<table>
<thead>
<tr>
<th>Trademarks</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic principles</td>
<td>2</td>
</tr>
<tr>
<td>Configuring function diagrams based on IEC</td>
<td>3</td>
</tr>
<tr>
<td>Configuring function diagrams based on VGB</td>
<td>4</td>
</tr>
<tr>
<td>Code generation based on IEC 61131</td>
<td>5</td>
</tr>
<tr>
<td>Administration</td>
<td>6</td>
</tr>
</tbody>
</table>

**SIEMENS**

**COMOS**

**Automation Logical**

**Operating Manual**

04/2012

A5E03778257-01
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
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

⚠️ CAUTION
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

⚠️ NOTICE
indicates that an unintended result or situation can occur if the relevant information is not taken into account.

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 Trademarks ................................................................................................................................................ 7

2 Basic principles.......................................................................................................................................... 9
  2.1 Aim..................................................................................................................................................... 9
  2.2 Block-specific definitions............................................................................................................ 9

3 Configuring function diagrams based on IEC.......................................................................................... 11
  3.1 Overview of the configuration in accordance with IEC ................................................................. 11
  3.2 Creating a function diagram........................................................................................................ 11
  3.3 Structure of an IEC function diagram....................................................................................... 12
  3.4 Basic functions of an IEC function diagram............................................................................ 13
  3.5 Placing a template in the function diagram........................................................................... 14
  3.6 Using the signal............................................................................................................................ 15
    3.6.1 Signal-specific definitions........................................................................................................ 15
    3.6.2 Overview of the signal types.................................................................................................. 16
    3.6.3 Checking the signal type......................................................................................................... 17
    3.6.4 Placing the signal.................................................................................................................... 17
    3.6.5 Display of the signal............................................................................................................. 18
    3.6.6 Navigating in the function diagram with signals.................................................................. 18
    3.6.7 Using the signal - cross-reference list................................................................................ 20
  3.7 Connector interlock due to constant signal value........................................................................ 22
  3.8 Using the function block................................................................................................................ 23
    3.8.1 Function block-specific definitions...................................................................................... 23
    3.8.2 Placing the function block.................................................................................................... 24
    3.8.3 Using the function block........................................................................................................ 24
  3.9 Placing the dynamic macro block............................................................................................... 26
  3.10 Editing and updating macro blocks.......................................................................................... 27
    3.10.1 Controlling und updating the display.................................................................................. 27
    3.10.2 Editing and checking the detail drawing ............................................................................ 28
    3.10.3 Editing inconsistency.......................................................................................................... 30
  3.11 Updating macro blocks in bulk.................................................................................................. 31
    3.11.1 Opening a tab....................................................................................................................... 31
    3.11.2 Initializing and updating tabs.............................................................................................. 31
    3.11.3 Preliminary work: Checking use and properties................................................................ 32
    3.11.4 Updating macro blocks in bulk........................................................................................... 33
    3.11.5 Information on bulk processing by means of an object query ........................................ 33
  3.12 Printing the function diagram with internal circuits................................................................... 34
  3.13 Connecting blocks.......................................................................................................................... 35
    3.13.1 Connection types for block connectors............................................................................. 35
    3.13.2 Connecting block connectors by means of a reference..................................................... 35
# Table of contents

3.13.3 Connecting block connectors by means of an intermediate variable ........................................ 36
3.13.4 Display of connections in the IEC function diagram .................................................................. 37
3.13.5 Using connections ......................................................................................................................... 38
3.13.6 Negating a connection .................................................................................................................. 39
3.14 Parameterizing blocks ...................................................................................................................... 41
3.15 Creating and numbering step sequences ......................................................................................... 41
3.15.1 Overview of the step sequence object ......................................................................................... 41
3.15.2 Creating step sequences (SFCs) .................................................................................................. 43
3.15.3 Numbering steps, transitions or actions ...................................................................................... 44
3.16 Placing and using the comment block ............................................................................................. 45
3.17 Interfaces ........................................................................................................................................ 46

4 Configuring function diagrams based on VGB .................................................................................. 47
4.1 Overview of VGB-compliant structures ............................................................................................ 47
4.2 Using signal in VGB .......................................................................................................................... 48
4.3 Display of connections in the VGB function diagrams ..................................................................... 48
4.4 Using macro block (auto symbol) ..................................................................................................... 49

5 Code generation based on IEC 61131 ................................................................................................. 51
5.1 Overview of code generation ............................................................................................................. 51
5.2 Overview of code objects ................................................................................................................ 52
5.2.1 Definition of "code classes" ........................................................................................................... 52
5.2.2 Definition of "Function diagram" .................................................................................................. 52
5.2.3 Definition of "Configuration" ......................................................................................................... 53
5.2.4 Definition of "Resource" ............................................................................................................... 54
5.2.5 Definition of "Task" ....................................................................................................................... 54
5.2.6 Definition of "Program organization unit" (POU) ....................................................................... 55
5.2.7 Signals (Variables) ....................................................................................................................... 55
5.2.8 Overview of the COMOS symbol tables ..................................................................................... 56
5.2.8.1 Basic principle of symbol table generation .......................................................................... 56
5.2.8.2 Basic principle of the partial update of symbol tables ............................................................... 57
5.2.8.3 Specific entries in the symbol table .......................................................................................... 57
5.3 Describing and exporting codes ....................................................................................................... 58
5.3.1 Preparing code generation for a block ......................................................................................... 58
5.3.2 Exporting the block ....................................................................................................................... 59
5.3.3 Preparing code generation for a program ................................................................................... 60
5.3.4 Exporting the program .................................................................................................................. 61

6 Administration ...................................................................................................................................... 63
6.1 Allowed connection type combinations ........................................................................................... 63
6.2 Checking and adjusting standard tables .......................................................................................... 64
6.2.1 Overview of the standard tables for function diagrams ............................................................... 64
6.2.2 Opening standard tables ............................................................................................................... 65
6.2.3 Combinations permitted by data hierarchy .................................................................................. 65
6.2.4 Adjusting standard tables ............................................................................................................ 67
6.3 Creating a function diagram template ............................................................................................... 68
6.3.1 Overview of report templates ....................................................................................................... 68
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.2</td>
<td>Creating a user-defined template for a function diagram</td>
<td>68</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Controlling the toolbar</td>
<td>69</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Reference: Variables in the signal symbol script</td>
<td>69</td>
</tr>
<tr>
<td>6.3.5</td>
<td>Reference: Variables in the options script of the function diagrams</td>
<td>70</td>
</tr>
<tr>
<td>6.4</td>
<td>Adjusting block base objects</td>
<td>71</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Overview of block base objects</td>
<td>71</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Creating a user-defined block</td>
<td>72</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Specifications for a connector interlock</td>
<td>72</td>
</tr>
<tr>
<td>6.4.4</td>
<td>Creating user-defined macro blocks</td>
<td>73</td>
</tr>
<tr>
<td>6.4.5</td>
<td>Creating user-defined dynamic blocks</td>
<td>73</td>
</tr>
<tr>
<td>6.4.6</td>
<td>Creating a template for a macro block</td>
<td>73</td>
</tr>
<tr>
<td>6.4.7</td>
<td>Creating a base object for the comment block</td>
<td>75</td>
</tr>
<tr>
<td>6.5</td>
<td>Reference: Script for VGB signals</td>
<td>75</td>
</tr>
<tr>
<td>6.6</td>
<td>Generating code templates</td>
<td>75</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Reference: Base objects for code generation</td>
<td>75</td>
</tr>
<tr>
<td>6.6.2</td>
<td>Creating text templates for a placeholder text</td>
<td>76</td>
</tr>
<tr>
<td>6.6.3</td>
<td>Reference: Symbolism in an own function group</td>
<td>77</td>
</tr>
<tr>
<td>6.6.4</td>
<td>Overview: Assigned function group</td>
<td>78</td>
</tr>
<tr>
<td>6.7</td>
<td>Integration of variants/control structures</td>
<td>79</td>
</tr>
<tr>
<td>6.7.1</td>
<td>Control structure #Exist-#Default-#EndExist</td>
<td>79</td>
</tr>
<tr>
<td>6.7.2</td>
<td>Control structure #If-#Else-#EndIf</td>
<td>80</td>
</tr>
<tr>
<td>6.7.3</td>
<td>Control structure #ForEach-#Next</td>
<td>81</td>
</tr>
<tr>
<td>6.7.4</td>
<td>Control structure #include</td>
<td>82</td>
</tr>
</tbody>
</table>
Trademarks

Registered trademark: COMOS®
2 Basic principles

2.1 Aim

The “Logical” module 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 standards IEC and VGB. The following projects can be used as examples:

- COMOS_FP (Planning example based on IEC 1131-1)
- COMOS_KKS (power station based on KKS standard; VGB R170 C)

2.2 Block-specific definitions

Definition of block

In COMOS, blocks are all objects that can be placed in a function diagram. There are the following types of placeable objects:

- Function blocks
- Transitions
- Steps
- Signals

Definition of block type

A block type is a generally valid, reusable function block. It is independent of the configured PLC, because its algorithm does not process any 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 accept the values that the block type requires in order to manage its task.
Basic principles

2.2 Block-specific definitions

- **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.
3.1 Overview of the configuration in accordance with IEC

Configuration using a function diagram includes the following basic steps:

- Creating a function diagram
- Placing a template in the function diagram
- Placing additional signals in the function diagram input and output area
- Placing additional blocks in the function diagram
- Connecting blocks and signals
- Creating step sequences

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

3.2 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

To create a function diagram, proceed as follows:

- From the "Position" or "Measurement function" context menu, select "New > EFF.001 Function diagram IEC".

  The function diagram is created in the Navigator underneath the position or measurement function.

The corresponding command is no longer available in the context menu once the function diagram has been created. In the COMOS default model, just one function diagram is situated beneath a position or measuring function.
3.3 Structure of an IEC function diagram

Structure

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

1. Toolbar
   The toolbar contains blocks that are prepared in the function diagram template.
2. Header
   The header displays the legend for the input and output signals.
3. Range for input signals/left slot
   Input signals transfer a value from a process to the function diagram for processing.
4. Work area
   You place and connect blocks in the work area.
5. Range for output signals/right slot
   Output signals return a value processed by the function diagram back to the process.
6. Document-defined symbol bar
   The document-defined symbol bar contains blocks that are assembled from the base data.
3.4 Basic functions of an IEC function diagram

Requirement
A function diagram has been created and is open.

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 following manuals:
- "COMOS Platform Getting Started", keyword "Interactive reports"
- "Reports - Basic Operation", keyword "Basic operation of the interactive report"

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.

You can find additional information on this topic in the "Reports - Basic Operation" manual, keyword "Symbol bar”.

Collision check
Objects cannot be placed on the function diagram so that they overlap each other. This also applies to groups.

The simple drawing elements (lines, circles and free texts) are not covered by the collision check.

Fixed symbol sizes
Blocks on function diagrams have a fixed size.

Sample data and library in the COMOS DB
The base objects for blocks are located in the base objects at the following location:
"@03 Structures > FUP Logical engineering > D Catalog FuE"
3.5 Placing a template in the function diagram

Introduction

Example templates for function diagrams based on the IEC standard are prepared in the COMOS DB. With the help of these templates, various standardized measurement and control tasks can be processed. In general, a template contains several already interconnected blocks.

Requirement

A function diagram has been created and is open.
Templates for function diagrams have been created in the base data.

Procedure

To insert a template in a function diagram, proceed as follows:

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

Alternative method

Alternatively, you can copy a function diagram into the properties of a "position". Open the "Properties" of the "Position" and select the function diagram template on the "eBlock > Copy In Function Diagram" tab. This creates the function diagram underneath the "Position".
3.6 Using the signal

3.6.1 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-hand slot or the right-hand slot.

- "Input" signal type
  Transfers a value from the process to the function diagram for processing. Example: A sensor transfers a measuring signal to the process control. You place the "Input signal" in the left-hand area of a function diagram.

- "Output" signal type
  The function diagram transfers a value back to the process. Example: A control block transfers a control signal to the process. Place the "Input signal" in the right-hand area of a function diagram.

See Overview of the signal types (Page 16).

- "System data" tab, "Data type"

The "Data type" at the signal 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 perform the change of 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-hand or right-hand 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.

### 3.6.2 Overview of the signal types

**"A Hardware"**

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

In the COMOS default model, these signals are available in the "02 Signal planning" folder and have been created within the framework of basic engineering. You can find more information on this topic in the "EI&C Operation" manual, keyword "Creating and designing functions".

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

**"C 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-hand slot)
- D Software Output (right-hand slot)

**"E 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-hand slot)
- F Variable Output (right-hand slot)

**"Function block (FB)"**

Only supported from COMOS 9.2 onward for compatibility reasons.

This signal type represents a value at a block connector. In the context of a unit, an FB signal is used to transfer values to or from a block.

**"Z Break"**

Only supported from COMOS 9.2 onward for compatibility reasons.
Break signals are used to distribute a function diagram across multiple interactive reports. Break signals are therefore comparable with the pair references from detail engineering charts. Break signals have a 1-to-n link:

- A break signal can be continued on n diagrams.
- A continued break signal may have exactly one source.

"Connector"

Only supported from COMOS 9.2 onward for compatibility reasons.

This signal type represents a buffer. In the context of a unit, a connector signal is used, for example, to connect blocks to several function diagrams.

3.6.3 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".

Checking the signal type

To check the signal type, proceed as follows:

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

3.6.4 Placing the signal

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".

A function diagram has been created and is open.

Procedure

To place a signal on a function diagram, proceed as follows:

1. Select the signal in the Navigator.
2. Place the signal in the function diagram using drag&drop.
   - Place an "Input" type signal in the left-hand slot.
   - Place an "Output" type signal in the right-hand slot.
   - Place a signal without an "Input" or "Output" type either in the left-hand or the right-hand slot.
     Drag the signal to the left-hand half of the function diagram to insert it into the left-hand slot.
     Drag the signal to the right-hand half of the function diagram to insert it into the right-hand slot.

See also

Structure of an IEC function diagram (Page 12)

3.6.5 Display of the signal

Requirement

A signal is placed on the function diagram.

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>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

① Device label
② Description
③ Cross-reference
④ Label or name of the signal
⑤ Location/channel
⑥ Signal
Display of multiple placing (dynamic cross-reference lists)

In the default setting, a cross-reference arrow shows that a signal is placed on different function diagrams:

In order to display cross-reference texts instead of the cross-reference arrow, follow these steps:

1. In the function diagram toolbar, activate the option "Show references".
2. Click "Apply" (the check mark button on the right).
3. Click on "Reevaluate document".

The ID of the report on which the signal is placed appears under the signal symbol:
To permanently enable the "Show references" option, follow these steps:

**Change the signal default setting**
1. Switch to the base object of the signal.
2. Open the "System data" tab in the properties.
3. Activate the "Show references" option.

**Change the default setting of the report**
1. Switch to the base object of the report.
2. Open the "System data" tab in the properties.
3. Activate the "Show references" option.

### 3.6.6 Navigating in the function diagram with signals

**Requirement**
- A function diagram has been created and is open.
- A signal is placed on the function diagram.

"Navigate" context menu for placed signals

Each signal knows all of its uses, as well as its template and its position in the navigator.
The following navigation options are offered for the use of signals:

- Context menu at the signal, "Navigate > Documents > <Usage>
  The <Usage> entry has the following format:
  <Group sheet number> <Document name> <Document description> <Slot>
You can find additional information on this topic in the manual "COMOS Platform Getting Started", keyword "Navigate".

### 3.6.7 Using the signal - cross-reference list

#### Requirement
A function diagram has been created and is open.
A signal is placed on the function diagram.

#### Signal cross-references (uses of the signal)

To check the signal usages, proceed as follows:

1. In the menu bar, select the command "Extra > Detail > Signal cross-references"
   In the detail area underneath the Navigator, the "Signal cross-references" tab opens.
2. Select a signal in the function diagram.
   In the "Signal cross-references" tab, all uses of the signal appear.
3. On the "Signal cross-references" tab, check the information on the signal.
   - "Source/target" column
     Arrow to the right: Signal is "Source".
     Arrow to the left: Signal is "sink".
   - "Name" column
     Name of the owner and name of the function diagram.
   - "Slot" column
     Line number of the slot, whereby the slot can be left or right.
4. Select an entry in the "Signal cross-references" list.
5. Select the "Open" command in the context menu.
   The corresponding function diagram opens and the placed signal is highlighted.
3.7 Connector interlock due to constant signal value

Requirement

The blocks have the classification "AV". See also chapter Specifications for a connector interlock (Page 72).

Aim

If a constant signal value is entered at an unconnected connector, this connector is interlocked and can no longer be connected.

The connector interlock is available for:

- Macro block

See also chapter Placing the dynamic macro block (Page 26).

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:

4. Overwrite the hash sign with the signal value.
5. Confirm with Return.

The connector can no longer be connected.

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 appears.
4. Delete the constant signal value.
5. Confirm with Return.

The hash sign appears in the input box.
The connector can be connected 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 disappears.
If the connection is terminated, the input box for the constant signal value appears again.

**Controlling connector visibility**

In the "Connector visibility" dialog, the interlocked connectors are displayed. For information on interlocking connectors, see Controlling and updating the display (Page 27).

### 3.8 Using the function block

#### 3.8.1 Function block-specific definitions

**Function block**

A function block is a completed subtask of a sequential control system. Function blocks 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.

A function block consists of the block header and the block interface.
A special case of function blocks are the dynamic blocks and macro blocks.

**Dynamic block**

A dynamic block has two properties:
- The block connectors can be hidden and displayed again in the function diagram.
- The distance of the connectors is controllable via the Properties.

Dynamic blocks are classified in the COMOS default model simultaneously as macro blocks.

**Macro block**

In the hierarchical logical engineering, a macro block is used. It encapsulates a complex function. If the macro block is placed on a higher-level function diagram, the enclosed function logic is available, but the display of the encapsulated function is greatly simplified.
Standard function

Standard functions represent subtasks that cannot be divided any further and which can very likely be used in every function diagram.

Examples:
- AND-gate
- OR gate

3.8.2 Placing the function block

Requirement

A function diagram has been created and is open.

Procedure

To insert a block into a function diagram, proceed as follows:

1. To insert a block form 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 to the function diagram.

Result

The block will be 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 underneath the function diagram.

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 of the block

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

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

Display of a dynamic block

A dynamic block is displayed as follows:

- Hiding and showing the connectors
  In the COMOS default model, a dynamic block is used only in conjunction with a macro block. See the following item, "Display of a macro block".

Display of a macro block

In the COMOS default model, a macro block is a dynamic block at the same time. A macro block has the following possibilities for display:

- Hiding and showing the connectors
  See chapter Creating user-defined dynamic blocks (Page 73).
- Simplified display of the lower-level function diagram
  The lower-level function diagram forms the complex internal structure of the macro block, or its "inner circuitry". For the inner circuitry, a strongly simplified display is generated, which can be connected and moved as usual.

3.8.3 Using the function block

Requirement

A function diagram has been created and is open.

At least one block is placed on the function diagram.

Copying the block

To copy a block, follow these steps:

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.
3.9 Placing the dynamic macro block

**Deleting the block**

To delete a block, follow these steps:

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.

**Placing the macro block**

To place a macro block in the function diagram, proceed as follows:

1. Select the macro block on the "Units" tab in the Navigator.
   The macro block is located underneath the function.
2. Alternatively, create a new macro block with the help of the example objects in the COMOS DB:
   - Go to the “Base objects” tab.
   - Search for the macro block base object.
   - Use drag&drop to drag the base object to the "Units" tab.
   - Select the macro block on the "Units" tab in the Navigator.
3. Drag the macro block to the function diagram using drag&drop.

**Checking the placed macro block**

Each macro block has a reference to a subordinate function diagram. To check the reference to the function diagram, proceed as follows:

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

**Macro block base object in the COMOS DB**

You can find the base objects of the IEC macro block at the following points:

"@03 Structures > FUP > D > 01 > AA > AD > 100 Macro block"
3.10 Editing and updating macro blocks

3.10.1 Controlling and updating the display

Requirement
A function diagram has been created and is open.
A macro block is placed.

Defining connector visibility
To specify the visible connectors, proceed as follows:
1. Select the desired macro block on the function diagram.
2. Click the "Options > Connector visibility" command in the context menu.
3. In the following window, select the connectors you want to be visible.
The symbol graphic is then displayed in an expanded or reduced manner in the function diagram.

Note
Connectors cannot be hidden if they are already connected in the current function diagram.

Updating macro blocks
To update the macro block, follow these steps:
1. Select the desired macro block on the function diagram.
2. Right-click on the block.
3. Click the "Options > Update macro block" command in the context menu.
Defining the connector distance
To define the distance between the connectors and thereby the size of the macro blocks, follow these steps:
1. Select the desired macro block on the function diagram.
2. Click the "Properties" command in the context menu.
3. Select the "Attributes > System" tab.
4. Select one of the following options for the distance:
   - Select relative distance
     In the "Connector distance mode" list, select the "Current grid size" entry.
     In the "Connector distance" list, select an entry.
     The minimum distance between connectors is four grid points.
   - Select absolute distance
     In the "Connector distance" list, select the "Simple" entry.
     In the "Connector distance mode" list, select an entry. For a 4 grid, the "4 coordinate points" setting corresponds to grid distance 1.

Displaying attributes in the placed macro block
In order to display attributes of the macro block on the function diagram in a symbol, proceed as follows:
1. Create the "USERPARAM" tab in the macro block base object.
2. Create attributes on the tab with the prefix "ADDITIONALP".
The attributes are entered at the bottom of the macro block in a box.

3.10.2 Editing and checking the detail drawing

Requirement
A function diagram has been created and is open.
A macro block is placed.

Displaying detail drawing
To display the detail drawing of a placed macro block, proceed as follows:
1. Select the desired macro block on the function diagram.
2. Right-click on the macro block.
3. Click the "Detail drawing > Display" command in the context menu.
The function diagram with the detail drawing is opened write-protected.
Changing the detail drawing

To change the detail drawing of a macro block, proceed as follows:

1. Select the macro block.
2. In the context menu, select the “Detail drawing > Change” command.
3. Select a project at the very top of 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.
   According to the COMOS default model, the following structure could be:
   "Current Project COMOS_FP > Templates > FUP Logical engineering > 02 Macro blocks"
5. Select the macro block.
6. Confirm the following prompt with "Yes":
   "Should the macro block be updated?"
   The update of an individual block follows the same rules as the bulk update. See chapter Updating macro blocks in bulk (Page 33).

Alternatively, the following context menu is available: "Options -> update macro block".

Editing the detail drawing (generating a copy)

To generate a variant of the macro block detail drawing, proceed as follows:

1. Select the macro block.
2. In the context menu, select the command “Detail drawing > Edit (generate copy)".
3. Select a tab at the very top of the dropdown menu.
4. Select one of the two options:
   – Project
     The project currently open.
   – Base project
     This option has the result that COMOS switches to the base project. If you do not have the required rights, COMOS cancels the editing process.
5. In the navigator, select a storage location.
   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 created in the location selected by you.

The copied function diagram, complete with the macro block detail drawing, is opened. The detail drawing can be changed and saved.
3.10.3 Editing inconsistency

Requirement

A function diagram has been created and is open.
A macro block is placed.

Causes of inconsistencies

An inconsistent macro block might have the following causes:

- The following request was confirmed with "No":
  “Should the macro block be updated?”
- 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, check for inconsistency as follows:

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

Correcting inconsistency automatically

To automatically correct inconsistency, proceed as follows:

1. Use the "Detail drawing > Change" command again.
2. Confirm the following request with "Yes":
   “Should the macro block be updated?”

Correcting inconsistency manually

To correct the inconsistency manually, proceed as follows:

1. Select the block on the function diagram.
2. Right-click on the object. 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 connectors 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 connectors 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 connectors being lost or rewired.

---

### 3.11 Updating macro blocks in bulk

#### 3.11.1 Opening a tab

- From the COMOS menu, select the "Plugins > Automation > Update macro blocks" command.

The "Update macro blocks" tab opens.

#### 3.11.2 Initializing and updating tabs

**Requirement**

In the unit design, several macro blocks are used on function diagrams.
Configuring function diagrams based on IEC

3.11 Updating macro blocks in bulk

Procedure

1. From the navigator, drag a folder or another object into the "Start object" field.
   In the COMOS default model, use a folder from the "Units" tab.
2. From the navigator, drag a macro block into the "macro block" field.
   In the COMOS default model, use a macro block from the "Units" tab.
   COMOS searches the template for this 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, proceed as follows:

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

3.11.3 Preliminary work: Checking use and properties

Requirement

In the unit design, several macro blocks are used on function diagrams.

Checking the use of macro blocks

1. Set the start fields as described.
   See chapter Initializing and updating tabs (Page 31).
2. Check the use in the "Used in document" column.
3. Check the hierarchical context of the macro blocks:
   - Select an entry in the list.
   - Select the "Navigate > Macro block" command from the context menu.

Checking entries in the control fields in advance

- Select the "Navigate" command in the context menu of a control field.

Checking macro blocks in advance

1. Select an entry from the list.
2. Select the "Properties" command in the context menu.
3. Check the macro block properties.
3.11.4 Updating macro blocks in bulk

Requirement

In the unit design, several macro blocks are used on function diagrams.

Configuring tabs

1. In the "New template" field, enter the template from which the macro blocks are to be derived in the future.

   In the COMOS default model, use a base project template from the "@Template Templates" branch. To do this, go to the "Base objects" tab in the navigator. The "@Template Templates" branch can be found right at the bottom of the tab.

2. In the list, select the affected macro blocks using the following options:
   - Selecting/clearing the checkbox
   - Context menu of the list: "Select all macro blocks"
   - Context menu of the list: "Select selected macro blocks"
   - Context menu of the list: "Deselect all macro blocks"

   The context menu of the list is only visible when at least one macro block is listed.

3. Optional: Activate the "Update placed macro blocks" 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 template is changed if the following applies:

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 new connection is closed again. If required, the connector is kinked in order to reach the new connection point.

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

Click "Start".

3.11.5 Information on bulk processing by means of an object query

Requirement

In the unit design, several macro blocks are used on function diagrams.

Alternative bulk processing by means of an object query

In the COMOSDB, macro blocks are classified with "MF". This allows for bulk search and editing of macro blocks with the help of an object query. The corresponding query is located in the base project at the following position:

"@02 General objects > 200 > FUP > Q > 04 Macro blocks template change"

You can find additional information on this topic in the "Queries" manual, keyword "Classification" and "Database search".

3.12 Printing the function diagram with internal circuits

Requirement

At least one macro block is used in a function diagram.

Printing the detail drawing

To print a function diagram including the detail drawing of the placed macro blocks, proceed as follows:

1. Activate the “Print macro block detail drawing” project option:
   - Open the properties of the project.
   - Switch to the “Options > ET/I&C Options 2” tab.
   - Activate the following option:
     "Settings" group, “Print macro block detail drawing” option.

2. Print the document via the print manager:
   COMOS menu "Documents > Print".

The detail drawing is not taken into account if the "Print" function is selected in the function diagram.
3.13 Connecting blocks

3.13.1 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 underneath each function diagram. The intermediate variable is an object.
  - Intermediate variables are used for the following tasks:
    - The connection is acquired in object queries.

3.13.2 Connecting block connectors by means of a reference

Requirement

A function diagram has been created and is open.

The blocks have been created.

Procedure

To connect blocks by means of a reference, proceed as follows:

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 and from the context menu of the inconsistent connection select the "Connection > Memorize [block output label]" command.
   "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.
5. Also create a half-open, inconsistent connection at the target block input.
6. Select the inconsistent connection and from the context menu of the inconsistent connection select the "Connection > Connect with <Block output label>" command.

Disconnecting a memorized connection
To disconnect a memorized connection, proceed as follows:
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".

3.13.3 Connecting block connectors by means of an intermediate variable

Requirement
Function diagrams have been created and opened.
The function diagrams contain blocks.

Procedure
To connect block connectors by means of an intermediate variable, proceed as follows:
1. Creating an object of the "Connector" signal type.
To create the "Connector" signal type, select the command "New > @IEC Signals > @ST Signal types > K Connector" from the context menu of the "Signal engineering" entry in the Navigator.

2. Open the properties of the "Connector" and enter a meaningful "description".

3. Drag&drop the "Connector" to 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.

4. Connect the "Connector" to the block output.

5. From the Navigator, drag&drop the "Connector" to the function diagram in which you want to connect the "Connector" to a block input. You can also place the "connector" on several function diagrams and, in this way, simultaneously connect it with different block inputs (signal targets).

6. Connect the "Connector" to the block input or the block inputs.

3.13.4 Display 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>Color</th>
<th>Description</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:

- [Name of the function diagram].<sheet number>.<slot number>

The following illustration shows the references of a connection across sheets:
3.13.5 Using connections

Checking and editing inconsistent connections

Inconsistent connections are not stored in the database and displayed as "not connected". To display details on the inconsistency, proceed as follows:

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

Signal tracking via the connections

To track a signal, proceed as follows:

1. Select the connection or the reference.
   - The connection is highlighted in color.
2. In the context menu, select one of the commands in "Navigate > ... ".
   - The target positions of the connection (for example, block connectors or base objects) are listed in this submenu.

Automatic routing of the connecting lines

The connections automatically search for the best route to the function diagram. If an object is moved, the connection is automatically re-routed.
In order to manually re-route connections on the function diagram, proceed as follows:

1. Select a connection or several connections.
2. From the context menu, select: "Options > Reroute".

During rerouting, the dynamic connector points are re-set depending on the position of the objects on the function diagram.

Dynamic connector points are connectors that can occur at any position on a connection line or component.

3.13.6 Negating a connection

Requirement

A function diagram has been created and is open.
A connection has been created at a block.

Procedure

To negate a connection, proceed as follows:

1. Select the connection.
2. In the context menu of the connection select the "Settings > Signal Is Negated" command.
Result

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 plan. 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 connecting dot.

See also

Connection types for block connectors (Page 35)
3.14 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.

Requirement

A function diagram has been created and is open.
The blocks have been created.

Procedure

To parameterize a block, proceed as follows:

1. Inserting constants into the function diagram
   - From the toolbar, drag&drop the constant to the function diagram.
   - Connect the output of the constant to the block input.
   - Select the constant and enter the required character string in the toolbar of the function diagram.

2. Assigning parameters to the range limits of the block
   - Open the properties of the block.
   - Enter the values for the range limits on the "Attributes" tab.

3.15 Creating and numbering step sequences

3.15.1 Overview of the step sequence object

Introduction

The objects for step sequence structuring are available in function diagrams in accordance with IEC.
The objects for step sequence structuring can, however, also be used in function diagrams in accordance with VGB. In this case, you also need to use the corresponding base objects.
You can either configure step sequences separately or in combination with other blocks.
Overview of the objects for step sequences

The templates for a step sequence are at the top of the function diagram in the toolbar.

- "01 Step"
  Connectors:
  - Input (top): Transition
  - Output (right): Action block
  - Output (bottom): Transition

- "02 Transition"
  Connectors:
  - Input (top): Step
  - Input (left): Signal (switching condition)
  - Output (bottom): Step

- "03 Action block"
  Classification: "FP > SC > AC"
  An "Action block" has an "Action" element.
  Connector:
  - Input (left): Step

- "04 Simultaneous branch"
  A simultaneous branch has only dynamic connection points.
  The simultaneous branch is scalable.

- "06 Alternative branching"
  An alternative branch has only dynamic connection points.
  An alternative branch is scalable.
Example

The following illustration shows a step sequence 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 sequences in the COMOS DB

"@03 Structures > FUP Logical engineering > D Catalog FuE > 01 IEC1131-3 > AA IEC131-3 > AC Step sequence"

3.15.2 Creating step sequences (SFCs)

Requirement

A function diagram in accordance with IEC has been created and opened.

Creating a step sequence

1. Activate the "Connection" tool.
2. Place the switching conditions on the left of the input area of the function diagram.
3. Create the following step sequence from top to bottom.
4. Place the actions in the function diagram output area.

Configure a "step"

In order to configure a "Step" object, proceed as follows:
1. Open the properties of the placed step.
2. Select the "Attributes > Step parameters" tab.
3. Enter the time.
4. Enter the step type:
   - The first step of a step sequence receives the "Initial step" step type.
   - All remaining steps receive the "Step" step type.

Connection rules

Along with the connection rules for blocks, the following connection rules for a step sequence apply:
- A "step" and a "transition" can only be connected alternately.
- An "action block" must be connected to a "step".
- A "transition" must be connected to a "signal" as a switching condition.
- In the same way as connections, branches can only be placed between "steps" and "transitions".

3.15.3 Numbering steps, transitions or actions

Requirement

A function diagram has been created and is open.
Step sequence elements are placed.
The corresponding objects are hierarchically classified as follows: "FP, SC, AC".

Consecutive numbering of steps, transitions or actions

In order to consecutively number steps, transitions or actions automatically, proceed as follows:
1. Select at least two report objects 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.
### 3.16 Placing and using the comment block

#### Requirement

The current COMOS DB is used.

A function diagram is open and it contains signals or function blocks.

The base object of the comment block is prepared. See chapter Creating a base object for the comment block (Page 75).

#### Selecting a comment block connected to a block

A comment block can be available for a signal or function block. To create a connected comment block, follow these steps:

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

#### Creating a free comment block

To create a free comment block, follow these steps:

1. Using drag&drop, place the “Comment block” object on the function diagram.
   An icon and a text block appear.
2. Edit the free comment block in exactly the same way as a connected comment block.

#### Editing the comment text in a comment block

1. Single-click twice (do not double-click) on the text area of the text block.
   Enter the comment. The Enter key generates an interruption in the text block.
2. Complete your entries by clicking with the mouse outside the text block on the report.

#### 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 in the function diagram.
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 can be found here:
"@03 Structures > FUP > D > 04 > 01 Comment block"

3.17 Interfaces

- Siemens: SIMATIC S7
  You can find additional information on this topic in the "COMOS Automation Interfaces" document, keyword "SIMATIC PCS 7 data interface".
- FD-based code generation
  See chapter "Code generation based on IEC 61131 (Page 51)"
4.1 Overview of VGB-compliant structures

Introduction

VGB function diagrams are mostly edited in the same way as IEC function diagrams.

Requirement

On the "Units" tab, a KKS-compliant unit structure must be created. In COMOS, the VGB logical engineering for KKS-compliant unit structures is supported.

In the KKS structure, a position must be created. The position has the following folder:

- "01 Basic engineering"
- "02 Signal engineering"

VGB master data

- Blocks
  
  Blocks for VGB are stored in the following position under the "Base objects" tab in the navigator:
  
  "@03 Structures > FUP > D > 02 VGB - R 170 C"

- Signals
  
  Signals for VGB are stored in the following position under the "Base objects" tab in the navigator:
  
  "@03 Structures > BAS > KKS > @U > 4 > KKS Device catalog"

  The VGB-compliant name masks are implemented in these signals.

Level-specific function diagrams

In COMOS, function diagrams are stored for the following levels:

- Cell level
- Overview level
- Individual level

In VGB, the function diagrams of the individual levels are only created under the function objects.

VGB function diagrams of the individual levels are stored in the base project in the navigator on the "Documents" tab at the following position:
4.2 Using signal in VGB

Requirement

VGB-compliant unit structures are created.
A VGB-compliant function diagram has been created.
In the "02 Signal engineering" folder, VGB-compliant signals are created.

Placing a signal in the function diagram

To create a signal on the VGB-compliant function diagram, proceed as follows:
1. Drag the signal using drag&drop onto the function diagram.
   The signal is placed in area 3 or 5 of the function diagram but is not yet aligned.
2. Select the symbol of the placed signal and move the symbol into the desired slot.

Connecting function block with signal

Use the connector tool to connect the function block with the signal.

4.3 Display of connections in the VGB function diagrams

Requirement

A VGB-compliant function diagram has been created and is open.
Display of connection lines to signals

In contrast to IEC, the connecting lines from the function blocks to the signals are only indicated in VGB:

Display of cross-page connections

If a block or a signal in the function diagram is connected in a across pages, the connection is graphically interrupted and receives a text reference at both interruption points.

This also applies in the event that two objects were previously connected on the same page and one of the objects was moved to another page.

4.4 Using macro block (auto symbol)

Requirement

VGB-compliant unit structures are created.

A VGB-compliant function diagram has been created and is open.

Placing the macro block

To create a macro block on the VGB-compliant function diagram, proceed as follows:

1. Select the following macro block:
   - Base project, "Units" tab:
     - "Templates > FUP Logical engineering > 02 macro blocks > 201 FuE assembly groups for VGB"

2. Use the macro blocks as described in the IEC chapter.

See Editing and updating macro blocks (Page 27).
Macro block base object in the COMOS DB

"@03 Structures > FUP > D > 02 > 01 > AF > 03 Macro block"
5

Code generation based on IEC 61131

5.1 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

In this case, text-based code templates are generated by means of the placeholder technique.

For the placeholder technique code modules 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

In this case, graphical planning helps to create a system-neutral and language-neutral code on a function diagram. The neutrally generated code is then exported specific to the target system.

The following target systems are supported:

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

5.2 Overview of code objects

5.2.1 Definition of "code classes"

Code classes were defined in order to enable the use of COMOS control code generation in any database structure. The code classes are entered as follows:
1. Open the properties of a block
2. Select the "System" tab
3. "CLASS CO class" attribute

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

The following classes are provided:

- Configuration
- Resource
- Task
- Program Organization Unit (POU)
- Function block
- Variable

5.2.2 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.
5.2.3 Definition of "Configuration"

A configuration according to IEC 61131 consists of at least one resource. The resource contains \(<1\ldots n>\) 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 the call sequence for single modules of a resource. The creation of the contained POUs 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 maps resources, tasks and program organization units. A call, or the according order of the text modules is generated via the connection information of the complete program course, depending on the target system.
5.2.4 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 a resource.

5.2.5 Definition of "Task"

A task is called cyclically and contains the list and order of the to be edited programs that were assigned to the task.

There are two ways to make this assignment:

- Via a link object
- Graphically via the connection information of the objects
5.2.6 **Definition of "Program organization unit" (POU)**

The superior object for a program or text module is characterized as a Program Organization Unit (POU).

Each POU is equivalent to a module that has to be generated in the target system, respectively a text module. This POU can contain several networks, logics/routines and function plans. 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 superior 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

With the help of the POU, delta handling between different code states can be performed.

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 modules or routine process definition.

5.2.7 **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:

- **"ADDRESS"**
  - Hardware address
  - The hardware address is optional.
- **"MA11"**
  - Symbolic address
  - The symbolic address has to be unique throughout the program and project.
- **"MA14"**
  - Data type
5.2 Overview of code objects

- "MA13"
  Comment
  The comment does not have to be set.

- "MA16"
  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 on 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>

5.2.8 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 later be used like direct hardware addresses in the projection process. 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.

5.2.8.1 Basic principle of symbol table generation
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.

To generate a symbol table, right-click on the symbol table object which is assigned to the corresponding control CPU (resource). Then select the "Symbol table > Update symbol table" command in the context menu.
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.

### 5.2.8.2 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.

### 5.2.8.3 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 part unit
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:
5.3 Describing and exporting codes

5.3.1 Preparing code generation for a block

Introduction

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

Procedure

1. Create a new position below the folder "04 I&C (instrumentation & control)".
2. Right-click on the position.
3. Select the "New > POE Program organization unit" command in the context menu.

4. The program organization unit can subsequently be described by a function diagram, a placeholder module or by a combination of a function diagram and a placeholder module.
   - To select a function diagram, click "New > PFFA.001 function diagram IEC" in the program organization unit context menu.
   - To select a placeholder module, click "New > Z1 Auto symbol", for example, in the program organization unit context menu.
5. Open the function diagram you previously created.
6. To interconnect the placeholder module, drag & drop it onto the function diagram. After interconnecting the placeholder module, you can trigger the code generation.

**Note**

The placeholder module can only be interconnected with signals. Pre- or post interconnection of the placeholder module can be carried out on its own function diagram page. The interconnection of a placeholder module is conducted via internal signals.

---

**5.3.2 Exporting the block**

**Requirement**

A POU is created.
Under the POU, a function diagram and a placeholder module are created.
The placeholder module has been placed in the function diagram and has been processed.

**Procedure**

Follow these steps:
1. Right-click on the program organization unit.
2. Select the "Export IEC 61131-3" command in the context menu.

With the procedure described above, only a single module can be described in terms of software. In order to describe a complete control program in terms of software, you have to assign a calling structure, "CFB Caller FB" and a "CPU-assigned PLC" to the new position.

### 5.3.3 Preparing code generation for a program

**Procedure**

In order to describe a complete control program in terms of software, you have to assign a calling structure, "CFB Caller FB" and a "CPU-assigned PLC" to the new position.

Follow these steps:

1. If the "eBlock > SW interface" tab already has a task for "Caller FB" and "Assigned PLC", all you need to do is assign the targets using drag&drop.

2. If the corresponding tasks are not available, you can assign these by right-clicking on the user interface.

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

5. With the help of the connection tool, connect the POU with the existing program organization units.

Having completed this interconnection, the code can be generated for the complete control program.

5.3.4 Exporting the program

Requirement

A configuration and function diagram has been created for configuration purposes.

A POU is created and edited on the configuration function diagram.

The eBlock for the software interface is edited.

Procedure

Having completed this interconnection, the code can be generated for the complete control program.

1. Right-click on "KFG1 configuration".
2. Select the "Export IEC 61131-3" command in the context menu.
5.3 Describing and exporting codes
6.1 Allowed connection type combinations

Interaction with the "@ConnectionPossibleS" standard table

The "Allowed connection type combinations" dialog is an alternative interface for editing the "@ConnectionPossibleS" standard table. All changes in the "Allowed connection type combinations" dialog are applied in the "@ConnectionPossibleS" standard table and vice versa.

The "Allowed connection type combinations" dialog and the "@ConnectionPossibleS" standard table are not permitted to be open at the same time because the interfaces are not updated immediately. This means that it would be possible that the user would see outdated details in one dialog when both are open.

Certain connection type combinations are fixed. For more information on this, see Combinations permitted by data hierarchy (Page 65).

Opening the "Allowed connection type combinations" dialog

1. Change to the system project.
   This is where the "@ConnectionPossibleS" standard table is located.
2. Select the "Administrator > Base data" command in the COMOS menu bar.

Changing the zoom of the table

The table occupies the space available in the working area. If the working area is made larger or smaller, the zoom changes correspondingly.

Defining individual connection type combinations

Click on a white box.

All check boxes on the diagonals are already selected and cannot be cleared.

An exception to this rule is: It is not permissible to connect an input with an input or an output with an output. For this reason, these check boxes are cleared on the diagonals and cannot be selected.

Defining multiple similar connection type combinations

Click on a white box and hold down the mouse button.

Holding the button down, drag the mouse over the white box.
6.2 Checking and adjusting standard tables

6.2.1 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:

- "@10 Attribute catalog > FUP Logical engineering"
  Contains standard tables for the parameterizing of function diagrams and blocks.

- "@10 Attribute catalog > FUP > Y > F004 Step type"
  Distinguishes between the initial step and the ("normal") steps. Used in:
  "@03 Structures > FUP > D > 01 > AA > AC > 01 Step", "Step parameter" tab. Effect: Another symbol is displayed by means of attributes with graphical relevance.

- "@10 Attribute catalog > FUP > Y > A001 > A001001 Definition mark"
  Used in:
  "@03 Structures > FUP > D > 01 > AA > AC > 03 Action block". An element is located underneath. The standard table on the "Layout" tab is used in this element.

- "@10 Attribute catalog > FUP > Y > F001 Layout functional components"
  Used in:
  "@03 Structures > FUP > D > 01 > AA > AD > A Analog monitoring" and in "@03 Structures > FUP > D > 01 > AA > AD > C Controller". Used on the "Layout" tab. Effect: A corresponding subsymbol is activated in the symbol, depending on the setting.

- "@10 Attribute catalog > FUP > Y > F002 > F002002001 Code line selection"
  Used in:
  "@03 Structures > FUP > D > 01 > AA > AD > S Split range". Used on the "SplitRange" tab, "Characteristic line type" attribute. Effect: The symbol is changed, depending on the setting.
6.2 Checking and adjusting standard tables

- "@10 Attribute catalog > FUP > Y > F002 > F002_001 Controller type"
  Used in:
  "@03 Structures > FUP > D > 01 > AA > AD > C Controller": Used on the "Regulator" tab (or Controller tab), "Regulator (controller) type” attribute.
- "IC Instrumentation & control I&C"
  Contains standard tables for parameterizing functions and signals.

Standard tables in the system project

In the system project, the following standard tables are used in function diagrams:
- "@ConnectionTypeF Contact point types (FC)"
  Defines the thickness and type of the connection lines in the step sequence.
- "@ConnectionTypeS 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.
  Located below this is:
  "@ConnectionPossibleS"
  Contains the connection rules for each data type.

6.2.2 Opening standard tables

In order to open a standard table with write access, switch to the base project or system project and select the following command: "Administrator > Base data > Standard tables".

6.2.3 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></td>
<td>ANY_INT</td>
<td>ANY_REAL</td>
<td></td>
</tr>
<tr>
<td>BOOL</td>
<td>INT</td>
<td>UINT</td>
<td>REAL</td>
</tr>
</tbody>
</table>
6.2 Checking and adjusting standard tables

<table>
<thead>
<tr>
<th>ANY</th>
<th>BYTE</th>
<th>SINT</th>
<th>USINT</th>
<th>LREAL</th>
<th>TIME_OF_DAY</th>
<th>STRING</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>DINT</td>
<td>UDINT</td>
<td></td>
<td>DATE_AND_TIME</td>
<td>Derived subtypes (including those with structure/field/enumerator declarations)</td>
<td></td>
</tr>
<tr>
<td>DWORD</td>
<td>LINT</td>
<td>ULINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LWORD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These connection combinations are also permitted when they are not included in the "@ConnectionPossibleS Contact point types (signals)" standard table.

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:

- "@ConnectionPossibleS Contact point types (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".
6.2 Checking and adjusting standard tables

6.2.4 Adjusting standard tables

Creating additional data types

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

- "S02 Data types"
  This standard table is used in the "Data Type" attribute of a signal. You can find this standard table under "IC Catalog I&C > S Signal".
- "@ConnectionTypeS 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 on the engineering side, the subtype of the associated connector will automatically be modified accordingly.

Configuring the connection lines

To configure connecting lines, use the following standard table:

- "@ConnectionTypeS Contact point types (signals)"

The following table shows the parameters of the "@ConnectionTypeS Contact point types (signals)" list:

<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>
6.3 Creating a function diagram template

6.3.1 Overview of report templates

Variables

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>CRp.@0.EF.EFF.EFF.001</td>
<td>IEC function diagram in A3 format.</td>
<td>Toolbar with standard IEC function blocks.</td>
</tr>
<tr>
<td>CRp.@0.EF.EFF.EFF.002</td>
<td>IEC function diagram in A3 format for step sequence configuration.</td>
<td>Toolbar with standard IEC step sequence elements.</td>
</tr>
<tr>
<td>CRp.@0.EF.EFP.EFP.002</td>
<td>Hardware-related signal list.</td>
<td>Evaluating report with a subordinate query.</td>
</tr>
<tr>
<td>CRp.@0.EF.EFP.EFP.003</td>
<td>Complete signal list for hardware and software.</td>
<td>Evaluating report with a subordinate query.</td>
</tr>
<tr>
<td>CRp.@0.EF.EFP.EFP.010</td>
<td>Cross-reference list. Lists additional placements of a signal.</td>
<td>Evaluating report with a subordinate query.</td>
</tr>
<tr>
<td>CRp.@0.EF.EFP.EFP.020</td>
<td>Sequence matrix for presenting sequence chains in tabular format.</td>
<td>Evaluating report with a subordinate query.</td>
</tr>
</tbody>
</table>

6.3.2 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, proceed as follows:

1. Open the function diagram properties.
2. Switch to the "Attributes > Title block" tab.
   
   Here, you define the objects whose values will be used for the metadata.
3. Provide a description of the function diagram.
4. Click the "OK" to save the metadata.

You can find additional information on this topic in the "COMOS Platform Administration" manual, keyword "Interactive reports".
6.3 Creating a function diagram template

6.3.3 Controlling the toolbar

"System data" tab

Blocks are available in the toolbar of the function diagram. To adjust the blocks offered, proceed as follows:

1. Open the properties of the interactive report.
2. Switch to the "Attributes > System data" tab.
3. Set the required objects in the groups "Step sequence", "Standard functions" and "Standard function blocks".

6.3.4 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 \text{height of entry} &lt; \text{height of report})</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 \text{height of entry} &lt; \text{height of report})</td>
</tr>
</tbody>
</table>
6.3 Creating a function diagram template

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SignalWidth</td>
<td>Specifies the width of a signal in the report.</td>
<td>The variable value is only evaluated for symbols without a graphical display.</td>
</tr>
<tr>
<td></td>
<td>Unit: Millimeters</td>
<td>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.</td>
<td>The variable value is only evaluated for symbols without a graphical display.</td>
</tr>
<tr>
<td></td>
<td>Unit: Millimeters</td>
<td>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.</td>
<td>The variable value is only evaluated for symbols without a graphical display.</td>
</tr>
<tr>
<td></td>
<td>Unit: Millimeters</td>
<td>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

![Diagram of SignalWidth and SignalWidth1](image)

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

Variables

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 underneath the document (unlike with ET reports). &quot;FUP&quot;-type documents always have multiple pages.</td>
</tr>
<tr>
<td>Symboltype</td>
<td>Specifies the standard you use for configuration in the function diagram.</td>
<td>Assign either the &quot;FUP_IEC&quot; or the &quot;FUP_VGB&quot; value to this variable.</td>
</tr>
</tbody>
</table>
### 6.4 Adjusting block base objects

#### 6.4.1 Overview of block base objects

**Introduction**

The base objects for blocks and templates are stored in the following position on the "Base objects" tab of the Navigator:

- "@03 Structures > FUP Logical engineering"

**Block families**

You can find the base objects for blocks for the individual templates under "D Catalog FuE".

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 connections: The user can determine the number of connectors in a variable way.

Examples:

- "@03 Structures > FUP > D > 01 > AA > AA > 400 > 401 AND-gate"
- "@03 Structures > FUP > D > 01 > AA > AA > 400 > 402 OR gate"
- "@03 Structures > FUP > D > 01 > AA > AA > 500 > 505 MUX Multiplexer"
- "@03 Structures > FUP > D > 01 > AA > AC > 04 Simultaneous branch"
- "@03 Structures > FUP > D > 01 > AA > AC > 06 Alternative branching"

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".

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGBConnectionLength</td>
<td>Controls the length of the horizontal cross line in a VGB connection line.</td>
<td>In VGB-compliant function diagrams, not all of the connection lines are continuous. Instead, as a rule only the vertical part of the connection is continuous, while the horizontal part is signified by a cross line.</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 on 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>
Templates for function diagrams

You can find the base objects for function diagram templates under "130 FuE assembly groups for IEC1131-3". Templates for standardized measurement and control plans are stored under this position.

See also

Placing a template in the function diagram (Page 14)

6.4.2 Creating a user-defined block

Introduction

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

- For a user-defined block, create a new base object node as well as a base object.
- In the base object, define a separate graphical display for each function diagram type.

You can find additional information on this topic in the "Basic Operation" manual, keyword "Symbol Designer".

6.4.3 Specifications for a connector interlock

Classification

The blocks have the functional classification "AV". You can find more information on this topic in the "Queries" manual, keyword "Update Classification".

Attributes at the base object

- "SYS0005 connector interlock field width" attribute
  Specifies the width of the text field in grid points.
  "@03 Structures > FUP > D > 01 > AA > AD > 100 > SYS > SYS0005 connector interlock field width"

Extended attribute at the connector

- "Locking"
  Controls the interlock of the connector.

See also

Connector interlock due to constant signal value (Page 22)
6.4 Adjusting block base objects

6.4.4 Creating user-defined macro blocks

Procedure
To create a user-defined block with the functional scope of a macro block, proceed as follows:

1. On the "System settings" tab of the corresponding base object, the "MF" classification must be set in the "Functional classification" list.
2. The "Attributes > System" tab must contain the following properties:
   - "BLK0001 Connector distance"
   - "MB001 Macro block"
3. To automate the calculation of function diagram symbols, you need to store the following symbol script on the "Symbols" tab:
   ```
   CreateObject("ComosGeoSymbols.MacroBlock").CreateScript Me, PARAM, "FUP_IEC", "???"
   ```

6.4.5 Creating user-defined dynamic blocks

Procedure
To create a user-defined dynamic block, proceed as follows:

1. On the "System settings" tab of the corresponding base object, the "DF" classification must be set in the "Functional classification" list.
2. The "Attributes > System" tab must contain the following property:
   - "BLK0001 Connector distance"
3. If you want the function diagram symbols to be automatically calculated, you must place the following symbol script on the "Symbols" tab:
   ```
   CreateObject("ComosGeoSymbols.MacroBlock").CreateScript Me, PARAM, "FUP_IEC", "???"
   ```

6.4.6 Creating a template for a macro block

Introduction
Both in the base project and in the planning project, function diagram templates can be created for macro blocks in the "@Template" branch.
General rules

The following rules must be regarded:

- A function diagram is created underneath the assembly element. This can be edited like any other standard function diagram.
- All objects placed on the function diagram should be arranged underneath the document.
- The use of any function block is permitted.
- The signals used should be assigned as “signal type” software input or software output.
- 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

To set the attributes of a macro block, proceed as follows:

1. Create a base object of the macro block type.
2. Open the macro block properties.
3. Select the "Attributes > System" tab.
4. To assign a template to the detail drawing, click the "..." button next to the "Macro block" field.
5. To graphically display the symbol, select the "single" or "double" grid from the "Connector distance" list.
6. Click "OK" to apply the settings.

Creating a new template object

1. Switch to the "@Template templates" branch.
2. Select "New > General > New object" in the context menu.
3. Select the "General" tab in the properties of the new object.
4. Set the macro block base object in the "Base object" field.

Designing detail drawing (creating internal function diagram)

2. Select the "Report" tab in the properties of the new document.
4. Edit the function diagram.
6.4.7 Creating a base object for the comment block

Creating the base object for the comment block

The base object can be found here:
"03 Structures > FUP > D > 04 > 01 Comment block"

A comment block has the functional classification "KB".

Adding a comment block to a block

To provide a signal or a function block with an optional comment block, proceed as follows:
1. Open the properties of the block and select the "Elements" tab.
2. In the Navigator, select the base object of the comment block.
3. Drag the base object of the comment block to the "Elements" tab.

6.5 Reference: Script for VGB signals

Script in the signal

The script from the OnChange event of the "FUP01.FUP0001" attribute of the VGB signal is transferred into the ChangeSignalName function in the ScriptLib.

The ChangeSignalName function checks a change of signal name and changes the CDevice Pointer.

6.6 Generating code templates

6.6.1 Reference: Base objects for code generation

Requirement

The following nodes are required in the base object of the base project:
- "@03 Structures > FUP Logical engineering > CO Code generating".

This node is already part of the ComosDB and contains all base objects that are needed for the control code generation.
6.6 Generating code templates

The following nodes are also needed:

- "@03 Structures > FUP Logical engineering > D Catalog FuE > 01 IEC1131-3 > AA IEC1131-3 > XX Organizational units > KFG Configuration"

  Contains the structure for describing configurations according to IEC 61131.

- "@03 Structures > FUP Logical engineering > D Catalog FuE > 01 IEC1131-3 > AA IEC1131-3 > XX Organizational units > POE 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 object, plus various placeholder templates. Via the templates you can create code modules to generate control code.

6.6.2 Creating text templates for a placeholder text

For each placeholder template you can create text templates for a placeholder text.

**Procedure**

To create a text template, proceed as follows:

1. Right-click on the placeholder template.
2. Select the "New > New document" command in the context menu.
3. On the "General" tab in the "Type" list, select the "NetTemplateEditor" document type.

4. Enter "FPC010 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 "Generating code templates (Page 75).

<table>
<thead>
<tr>
<th>General</th>
<th>Attributes</th>
<th>NetTemplateEditor</th>
<th>Revisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source code</td>
<td>System information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target system</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware:</td>
<td>CoDeSys</td>
<td>Version:</td>
<td>v01</td>
</tr>
</tbody>
</table>

```
("... CDS AML ...")
LD (CT:RUN)
ANDN ((CT):Close_FULL)
OR (CT:Close_EMPTY)
ST (CO:OK)
LDN (CO:FULL)
AND (CH:EMPTY)
ST (CO:PUMPE)
```

**Note**

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

### 6.6.3 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 it is necessary, for example, to place the resulting symbols in quotation marks, these must be set outside the brackets.

**Example**

```
{FGDB} -> Placeholder for the local FGDB symbol.

{B1MV} -> Placeholder for the local B1MV symbol.
```

Next to the references on a symbol entry you can also set references to attributes in the structure.

The syntax is as follows:

```
{[Marker]: [Reference]}
```

The reference is specified as follows:
Tab name.Attribute 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 Reference to the attribute "SYS008" of the variables group

This reference can also access the "SYS008" attribute of a referenced variables group by means of the specification of a software interface.

\{A: HE.FGName\}

Markers can also be created for symbols for in- and output connectors on a function plan. This functionality is only used for the FD-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 connector</td>
</tr>
<tr>
<td>&quot;CO&quot;</td>
<td>Output connector</td>
</tr>
</tbody>
</table>

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

**Example**

\{CI: Input1\}
\{CO: Output1\}

**6.6.4 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\} \rightarrow Reference to a signal of an assigned fill-level measurement.

Following illustration displays an excerpt from a module template with the described syntax.
6.7 Integration of variants/control structures

Introduction

To minimize the amount and therefore maintenance load of network templates there is an option that helps to easily integrate variants into 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 key words is not case-sensitive.

6.7.1 Control structure #Exist-#Default-#EndExist

Syntax

```c
//#Exist([Marker:]Condition)
.
.
.
///#Default]
.
.
.
//#EndExist
```

The command below the code is not commented out if the #Exist command returns a true result. If the result is false the command is commented out.

The code underneath the optional #Default command is always used if the #Exist condition is false.
The #EndExist command completes the Exist block. Following markers can be used:

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

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

If no marker is used it is checked if there is a symbol in the symbol table.

### 6.7.2 Control structure #If-#Else-#EndIf

**Syntax**

```
#if([Marker:]Operand1 [Operator] [[Marker:]Operand2])
  .
  .
  .
  //#Else
  .
  .
  //#Endif
```

The #If-#Else-#EndIf control structure is used if certain program parts should only be used under certain conditions. The to be evaluated condition must always return a Boolean value. If the result of the evaluation is true, the code below the #If command is not commented out. The code below the optional #Else command is always used if the #If condition is false.

The #EndIf command completes the control structure.

Following operators are currently supported:

<table>
<thead>
<tr>
<th>Comparison operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a == b</code></td>
<td>The result of the operation is true if <code>a</code> equals <code>b</code>.</td>
</tr>
<tr>
<td><code>a != b</code></td>
<td>The result of the operation is true if <code>a</code> does not equal <code>b</code>.</td>
</tr>
<tr>
<td><code>a &gt; b</code></td>
<td>The result of the operation is true if <code>a</code> is greater than <code>b</code> (numeric data types).</td>
</tr>
<tr>
<td><code>a &lt; b</code></td>
<td>The result of the operation is true if <code>a</code> is smaller than <code>b</code> (numeric data types).</td>
</tr>
</tbody>
</table>
6.7 Integration of variants/control structures

<table>
<thead>
<tr>
<th>a &gt;= b</th>
<th>The result of the operation is true if a is greater than or equal to b (numeric data types).</th>
</tr>
</thead>
<tbody>
<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 still perform a check regarding the values "true" or "1".

**Example**

```c
//If(1)
  .
//#Else
  .
//#Endif
```

This can e.g. be used to comment out complete code passages.

Following markers can be used:

- Marker for attribute access:

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

### 6.7.3 Control structure #ForEach-#Next

**Syntax with operand**

```c
//#ForEach(Operand)
...
//#Next
```

The #ForEach-#Next control structure is used, in dependency of an operator, to insert a number of equal program parts. For each element of the operator a program part with the according count setting is added.

```c
//#ForEach ({B1TK*})
...
U     "{STA.B1TK#}";
=     "(FGDB_ZI_B1TK#)";  -> Refers to an assembly part control in the data block

//#Next
```

**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 rule for the creation can be described as follows:

```c
//#ForEach (B:SB*)
//#include(SF\NT17)
```
According to the rule:

For each back pointer of an SB* task

Go to the source in folder SF and integrate NT17

### 6.7.4 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 operator is copied completely to the position of the command. This case is needed to implement complete text blocks.

The file path has to be indicated at the start of the function 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 command.