Copyright

The data in this document may not be altered or amended without special notification from ETAS GmbH. ETAS GmbH undertakes no further obligation in relation to this document. The software described in it can only be used if the customer is in possession of a general license agreement or single license. Using and copying is only allowed in concurrence with the specifications stipulated in the contract.

Under no circumstances may any part of this document be copied, reproduced, transmitted, stored in a retrieval system or translated into another language without the express written permission of ETAS GmbH.

© Copyright 2013 ETAS GmbH, Stuttgart

The names and designations used in this document are trademarks or brands belonging to the respective owners.

Document EC010015 V6.2 R01 EN - 05.2013
Contents

1 Introduction ......................................................... 5
  1.1 Safety Advice .................................................. 5
    1.1.1 Correct Use .............................................. 5
    1.1.2 Labeling of Safety Instructions ......................... 5
    1.1.3 Demands on the Technical State of the Product ....... 5
  1.2 Manual Structure ............................................. 6
  1.3 Typographic Conventions ................................. 6

2 Operators in Block Diagrams .................................. 7
  2.1 Arithmetic Operators ....................................... 7
  2.2 Comparison Operators ..................................... 8
  2.3 Logical Operators .......................................... 9
  2.4 Multiplex Operator ....................................... 9
  2.5 Case Operator ............................................. 9
  2.6 Miscellaneous Operators ................................. 10

3 Control Flow Elements in Block Diagrams ..................... 13
  3.1 If...Then ................................................... 13
  3.2 If...Then...Else .......................................... 13
  3.3 Switch ..................................................... 14
  3.4 While ....................................................... 14
  3.5 Break ....................................................... 15

4 Elements in Block Diagrams ................................... 17
  4.1 Scalar Elements ........................................... 17
    4.1.1 Variables ............................................... 17
    4.1.2 Parameters ........................................... 18
    4.1.3 Real-Time Elements ................................. 18
      Messages ................................................... 18
      Resources .................................................. 19
1 Introduction

ASCET provides an innovative solution for the functional and software development of modern embedded software systems. ASCET supports every step of the development process with a new approach to modelling, code generation and simulation, thus making higher quality, shorter innovation cycles and cost reductions a reality.

This manual lists the icons and symbols that appear in classes and modules specified as ASCET block diagrams.

1.1 Safety Advice

Please adhere to the Product Liability Disclaimer (ETAS Safety Advice) and to the following safety instructions to avoid injury to yourself and others as well as damage to the device.

1.1.1 Correct Use

ETAS GmbH cannot be made liable for damage which is caused by incorrect use and not adhering to the safety instructions.

1.1.2 Labeling of Safety Instructions

The safety instructions contained in this manual are shown with the standard danger symbol shown below:

![Danger Symbol]

The following safety instructions are used. They provide extremely important information. Read this information carefully.

**WARNING!**

Indicates a possible medium-risk danger which could lead to serious or even fatal injuries if not avoided.

**CAUTION!**

Indicates a low-risk danger which could result in minor or less serious injury or damage if not avoided.

**NOTICE**

Indicates behavior which could result in damage to property.

1.1.3 Demands on the Technical State of the Product

The following special requirements are made to ensure safe operation:

- Take all information on environmental conditions into consideration before setup and operation (see the documentation of your computer, hardware, etc.).
Further safety advice is given in the ASCET V6.2 safety manual (ASCET Safety Manual.pdf) available on your installation disk, in the ETASManuals\ASCET V6.2 folder on your computer or in the download center of the ETAS web site.

1.2 Manual Structure

The ASCET “Block Diagram Icon Reference” manual contains the following chapters:

- "Introduction" (this chapter)
  This chapter contains general information such as documentation conventions.
- "Operators in Block Diagrams"
  This chapter lists all block diagram operators and their icons.
- "Control Flow Elements in Block Diagrams"
  This chapter lists all control flow elements and their icons.
- "Elements in Block Diagrams"
  This chapter lists all block diagram elements and their icons.
- "Miscellaneous Icons"
  This chapter lists the remaining icons and their meaning.

1.3 Typographic Conventions

The following typographic conventions are used in this manual:

Select **File → Open**. Menu commands are shown in **blue boldface**.

Click **OK**. Names of buttons and options are shown in **blue boldface**.

Press **<ENTER>**. Keyboard commands are shown in angled brackets and **CAPITALS**.

The “Open File” dialog window opens. Names of program windows, dialog windows, fields, etc. are shown in quotation marks.

Select the file **setup.exe**. Text in drop-down lists on the screen, program code, as well as path- and file names are shown in the **Courier** font.

A **distribution** is always a one-dimensional table of sample points.

The OSEK group (see [http://www.osekvdx.org/](http://www.osekvdx.org/)) has developed certain standards.

General emphasis and new terms are set in **italics**.

Links to internet documents are set in **blue, underlined** font.

Important notes for the users are presented as follows:

**Note**

**Important note for users.**
2 Operators in Block Diagrams

This chapter lists the operators available in ASCET block diagrams.

- arithmetic operators (section 2.1 on page 7)
- comparison operators (section 2.2 on page 8)
- logical operators (section 2.3 on page 9)
- multiplex operator (section 2.4 on page 9)
- case operator (section 2.5 on page 9)
- miscellaneous operators (section 2.6 on page 10)

2.1 Arithmetic Operators

The meaning of the operators is the same as in ESDL. The following operators are available:

- Addition, Subtraction, Multiplication, Division, Modulo

The addition and multiplication operators can have 2 to 20 arguments. The subtraction, division and modulo operators have only two arguments.

<table>
<thead>
<tr>
<th>name</th>
<th>palette / toolbar / &quot;Navigation&quot; tab icon</th>
<th>block diagram icon</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td></td>
<td></td>
<td>Returns the sum of the inputs.</td>
</tr>
<tr>
<td>Subtraction</td>
<td></td>
<td></td>
<td>Returns the difference between the upper and the lower input.</td>
</tr>
<tr>
<td>Multiplication</td>
<td></td>
<td></td>
<td>Returns the product of the inputs.</td>
</tr>
<tr>
<td>Division</td>
<td></td>
<td></td>
<td>Returns the quotient of the upper and the lower input.</td>
</tr>
<tr>
<td>Modulo</td>
<td></td>
<td></td>
<td>Returns the remainder of the division upper / lower input.</td>
</tr>
</tbody>
</table>
2.2 Comparison Operators

The comparison operators are identical to their counterparts in ESDL. The following comparison operators are available:

- Greater, Smaller, Smaller or Equal, Greater or Equal, Equal, Not Equal

Each operator has 2 arguments. The Equal and Not Equal operators can be applied to arithmetic and non-arithmetic elements.

<table>
<thead>
<tr>
<th>name</th>
<th>palette / toolbar / &quot;Navigation&quot; tab icon</th>
<th>block diagram icon</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greater</td>
<td>🔄</td>
<td>⬅️</td>
<td>true if the upper input is greater than the lower input. Returns false otherwise.</td>
</tr>
<tr>
<td>Smaller</td>
<td>⬅️</td>
<td>🔄</td>
<td>true if the upper input is smaller than the lower input. Returns false otherwise.</td>
</tr>
<tr>
<td>Smaller or Equal</td>
<td>⬅️</td>
<td>🔄</td>
<td>true if the upper input is smaller than or equal to the lower input. Returns false otherwise.</td>
</tr>
<tr>
<td>Greater or Equal</td>
<td>🔄</td>
<td>⬅️</td>
<td>true if the upper input is greater than or equal to the lower input. Returns false otherwise.</td>
</tr>
<tr>
<td>Equal</td>
<td>⬅️</td>
<td>🔄</td>
<td>true if the upper input is equal to the lower input. Returns false otherwise.</td>
</tr>
<tr>
<td>Not Equal</td>
<td>🔄</td>
<td>⬅️</td>
<td>true if the upper input is not equal to the lower input. Returns false otherwise.</td>
</tr>
</tbody>
</table>
2.3 Logical Operators

The logical operators are identical to their counterparts in ESDL. The following logical operators are available:

- And, Or, Not

The And and Or operators can have 2 to 20 arguments, the Not operator has one argument.

<table>
<thead>
<tr>
<th>name</th>
<th>palette / toolbar / &quot;Navigation&quot; tab icon</th>
<th>block diagram icon</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>And</td>
<td><img src="image" alt="And Icon" /></td>
<td><img src="image" alt="And Block Diagram" /></td>
<td>Returns true if all inputs are true. Returns false if at least one input is false.</td>
</tr>
<tr>
<td>Or</td>
<td><img src="image" alt="Or Icon" /></td>
<td><img src="image" alt="Or Block Diagram" /></td>
<td>Returns true if at least one input is true. Returns false if all inputs are false.</td>
</tr>
<tr>
<td>Not</td>
<td><img src="image" alt="Not Icon" /></td>
<td><img src="image" alt="Not Block Diagram" /></td>
<td>Returns true(false) if the input is false(true).</td>
</tr>
</tbody>
</table>

2.4 Multiplex Operator

The conditional operator ( ? : ) is named Multiplex operator (for short: Mux) in the graphical representation.

The multiplex operator can be used with 2 to 20 arguments, however, the identical functionality of a multi-argument Mux operator can be built as a cascade of several 2-argument Mux operators.

<table>
<thead>
<tr>
<th>name</th>
<th>palette / toolbar / &quot;Navigation&quot; tab icon</th>
<th>block diagram icon</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mux</td>
<td><img src="image" alt="Mux Icon" /></td>
<td><img src="image" alt="Mux Block Diagram" /></td>
<td>The input on the top is the condition, the inputs on the left are the values. The Mux operator returns the upper (lower) left input if the condition is false (true).</td>
</tr>
</tbody>
</table>

2.5 Case Operator

The Case operator is a special case of the conditional operator. It does not take a logical value, but a switch value of discrete type. The Case operator has \( n \) arguments (\( n = 2 \ldots 20 \)), \( n - 1 \) of which are numbered consecutively. The last argument is the default case.
Depending on the switch value, one of the arguments is selected. If the switch value is 1, the first argument is returned, if it is 2 the second is returned, and so on. If the switch value is less than 1, or \( n \), or larger than \( n \), the last argument is returned.

### 2.6 Miscellaneous Operators

**Max and Min Operators:** The Max and Min operators can have 2 to 20 arguments; they can be applied only to arithmetic elements.

<table>
<thead>
<tr>
<th>Name</th>
<th>Palette / Toolbar / Block Diagram</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case</td>
<td>palette / toolbar / &quot;Navigation&quot; tab icon</td>
<td>The input on the top is the switch value, the inputs on the left are the arguments.</td>
</tr>
</tbody>
</table>

**Max**

Returns the largest input value.

**Min**

Returns the smallest input value.

**Between Operator:** The *Between* operator checks if the argument value (upper input) lies between the limiters min (middle input) and max (lower input). If this is the case, the logical return value is true, otherwise it is set to false.

<table>
<thead>
<tr>
<th>Name</th>
<th>Palette / Toolbar / Block Diagram</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between</td>
<td>palette / toolbar / &quot;Navigation&quot; tab icon</td>
<td>Returns true(false) if the argument lies (does not lie) between min and max.</td>
</tr>
</tbody>
</table>
**Absolute Operator:** Argument and return value of the *Absolute* operator have to be both either cont or discrete.

<table>
<thead>
<tr>
<th>name</th>
<th>palette / toolbar / &quot;Navigation&quot; tab icon</th>
<th>block diagram icon</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute</td>
<td><img src="image" alt="Absolute Icon" /></td>
<td><img src="image" alt="Absolute Diagram" /></td>
<td>Returns the absolute value of the argument.</td>
</tr>
</tbody>
</table>

**Negation Operator:** Argument and return value of the *Negation* operator can be cont or discrete; if the argument is cont, the type of the return value should be the same.

<table>
<thead>
<tr>
<th>name</th>
<th>palette / toolbar / &quot;Navigation&quot; tab icon</th>
<th>block diagram icon</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation</td>
<td><img src="image" alt="Negation Icon" /></td>
<td><img src="image" alt="Negation Diagram" /></td>
<td>Returns the negative value of the argument.</td>
</tr>
</tbody>
</table>
# Control Flow Elements in Block Diagrams

This chapter lists the control flow elements available in ASCET block diagrams. The following control flow statements are available in block diagrams:

- If...Then (section 3.1 on page 13)
- If...Then...Else (section 3.2 on page 13)
- Switch (section 3.3 on page 14)
- While (section 3.4 on page 14)
- Break Statement (section 3.5 on page 15)

All control flow statements except Break evaluate a logical expression and, depending on the result, activate a control flow branch which may contain several statements. The statements represented by sequence calls are connected to the control flow by connectors.

The Break statement can be used to exit immediately from each of the other control flow elements and return to another enclosing statement or to the remainder of the model.

## 3.1 If...Then

The **If...Then** statement evaluates a logical expression. The control flow output is connected to one or more sequence calls which are triggered whenever the control flow branch is activated. Whenever the input expression evaluates to **true**, the connected sequence calls are executed.

<table>
<thead>
<tr>
<th>Name</th>
<th>palette / toolbar / block diagram</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>If...Then</td>
<td><img src="icon.png" alt="Icon" /></td>
<td>Activates a control flow branch (connected to the output) if the input is <strong>true</strong>.</td>
</tr>
</tbody>
</table>

## 3.2 If...Then...Else

**If...Then...Else** is similar to If...Then, but has two control flow branches. Depending on the value of the logical expression, one of the branches is executed.

<table>
<thead>
<tr>
<th>Name</th>
<th>palette / toolbar / block diagram</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>If...Then...Else</td>
<td><img src="icon.png" alt="Icon" /></td>
<td>Activates the control flow branch on the right (connected to the output) if the input is <strong>true</strong>. Activates the control flow branch at the bottom if the input is <strong>false</strong>.</td>
</tr>
</tbody>
</table>
3.3 Switch

The Switch construct is similar to the Case operator (cf. section 2.5 on page 9). A Switch evaluates a signed discrete or unsigned discrete value and, depending on that value, activates different control flow branches. These branches are separated from each other, so that a “fall through” like in the switch construct in C is not possible.

A switch can have 2 to 20 branches.

For each alternative, the value for the branch can be defined by the user. The last branch at the bottom is the default branch.

<table>
<thead>
<tr>
<th>Name</th>
<th>palette / toolbar / “Navigation” tab icon</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch</td>
<td>![ Switch Icon ]</td>
<td>Activates the control flow branch whose value is the same as the input value. Activates the default branch (at the bottom) if no branch value equals the input.</td>
</tr>
</tbody>
</table>

3.4 While

The While loop is the only loop construct available in block diagrams. Care has to be taken to avoid infinite loops or loops unsuitable for real-time applications.

<table>
<thead>
<tr>
<th>Name</th>
<th>palette / toolbar / “Navigation” tab icon</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>While</td>
<td>![ While Icon ]</td>
<td>Activates the control flow branch if the input value is true. The operation is executed as long as the input value remains true; the input value should be manipulated in the while loop.</td>
</tr>
</tbody>
</table>
3.5 Break

The `break` operator in the block diagram editor behaves similar to a C language return statement.

**Note**

The `break` operator in the block diagram editor behaves differently from the `break` statement in ESDL.

<table>
<thead>
<tr>
<th>Name</th>
<th>palette / toolbar / &quot;Navigation&quot; tab icon</th>
<th>block diagram icon</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| Break | ![Break Icon](image) | ![Break Icon](image) | In **method**: Causes an immediate return from the method. The user is responsible for the correct setting of any return values before break is executed.  
In **process**: Causes a deferred exit (i.e. all send messages are sent before the exit occurs). |
4 Elements in Block Diagrams

This chapter lists the elements available in ASCET block diagrams.

- section 4.1 "Scalar Elements" on page 17
- section 4.2 "Composite Elements" on page 21
- section 4.3 "Complex Elements (Included Components)" on page 26
- section 4.4 "Signature Elements" on page 26
- section 4.5 "Miscellaneous Elements" on page 29

4.1 Scalar Elements

Several scalar elements are available in ASCET block diagrams:

- variables (section 4.1.1 on page 17)
- parameters (section 4.1.2 on page 18)
- real-time elements (section 4.1.3 on page 18)
- literals (section 4.1.4 on page 20)
- constants and system constants (section 4.1.5 on page 20)

4.1.1 Variables

Five types of scalar variables are available in ASCET block diagrams:

- Logic, Signed discrete, Unsigned discrete, Continuous, Enumeration

The "Elements" palette provides a button for each variable, the "Elements" toolbar provides a single button with a sub-menu for the type of the numeric variable.

In the block diagram, the type of a variable is not visible, only kind (variable, virtual variable) and scope (exported/imported/local) and some properties (virtual/nonvirtual, volatile/nonvolatile); see chapter 5 on page 33 for details.

The type of a variable is shown in the "Tree" pane.

<table>
<thead>
<tr>
<th>name</th>
<th>palette icon</th>
<th>block diagram icon</th>
<th>Tree pane icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Variable</td>
<td><img src="logic.png" alt="Logic Icon" /></td>
<td><img src="logic.png" alt="Logic Icon" /></td>
<td><img src="logic.png" alt="Logic Icon" /></td>
</tr>
<tr>
<td>Signed Discrete Variable</td>
<td><img src="" alt="Signed Discrete Icon" /></td>
<td><img src="" alt="Signed Discrete Icon" /></td>
<td><img src="" alt="Signed Discrete Icon" /></td>
</tr>
<tr>
<td>Unsigned Discrete Variable</td>
<td><img src="" alt="Unsigned Discrete Icon" /></td>
<td><img src="" alt="Unsigned Discrete Icon" /></td>
<td><img src="" alt="Unsigned Discrete Icon" /></td>
</tr>
<tr>
<td>Continuous Variable</td>
<td><img src="cont.png" alt="Continuous Icon" /></td>
<td><img src="cont.png" alt="Continuous Icon" /></td>
<td><img src="cont.png" alt="Continuous Icon" /></td>
</tr>
<tr>
<td>Enumeration Variable</td>
<td><img src="enum.png" alt="Enumeration Icon" /></td>
<td><img src="enum.png" alt="Enumeration Icon" /></td>
<td><img src="enum.png" alt="Enumeration Icon" /></td>
</tr>
</tbody>
</table>

a. The patterns of the red squares indicate the scope (cf. section 5.1) and some properties (cf. section 5.2) of the variables.
4.1.2 Parameters

Five types of scalar parameters are available in ASCET block diagrams:

- Logic, Signed discrete, Unsigned discrete, Continuous, Enumeration

The "Elements" palette provides a button for each parameter, the "Elements" toolbar provides a single button with a sub-menu for the type of the numeric parameter.

In the block diagram, the type of a parameter is not visible, only kind and scope (exported/imported/local) and some properties (e.g., normal/dependent); see chapter 5 on page 33 for details.

The type of a parameter is shown in the "Tree" pane.

<table>
<thead>
<tr>
<th>name</th>
<th>palette icon</th>
<th>block diagram icon</th>
<th>Tree pane icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Parameter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signed Discrete Parameter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unsigned Discrete Parameter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continuous Parameter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enumeration Parameter</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a. The patterns of the blue circles indicate the scope (cf. section 5.1) and some properties (cf. section 5.2) of the parameter.

4.1.3 Real-Time Elements

Messages

Messages form the input and output variables of processes and are used for interprocess communication. Three types of messages are available in ASCET block diagrams:

- Receive messages, Send messages, Send & Receive messages

Messages can be of the same type as variables and parameters, i.e, logic, signed/unsigned discrete, continuous, enumeration. As for variables and parameters, the message type is not visible in the diagram. In the "Tree" pane, no special icon indicates the message type, it is visible only textually in the "Outline" tab.
All messages can be of scope exported or imported, only send & receive messages can be of scope local (cf. section 5.1 on page 33).

<table>
<thead>
<tr>
<th>name</th>
<th>palette / toolbar icon</th>
<th>block diagram icon</th>
<th>Tree pane icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive Message</td>
<td></td>
<td><img src="image" alt="msg_receive_i" /></td>
<td>![msg_r_e::msg[cont]]</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="image" alt="msg_receive_e" /></td>
<td>![msg_r_l::msg[cont]]</td>
</tr>
<tr>
<td>Send Message</td>
<td></td>
<td><img src="image" alt="msg_send_i" /></td>
<td>![msg_s_e::msg[cont]]</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="image" alt="msg_send_e" /></td>
<td>![msg_s_l::msg[disc]]</td>
</tr>
<tr>
<td>Send &amp; Receive Message</td>
<td></td>
<td><img src="image" alt="msg_sendreceive_i" /></td>
<td>![msg_sr_e::msg[cont]]</td>
</tr>
<tr>
<td></td>
<td></td>
<td><img src="image" alt="msg_sendreceive_e" /></td>
<td>![msg_sr_l::msg[udisc]]</td>
</tr>
</tbody>
</table>

**Resources**

A resource represents a part of an application that can only be used exclusively. In order to access a resource, there are two methods:

- `void reserve()`: the resource is reserved, i.e. access to it is blocked.
- `void release()`: the resource is released, i.e. access to it is granted again.
**dT Parameter**

In control engineering applications the result of the calculations within a component often depends on the value of the sampling rate. ASCET provides the system parameter $dT$ for uniformly describing the algorithms for all sampling rates. The value of this parameter is provided by the operating system and represents the time difference since the last activation of the currently active task.

In a block diagram component, $dT$ is of scope local. In a project, $dT$ is of scope exported.

<table>
<thead>
<tr>
<th>name</th>
<th>palette / toolbar icon</th>
<th>block diagram icon</th>
<th>Tree pane icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>$dT$</td>
<td><img src="image" alt="Icon" /></td>
<td><img src="image" alt="Icon" /></td>
<td>$dT::dT$</td>
</tr>
</tbody>
</table>

$dT$ (in block diagram)

$dT::dT$ (in project)

### 4.1.4 Literals

Literals are strings that represent a fixed value of a basic scalar type which can be used in any expression. The value of a literal is either a number (discrete or continuous), a character string, or one of the values `true` or `false` (logical).

Five literals are predefined in ASCET block diagrams:

- Enumeration literal, Logic literal `true`, Logic literal `false`, Continuous literal `0.0`, Continuous literal `1.0`

| name               | palette / toolbar icon | block diagram icon | "Navigation" tab icon
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration Literal</td>
<td><img src="image" alt="Icon" /></td>
<td><img src="image" alt="Icon" /></td>
<td><code>string</code></td>
</tr>
<tr>
<td>Logic Literal <code>true</code></td>
<td><img src="image" alt="Icon" /></td>
<td><img src="image" alt="Icon" /></td>
<td><code>true</code></td>
</tr>
<tr>
<td>Logic Literal <code>false</code></td>
<td><img src="image" alt="Icon" /></td>
<td><img src="image" alt="Icon" /></td>
<td><code>false</code></td>
</tr>
<tr>
<td>Continuous Literal <code>0.0</code></td>
<td><img src="image" alt="Icon" /></td>
<td><img src="image" alt="Icon" /></td>
<td><code>0.0</code></td>
</tr>
<tr>
<td>Continuous Literal <code>1.0</code></td>
<td><img src="image" alt="Icon" /></td>
<td><img src="image" alt="Icon" /></td>
<td><code>1.0</code></td>
</tr>
</tbody>
</table>

a. Literals are not shown in the "Outline" tab.

### 4.1.5 Constants and System Constants

Constants and system constants cannot be created via palette or toolbar buttons. They are created with the parameter or variable buttons (cf. section 4.1.1 and section 4.1.2) and the appropriate selection (i.e. `Constant` or `System Constant`) in the "Kind" combo box of the properties editor.

Constants and system constants can be of scope exported, imported or local; they can be of the types logic, signed/unsigned discrete, continuous, enumeration (see also chapter 5 on page 33).
The representation of constants and system constants in the “Tree” pane is similar to that of parameters (section 4.1.2), except for the overlay icon ( = or #).

<table>
<thead>
<tr>
<th>Name</th>
<th>Palette / Toolbar Icon</th>
<th>Block Diagram Icona</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>n/a</td>
<td><img src="https://example.com/icon.png" alt="Icon" /></td>
<td>Constants are created as a define statement in the generated C code. However, they are not necessarily explicitly visible in the generated code.</td>
</tr>
<tr>
<td>System Constant</td>
<td>n/a</td>
<td><img src="https://example.com/icon.png" alt="Icon" /></td>
<td>System constants are used like constants, and also created as define statements. Unlike constants, system constants can be implemented. They are always explicitly visible in the generated code.</td>
</tr>
</tbody>
</table>

Remarks

a. The patterns of the blue circles indicate the scope (cf. section 5.1) and some properties (cf. section 5.2) of the constants.

4.2 Composite Elements

Several composite, i.e. non-scalar elements are available in ASCET block diagrams:

- arrays (section 4.2.1 on page 21)
- matrices (section 4.2.2 on page 22)
- characteristic lines and maps (section 4.2.3 on page 23)

Composite elements can be variables or parameters of any scope (cf. section 5.1). The tables

4.2.1 Arrays

An array is a one-dimensional, indexed set of variables or parameters which have the same scalar type (logic, signed/unsigned discrete, continuous). This type is not visible in the diagram. In the “Tree” pane, no special icon indicates the array type, it is visible only textually in the “Outline” tab.

The position of a scalar value within an array is indicated by its associated index value which must be of the type unsigned discrete.

The interface of an array consists of the following methods:

- `void setAt(<scalar type> a, udisc i)`: The assignment of the scalar value `a` (input at the left side) to the position `i` (left input at the bottom) in the array.
- `scalar type getAt(udisc i)`: Returns the value (output at the right) at position `i` (the right input at the bottom) of the array.
4.2.2 Matrices

A matrix is a two-dimensional, indexed set of variables or parameters which have the same scalar type (logic, signed/unsigned discrete continuous). This type is not visible in the diagram. In the "Tree" pane, no special icon indicates the matrix type, it is visible only textually in the "Outline" tab.

The position of a scalar value within a matrix is indicated by its associated X and Y index values, which must be of the type unsigned discrete.

The interface of an array consists of the following methods:

- void setAt(<scalar type> a, udisc ix, udisc iy): The assignment of the scalar value a (input at the left side) to the position ix (left input at the bottom) / iy (left input at the top) in the array.
- scalar type getAt(udisc i): Returns the value (output at the right) at position jx (right input at the bottom) / jy (right input at the top) of the matrix.
4.2.3 Characteristic Lines and Maps

To support nonlinear control engineering, characteristic lines and maps are available in ASCET block diagrams. They are used to describe a value in dependence of one or two other values.

Characteristic lines and maps are available in the varieties normal, fixed, and group.

A fixed characteristic line/map has an equidistant distribution, i.e. the sample points have a constant distance from each other.

A group characteristic line/map does not contain a sample point distribution, but references an external distribution (cf. page 24) of sample points.

**Characteristic Lines**

A characteristic line is represented as a one-dimensional table of sample points, each of which is associated with a sample value. The sample points represent the X axis of a function graph, the sample values represent the curve being described.

The "Elements" palette provides a combo box and a button for characteristic lines, the "Elements" toolbar provides a single button with a submenu for the variety.

In the block diagram, the type of a characteristic line is not visible, only kind, scope (exported/imported/local) and some properties; see chapter 5 on page 33 for details.

The types of sample points and values of the characteristic line are not visible in the diagram. In the "Tree" pane, no special icons indicate the types, they are visible only textually in the "Outline" tab.

Green (red) arrows indicate that the sample point distribution is strictly monotonic increasing (decreasing). No arrow is shown if strict monotony is not given.

The interface of a characteristic line consists of the following methods:

- void search (<arithmetic type> a): The supporting points surrounding a (input at the left) are searched, and the interpolation factors are computed.
  - Available for normal/fixed characteristic lines and distributions of group characteristic lines.
- <arithmetic type> interpolate(): This method interpolates the value of the characteristic line from the interpolation factors and the value points at the associated supporting points. The result is returned (output on the right).
- <arithmetic type> getAt (<arithmetic type> a): is the combination of the search and interpolate method.
  - Not available for group characteristic lines.
## Characteristic Maps

A characteristic map is represented as a two-dimensional table of sample points; each pair of sample points is associated with a sample value.

The "Elements" palette provides a combo box and a button for characteristic maps, the "Elements" toolbar provides a single button with a submenu.

In the block diagram, the type of a characteristic map is not visible, only kind, scope (exported/imported/local) and some properties; see chapter 5 on page 33 for details.

The types of sample points and values of the characteristic map are not visible in the diagram. In the "Tree" pane, no special icons indicate the types, they are visible only textually in the "Outline" tab.

Green (red) arrows indicate that the X sample point distribution is strictly monotonic increasing (decreasing). No arrow is shown if strict monotony is not given.

<table>
<thead>
<tr>
<th>name</th>
<th>palette icon</th>
<th>block diagram icon</th>
<th>Tree pane icon</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OneD Table</strong></td>
<td><img src="image1" alt="Normal" /></td>
<td><img src="image2" alt="CharLine" /></td>
<td><img src="image3" alt="CharLine::ID[cont-&gt;cont]" /> <img src="image4" alt="CharLineV::ID[cont-&gt;cont]" /></td>
</tr>
<tr>
<td>(normal); strictly monotonic increasing</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td><strong>OneD Table</strong></td>
<td><img src="image5" alt="Fixed" /></td>
<td><img src="image6" alt="fixedCharLine" /></td>
<td></td>
</tr>
<tr>
<td>(fixed)</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OneD Table</strong></td>
<td><img src="image7" alt="Group" /></td>
<td><img src="image8" alt="groupByLine" /></td>
<td></td>
</tr>
<tr>
<td>(group)</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Distribution</strong></td>
<td><img src="image9" alt="Dist" /></td>
<td><img src="image10" alt="X_distr" /></td>
<td><img src="image11" alt="X_distr::distrib[cont]" /></td>
</tr>
<tr>
<td>(required for group char. line)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For the icon differences induced by scope, see section 5.1 on page 33.
The monotony behavior of the Y sample point distribution is not shown in the block diagram.

The interface of a characteristic map consists of the following methods:

- **void search ((arithmetic type) a, (arithmetic type) b):** The supporting points surrounding a and b (inputs at the left) are searched, and the interpolation factors are computed. Available for normal/fixed characteristic maps and distributions of group characteristic maps.

- **(arithmetic type) interpolate():** This method interpolates the value of the characteristic map from the interpolation factors and the value points at the associated supporting points. The result is returned (output on the right).

- **(arithmetic type) getAt ((arithmetic type) a, (arithmetic type) b):** The combination of the search and interpolate method.
  Not available for group characteristic maps.

<table>
<thead>
<tr>
<th>name</th>
<th>palette icon</th>
<th>block diagram icon</th>
<th>Tree pane icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwoD Table (normal); X axis strictly monotonic increasing</td>
<td>![Normal Icon]</td>
<td>![normalCharMap Icon]</td>
<td>![CharMap::2D[cont,cont-&gt;cont] Icon]</td>
</tr>
<tr>
<td>TwoD Table (normal); X axis strictly monotonic decreasing</td>
<td>![CharMapDown Icon]</td>
<td>![charMapDown Icon]</td>
<td>![charMapV::2D[cont,cont-&gt;cont] Icon]</td>
</tr>
<tr>
<td>TwoD Table (normal); no strict monotony for X axis</td>
<td>![charMapMisc Icon]</td>
<td>![charMapMisc Icon]</td>
<td></td>
</tr>
<tr>
<td>TwoD Table (fixed)</td>
<td>![Fixed Icon]</td>
<td>![fixedCharMap Icon]</td>
<td></td>
</tr>
<tr>
<td>TwoD Table (group)</td>
<td>![Group Icon]</td>
<td>![grpMap(X.dis,Y.dis) Icon]</td>
<td></td>
</tr>
<tr>
<td>Distribution (2 required for group char. map)</td>
<td>![Distribution Icon]</td>
<td>![X_dis::distrib[cont] Icon]</td>
<td></td>
</tr>
</tbody>
</table>

For the icon differences induced by scope, see section 5.1 on page 33.
4.3 Complex Elements (Included Components)

In a block diagram, an included component, or complex element, is represented by the component's layout.

In the "Tree" pane, an included component is represented by its component type icon.

<table>
<thead>
<tr>
<th>component type</th>
<th>icon</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>module&lt;sup&gt;a&lt;/sup&gt;</td>
<td><img src="image" alt="Module_BDE::Module_BDE" /></td>
<td>Modules can only be included in other modules.</td>
</tr>
<tr>
<td>class&lt;sup&gt;a&lt;/sup&gt;</td>
<td><img src="image" alt="Class_BDE::Class_BDE" /></td>
<td></td>
</tr>
<tr>
<td>state machine</td>
<td><img src="image" alt="State_Machine::State_Machine" /></td>
<td>State machines, Boolean tables and conditional tables are special classes.</td>
</tr>
<tr>
<td>Boolean table</td>
<td><img src="image" alt="Class_BoolTab::Class_BoolTab" /></td>
<td></td>
</tr>
<tr>
<td>conditional table</td>
<td><img src="image" alt="Class_CondTab::Class_CondTab" /></td>
<td></td>
</tr>
<tr>
<td>record</td>
<td><img src="image" alt="Record::Record" /></td>
<td></td>
</tr>
</tbody>
</table>

<sup>a</sup> The small blue boxes with letters denote the item type, i.e. Block diagram, C code, or ESDL.

4.4 Signature Elements

ASCET block diagrams can contain elements of method and process signatures.

- Method signature elements (section 4.4.1 on page 26)
- process signature elements (section 4.4.2 on page 29)

4.4.1 Method Signature Elements

The signature elements of a method can only be created in the signature editor. A method can have the following signature elements:

- arguments (*Method Arguments* on page 27)
- local variables (*Local Variables (Method)* on page 28)
- return value (*Return Value* on page 28)
### Method Arguments

<table>
<thead>
<tr>
<th>Argument type</th>
<th>block diagram icon</th>
<th>Icon in Tree pane and signature editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>scalar</td>
<td><img src="image1" alt="scalar icons" /></td>
<td><img src="image2" alt="signature icons" /></td>
</tr>
<tr>
<td>enumeration</td>
<td><img src="image3" alt="enumeration icon" /></td>
<td><img src="image4" alt="signature icon" /></td>
</tr>
<tr>
<td>composite</td>
<td><img src="image5" alt="composite icons" /></td>
<td><img src="image6" alt="signature icons" /></td>
</tr>
<tr>
<td>complex</td>
<td><img src="image7" alt="complex icon" /></td>
<td><img src="image8" alt="signature icon" /></td>
</tr>
</tbody>
</table>
### Local Variables (Method)

<table>
<thead>
<tr>
<th>variable type</th>
<th>block diagram icon</th>
<th>icon in Tree pane and signature editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>scalar</td>
<td><img src="image1" alt="Icon" /></td>
<td><img src="image2" alt="Icon" /></td>
</tr>
<tr>
<td>enumeration</td>
<td><img src="image3" alt="Icon" /></td>
<td><img src="image4" alt="Icon" /></td>
</tr>
<tr>
<td>composite</td>
<td><img src="image5" alt="Icon" /></td>
<td><img src="image6" alt="Icon" /></td>
</tr>
<tr>
<td>complex</td>
<td><img src="image7" alt="Icon" /></td>
<td><img src="image8" alt="Icon" /></td>
</tr>
</tbody>
</table>

### Return Value

<table>
<thead>
<tr>
<th>return value type</th>
<th>block diagram icon</th>
<th>icon in Tree pane and signature editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>scalar</td>
<td><img src="image9" alt="Icon" /></td>
<td><img src="image10" alt="Icon" /></td>
</tr>
<tr>
<td>enumeration</td>
<td><img src="image11" alt="Icon" /></td>
<td><img src="image12" alt="Icon" /></td>
</tr>
<tr>
<td>composite</td>
<td><img src="image13" alt="Icon" /></td>
<td><img src="image14" alt="Icon" /></td>
</tr>
<tr>
<td>complex</td>
<td><img src="image15" alt="Icon" /></td>
<td><img src="image16" alt="Icon" /></td>
</tr>
</tbody>
</table>
4.4.2 Process Signature Elements

The signature elements of a process can only be created in the signature editor. A process can have the following signature elements:

- local variables

### variable type | block diagram icon | icon in Tree pane and signature editor
---|---|---
scalar | ![Scalar Icon](image) | ![Scalar Icon](image)
enumeration | ![Enumeration Icon](image) | ![Enumeration Icon](image)
composite | ![Composite Icon](image) | ![Composite Icon](image)
complex | ![Complex Icon](image) | ![Complex Icon](image)

4.5 Miscellaneous Elements

Several other elements are available in ASCET block diagrams:

- implementation casts (section 4.5.1 on page 29)
- hierarchies (section 4.5.2 on page 30)
- self (section 4.5.3 on page 30)
- comments (section 4.5.4 on page 31)

4.5.1 Implementation Casts

*Implementation casts* provide the user with the ability to influence the implementation of intermediate results within arithmetic chains. This allows the user to display knowledge regarding particular physical correlations (for example, that a specific range of values is not exceeded at a defined point in the model) in the model, without requiring the allocation of physical memory.

Implementation casts cannot be used in conjunction with logical elements.
### 4.5.2 Hierarchies

In order to structure a block diagram, *graphical hierarchies* can be used. Graphical hierarchies do not influence the semantics of a block diagram, but are used for structuring only.

A hierarchy contains a part of the block diagram. At its parent level of the diagram, it is visible only as a symbol. The lines that cross the border of the hierarchy, i.e. that connect elements inside the hierarchy with those outside, are represented by pins.

<table>
<thead>
<tr>
<th>name</th>
<th>palette / toolbar icon</th>
<th>block diagram icon</th>
<th>&quot;Navigation&quot; tab icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy</td>
<td>![Hierarchy Icon]</td>
<td>![Hierarchy Diagram]</td>
<td>![Hierarchy Tab Icon]</td>
</tr>
</tbody>
</table>

**Inpin** *(none)*

**Outpin** *(none)*

a. Hierarchies, their inpins and outpins are not shown in the "Outline" tab.

### 4.5.3 Self

The *Self* element is a reference to the currently edited component itself.

<table>
<thead>
<tr>
<th>name</th>
<th>palette / toolbar icon</th>
<th>block diagram icon</th>
<th>&quot;Navigation&quot; tab icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self</td>
<td>![Self Icon]</td>
<td>![Self Diagram]</td>
<td>![Self Tab Icon]</td>
</tr>
</tbody>
</table>

a. The Self element is not shown in the "Outline" tab.
b. The icon depends on the component type (see also section 4.3)
4.5.4 Comments

ASCET block diagrams can include textual comments.

| name    | toolbar button | block diagram icon (= comment text) | "Navigation" tab icon

| Comment | ![Comment Icon] | This is a comment. | ![Comment Icon] |

a. Comments are not shown in the "Outline" tab.
5 Miscellaneous Icons

This chapter lists various icons used for different purposes in block diagrams.

- section 5.1 "Scope Icons" on page 33
- section 5.2 "Icons for Properties" on page 34
- section 5.3 "Icons in the "Tree" Pane" on page 35
- section 5.4 "Icons in the "General" Toolbar" on page 36
- section 5.5 "Icons in Tab Labels" on page 37

5.1 Scope Icons

The following scopes are available for ASCET elements:

- imported, exported, local, method-/process-local

Each scope is represented by a pattern on the basic variable or parameter icon.

<table>
<thead>
<tr>
<th>scope</th>
<th>definition</th>
<th>icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>exported</td>
<td>Exported elements are defined in one component and can be accessed by all other components by importing that element.</td>
<td>![icon]</td>
</tr>
<tr>
<td>imported</td>
<td>Imported elements are defined in another component or project, but can be used in the component that imports them.</td>
<td>![icon]</td>
</tr>
<tr>
<td>local</td>
<td>Local elements can only be used within the component that defines them, i.e. in all methods or processes of that component.</td>
<td>![icon]</td>
</tr>
<tr>
<td>method-/process-local&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Method-/process-local elements can only be used in the method/process that define them.</td>
<td>![icon]</td>
</tr>
</tbody>
</table>

a. variables only

The scope icons appear in various places, e.g., the "Tree" pane or the block diagram.
5.2 Icons for Properties

Several element properties are marked by overlay icons to the basic variable or parameter icon.

<table>
<thead>
<tr>
<th>property</th>
<th>definition</th>
<th>block diagram icon</th>
<th>Tree pane icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference(^a)</td>
<td>Marks a reference type (i.e. composite or complex element) as explicit reference.</td>
<td><img src="image" alt="referenceArray icon" /></td>
<td><img src="image" alt="referenceClass icon" /></td>
</tr>
<tr>
<td>virtual</td>
<td>Virtual variables/parameters are only available in the specification platform, they bear no relevance for code generation.</td>
<td><img src="image" alt="virtualVariable icon" /></td>
<td><img src="image" alt="virtualParam icon" /></td>
</tr>
<tr>
<td>dependent(^b)</td>
<td>Model parameters can be connected to other system or model parameters via a mathematical dependency.</td>
<td><img src="image" alt="parEnv icon" /></td>
<td></td>
</tr>
<tr>
<td>non-volatile(^c)</td>
<td>In the ECU, the element is placed in the non-volatile memory.</td>
<td><img src="image" alt="Nvariable icon" /></td>
<td></td>
</tr>
</tbody>
</table>

\(^a\) only available for variables  
\(^b\) only available for parameters  
\(^c\) overlay icon only shown for variables and messages
5.3 Icons in the "Tree" Pane

Most icons that appear in the tabs of the "Tree" pane have already been mentioned. The remaining ones are listed here.

<table>
<thead>
<tr>
<th>Location</th>
<th>Icon</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Outline&quot; tab</td>
<td><img src="image1" alt="Image" /></td>
<td>- public diagram, currently not loaded (left) / loaded (right, blue rim)</td>
</tr>
<tr>
<td></td>
<td><img src="image2" alt="Image" /></td>
<td>- private diagram, currently not loaded (left) / loaded (right, blue rim)</td>
</tr>
<tr>
<td></td>
<td><img src="image3" alt="Image" /></td>
<td>- additional marker for the currently loaded diagram</td>
</tr>
<tr>
<td></td>
<td><img src="image4" alt="Image" /></td>
<td>- method, public(^a) (left) or private (right)</td>
</tr>
<tr>
<td></td>
<td><img src="image5" alt="Image" /></td>
<td>- process(^b) (always public)</td>
</tr>
<tr>
<td></td>
<td><img src="image6" alt="Image" /></td>
<td>- indicates the process/method most recently used</td>
</tr>
<tr>
<td>&quot;Navigation&quot; tab</td>
<td><img src="image7" alt="Image" /></td>
<td>- root node for the &quot;Graphic Blocks&quot; tree structure</td>
</tr>
<tr>
<td></td>
<td><img src="image8" alt="Image" /></td>
<td>- root node for the &quot;Sequence Calls&quot; tree structure</td>
</tr>
</tbody>
</table>

a. A similar icon marks the "Methods" tab in the "Browse" view (cf. section 5.5).
b. In modules only

Each tab in the tree pane contains some buttons.

<table>
<thead>
<tr>
<th>Button</th>
<th>Icon</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change filter</td>
<td><img src="image9" alt="Image" /></td>
<td><em>Outline</em> and &quot;Navigation&quot; tab only; can be used to filter the tabs.</td>
</tr>
<tr>
<td>settings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change sort</td>
<td><img src="image10" alt="Image" /></td>
<td>&quot;Outline&quot; tab only; can be used to sort the tab.</td>
</tr>
<tr>
<td>criteria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expand All</td>
<td><img src="image11" alt="Image" /></td>
<td></td>
</tr>
<tr>
<td>Collapse All</td>
<td><img src="image12" alt="Image" /></td>
<td></td>
</tr>
<tr>
<td>Search the Tree</td>
<td><img src="image13" alt="Image" /></td>
<td>&quot;Outline&quot; and &quot;Navigation&quot; tab only; can be used (together with the &quot;Search&quot; field) to search for a particular item.</td>
</tr>
</tbody>
</table>
5.4 Icons in the "General" Toolbar

Besides default buttons such as **Save** or **Print**, the "General" toolbar contains the following buttons:

<table>
<thead>
<tr>
<th>button name</th>
<th>icon</th>
<th>explanation</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch to Connection Mode</td>
<td></td>
<td>starts the element connection mode</td>
<td></td>
</tr>
<tr>
<td>Redraw</td>
<td></td>
<td>reloads the current diagram</td>
<td></td>
</tr>
<tr>
<td>Edit Component Data</td>
<td></td>
<td>opens the data editor for the edited component</td>
<td>A similar icon (without pencil) marks the &quot;Data&quot; tab in the &quot;Browse&quot; view.</td>
</tr>
<tr>
<td>Edit Component Implementation</td>
<td></td>
<td>opens the implementation editor for the edited component</td>
<td>A similar icon (without pencil) marks the &quot;Implementation&quot; tab in the &quot;Browse&quot; view.</td>
</tr>
<tr>
<td>Edit Default Project</td>
<td></td>
<td>opens the default project for the edited component</td>
<td></td>
</tr>
<tr>
<td>Tool Options</td>
<td></td>
<td>opens the ASCET options window</td>
<td></td>
</tr>
<tr>
<td>Insert Component</td>
<td></td>
<td>inserts a component as complex element</td>
<td></td>
</tr>
<tr>
<td>Insert Method</td>
<td></td>
<td>creates a new method in the current diagram</td>
<td>A similar icon (without +) marks the &quot;Methods&quot; tab in the &quot;Browse&quot; view.</td>
</tr>
<tr>
<td>Insert Process</td>
<td></td>
<td>creates a new method in the current diagram</td>
<td>Only available if the edited component was opened from the including component.</td>
</tr>
<tr>
<td>Browse to Parent Component</td>
<td></td>
<td>opens an editor for the including component</td>
<td></td>
</tr>
<tr>
<td>Generate Code</td>
<td></td>
<td>generates code for the edited component</td>
<td></td>
</tr>
<tr>
<td>Compile generated Code</td>
<td></td>
<td>compiles generated code for the edited component</td>
<td></td>
</tr>
<tr>
<td>Open Experiment</td>
<td></td>
<td>generates and compiles code for the edited component and starts the offline experiment</td>
<td></td>
</tr>
</tbody>
</table>
## 5.5 Icons in Tab Labels

The following table lists the icons used in the labels of the various tabs of the block diagram editor.

<table>
<thead>
<tr>
<th>location</th>
<th>icon</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Outline&quot; tab</td>
<td><img src="image" alt="Outline" /></td>
<td></td>
</tr>
<tr>
<td>&quot;Navigation&quot; tab</td>
<td><img src="image" alt="Navigation" /></td>
<td></td>
</tr>
<tr>
<td>&quot;Database&quot; tab</td>
<td><img src="image" alt="Database" /></td>
<td></td>
</tr>
<tr>
<td>&quot;Specification&quot; tab</td>
<td><img src="image" alt="Specification" /></td>
<td>In the editor, this tab label is displayed vertically.</td>
</tr>
<tr>
<td>&quot;Browse&quot; tab</td>
<td><img src="image" alt="Browse" /></td>
<td>In the editor, this tab label is displayed vertically.</td>
</tr>
<tr>
<td>&quot;Elements&quot; tab</td>
<td><img src="image" alt="Elements" /></td>
<td></td>
</tr>
<tr>
<td>&quot;Data&quot; tab</td>
<td><img src="image" alt="Data" /></td>
<td></td>
</tr>
<tr>
<td>&quot;Implementation&quot; tab</td>
<td><img src="image" alt="Implementation" /></td>
<td></td>
</tr>
<tr>
<td>&quot;Methods&quot; tab</td>
<td><img src="image" alt="Methods" /></td>
<td></td>
</tr>
<tr>
<td>&quot;Layout&quot; tab</td>
<td><img src="image" alt="Layout" /></td>
<td></td>
</tr>
</tbody>
</table>
6 **ETAS Contact Addresses**

### ETAS HQ

ETAS GmbH

Borsigstraße 14              Phone: +49 711 89661-0
70469 Stuttgart              Fax: +49 711 89661-106
Germany                        WWW: [www.etas.com](http://www.etas.com)

### ETAS Subsidiaries and Technical Support

For details of your local sales office as well as your local technical support team and product hotlines, take a look at the ETAS website:

Index

A
Absolute 11
Addition 7
And 9
Arithmetic Operators 7
Array 21

B
Between 10
Break 15

C
Case 10
Case Operator 9
characteristic line 23
fixed 24
group 24
characteristic map 24
fixed 25
group 25
comment 31
Comparison Operators 8
complex element 26

D
composite elements 21–25
array 21
characteristic line 23
characteristic map 24
distribution 24, 25
matrix 22
oneD table 24
twoD table 25
Constant 20
Continuous Literal 0.0 20
Continuous Literal 1.0 20
Continuous Parameter 18
Continuous Variable 17
collection flow elements 13–15
break 15
if...then 13
if...then...else 13
switch 14
while 14

D
Distribution 24, 25
Division 7
dT 20
dT Parameter 20
Index

E
- elements 17–31
  - array 21
  - characteristic line 23
  - characteristic map 24
  - comment 31
  - complex ~ 26
  - composite 21
  - constants 20
  - distribution 24, 25
dT parameter 20
- hierarchy 30
- implementation cast 29
- included component 26
- literals 20
- matrix 22
- messages 18
- method argument 27
- method signature 26
- method-local variable 28
- miscellaneous 29
- oneD table 24
- parameters 18
- process signature 29
- real-time elements 18
- receive message 19
- resource 19
- return value 28
- scalar 17
- self 30
- send & receive message 19
- send message 19
- signature ~ 26
- system constants 20
twoD table 25
- variables 17

Enumeration Literal 20
Enumeration Parameter 18
Enumeration Variable 17
Equal 8
ETAS Contact Addresses 39

G
- General toolbar 36
- Greater 8
- Greater or Equal 8

H
- hierarchy 30

I
- If...Then 13

If...Then...Else 13
implementation cast 29
included component 26

L
- Literals 20
- local variable
  - method 28
  - process 29
- Logic Literal false 20
- Logic Literal true 20
- Logic Parameter 18
- Logic Variable 17
- Logical Operators 9

M
- Matrix 22
- Max 10
- Messages 18, 20
- method argument 27
- method signature elements 26–28
  - argument 27
  - local variable 28
  - return value 28
- Min 10
- miscellaneous 33–37
  - General toolbar 36
  - properties 34
  - scope 33
  - tab label 37
  - Tree pane 35
- miscellaneous elements 29–31
  - comment 31
  - hierarchy 30
  - implementation cast 29
  - self 30
- Miscellaneous Operators 10
- Modulo 7
- Multiplex Operator 9
- Multiplication 7
- Mux 9

N
- Navigation tab 35
- Negation 11
- Not 9
- Not Equal 8

O
- OneD Table 24
operators 7–11
  absolute 11
  addition 7
  and 9
  between 10
  case 10
  division 7
  equal 8
  greater 8
  greater or equal 8
  max 10
  min 10
  modulo 7
  multiplication 7
  mux 9
  negation 11
  not 9
  not equal 8
  or 9
  smaller 8
  smaller or equal 8
  subtraction 7
Or 9
Outline tab 35

P
Parameters 18
process signature elements 29
Product liability disclaimer 5
properties icons 34

R
real-time elements 18
  dT parameter 20
  messages 18
  receive message 19
  resource 19
  send & receive message 19
  send message 19
Receive Message 19
Resource 19
return value 28

S
Safety Instructions
  technical state 5
scalar elements 17–21
  constants 20
  continuous literal 0.0 20
  continuous literal 1.0 20
  continuous parameter 18
  continuous variable 17
dT parameter 20
enumeration literal 20
enumeration parameter 18
enumeration variable 17
literals 20
logic literal false 20
logic literal true 20
logic parameter 18
logic variable 17
messages 18
parameters 18
real-time elements 18
receive message 19
send & receive message 19
send message 19
signed discrete parameter 18
signed discrete variable 17
system constants 20
unsigned discrete parameter 18
unsigned discrete variable 17
variables 17
scope icon 33
self 30
Send & Receive Message 19
Send Message 19
signature elements 26–29
  method argument 27
  method-local variable 28
  process-local variable 29
  return value 28
Signed Discrete Parameter 18
Signed Discrete Variable 17
Smaller 8
Smaller or Equal 8
Subtraction 7
Switch 14
System Constant 20

T
tab label 37
Tree pane 35
  buttons 35
TwoD Table 25

U
Unsigned Discrete Parameter 18
Index

Unsigned Discrete Variable 17

V
Variables 17

W
While 14