ETAS

ASCET V6.3
Icon Reference Guide

DRIVING | EMBEDDED EXCELLENCE

Copyright

Document EC010015 V6.3 R01 EN - 12.2014

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 6
1.2 Manual Structure 6
1.3 Typographic Conventions 6

2 Operators in Block Diagrams 8
2.1 Arithmetic Operators 8
2.2 Comparison Operators 8
2.3 Logical Operators 9
2.4 Multiplex Operator 10
2.5 Case Operator 10
2.6 Miscellaneous Operators 11

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 14

4 Elements in Block Diagrams 16
4.1 Scalar Elements 16
4.1.1 Variables 16
4.1.2 Parameters 17
4.1.3 Real-Time Elements 17
Messages 17
Resources 18

dT Parameter 19
4.1.4 Literals 19
4.1.5 Constants and System Constants 19
4.2 Composite Elements 20
4.2.1 Arrays 20
4.2.2 Matrices 21
4.2.3 Characteristic Lines and Maps 22
Characteristic Lines 22
Characteristic Maps 23
4.3 Complex Elements (Included Components) 25
4.4 Signature Elements 26
4.4.1 Method Signature Elements 26
Method Arguments 27
Local Variables (Method) 28
Return Value 28
4.4.2 Process Signature Elements 28
4.5 Miscellaneous Elements 29
4.5.1 Implementation Casts 29
4.5.2 Hierarchies 30
4.5.3 Self 30
4.5.4 Comments 31
5 Miscellaneous Icons 32
5.1 Scope Icons 32
5.2 Icons for Properties 33
5.3 Icons in the "Tree" Pane 34
5.4 Icons in the "General" Toolbar 35
5.5 Icons in Tab Labels 36
6 ETAS Contact Addresses 37
ETAS HQ 37
ETAS Subsidiaries and Technical Support 37
Index 39

1 Introduction

INTECRIO provides an innovative solution for the functional and software development of modern embedded software systems. INTECRIO 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 INTECRIO 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:

⚠

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 V6.3 safety manual (`ASCET Safety Manual.pdf`) available at ETAS upon request.

1.2 Manual Structure

The INTECRIO "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.	General emphasis and new terms are set in *italics*.
The OSEK group (see http://www.osekvdx.org/) has developed certain standards.	Links to internet documents are set in blue, underlined font.

Important notes for the users are presented as follows:

Tip

Important note for users.

2 Operators in Block Diagrams

This chapter lists the operators available in INTECRIO block diagrams.

- arithmetic operators (section 2.1 on page 8)
- comparison operators (section 2.2 on page 8)
- logical operators (section 2.3 on page 9)
- multiplex operator (section 2.4 on page 10)
- case operator (section 2.5 on page 10)
- miscellaneous operators (section 2.6 on page 11)

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.

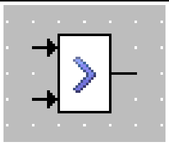
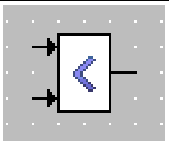
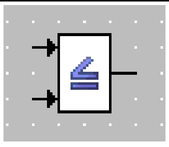
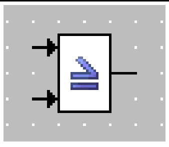
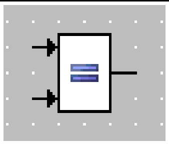
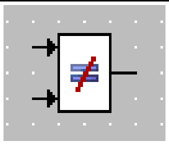
name	**palette / toolbar / "Navigation" tab icon**	**block diagram icon**	**remarks**
Addition	+	+	Returns the sum of the inputs.
Subtraction	−	−	Returns the difference between the upper and the lower input.
Multiplication	×	×	Returns the product of the inputs.
Division	÷	÷	Returns the quotient of the upper and the lower input.
Modulo	%	%	Returns the remainder of the division upper / lower input.

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.

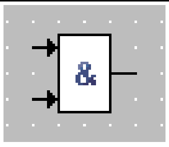
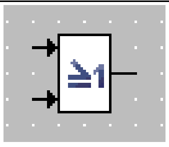
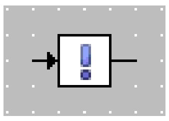
name	palette / toolbar / "Navigation" tab icon	block diagram icon	remarks
Greater	>		Returns `true` if the upper input is greater than the lower input. Returns `false` otherwise.
Smaller	<		Returns `true` if the upper input is smaller than the lower input. Returns `false` otherwise.
Smaller or Equal	≦		Returns `true` if the upper input is smaller than or equal to the lower input. Returns `false` otherwise.
Greater or Equal	≧		Returns `true` if the upper input is greater than or equal to the lower input. Returns `false` otherwise.
Equal	=		Returns `true` if the upper input is equal to the lower input. Returns `false` otherwise.
Not Equal	≠		Returns `true` if the upper input is not equal to the lower input. Returns `false` otherwise.

2.3 Logical Operators

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

- And, Or, Not

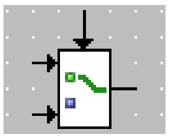
The And and Or operators can have 2 to 20 arguments, the Not operator has one argument.

name	palette / toolbar / "Navigation" tab icon	block diagram icon	remarks
And	&		Returns `true` if all inputs are `true`. Returns `false` if at least one input is `false`.
Or	≥1		Returns `true` if at least one input is `true`. Returns `false` if all inputs are false.
Not	!		Returns `true` (`false`) if the input is `false` (`true`).

2.4 Multiplex Operator

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

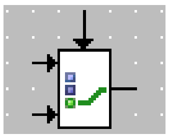
The multiplex operator can be used with 2 to 20 arguments, however, the identical functionality of a multi-argument Mux operator can be built as a cascade of several 2-argument Mux operators.

name	palette / toolbar / "Navigation" tab icon	block diagram icon	remarks
Mux			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`).

2.5 Case Operator

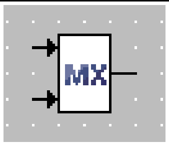
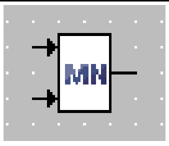
The *Case* operator is a special case of the conditional operator. It does not take a logical value, but a switch value of discrete type. The Case operator has `n` arguments (`n = 2..20`), `n-1` of which are numbered consecutively. The last argument is the default case.

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.

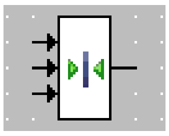
name	palette / toolbar / "Navigation" tab icon	block diagram icon	remarks
Case			The input on the top is the switch value, the inputs on the left are the arguments.

2.6 Miscellaneous Operators

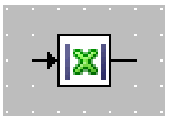
Max and Min Operators: The Max and Min operators can have 2 to 20 arguments; they can be applied only to arithmetic elements.

name	palette / toolbar / "Navigation" tab icon	block diagram icon	remarks
Max	MX		Returns the largest input value.
Min	MN		Returns the smallest input value.

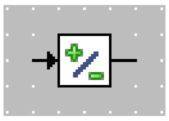
Between Operator: The *Between* operator checks if the argument value (upper input) lies between the limiters min (middle input) and max (lower input). If this is the case, the logical return value is true, otherwise it is set to false.

name	palette / toolbar / "Navigation" tab icon	block diagram icon	remarks
Between			Returns `true` (`false`) if the argument lies (does not lie) between min and max.

Absolute Operator: Argument and return value of the *Absolute* operator have to be both either cont or discrete.

name	palette / toolbar / "Navigation" tab icon	block diagram icon	remarks
Absolute			Returns the absolute value of the argument.

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.

name	palette / toolbar / "Navigation" tab icon	block diagram icon	remarks
Negation			Returns the negative value of the argument.

3 Control Flow Elements in Block Diagrams

This chapter lists the control flow elements available in INTECRIO block diagrams.

The following control flow statements are available in block diagrams:

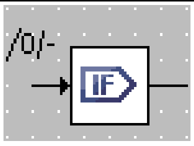
- If...Then (section 3.1 on page 13)
- If...Then...Else (section 3.2 on page 13)
- Switch (section 3.3 on page 14)
- While (section 3.4 on page 14)
- Break Statement (section 3.5 on page 14)

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.

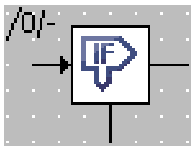
3.1 If...Then

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.

Name	palette / toolbar / "Navigation" tab icon	block diagram icon	Remarks
If...Then			Activates a control flow branch (connected to the output) if the input is `true`.

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.

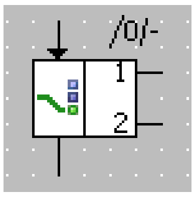
Name	palette / toolbar / "Navigation" tab icon	block diagram icon	Remarks
If...Then...Else			Activates the control flow branch on the right (connected to the output) if the input is `true`. Activates the control flow branch at the bottom if the input is `false`.

3.3 Switch

The *Switch* construct is similar to the Case operator (cf. section 2.5 on page 10). 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.

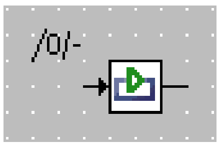
A switch can have 2 to 20 branches.

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

Name	palette / toolbar / "Navigation" tab icon	block diagram icon	Remarks
Switch			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.

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.

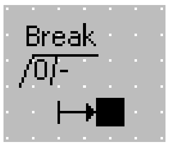
Name	palette / toolbar / "Navigation" tab icon	block diagram icon	Remarks
While			Activates the control flow branch if the input value is `true`. The operation is executed as long as the input value remains `true`; the input value should be manipulated in the while loop.

3.5 Break

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

Tip

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

Name	palette / toolbar / "Navigation" tab icon	block diagram icon	Remarks
Break			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 INTECRIO block diagrams.

- section section 4.1 "Scalar Elements" on page 16
- section section 4.2 "Composite Elements" on page 20
- section section 4.3 "Complex Elements (Included Components)" on page 25
- section section 4.4 "Signature Elements" on page 26
- section section 4.5 "Miscellaneous Elements" on page 29

4.1 Scalar Elements

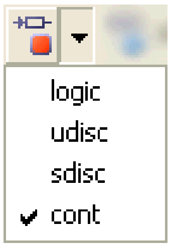
Several scalar elements are available in INTECRIO block diagrams:

- variables (section 4.1.1 on page 16)
- parameters (section 4.1.2 on page 17)
- real-time elements (section 4.1.3 on page 17)
- literals (section 4.1.4 on page 19)
- constants and system constants (section 4.1.5 on page 19)

4.1.1 Variables

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

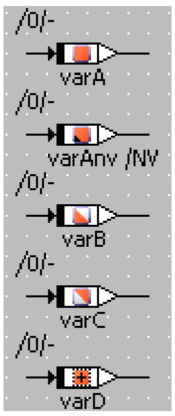
- Logic, Signed discrete, Unsigned discrete, Continuous, Enumeration



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 32 for details.

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

name	palette icon	block diagram icon[a]	Tree pane icon
Logic Variable	L🟥		L 🟥 l_var::log
Signed Discrete Variable	S🟥		S 🟥 s_var::sdisc
Unsigned Discrete Variable	U🟥		U 🟥 u_var::udisc
Continuous Variable	C🟥		C 🟥 c_var::cont
Enumeration Variable	E🟥		E 🟥 e_var::enum_s

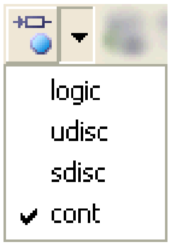
a. The patterns of the red squares indicate the scope (cf. section 5.1) and some properties (cf. section 5.2) of the variables.

4.1.2 Parameters

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

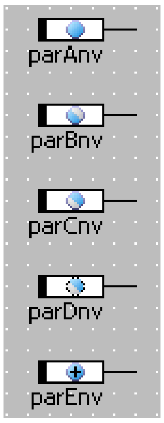
- Logic, Signed discrete, Unsigned discrete, Continuous, Enumeration

The "Elements" palette provides a button for each parameter, the "Elements"toolbar provides a single button with a submenu 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 32 for details.

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

name	palette icon	block diagram icon[a]	Tree pane icon
Logic Parameter		 parAnv	l_par::log
Signed Discrete Parameter		parBnv	s_par::sdisc
Unsigned Discrete Parameter		parCnv	u_par::udisc
Continuous Parameter		parDnv	c_par::cont
Enumeration Parameter		parEnv	e_par::enum_s

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

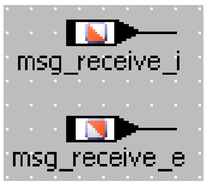
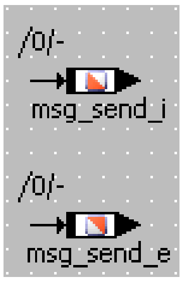
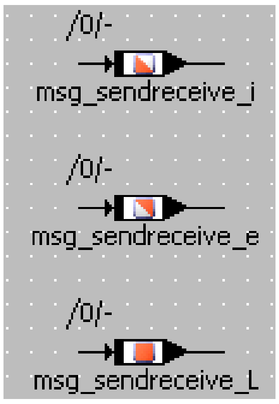
Messages

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

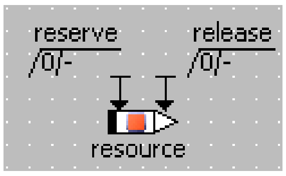
All messages can be of scope exported or imported, only send & receive messages can be of scope local (cf. section 5.1 on page 32).

name	palette / toolbar icon	block diagram icon	Tree pane icon
Receive Message	✉		✉ 🟥 msg_r_e::mesg[cont] ✉ 🟥 msg_r_i::mesg[cont]
Send Message	✉		✉ 🟥 msg_s_e::mesg[cont] ✉ 🟥 msg_s_i::mesg[sdisc]
Send & Receive Message	✉	 /0/- msg_sendreceive_i /0/- msg_sendreceive_e /0/- msg_sendreceive_L	✉ 🟥 msg_sr_e::mesg[cont] ✉ 🟥 msg_sr_i::mesg[log] ✉ 🟥 msg_sr_L::mesg[udisc]

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.

name	palette / toolbar icon	block diagram icon	Tree pane icon
Resource	🔑		🔑 🟥 resource::resource

dT Parameter

In control engineering applications the result of the calculations within a component often depends on the value of the sampling rate. INTECRIO 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.

name	palette / toolbar icon	block diagram icon	Tree pane icon
dT		dT	dT::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 INTECRIO 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[a]
Enumeration Literal	ENUM	string	ENUM string
Logic Literal true	TRUE	true	ENUM true
Logic Literal false	FALSE	false	ENUM false
Continuous Literal 0.0	0.0	0.0	ENUM 0.0
Continuous Literal 1.0	1.0	1.0	ENUM 1.0

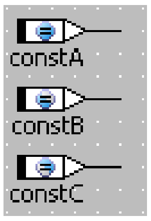
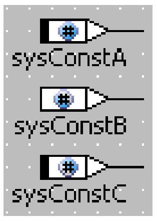
a. Literals are not shown in the "Outline" tab.

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 32).

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 **#**).

name	**palette / toolbar icon**	**block diagram icon**[a]	**Remarks**
Constant	n/a		Constants are created as a define statement in the generated C code. However, they are not necessarily explicitly visible in the generated code.
System Constant	n/a		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.

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 INTECRIO block diagrams:

- arrays (section 4.2.1 on page 20)
- matrices (section 4.2.2 on page 21)
- characteristic lines and maps (section 4.2.3 on page 22)

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

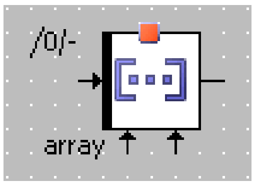
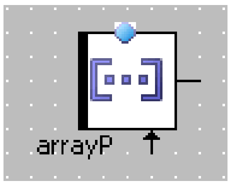
4.2.1 Arrays

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.

name	palette / toolbar icon	block diagram icon	Tree pane icon
Array (variable)			🟥 array::array[cont]
Array (parameter)			🔵 arrayP::array[cont]

For the icon differences induced by scope, see section 5.1 on page 32.

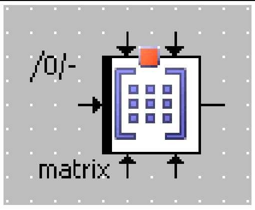
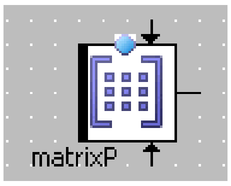
4.2.2 Matrices

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.

name	palette / toolbar icon	block diagram icon	Tree pane icon
Matrix (variable)			🟥 matrix::mat[cont]
Matrix (parameter)			🔵 matrixP::mat[cont]

For the icon differences induced by scope, see section 5.1 on page 32.

4.2.3 Characteristic Lines and Maps

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

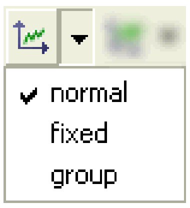
Characteristic lines and maps are available in the varieties *normal*, *fixed*, and *group*.

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 23) 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 32 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:

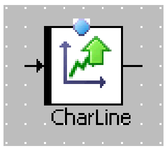
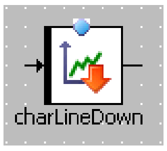
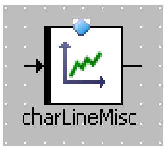
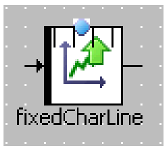
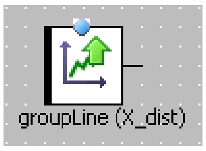
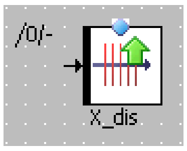
- `void search (`*`<arithmetic type>`*` a)`: The supporting points surrounding `a` (input at the left) are searched, and the interpolation factors are computed.

  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.

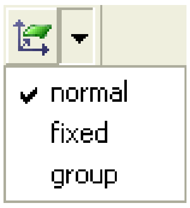
  Not available for group characteristic lines.

name	palette icon	block diagram icon	Tree pane icon
OneD Table (normal); strictly monotonic increasing	Normal +		CharLine::1D[cont->cont] CharLineV::1D[cont->cont]
OneD Table (normal); strictly monotonic decreasing			
OneD Table (normal); no strict monotony			
OneD Table (fixed)	Fixed +		
OneD Table (group)	Group +		
Distribution (required for group char. line)			X_dis::distrib[cont]

For the icon differences induced by scope, see section 5.1 on page 32.

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 32 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.

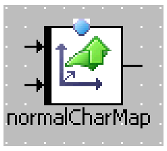
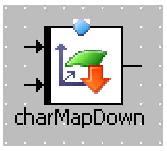
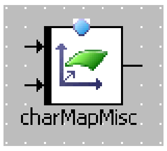
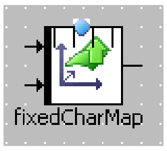
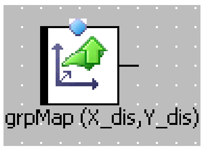
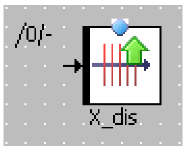
Green (red) arrows indicate that the X sample point distribution is strictly monotonic increasing (decreasing). No arrow is shown if strict monotony is not given.

The monotony behavior of the Y sample pont 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.

name	palette icon	block diagram icon	Tree pane icon
TwoD Table (normal); X axis strictly monotonic increasing	Normal +		CharMap::2D[cont,cont->cont] charMapV::2D[cont,cont->cont]
TwoD Table (normal); X axis strictly monotonic decreasing			
TwoD Table (normal); no strict monotony for X axis			
TwoD Table (fixed)	Fixed +		
TwoD Table (group)	Group +		
Distribution (2 required for group char. map)			X_dis::distrib[cont]

For the icon differences induced by scope, see section 5.1 on page 32.

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.

component type	icon	remarks
module[a]	🟥 Module_BDE::Module_BDE 🟥 Module_C::Module_C 🟥 Module_ESDL::Module_ESDL	Modules can only be included in other modules.
class[a]	🟥 Class_BDE::Class_BDE 🟥 Class_C::Class_C 🟥 Class_ESDL::Class_ESDL	
state machine	🟥 State_Machine::State_Machine	State machines, Boolean tables and conditional tables are special classes.
Boolean table	🟥 Class_BoolTab::Class_BoolTab	
conditional table	🟥 Class_CondTab::Class_CondTab	
record	🟥 Record::Record	

a. The small blue boxes with letters denote the item type, i.e. **B**lock diagram, **C** code, or **E**SDL.

4.4 Signature Elements

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

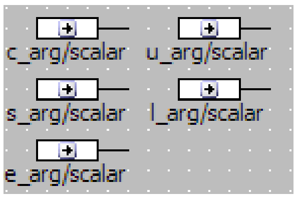
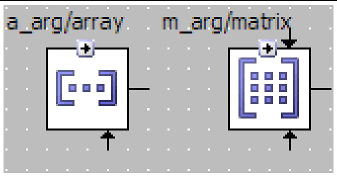
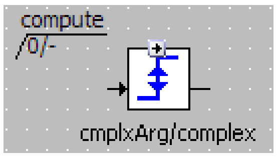
- Method signature elements (section 4.4.1 on page 26)
- process signature elements (section 4.4.2 on page 28)

4.4.1 Method Signature Elements

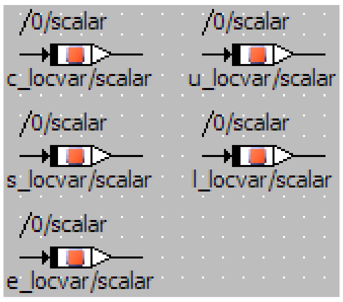
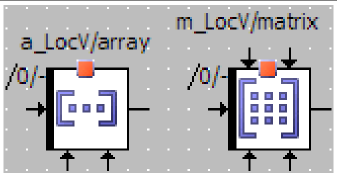
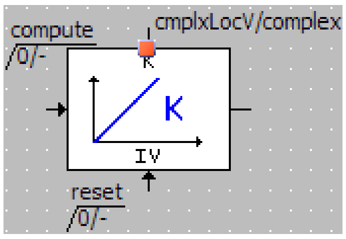
The signature elements of a method can only be created in the signature editor. A method can have the following signature elements:

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

Method Arguments

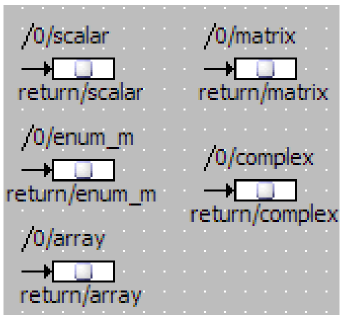
Argument type	block diagram icon	Icon in Tree pane and signature editor
scalar	 c_arg/scalar u_arg/scalar s_arg/scalar l_arg/scalar e_arg/scalar	c_arg::cont [In] s_arg::sdisc [In] u_arg::udisc [In] l_arg::log [In]
enumeration		e_arg::Enumeration [In]
composite		a_arg::array[cont] [In] m_arg::mat[cont] [In]
complex		cmplxArg::EdgeBi [InOut]

Local Variables (Method)

variable type	block diagram icon	icon in Tree pane and signature editor
scalar	 /0/scalar c_locvar/scalar /0/scalar u_locvar/scalar /0/scalar s_locvar/scalar /0/scalar l_locvar/scalar /0/scalar e_locvar/scalar	🟥 c_locvar::cont 🟥 s_locvar::sdisc 🟥 u_locvar::udisc 🟥 l_locvar::log
enumeration		🟥 e_locvar::Enumeration
composite		🟥 arrayLocvar::array[cont] 🟥 matrixLocvar::mat[cont]
complex	 compute /0/- cmplxLocV/complex K K IV reset /0/-	🟥 cmplxLocV::IntegratorK

Return Value

return value type	block diagram icon	icon in Tree pane and signature editor

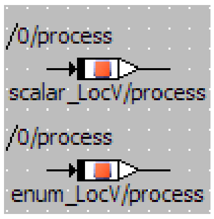
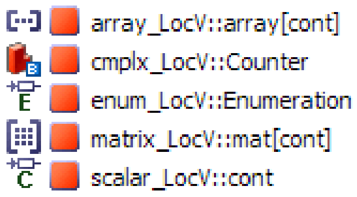
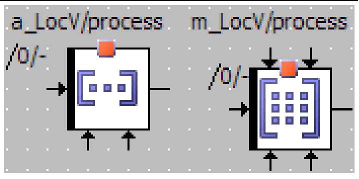
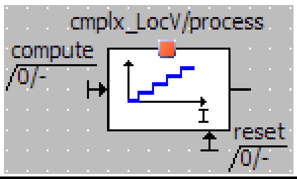
scalar	 /0/scalar
	return/scalar
	/0/matrix
	return/matrix
	/0/enum_m
	return/enum_m
	/0/complex
	return/complex
	/0/array
	return/array
		⬜ return::cont

enumeration		⬜ return::Enumeration
composite		⬜ return::array[cont] ⬜ return::mat[cont]
complex		⬜ return::CountDown

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		
enumeration		
composite		
complex		

array_LocV::array[cont]
cmplx_LocV::Counter
enum_LocV::Enumeration
matrix_LocV::mat[cont]
scalar_LocV::cont

4.5 Miscellaneous Elements

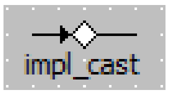
Several other elements are available in INTECRIO block diagrams:

- 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.

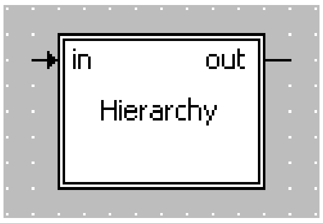
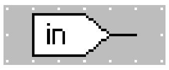
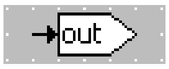
Implementation casts cannot be used in conjunction with logical elements.

name	palette / toolbar icon	block diagram icon	Tree pane icon
Implementation Cast			impl_cast::cont

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.

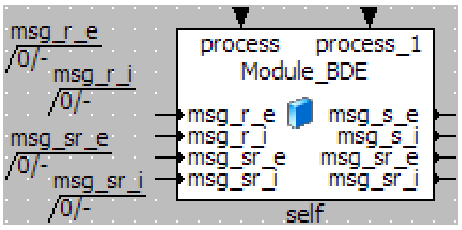
A hierarchy contains a part of the block diagram. At its parent level of the diagram, it is visible only as a symbol. The lines that cross the border of the hierarchy, i.e. that connect elements inside the hierarchy with those outside, are represented by pins.

name	**palette / toolbar icon**	**block diagram icon**	**"Navigation" tab icon**[a]
Hierarchy			Hierarchy
Inpin	*(none)*		in
Outpin	*(none)*		out

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.

name	**palette / toolbar icon**	**block diagram icon** (= layout of currently edited component)	**"Navigation" tab icon**[a,b]
Self		 msg_r_e /0/- msg_r_i /0/- msg_sr_e /0/- msg_sr_i /0/- process process_1 Module_BDE msg_r_e msg_r_i msg_sr_e msg_sr_i msg_s_e msg_s_i msg_sr_e msg_sr_i self	self

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

INTECRIO block diagrams can include textual *comments*.

name	**toolbar button**	**block diagram icon** (= comment text)	**"Navigation" tab icon**[a]
Comment	ABC	This is a comment.	ABC Comment

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 section 5.1 "Scope Icons" on page 32
- section section 5.2 "Icons for Properties" on page 33
- section section 5.3 "Icons in the "Tree" Pane" on page 34
- section section 5.4 "Icons in the "General" Toolbar" on page 35
- section section 5.5 "Icons in Tab Labels" on page 36

5.1 Scope Icons

The following scopes are available for INTECRIO elements:

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

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

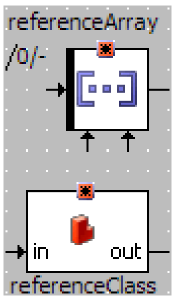
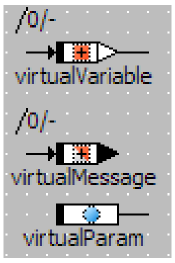
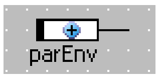
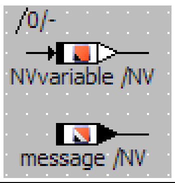
scope	definition	icon
exported	Exported elements are defined in one component and can be accessed by all other components by importing that element.	◪ ◓
imported	Imported elements are defined in another component or project, but can be used in the component that imports them.	◩ ◒
local	Local elements can only be used within the component that defines them, i.e. in all methods or processes of that component.	🟥 🔵
method-/ process-local[a]	Method-/process-local elements can only be used in the method/process that define them.	🟥

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.

property	definition	block diagram icon	Tree pane Icon
reference[a]	Marks a reference type (i.e. composite or complex element) as explicit reference.		🟥
virtual	Virtual variables/parameters are only available in the specification platform, they bear no relevance for code generation.	 /0/- virtualVariable /0/- virtualMessage virtualParam	🟥 ●
dependent[b]	Model parameters can be connected to other system or model parameters via a mathematical dependency.		●
non-volatile[c]	In the ECU, the element is placed in the non-volatile memory.		🟥

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.

location	icon	explanation
"Outline" tab	▣ ▣	*public diagram*, currently not loaded (left) / loaded (right, blue rim)
	▣ ▣	*private diagram*, currently not loaded (left) / loaded (right, blue rim)
	✎	additional marker for the currently loaded diagram
	▣ ▣	*method*, public[a] (left) or private (right)
	◎	*process*[b] (always public)
	#	indicates the process/method most recently used
"Navigation" tab	⊞	root node for the "Graphic Blocks" tree structure
	#	root node for the "Sequence Calls" tree structure

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.

button	icon	remarks
Change filter settings	▼	"Outline" and "Navigation" tab only; can be used to filter the tabs.
Change sort criteria	↓AZ	"Outline" tab only; can be used to sort the tab.
Expand All	⊞	
Collapse All	⊟	
Search the Tree	◎◎	"Outline" and "Navigation" tab only; can be used (together with the "Search" field) to search for a particular item.

5.4 Icons in the "General" Toolbar

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

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

button name	icon	explanation	remarks
Set Zoom to Page		sets the zoom factor so that the entire first page is shown	
Set Zoom to 100%		sets the zoom factor to 100%	
Set Zoom to Fit		sets the zoom factor so that the entire diagram is shown	

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.

location	icon	remarks
"Outline" tab	Outline	
"Navigation" tab	Navigation	
"Database" tab	Database	
"Specification" tab	Specification	In the editor, this tab label is displayed vertically.
"Browse" tab	Browse	In the editor, this tab label is displayed vertically.
"Elements" tab	Elements	
"Data" tab	Data	
"Implementation" tab	Implementation	
"Methods" tab	Methods	
"Layout" tab	Layout	

6 ETAS Contact Addresses

ETAS HQ

ETAS GmbH

Borsigstraße 14	Phone:	+49 711 3423-0
70469 Stuttgart	Fax:	+49 711 3423-2106
Germany	WWW:	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:

ETAS subsidiaries	WWW:	www.etas.com/en/contact.php
ETAS technical support	WWW:	www.etas.com/en/hotlines.php

Index

A
Absolute 12
Addition 8
And 10
Arithmetic Operators 8
Array 20

B
Between 11
Break 14

C
Case 11
Case Operator 10
characteristic line 22
  fixed 23
  group 23
characteristic map 23
  fixed 25
  group 25
comment 31
Comparison Operators 8
complex element 25
composite elements 20–25
  array 20
  characteristic line 22
  characteristic map 23
  distribution 23, 25
  matrix 21
  oneD table 23
  twoD table 25
Constant 19
Continuous Literal 0.0 19
Continuous Literal 1.0 19
Continuous Parameter 17
Continuous Variable 16
control flow elements 13–15
  break 14
  if...then 13
  if...then...else 13
  switch 14
  while 14

D
Distribution 23, 25
Division 8
dT 19
dT Parameter 19

E
elements 16–31
- array 20
- characteristic line 22
- characteristic map 23
- comment 31
- complex ~ 25
- composite 20
- constants 19
- distribution 23, 25
- dT parameter 19
- hierarchy 30
- implementation cast 29
- included component 25
- literals 19
- matrix 21
- messages 17
- method argument 27
- method signature 26
- method-local variable 28
- miscellaneous 29
- oneD table 23
- parameters 17
- process signature 28
- real-time elements 17
- receive message 18
- resource 18
- return value 28
- scalar 16
- self 30
- send & receive message 18
- send message 18
- signature ~ 26
- system constants 19
- twoD table 25
- variables 16

Enumeration Literal 19
Enumeration Parameter 17
Enumeration Variable 16
Equal 9
ETAS Contact Addresses 37

G
General toolbar 35
Greater 9
Greater or Equal 9

H
hierarchy 30

I
If...Then 13
If...Then...Else 13
implementation cast 29
included component 25

L
Literals 19
local variable
- method 28
- process 28

Logic Literal false 19
Logic Literal true 19
Logic Parameter 17
Logic Variable 16
Logical Operators 9

M
Matrix 21
Max 11
Messages 17, 19
method argument 27
method signature elements 26–28
- argument 27
- local variable 28
- return value 28

Min 11
miscellaneous 32–36
- General toolbar 35
- properties 33
- scope 32
- tab label 36
- Tree pane 34

miscellaneous elements 29–31
- comment 31
- hierarchy 30
- implementation cast 29
- self 30

Miscellaneous Operators 11
Modulo 8
Multiplex Operator 10
Multiplication 8
Mux 10

N
Navigation tab 34
Negation 12
Not 10
Not Equal 9

O
OneD Table 23

operators 8–12
  absolute 12
  addition 8
  and 10
  between 11
  case 11
  division 8
  equal 9
  greater 9
  greater or equal 9
  max 11
  min 11
  modulo 8
  multiplication 8
  mux 10
  negation 12
  not 10
  not equal 9
  or 10
  smaller 9
  smaller or equal 9
  subtraction 8
Or 10
Outline tab 34

P
Parameters 17
process signature elements 28
Product liability disclaimer 5
properties icons 33

R
real-time elements 17
  dT parameter 19
  messages 17
  receive message 18
  resource 18
  send & receive message 18
  send message 18
Receive Message 18
Resource 18
return value 28

S
Safety Instructions
  technical state 6
scalar elements 16–20
  constants 19
  continuous literal 0.0 19
  continuous literal 1.0 19
  continuous parameter 17
  continuous variable 16
  dT parameter 19
  enumeration literal 19
  enumeration parameter 17
  enumeration variable 16
  literals 19
  logic literal false 19
  logic literal true 19
  logic parameter 17
  logic variable 16
  messages 17
  parameters 17
  real-time elements 17
  receive message 18
  send & receive message 18
  send message 18
  signed discrete parameter 17
  signed discrete variable 16
  system constants 19
  unsigned discrete parameter 17
  unsigned discrete variable 16
  variables 16
scope icon 32
self 30
Send & Receive Message 18
Send Message 18
signature elements 26–29
  method argument 27
  method-local variable 28
  process-local variable 28
  return value 28
Signed Discrete Parameter 17
Signed Discrete Variable 16
Smaller 9
Smaller or Equal 9
Subtraction 8
Switch 14
System Constant 19

T
tab label 36
Tree pane 34
  buttons 34
TwoD Table 25

U
Unsigned Discrete Parameter 17
Unsigned Discrete Variable 16

V
Variables 16

W
While 14