td>Controls the page break.</td>
<td>In contrast to other interactive reports, function diagrams have several pages.</td>
</tr>
</tbody>
</table>

See also

Reference: Variables in the signal symbol script (Page 67)
4.1.6 Adjusting block base objects

4.1.6.1 Overview of block base objects

Block families

The base objects for blocks are located in the Navigator on the "Base objects" tab under the following node:

"@30 > M40 > A50 > A10 > A20 > A10 > A10 IEC 61131-3"

The blocks are grouped into block families. A block family contains blocks with similar functionality, for example, control blocks.

IEC knows function blocks with an undefined number of connectors. Such blocks are also prepared in COMOS with a varying number of connectors. The user can determine the number of connectors in various ways.

Examples:

- "@30 > M40 > A50 > A10 > A20 > A10 > A10 > A10 > A40 > A10 AND AND-gate"
- "@30 > M40 > A50 > A10 > A20 > A10 > A10 > A10 > A40 > A20 OR OR-gate"
- "@30 > M40 > A50 > A10 > A20 > A10 > A10 > A50 > A50 MUX Multiplexer"
- "@30 > M40 > A50 > A10 > A20 > A10 > A10 > A30 > A50 Parallel branch"
- "@30 > M40 > A50 > A10 > A20 > A10 > A10 > A30 > A60 Alternative branch"

This is carried out by means of a grab point, which can be used to change the size of the function block. This grab point becomes visible when the symbol is marked twice. Depending on the size, connectors are generated automatically with grid spacing of "four".

See also

Placing a template in the function diagram (Page 14)

4.1.6.2 Creating a user-defined block

Introduction

To create a user-defined block, use the Symbol Designer.

Procedure

1. For a user-defined block, create a new base object node as well as a new base object.
2. Define a separate graphical representation for each function diagram type in the base object.
   You can find more information on this topic in the "COMOS Platform Administration" manual, keyword "Defining symbols in Symbol Designer".
4.1.6.3 Specifications for a connector interlock

Attributes at the base object

- Attribute
  
  "@10 > A60 > A20 > A10 > A10 > A40 > Y00T00001 > Y00A03973 Field length connection locking"

  Specifies the width of the text field in grid points.

Extended attribute at the connector

- "Locking"

  Controls the interlock of the connector.

See also

  Link to Locking a connector by means of a constant signal value (Page 23)

4.1.6.4 Creating user-defined macro blocks

Introduction

You can create a user-defined block with the scope of functions of a macro block.

Requirement

- The following properties are available in the "Attributes > System data" tab:
  - "Y00A00514 Connector distance"
  - "Y00A00540 Macro block"

Procedure

1. Open the properties of the respective base object.
2. Select the "Classification" tab.
3. Enter the following classification in the "Functional classification" field: "O2 A780 A510 B960"
4. To automate the calculation of function diagram symbols, store the following symbol script in the "Symbols" tab:

   CreateObject("ComosGeoSymbols.MacroBlock").CreateScript Me, PARAM, "M42_P1", "???")
4.1 Managing function diagrams based on IEC

4.1.6.5 Creating user-defined dynamic blocks

Requirement

- The following property is available in the "Attributes > System data" tab:
  - "Y00A00514 Connector distance"

Procedure

1. Open the properties of the respective base object.
2. Select the "Classification" tab.
3. Enter the following classification in the "Functional classification" field: "O2 A510"
4. If you want the function diagram symbols to be calculated automatically, store the following symbol script in the "Symbols" tab:
   ```javascript
   CreateObject("ComosGeoSymbols.MacroBlock").CreateScript Me, PARAM, "M42_P1", "??"
   ```

4.1.6.6 Creating a template for a macro block

Introduction

Function diagram templates can be created for macro blocks in the "@Template" branch in both the base project and the engineering project.

General rules

Adhere to the following rules:

- A function diagram is created below the assembly element. Edit this function diagram like any standard function diagram.
- All objects placed on the function diagram should be arranged below the document.
- The use of any function block is permitted.
- The signals used should be assigned software input or software output as "signal type".
- Only connected signals are evaluated.
- The data type on the resulting macro block connection is calculated from the data type of the signal used.
Creating a new macro block base object

1. Create a base object of the "Macro block" type. Alternatively, copy the macro block example into the database: 
   "@30 > M40 > A50 > A10 > A20 > A10 > A10 > A40 > A10 Macro block".
2. Open the macro block properties.
3. Select the "Attributes > System data" tab.
4. To assign a template to the detail drawing, click the "..." button next to the "Macro block" field.
5. To set the grid spacing for the graphic display of the symbol, select the "Simple" or "Double" entry from the "Connector distance" list.
6. Click "OK".

Creating a new template object

1. Select the following branch: 
   "@Template Templates".
2. Select the "New > General > New object" command in the context menu.
3. Open the properties of the new object.
4. Select the "General" tab.
5. Set the macro block base object in the "Base object" field.

Designing detail drawing (creating internal function diagram)

1. Select the "New > General > New document" command in the context menu of the new template object.
2. Open the properties of the new document.
3. Select the "Report" tab.
4. Set the function diagram report template in the "Report template" field.
5. Edit the function diagram.

4.1.6.7 Creating a base object for the comment block

Introduction

You can provide a signal or a function block with an optional comment block.

Creating the base object for the comment block

1. You can find the base object under the following node: 
   "@10 > A60 > A20 > A10 > A90 > A10 Comment block".
   A comment block has the functional classification "B970".
Adding a comment block to a block

1. Open the properties of the block.
2. Select the "Elements" tab.
3. In the Navigator, select the base object of the comment block.
4. Drag&drop the base object of the comment block onto the "Elements" tab.

4.1.7 Checking and adjusting standard tables

4.1.7.1 Overview of the standard tables for function diagrams

Standard tables in the base project

In the base project, the following standard tables are used in function diagrams:

- "@40 > Y00 Standard table for attributes"
  Contains standard tables for the parameterizing of function diagrams and blocks.

- "@40 > Y00 > A10 > A30 > Y00N0060 Type of step"
  Distinguishes between the initial step and the ("normal") steps.
  Used in:
  "@10 > A60 > A20 > A10 > A10 > A30 > A10 Step", "Step parameters" tab
  Effect: Another symbol is displayed by means of attributes with graphical relevance.

- "@40 > Y00 > A10 > A30 > Y00N0054 Definition mark"
  Used in:
  "@10 > A60 > A20 > A10 > A10 > A30 > A40 Action", "Layout" tab

- "@40 > Y00 > A10 > A30 > Y00N0051 Code line selection"
  Used in:
  "@10 > A60 > A20 > A10 > A10 > A40 > A90 S Split range"
  Used in the "Split range" tab, "Characteristic curve type" attribute.
  Effect: The symbol is changed, depending on the setting.

- "@40 > Y00 > A10 > A30 > Y00N0053 Controller type"
  Used in:
  "@10 > A60 > A20 > A10 > A10 > A40 > A50 Controller"
  Used in the "Regulator" tab (or Controller tab), "Regulator (controller) type" attribute.
Standard tables in the system project

In the system project, the following standard tables are used in function diagrams:

- "@10 > M00 > A10 > Y10M00N00006 > Y10M00N00006F Contact point types (FC)"
  Defines the thickness and type of the connection lines in the step sequence.

- "@10 > M00 > A10 > Y10M00N00006 > Y10M00N00006S Contact point types (signals)"
  Contains data types for block connectors. Connectors are created in the base object and selected from the properties window for the connectors of the "Signal" type. All useable data types are defined here.

- "@10 > M00 > A10 > Y10M00N00007 > Y10M00N00007S Contact point type options (signals)"
  Contains the connection rules for the individual data types of the signals.

See also

Opening standard tables (Page 74)

4.1.7.2 Opening standard tables

Procedure

1. In order to open a standard table with write access, open the base project or system project.
2. Select the menu command "Administrator > Base data > Standard tables".

See also

Adjusting standard tables (Page 75)

4.1.7.3 Combinations permitted by data hierarchy

Compatible data types (permissible subtype connections)

In the default settings, the following subtype connection combinations are permitted:

- Block inputs and block outputs with the same data type
- Combinations of the following hierarchy:

<table>
<thead>
<tr>
<th>ANY</th>
<th>ANY_BIT</th>
<th>ANY_NUM</th>
<th>ANY_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOLEAN</td>
<td>INT</td>
<td>UINT</td>
<td>REAL</td>
</tr>
<tr>
<td>BYTE</td>
<td>SINT</td>
<td>USINT</td>
<td>LREAL</td>
</tr>
</tbody>
</table>
These connection combinations are also permitted when they are not included in the following standard table:

"@10 > M00 > A10 > Y10M00N00007 > Y10M00N00007S Contact point type options (signals)"

The following figure shows the classification of the data types from priority 1 to 4:

Configuring compatible data types (enables additional connection combinations)

To allow subtypes to be connected, use the following standard table:

- "@10 > M00 > A10 > Y10M00N00007 > Y10M00N00007S Contact point type options (signals)"
  This standard table controls which subtypes may be interconnected. Each subtype in "Value 2" may be connected to the corresponding subtype in "Value 1".

4.1.7.4 Adjusting standard tables

Creating additional data types

To create additional data types, use the following standard tables:

- "@40 > Y00 > A10 > A20 > Y00N00044 Signal type"
  This standard table is used in the "Data Type" attribute of a signal.

- "@10 > M00 > A10 > Y10M00N00006 > Y10M00N00006S Contact point types (signals)"
  This standard table is used in the "Subtype" attribute of a connector.
If, for example, you change the data type of a signal at the engineering end, the subtype of the associated connector will automatically be modified accordingly.

Configuring the connection lines

To configure connecting lines, use the following standard table:

- "@10 > M00 > A10 > Y10M00N00006 > Y10M00N00006S Contact point types (signals)"

The following table shows the parameters of this standard table:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value 1</td>
<td>Data type of the connector</td>
</tr>
<tr>
<td>Value 2</td>
<td>Line width in point</td>
</tr>
<tr>
<td>Value 3</td>
<td>Line type: Value range from 1 to 9</td>
</tr>
<tr>
<td></td>
<td>- 1: solid line</td>
</tr>
<tr>
<td></td>
<td>- 2: dashed line</td>
</tr>
<tr>
<td></td>
<td>- 9: dash-dot-dash line</td>
</tr>
<tr>
<td>Value 8</td>
<td>Data type priority: The higher the value, the higher the priority</td>
</tr>
<tr>
<td></td>
<td>- 1-4: Standard data types</td>
</tr>
<tr>
<td></td>
<td>- 5 and higher: User-defined data type</td>
</tr>
</tbody>
</table>

4.2 Managing code generation based on IEC 61131

4.2.1 Reference: Base objects for code generation

You require the following objects

- The objects under the node "@10 > A60 > A20 > A10 > A90 Organizational units".
- The objects under the node "@10 > A60 > A20 > A10 > A90 Miscellaneous base units".
- The queries under the node "@20 > A70 > Y10 > M42 Logical".

You also require the following nodes:

- "@30 > M40 > A50 > A10 > A20 > A10 > A10 > A90 Organizational units"
  Contains the structure for describing configurations based on IEC 61131.
- "@30 > M40 > A50 > A10 > A20 > A10 > A10 > A90 > A30 POU Program organization unit"
  Contains the structural description of how a program unit can be set up. Here, you will find a function diagram, which you can create in an engineering project, plus a placeholder template. Via the template, you can create code blocks to generate control code.
4.2.2 Creating text templates for a placeholder text

**Introduction**

You can create text templates for a placeholder text for each placeholder template.

**Procedure**

1. Select the "New > New document" command in the context menu of the placeholder template.
2. Select the "General" tab.
3. To specify the document type, select the "NetTemplateEditor" entry from the "Type" list.
4. Enter "@30 > M40 > A50 > A10 > A20 > A10 > A20 > A30 > A60 NT Network templates" as the base object.
5. On the "Attribute" tab, select the hardware, version and language.
6. Enter the text template in the "Template" attribute.

See also chapter Managing code generation based on IEC 61131 (Page 76).

![Template example](image)

**Note**

All templates created under a block must have the same number and type of connectors.

**See also**

[Integration of variants/control structures](Page 79)
4.2.3 Reference: Symbolism in an own function group

Basically you differentiate between symbolic names in an own function group and references to neighboring function groups. In the placeholder for data blocks or variables of the own function group, the local symbols are entered without a function group reference.

Texts that are to be replaced will hereafter be marked with curly brackets. The text in the brackets will be replaced. If, for example, the resulting symbols must be enclosed in quotes, place these symbols outside the brackets.

Example

\{FGDB\} \rightarrow \text{Placeholder for the local FGDB symbol.}
\{B1MV\} \rightarrow \text{Placeholder for the local B1MV symbol.}

In addition to references to a symbol entry, you can also reference attributes in the structure.

The syntax is as follows:
\{[\text{Marker}]: \text{[Reference]}\}

The reference is specified as follows:
"\text{Tab}\.\text{Attribute}_\text{name}"

Following entries are defined as markers:

<table>
<thead>
<tr>
<th>Marker</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Attribute of the variable group</td>
</tr>
<tr>
<td>OA</td>
<td>Attribute of the owner object</td>
</tr>
<tr>
<td>UA</td>
<td>Attribute of the overlaying unit</td>
</tr>
<tr>
<td>DA</td>
<td>Attribute of the overlaying Device</td>
</tr>
</tbody>
</table>

Example

\{OA: ADR.AdrIn\}

Currently there is an exception when using:

\{A: FGName\} \rightarrow \text{Reference to the following attribute of the variable group:}
"SYS008"

This reference can also access this attribute of a referenced variable group using the specification of a software interface.

\{A: HE.FGName\}

This functionality is only used for the FUP-based code generation. Following marker entries are used here:

<table>
<thead>
<tr>
<th>Connectors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;CI&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>&quot;CO&quot;</td>
<td>Output</td>
</tr>
</tbody>
</table>

In this case, the connector name is specified instead of the symbol.

Inputs and outputs can also be described as markers for symbol creation on a function diagram.
Example

{CI: Input1}
{CO: Output1}

4.2.4 Overview: Assigned function group

For placeholders that contain a reference to an assigned neighboring function group, an additional reference to the according function group type is added. The syntax inside the curly bracket first of all contains a reference to the function group. This reference is namely stated via the action object. The entry of the local symbolic name is divided by a point.

Example: {L.I} -> Reference to a signal of an assigned fill-level measurement.

4.2.5 Integration of variants/control structures

Requirement

- Placeholder texts are available.
  See chapter Creating text templates for a placeholder text (Page 77).

Introduction

To minimize the number of network templates and thus the maintenance work involved, there is an option that helps to easily integrate variants in a network template. In reference to commonly used programming languages control structures are integrated. The control structure serves to prepare certain program parts dynamically. The syntax of keywords is not case sensitive.

See also

Control structure #Exist-#Default-#EndExist (Page 79)
Control structure If-Else-EndIf (Page 80)
Control structure #ForEach-#Next (Page 81)
Control structure #include (Page 82)

4.2.5.1 Control structure #Exist-#Default-#EndExist

Syntax

```plaintext
//Exist([Marker:]Condition)
.
.
.
[//Default]
```
If the #Exist instruction results in "true", the code below the instruction is not commented out. If the result is "false", the code is commented out. The code below the optional #Default instruction is used if the #Exist condition is false. The #EndExist instruction completes the Exist block.

Following markers can be used:

- Marker I (external interface):
  The instruction checks if a reference to an interface exists.
  Example:
  ```plaintext
  //Exist(I:BS2)
  ```

- Marker S (symbol):
  The instruction checks if the symbol is located in the symbol table.
  Example:
  ```plaintext
  //Exist(S:B1TK1)
  //Exist(S:BS1.B1TK1)
  ```

If no marker is used, a check is performed to determine if there is a symbol in the symbol table.

See also
Integration of variants/control structures (Page 79)

4.2.5.2 Control structure #If-#Else-#EndIf

Syntax
```plaintext
//#If([Marker:]Operand1 [Operator] [[Marker:]Operand2])
  .
  .
  .
  //[#Else]
  .
  .
  .
  //#Endif
```

Only use the #If-#Else-#Endif control structure if certain program sections are to be used only under a specific condition. The to be evaluated condition must always return a Boolean value.

Is the result of the evaluation is true, the code below the #If instruction is not commented out. The code below the optional #Else instruction is used if the #If condition is false.

The #EndIf instruction completes the control structure.
The following comparison operators are currently supported:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a == b</td>
<td>The result of the operation is true if a equals b.</td>
</tr>
<tr>
<td>a != b</td>
<td>The result of the operation is true if a does not equal b.</td>
</tr>
<tr>
<td>a &gt; b</td>
<td>The result of the operation is true if a is greater than b (numeric data types).</td>
</tr>
<tr>
<td>a &lt; b</td>
<td>The result of the operation is true if a is smaller than b (numeric data types).</td>
</tr>
<tr>
<td>a &gt;= b</td>
<td>The result of the operation is true if a is greater than or equal to b (numeric data types).</td>
</tr>
<tr>
<td>a &lt;= b</td>
<td>The result of the operation is true if a is smaller than equals b (numeric data types).</td>
</tr>
</tbody>
</table>

You can additionally check for the values "true" or "1".

**Example**
```
//If(1)
./
//Else
./
//Endif
```

This can be used, for example, to comment out complete code passages.

You can use the following markers:
- Markers for attribute access

**Examples**
```
//If(HE.A:FGName!="")
//If(OA:SYS.SYS008="Test")
//If(DA:ADR.ADRIN>4)
```

**See also**
[Integration of variants/control structures](Page 79)

4.2.5.3 Control structure #ForEach-#Next

**Syntax with operand**
```
//#ForEach(Operand)
.
.
//#Next
```

Use the #ForEach-#Next control structure to insert a number of identical program sections depending on an operator. For each element of the operator a program part with the according count setting is added.
```
//#ForEach ({B1TK*})
.
.
```
Syntax with rules

Further usage can arise for the composition description of a module from multiple templates. In the header specification of the block to be generated, the code for the creation can be described as follows:

```c
//#ForEach (B:SB*)
//#include(SF\NT17)
//#Next
```

According to the rule:

For each back pointer of an SB* task

Go to the source in folder SF and integrate NT17

See also

Integration of variants/control structures  (Page 79)

4.2.5.4 Control structure #include

Syntax

```c
//#include(Operand)
```

The use of include structures is used to embed different text blocks.

Example 1

```c
//#include(SF/NT17)
```

In this case, the ST specification of an object from the file path in the operand is copied completely to the position of the instruction. This case is required to copy in complete text blocks.

You specify the path starting at the functional group object.

Example 2

```c
//#include("UC "{A:A.Label}" ;")
```

In this usage the to be built string is indicated in the operator. Therefore the already known mechanisms from the placeholder technique are available. The to be built text line is generated instead of the include instruction.
See also

Integration of variants/control structures (Page 79)
4.2 Managing code generation based on IEC 61131