

# MIKE ECO Lab

Numerical Lab for Ecological and  
Agent Based Modelling

User Guide





## PLEASE NOTE

### **COPYRIGHT**

This document refers to proprietary computer software which is protected by copyright. All rights are reserved. Copying or other reproduction of this manual or the related programs is prohibited without prior written consent of DHI. For details please refer to your 'DHI Software Licence Agreement'.

### **LIMITED LIABILITY**

The liability of DHI is limited as specified in your DHI Software Licence Agreement:

In no event shall DHI or its representatives (agents and suppliers) be liable for any damages whatsoever including, without limitation, special, indirect, incidental or consequential damages or damages for loss of business profits or savings, business interruption, loss of business information or other pecuniary loss arising in connection with the Agreement, e.g. out of Licensee's use of or the inability to use the Software, even if DHI has been advised of the possibility of such damages.

This limitation shall apply to claims of personal injury to the extent permitted by law. Some jurisdictions do not allow the exclusion or limitation of liability for consequential, special, indirect, incidental damages and, accordingly, some portions of these limitations may not apply.

Notwithstanding the above, DHI's total liability (whether in contract, tort, including negligence, or otherwise) under or in connection with the Agreement shall in aggregate during the term not exceed the lesser of EUR 10.000 or the fees paid by Licensee under the Agreement during the 12 months' period previous to the event giving rise to a claim.

Licensee acknowledge that the liability limitations and exclusions set out in the Agreement reflect the allocation of risk negotiated and agreed by the parties and that DHI would not enter into the Agreement without these limitations and exclusions on its liability. These limitations and exclusions will apply notwithstanding any failure of essential purpose of any limited remedy.







# CONTENTS

<b>1</b>	<b>About This Guide</b>	<b>9</b>
1.1	Purpose	9
1.2	Assumed User Background	9
<b>2</b>	<b>Introduction</b>	<b>11</b>
<b>3</b>	<b>Getting Started</b>	<b>13</b>
3.1	Defining a MIKE ECO Lab Template	13
3.1.1	Specifying state variables	14
3.1.2	Specifying constants	16
3.1.3	Specifying forcings	18
3.1.4	Specifying auxiliary variables	20
3.1.5	Specifying processes	23
3.1.6	Specifying derived outputs	26
<b>4</b>	<b>Dialogue Overview</b>	<b>29</b>
4.1	MIKE ECO Lab Setup	29
4.1.1	Common usage of the editor	29
4.1.2	General page	30
4.1.3	Overview page	30
4.2	Miscellaneous	31
4.2.1	Visibility groups	33
4.3	State Variables	36
4.3.1	State variables - details	37
4.4	Constants	39
4.4.1	Constants - details	40
4.5	Forcings	42
4.5.1	Forcings - details	43
4.6	Auxiliaries	45
4.6.1	Auxiliaries - details	46
4.7	Processes	48
4.7.1	Processes - details	49
4.8	Derived Outputs	51
4.8.1	Derived outputs - details	52
4.9	Particle Classes	53
4.9.1	Particle classes - details	54
4.9.2	Particle classes state variables	55
4.9.3	Particle classes constants	58
4.9.4	Restricted area search	61



4.9.5	Arithmetic expressions . . . . .	63
4.9.6	Horizontal movement vectors . . . . .	65
4.9.7	Downward movement . . . . .	67

<b>5</b>	<b>Reference Manual . . . . .</b>	<b>69</b>
5.1	Identifier . . . . .	69
5.1.1	Examples . . . . .	69
5.1.2	Identifier naming scheme . . . . .	69
5.2	Structure/Elements of a MIKE ECO Lab Template . . . . .	70
5.2.1	Eulerian framework . . . . .	71
5.2.2	Lagrangian framework . . . . .	72
5.3	Process Orientated Modelling with MIKE ECO Lab . . . . .	72
5.3.1	State variables . . . . .	73
5.3.2	Constants . . . . .	74
5.3.3	Forcings . . . . .	75
5.3.4	Auxiliary variables . . . . .	76
5.3.5	Processes . . . . .	77
5.3.6	Derived outputs . . . . .	78
5.4	Scopes . . . . .	78
5.5	Spatial Variabilities . . . . .	80
5.6	Expressions . . . . .	80
5.6.1	Control structures . . . . .	80
5.7	Built-in Constants . . . . .	82
5.8	Built-in Forcings . . . . .	83
5.9	Built-in Functions . . . . .	85
5.9.1	Standard mathematical functions . . . . .	85
5.9.2	Trigonometric functions . . . . .	90
5.9.3	Date/time functions . . . . .	94
5.9.4	Random number functions . . . . .	98
5.9.5	Biological/aquatic domain functions . . . . .	101
5.9.6	Special ABM functions . . . . .	107
5.9.7	Special functions to load/save reference values . . . . .	111
5.9.8	Averaging functions for concentration variables . . . . .	113
5.9.9	Table functions . . . . .	114
5.9.10	pH functions . . . . .	115
5.10	Agent Based Modelling with MIKE ECO Lab . . . . .	123
5.10.1	Structure of an ABM in MIKE ECO Lab . . . . .	123
5.10.2	Inside a particle class . . . . .	124
5.10.3	Individual properties . . . . .	129
5.10.4	Spatial representation . . . . .	132
5.10.5	Movement . . . . .	132
5.10.6	Movement projection . . . . .	133
5.10.7	Special built-in ABM functions . . . . .	133
5.10.8	Sensing functions – perception of environment/particles . . . . .	134
5.10.9	Interaction with other elements . . . . .	142
5.10.10	The XML track format . . . . .	147



<b>6</b>	<b>Error Messages and Codes</b> . . . . .	<b>151</b>
	<b>Index</b> . . . . .	<b>155</b>





# 1 About This Guide

## 1.1 Purpose

The main purpose of this User Guide is to get you started in the use of MIKE ECO Lab, for defining new MIKE ECO Lab templates or editing existing templates describing ecosystems. This User Guide is complemented by the Online Help.

## 1.2 Assumed User Background

Although the MIKE ECO Lab interface for developing new MIKE ECO Lab templates has been designed carefully with emphasis on a logical and user-friendly interface without compromising on flexibility and functionality, and although the User Guide and Online Help contain modelling procedures and reference material, basic knowledge about ecological modelling and ecosystems is always needed in any practical application. All knowledge about ecosystems is found in the MIKE ECO Lab templates and the model developers head. The MIKE ECO Lab software itself has no knowledge of any ecosystem before loading or creating a new template, and it is possible for the model developer to make templates that can be calculated, but make no sense. The interface is designed to catch some of these descriptions, which from a logical point of view cannot take place, but the reliability of the new MIKE ECO Lab templates, which are created by the model developer, is the model developer's own responsibility.

In this case, 'basic knowledge about ecological modelling and ecosystems' means a background in environmental science problems, which is sufficient for you to be able to know what processes take place and are important in the ecosystem that MIKE ECO Lab should describe. This User Guide is not intended as a substitute for a basic knowledge of the area in which you are working: ecological modelling.





## 2 Introduction

MIKE ECO Lab is a numerical lab for Ecological Modelling. It is an open and generic tool for customising aquatic ecosystem models to describe water quality, eutrophication, heavy metals and ecology. The module is mostly used for modelling water quality as part of an Environmental Impact Assessment (EIA) of different human activities, but the tool is also applied in aquaculture, e.g. for optimisation the production of fish, seagrasses and mussels. Another application area is in online forecasts of water quality.

The demand for tailor made ecosystem descriptions is great, because ecosystems vary. The strength of this tool is the easy modification and implementation of mathematical descriptions of ecosystems into the hydrodynamic engines of DHI.

The user can use predefined MIKE ECO Lab templates containing the mathematical descriptions of ecosystems or can choose to develop own model templates. A template is independent of grid systems and the same template can be loaded in MIKE HYDRO, MIKE 11, MIKE 21, MIKE 3, MIKE 21 FM, MIKE 3 FM and Coupled Model FM. The template can describe dissolved substances, particulate matter of dead or living material, living biological organisms and other components (all referred to as state variables in this context).

The module is developed to describe chemical, biological, ecological processes and interactions between state variables and also the physical process of sedimentation of components can be described. State variables included in MIKE ECO Lab can either be transported by advection-dispersion processes based on hydrodynamics, or have a more fixed nature (e.g. rooted vegetation or mussels).







## 3 Getting Started

The user has the option of

1. Using predefined DHI supported templates
2. Creating own templates
3. Modifying existing templates

If the user chooses to use the predefined DHI supported templates directly, it is not necessary to read this User Guide, but instead the User Guide for MIKE 11/21/3 ECO Lab can be relevant for information about running simulations with MIKE ECO Lab.

This manual is about creating new templates and editing existing templates.

If the user wishes to create a new template: Open MIKE Zero and choose: File | New | MIKE ECO Lab.

If the user wishes to modify an existing MIKE ECO Lab template, simply click on the file in the File Manager and the MIKE ECO Lab user interface will open the chosen file. The MIKE Zero installation program has copied examples of MIKE ECO Lab templates to your hard disk (the templates has the file extension \*.ecolab): see under the path where MIKE Zero was installed (.\\DHI\\2017\\MIKE Zero\\Templates\\ECOLab\\\*.ecolab).

These files can be used as they are or be modified. They can also just be used for inspiration.

### 3.1 Defining a MIKE ECO Lab Template

A MIKE ECO Lab template contains a mathematical description or a set of ordinary differential equations of an ecosystem including the processes affecting the ecosystem. In MIKE ECO Lab this mathematical description is divided into 7 types of components, which can be used to describe the system. The component types are:

1. State variables
2. Constants
3. Forcings
4. Auxiliary Variables
5. Processes
6. Derived Output
7. Particle classes / Agent based models

### 3.1.1 Specifying state variables

Step one for the model developer in the development process of the model is to decide which state variables that are necessary to include in the model for describing a certain ecosystem. State variables represent those variables that describe the state of the ecosystem and that the user wants to predict the state of.

Thus, for example, if the model developer wants to make a model of Biological Oxygen Demand (BOD) and Dissolved Oxygen (DO) in the water column it would be necessary to specify BOD and DO as state variables in the MIKE ECO Lab user interface.

To create a new state variable in MIKE ECO Lab use the context menu (right-mouse button) on the structure-tree in the left navigation panel or the 'new item' icon in the individual list views on the right panel.

A dialogue as shown in Figure 3.1 will appear.

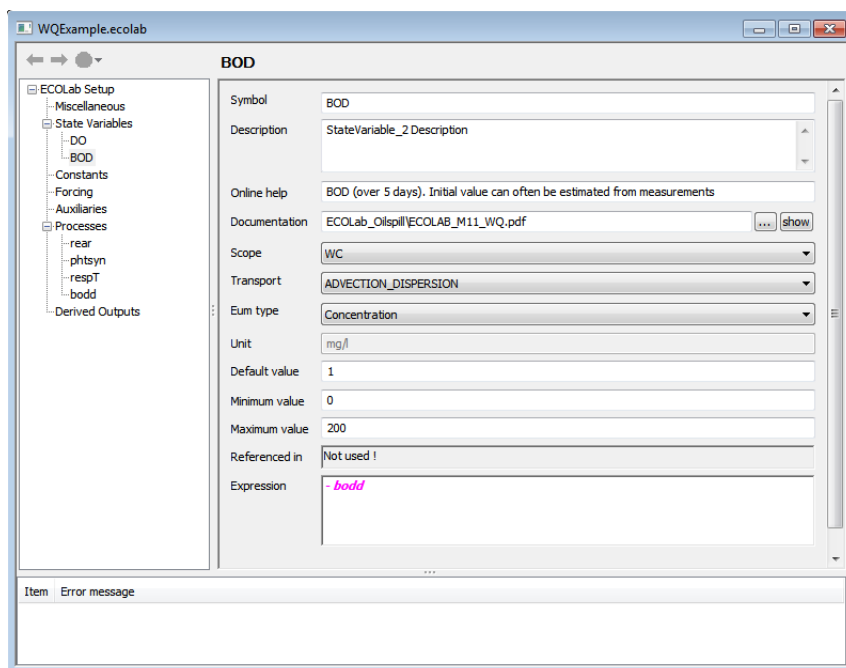


Figure 3.1 Creating a new State Variable in the MIKE ECO Lab dialogue

The user should specify different characteristics about the state variable (cf. Section 5.1 Identifier).



The name should not be too long, as the name specified in this box will be used later when referring to this state variable in user specified expressions. If the names are short, the expressions become shorter and easier to read.



The user can specify a longer description than the above name in the 'Description' attribute. The text written in this box will also be printed in the result files of the MIKE 11/21/3 ECO Lab simulations. The text will also appear as 'fly by' text in the menus for the MIKE 11/21/3 ECO Lab.

The model developer has the option of publishing further information or documentation of the state variable by specifying an address or a path to a website or document in the box called 'Documentation'. By choosing 'Show' a browser will show the documentation.

An optional short text (256 characters) can also be supplied in the box called 'Online help'.

Under the attribute 'Scope' it has to be specified where in the water environment the state variable can be found. There are 4 options:

1. WC Water Column
2. WS Water Surface
3. WB Water Bed
4. SED Sediment

The specification in this attribute is most important in multilayered grid systems like MIKE 3, because if a state variable is specified with scope WB, WS or SED, this state variable will in the 3D model only be a 2D state variable, but if specified with scope WC, the 3D model will reserve memory for a 3D state variable. The attribute is also introduced to avoid MIKE ECO Lab models introducing impossible descriptions. For example, it is not possible to have a process interacting with the atmosphere in the sediment.

Under the 'Transport' attribute it has to be chosen if the state variable should be transported by the advection-dispersion module of the MIKE System. Examples of this kind of state variable are usually found in the pelagic and could for instance be phytoplankton. The alternative is to have a state variable that has a fixed location. That would typically be different seagrasses, or mussels that both have a stationary nature.

There are 3 unit types available for state variables in MIKE ECO Lab:

1. Concentrations, with default unit mg/l, the unit is compatible with the EUM system
2. Mass per area, with default unit g/m<sup>2</sup>, the unit is compatible with the EUM system
3. Undefined, with a user specified text string as unit

The user must specify a default value for the state variable. The value should be the expected value for the state variable. The specified default value is suggested as initial conditions in the MIKE 11/21/3 ECO Lab.



The user must also specify a realistic range in which the state variable can be found. If the user specifies initial conditions in the MIKE 11/21/3 ECO Lab that are outside this interval the user will be warned.

State variables defined in the MIKE ECO Lab model are always available for output in the M11/21/3 ECO Lab menus.

The Expression box contains the ordinary differential equation for the state variable. This differential equation summarises the processes that influence the state variable.

The only legal arguments in the state variable expression are:

1. Process names
2. Stoichiometric numbers that can be multiplied in the processes

### 3.1.2 Specifying constants

Constants are used as arguments in the mathematical expressions of processes in a MIKE ECO Lab model. They are constant in time, but can vary in space. Typical examples of constants are different specific rate coefficients, exponents, half saturation concentrations and also universal constants such as gas constant and atomic weights.

Thus, for example, if the model developer wants to make a model of BOD and DO in the water column it could be relevant to specify a process for BOD degradation. This process could be described by an 1st order degradation expression that uses a specific rate for degradation. This specific rate could be described as a Constant in MIKE ECO Lab.

To create a new constant in MIKE ECO Lab use the context menu (right-mouse button) on the structure-tree in the left navigation panel or the 'new item' icon in the individual list views on the right panel.

A dialogue as shown in Figure 3.2 will appear.

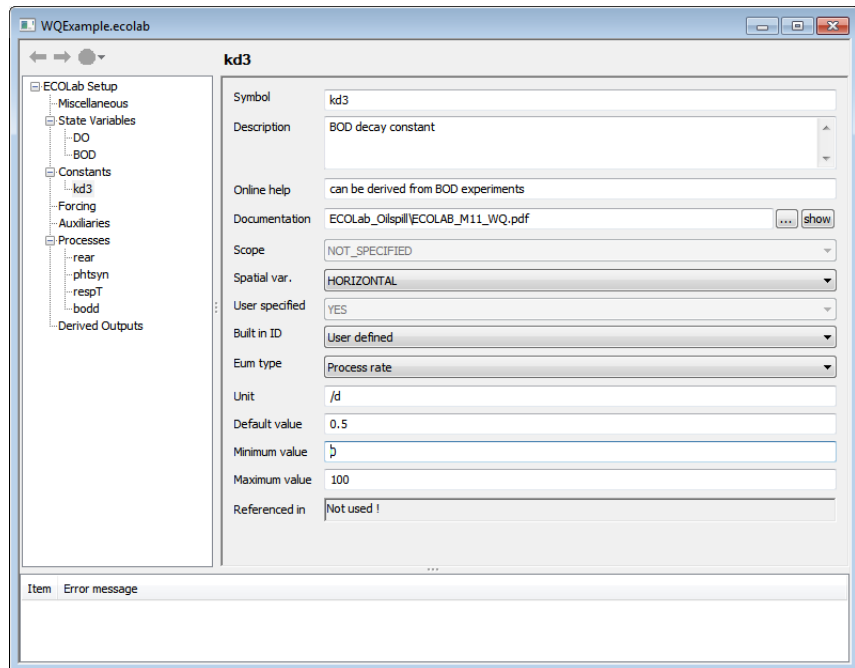


Figure 3.2 Creating a new Constant in the MIKE ECO Lab dialogue. In this case a 1st order decay rate for BOD

The user should specify different characteristics about the included constants.

The 'Constant name' attribute should be filled with a short describing name.



The name should not be too long because the name specified in this box is used later when referring to this constant in user specified expressions. If the names are short, the expressions become shorter and easier to read.

The user can specify a longer description than the above name in the 'Description' attribute. The text will appear as 'fly by' text in the menus for the MIKE 11/21/3 ECO Lab.

The model developer has the option of publishing further information or documentation of the constant by specifying an address or path to a website or document in the box called 'Documentation'. By choosing 'Show' a browser will show the documentation.

An optional short text (256 characters) can also be supplied in the box called 'Online help'.

Under the attribute 'Spatial Variation' it has to be specified which variation pattern the constant has. There are 3 options:



1. None: Global, with same value in all grid points
2. Horizontal: Possibly different values in each horizontal grid point, but without variation in water column
3. Horizontal and Vertical: Possible different values in all grid points. Relevant for multilayered grid systems

The attribute is introduced to avoid wasting computer memory, so that for instance a global constant only allocates memory for one number.

The user has the option of using built-in constants that are specified elsewhere in the MIKE system. The built-in constants are available, when choosing the constant as not being user specified. See Built-in Constants in Section 5.7.

There are 2 unit types available for constants in MIKE ECO Lab:

1. 1st order rate, with default unit: per day, the unit is compatible with the EUM system
2. Undefined, with a user specified text string as unit

The user must specify a default value for the constant. The value should be the expected value for the constant. The specified default value is also suggested as the default value for the constant in the MIKE 11/21/3 ECO Lab menus.

The user can also specify a realistic range if available where the constants are usually found. If the user specifies constants in the MIKE 11/21/3 ECO Lab menus that are outside this interval, the user will be warned.

### 3.1.3 Specifying forcings

Forcings are used as arguments in the mathematical expressions of processes in a MIKE ECO Lab model. They can vary in time and space. They represent variables of an external nature that affect the ecosystem. Typical examples of forcings are temperature, solar radiation and wind.

Thus, for example, if the model developer wants to make a model of BOD and DO in the water column it could be relevant to specify a process for BOD degradation. This process could be described by an expression dependent on temperature. This temperature could be described as a Forcing in MIKE ECO Lab.

To create a new forcing in MIKE ECO Lab use the context menu (right-mouse button) on the structure-tree in the left navigation panel or the 'new item' icon in the individual list views on the right panel.

A dialogue as shown in Figure 3.3 will appear.



Item	Error message

Figure 3.3 Creating a new Forcing in the MIKE ECO Lab dialogue. In this case a built-in forcing for temperature

The user should specify different characteristics about the included forcing.

The 'Forcing name' attribute should be filled with a short describing name.



The name should not be too long because the name specified in this box is used later when referring to this forcing in user specified expressions. If the names are short, the expressions become shorter and easier to read.

The user can specify a longer description than the above name in the 'Description' attribute. The text will appear as 'fly by' text in the menus for the MIKE 11/21/3 ECO Lab.

The model developer has the option of publishing further information or documentation of the forcing by specifying an address or a path to a homepage or document in the box called 'Documentation'. By choosing 'Show' a browser will show the documentation.

An optional short text can also be supplied in the box called 'Online help'.

Under the attribute 'Spatial Variation' it has to be specified which variation pattern the forcing has. There are 3 options:

1. None: Global, with same value in all grid points
2. Horizontal: Possibly different values in each horizontal grid point, but without variation in water column

3. Horizontal and Vertical: Possible different values in all grid points. Relevant for multilayered grid systems

The attribute is introduced to avoid wasting computer memory, so that a global forcing only allocates memory for one number.

The user has the option of using built-in forcings that are specified or calculated elsewhere in the MIKE system (cf. Section 5.8). The built-in forcings are available when choosing the forcing as not being user specified.

There are 2 unit types available for forcings in MIKE ECO Lab:

1. 1st order rate, with default unit: per day, the unit is compatible with the EUM system
2. Undefined, with a user specified text string as unit

The user must specify a default value for the forcing. The value should be the expected value for the forcing. The specified default value is also suggested as the default value for the forcing in the MIKE 11/21/3 ECO Lab menus.

The user can also specify a realistic range if available where the forcings are usually found. If the user specifies forcings in the MIKE 11/21/3 ECO Lab menus that are outside this interval, the user will be warned.

### 3.1.4 Specifying auxiliary variables

Auxiliary variables are mostly arguments in mathematical expressions' processes in a MIKE ECO Lab model, but sometimes they are only used for specifying results directly.

Thus, for example, if the model developer wants to make a model of BOD and DO in the water column it could be relevant to specify a process for BOD degradation. This process could be described by an expression dependent on temperature. This temperature could be described as a Forcing in MIKE ECO Lab.

To create a new auxiliary variable in MIKE ECO Lab use the context menu (right-mouse button) on the structure-tree in the left navigation panel or the 'new item' icon in the individual list views on the right panel.

A dialogue as shown in Figure 3.4 will appear.



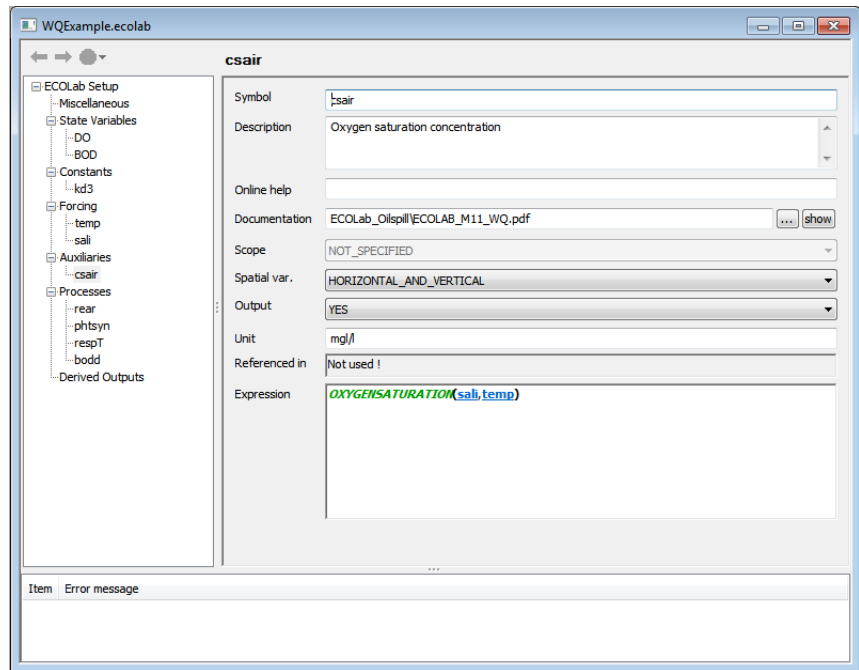


Figure 3.4 Creating a new Auxiliary Variable in the MIKE ECO Lab dialogue. A list of possible arguments including the built-in functions can be found in the manual/online help.

The user should specify different characteristics about the auxiliary variable.

The 'Variable name' attribute should be filled with a short describing name.



The name should not be too long, because the name specified in this box is used later when referring to this auxiliary variable in user specified expressions. If the names are short, the expressions become shorter and easier to read.

The user can specify a longer description than the above name in the description attribute. The text written in this box will also be printed in the result files of the MIKE 11/21/3 ECO Lab simulations. The text will also appear as 'fly by' text in the menus for the MIKE 11/21/3 ECO Lab.

The model developer has the option of publishing further information or documentation of the auxiliary variable by specifying an address or a path to a website or document in the box called 'Documentation'. By choosing 'Show' a browser will show the documentation.

An optional short text (256 characters) can also be supplied in the box called 'Online help'. This text is at the moment only used as reference and record of the MIKE ECO Lab model.



It can be specified in the 'Output' box that the auxiliary variable should be available for output in the M11/21/3 ECO Lab model menus. If 'YES' is chosen the user will be prompted in the M11/21/3 ECO Lab menus and asked if the auxiliary variable should be included in the result file.

Under the attribute 'Spatial Variation' it has to be specified which variation pattern the auxiliary variable has.

There are 3 options:

1. None: Global, with same value in all grid points
2. Horizontal: Possibly different values in each horizontal grid point, but without variation in water column
3. Horizontal and Vertical: Possible different values in all grid points. Relevant for multilayered grid systems

The attribute is introduced to avoid wasting computer memory and redundant calculations, so that for instance a global auxiliary variable only allocates memory for one number and is only calculated once per time step.

It is possible to specify user specified text as unit description for auxiliary variables in MIKE ECO Lab.

The Expression box should be filled with the mathematical formulation of the auxiliary variable.

The expression is formulated by a series of arguments, State Variables, Constants, Forcings, Auxiliary variables separated by operators.

The following argument types are possible to use in auxiliary variable expressions:

1. Numbers
2. Already defined names of state variables, constants, forcings, and auxiliary variables
3. Mathematical functions
4. Built-in functions
5. Logical functions
6. Reserved words

The following operators are possible to use in expressions:

1. Addition: +
2. Subtraction: -
3. Multiplication: \*



#### 4. Division: /

In addition to the above-mentioned, the expressions must follow a set of semantic rules for how the arguments can be combined. Of course, the expressions must also follow the syntax rules. See *Reference Manual for MIKE ECO Lab* for details about these rules.

The order in which the expressions in auxiliary variables are defined is also the order in which they will be calculated. Arguments in an expression defined after the expression itself is therefore not legal. Therefore it can often be necessary to change the order in which the variables are defined when creating a template.

### 3.1.5 Specifying processes

Processes describe the transformations that affect the state variables. That means processes are used as arguments in the differential equations that MIKE ECO Lab solves to determine the state of the state variables. These differential equations, which describe the interaction between state variables and processes are specified under 'State Variables'.

To create a new process in MIKE ECO Lab use the context menu (right-mouse button) on the structure-tree in the left navigation panel or the 'new item' icon in the individual list views on the right panel.

A dialogue as shown in Figure 3.5 will appear.

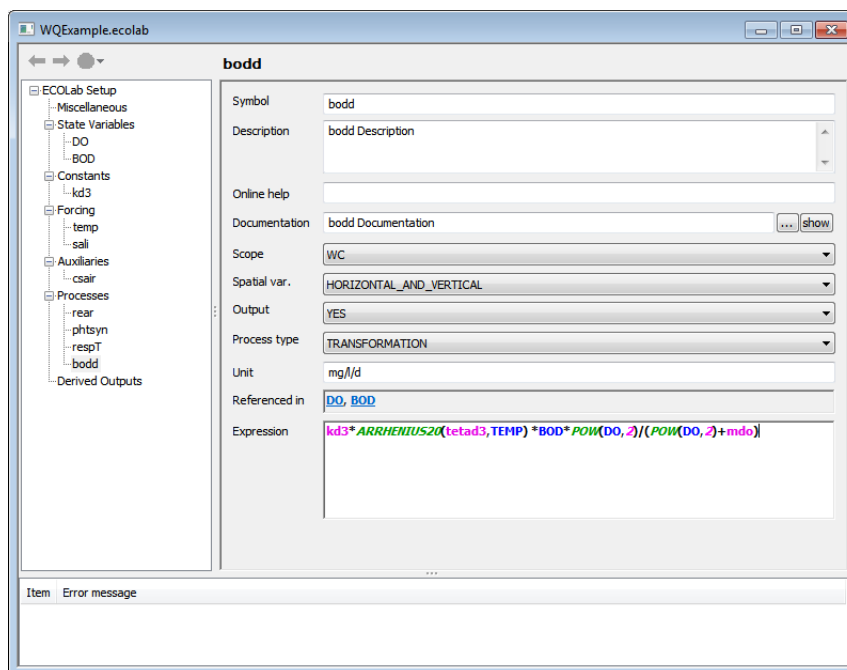


Figure 3.5 Creating a new Process in the MIKE ECO Lab dialogue. In this case it is a process for BOD degradation. Note that a built-in function (ARRHENIUS20) is used in the expression box along with other arguments.

The user should specify different characteristics about the process.

The 'Process name' attribute should be filled with a short describing name.



The name should not be too long, because the name specified in this box is used later when referring to this process in user specified expressions. If the names are short the expressions become shorter and easier to read.

The user can specify a longer description than the above name in the description attribute. The text written in this box will also be printed in the result files of the MIKE 11/21/3 ECO Lab simulations. The text will also appear as 'fly by' text in the menus for the MIKE 11/21/3 ECO Lab menus.

The model developer has the option of publishing further information or documentation of the process by specifying an address or path to a website or document in the box called 'Documentation'. By choosing 'Show' a browser will show the documentation.

An optional short text can also be supplied in the box called 'Online help'.

It can be specified in the 'Output' box that the process should be available for output in the M11/21/3 ECO Lab model menus. If 'YES' is chosen the user will be prompted in the M11/21/3 ECO Lab menus and asked if the process should be included in the result file.



Under the attribute 'Spatial variation' it has to be specified which variation pattern the process has. There are 3 options:

1. None: Global, with same value in all grid points
2. Horizontal: Possibly different values in each horizontal grid point, but without variation in water column
3. Horizontal and Vertical: Possibly different values in all grid points. Relevant for multilayered grid systems

The attribute is introduced to avoid wasting computer memory and redundant calculations, so that for instance a global process only allocates memory for one number and is only calculated once per time step.

It is possible to specify user specified text as unit description for processes in the MIKE ECO Lab model.

The Expression box should be filled with the mathematical formulation of the process.

The expression is formulated by a series of arguments, mathematical functions, built-in functions, logical functions and reserved words separated by operators.

The following arguments are possible to use in process expressions:

1. Numbers
2. Already defined names of state variables, constants, forcings, auxiliary variables and Processes
3. Mathematical functions
4. Built-in functions
5. Logical functions
6. Reserved words

The following operators are possible to use in expressions:

1. Addition: +
2. Subtraction: -
3. Multiplication: \*
4. Division: /

In addition to the above-mentioned, the expressions must follow a set of semantic rules for how the arguments can be combined. Of course, the expressions must also follow the defined syntax rules. See *Reference Manual for MIKE ECO Lab* for details about these rules.

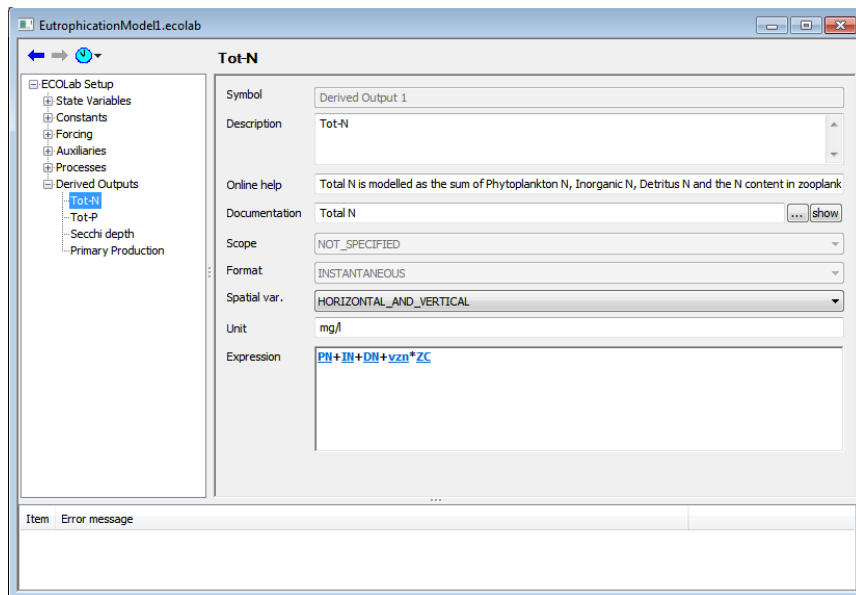
The order in which the expressions in processes are defined is also the order in which they will be calculated. Arguments in an expression defined after the expression itself is therefore not legal. Therefore it can often be necessary to change the order in which the variables are defined when creating a template.

### 3.1.6 Specifying derived outputs

Derived outputs are outputs that as the name indicates are derived based on other model results. An example could be an estimate of Total N, which summarises the state variables containing nitrogen. The difference between calculating Total N under 'Derived Outputs' and 'Auxiliary Variables' is that if state variables are included in the expressions, the values will be based on values from the previous time step in the auxiliary variables, but in derived output the state variables will have updated values for present time step. That is because the calculation of the derived output is done after the differential equation has been integrated.

To create a new derived output in MIKE ECO Lab use the context menu (right-mouse button) on the structure-tree in the left navigation panel or the 'new item' icon in the individual list views on the right panel.

A dialogue as shown in Figure 3.6 will appear.



**Figure 3.6** Creating a new Derived Output in the MIKE ECO Lab dialogue. In this case it is a derived output for Total N that summarises the nitrogen containing state variables in the water column

The user should specify different characteristics about the derived output.



Derived outputs do not have a name attribute, so derived outputs cannot be referred to in other expressions.

The user can specify a longer description in the 'Description' attribute. The text written in this box will be printed in the result files of the MIKE 11/21/3 ECO Lab simulations. The text will also appear as 'fly by' text in the menus for the MIKE 11/21/3 ECO Lab.

The model developer has the option of publishing further information or documentation of the derived output specifying an address or path to a website or document in the box called 'Documentation'. By choosing 'Show' a browser will show the documentation.

An optional short text can also be supplied in the box called 'Online help'. This text is at the moment only used as reference and record of the MIKE ECO Lab model.

Under the attribute 'Spatial Variation' the variation pattern of the derived output must be specified. There are three options:

1. None: Global, with same value in all grid points
2. Horizontal: Possibly different values in each horizontal grid point, but without variation in water column
3. Horizontal and Vertical: Possible different values in all grid points. Relevant for multilayered grid systems

The attribute is introduced to avoid wasting computer memory and redundant calculations, so that for instance a global derived output only allocates memory for one number and is only calculated once per time step.

If derived outputs are defined, they will always be available in the M11/21/3 ECO Lab menus, when specifying the result file.

It is possible to specify user specified text as unit description for a derived output in MIKE ECO Lab.

The expression box should be filled with the mathematical formulation of the derived output.

The expression is formulated by a series of arguments, mathematical functions, built-in functions, logical functions and reserved words separated by operators.

The following arguments are possible to use in expressions:

1. Numbers
2. Already defined names of state variables, constants, forcings, auxiliary variables and processes. Note that the values for the state variables will be based on values after integration has taken place



3. Mathematical functions
4. Built-in Functions
5. Logical functions
6. Reserved words

The following operators are possible to use in expressions:

1. Addition: +
2. Subtraction: -
3. Multiplication: \*
4. Division: /

In addition to the above-mentioned, the expressions must follow a set of semantic rules for how the arguments can be combined. Of course, the expressions must also follow the defined syntax rules. See *Reference Manual for MIKE ECO Lab* for details about these rules.





## 4 Dialogue Overview

### 4.1 MIKE ECO Lab Setup

#### 4.1.1 Common usage of the editor

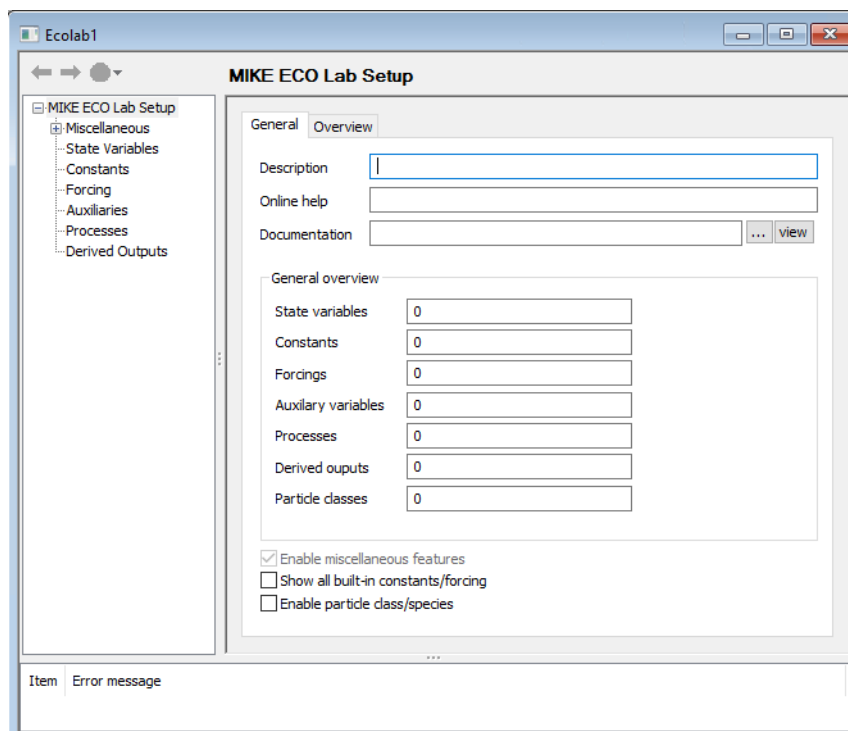


Figure 4.1 General page MIKE ECO Lab Setup

The 'General' page lists settings describing common properties of the current template. The user can supply a description, an online help and documentation. When creating a new template and supplying a valid link to an external file for the documentation, this link will be used as default for all subsequent created elements. Further the page lists as information the number of all currently defined elements in a table like structure.

The left panel shows the current MIKE ECO Lab template as a tree like structure, representing all defined items (state variables, constants etc.). See Section 5.2 for more details. The right panel is used for detail information and user interaction. On the right side there is a list of all items of the current selected category (i.e. all defined state variables etc.) or details for the current selected item. The lower panel is used for displaying messages (errors, warnings and hints). A double click on one of the messages will open the details page of that item (if available).

There is a chronic-navigation toolbar on the upper left top. You can use this toolbar to move back and for between already visited items.

To add new items use the context menu (right-mouse button) on the structure-tree in the left navigation panel or the 'new item' icon in the individual list views on the right panel.

### 4.1.2 General page

On this page the number of current defined items in each category is listed and you can supply some general information.

The checkbox 'Enable miscellaneous features' is automatically set for new created templates. Templates created by the MIKE ECO Lab editor from releases prior to MIKE by DHI 2012 and the referring part of the template is not available (cf. Section 4.2).

With the checkbox 'Enable particle class' the possibility to use ABM modelling is added to the MIKE ECO Lab template. By default only the standard process orientated MIKE ECO Lab modelling is enabled.

### 4.1.3 Overview page

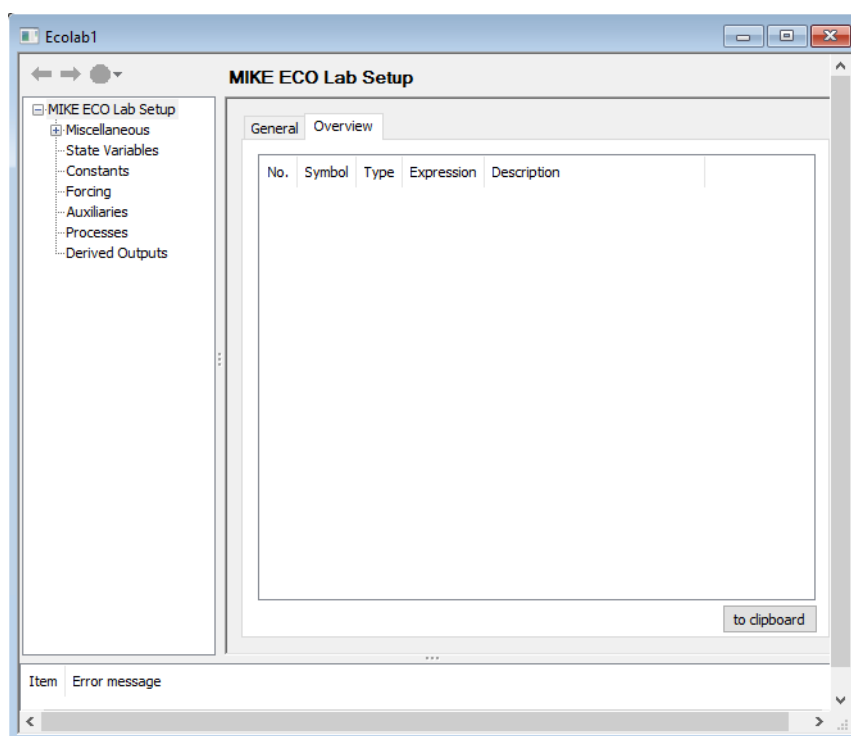


Figure 4.2 Overview page MIKE ECO Lab Setup



This overview lists all defined elements within the template. It shows a condensed overview citing the symbol name, type and expression. Depending on the type either the default value or the current expression is listed along with the description text. You can navigate to an item definition by a double-click.

## 4.2 Miscellaneous

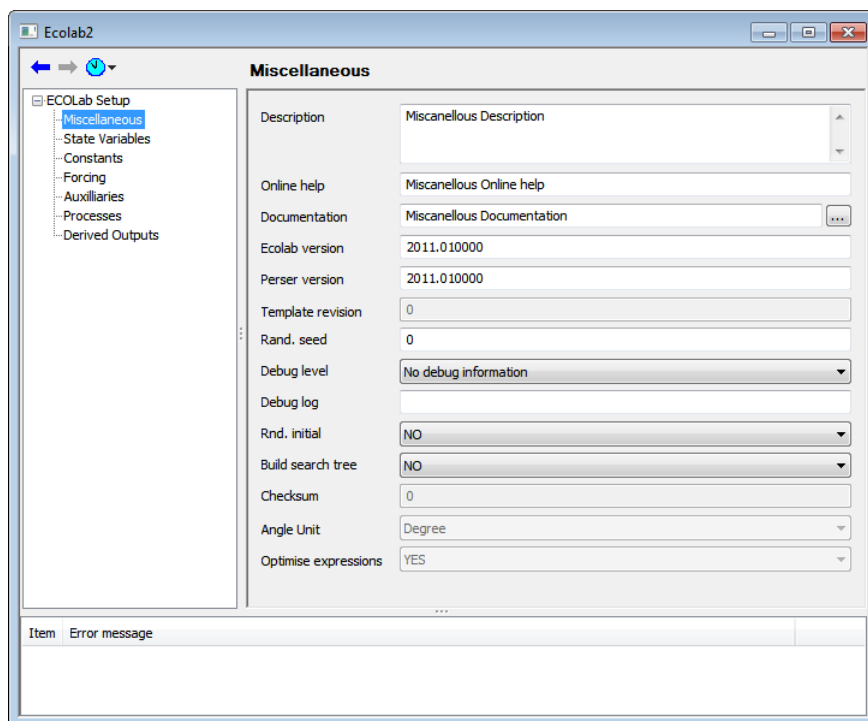


Figure 4.3 Miscellaneous page

This page is only available when a template has been created with MIKE by DHI Release 2012 or later. It can be used to adjust several fields influencing the MIKE ECO Lab computational core.

Below some generic fields are listed, which can be used to specify additional information:

### MIKE ECO Lab version

Specifies the version of the MIKE ECO Lab core. The version string is decoded as 'major release.minor release level'. Some features may not be available in all MIKE ECO Lab versions/releases and a warning message is printed in the log file when a version mismatch is detected.

### Parser version

Specifies the version of the MIKE ECO Lab parser. The version string is decoded as 'major release.minor release level'. Some features may not be available in all MIKE ECO Lab versions/releases and a warning message is printed in the log file when a version mismatch is detected.

### Template revision

This field is automatically updated whenever a template is saved, the user can not change this value manually. If the template revision could not be detected the value '-1' is shown. The template revision is printed in the logfile and can be used for quality control issues.

### RAND seed

This element controls the state of the pseudo random generator. It defines the so-called seed value (cf. Section 5.9.4). If you specify a seed value different from zero a reproducible sequence of random numbers is generated. This is especially useful during template development and debugging of ABM templates. For a production run a seed value of '0' should be used. This will seed the pseudo random generator with the current time and lead to an un-reproducible sequence of random numbers. A message on the seed status is printed in the log file.

### Debug level

MIKE ECO Lab can print various debug information. Normally no debug information is logged ('no debug information') but the verbosity can be set from 'Debug level 1' (few debug messages) to 'Debug level 6' (comprehensive debug information). Be aware that higher debug levels will produce a large amount of debug information.

### Debug log

If debug information is generated it can be saved to an additional file in the setup directory. the debug information may also be found in the simulation log file.

### ABM random bearing

This option is only active when particle classes are defined. If set to 'YES' the new released particles will have a random bearing/heading. When set to 'NO' newly released particles will look northwards (  $0^\circ$  ).

### ABM search tree

This option is only active when particle classes are defined. It is only relevant if area search functions on lagrange particles are used (cf. Section 5.10.8). If set to 'YES' each time step a spatial search tree constructed to query the particle neighbourhood. Depending on the setup this can greatly enhance the computational speed also there is a time overhead for the construction of the search tree.

### Checksum

Internal value, no user input possible.



### Angle unit

With this option the default angle unit for trigonometric functions is set. In MIKE ECO Lab versions prior to MIKE by DHI Release 2012 trigonometric functions in MIKE ECO Lab were using radians as angle unit but flow directions were always given in degree. Since Release 2012 the default angle unit for trigonometric functions is also 'degree' for new created templates.

### Optimise expressions

MIKE ECO Lab can optimise expressions by pre-computing some constant expressions parts. Depending on the expression complexity of this can speed up the computation.

## 4.2.1 Visibility groups

Since the 2017 release MIKE ECO Lab offers the possibility to assign the elements of a MIKE ECO Lab template (State variables, constants etc.) to user defined visibility groups and levels inside these groups. This allows the user to work with a selected sub-set of variables, i.e. simplifies the design and use of complex templates. Both the editor as well as selected application modules (MIKE FM interface) can use this grouping and visibility information to simplify the work with selected set of elements.

A visibility group should encompass all elements that are relevant for a functional/thematic group, i.e. a sub-model inside the template, e.g. "Phytoplankton", "Respiration" or "Bed process". Inside each group levels can be used to define a further hierarchy. Here the first level defines the most basic/general elements whereas higher levels define higher hierarchic orders (see Figure 4.4). If needed, the levels can be "expanded" when working with them, i.e. include all elements from lower levels as well. A possible use would be to include all fixed/stoichiometric constants in level 1, rarely changing constants in level 2 and always modified constants in level 3. As it is sometimes difficult to assign one element to exactly one level or group, each MIKE ECO Lab element can be assigned to multiple groups or levels inside the same group.

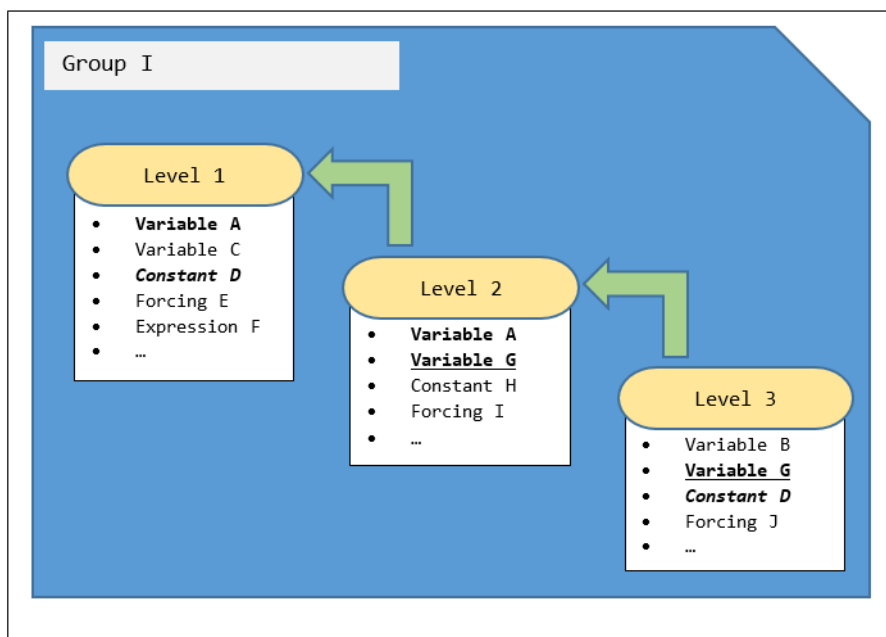


Figure 4.4 Schematic organisation of visibility groups and hierarchy levels. Elements occurring in more than one level are marked bold.

To add or remove a visibility group use the context menu or buttons of the group overview table. To add, remove or define visibility levels use the group overview details page (see Figure 4.5). The group overview is divided in the upper generic group name/description edit fields, a level table (listing all defined levels) and the element assignment tables at the bottom. A newly created group does not have any levels. Before you can assign any MIKE ECO Lab element to the group, you have to add appropriate levels to that group. Please note that it is not possible to move/resort levels or groups but you can add new and remove existing groups and levels.

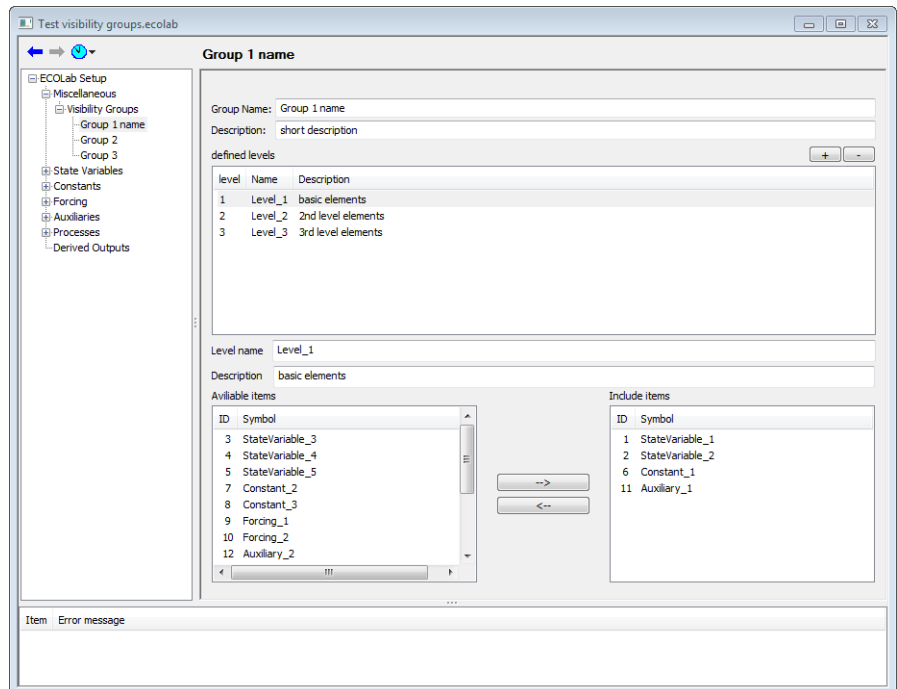


Figure 4.5 Example for defining a visibility group with three hierarchical levels

### Group name

Generic name for the current active group

### Group description

Short description for the current active group

### Defined levels

This table lists all defined hierarchy levels. To add a new or remove the selected level press the +/- button on the right top

### Level Name

Generic name for the current active/selected level

### Level Description

Short description for the current active/selected level

### Available variables

On the left side all available MIKE ECO Lab variables are listed whereas the right table shows all MIKE ECO Lab variables that are members of the current selected level. To move a variable from one side to the other use the arrow buttons.

## 4.3 State Variables

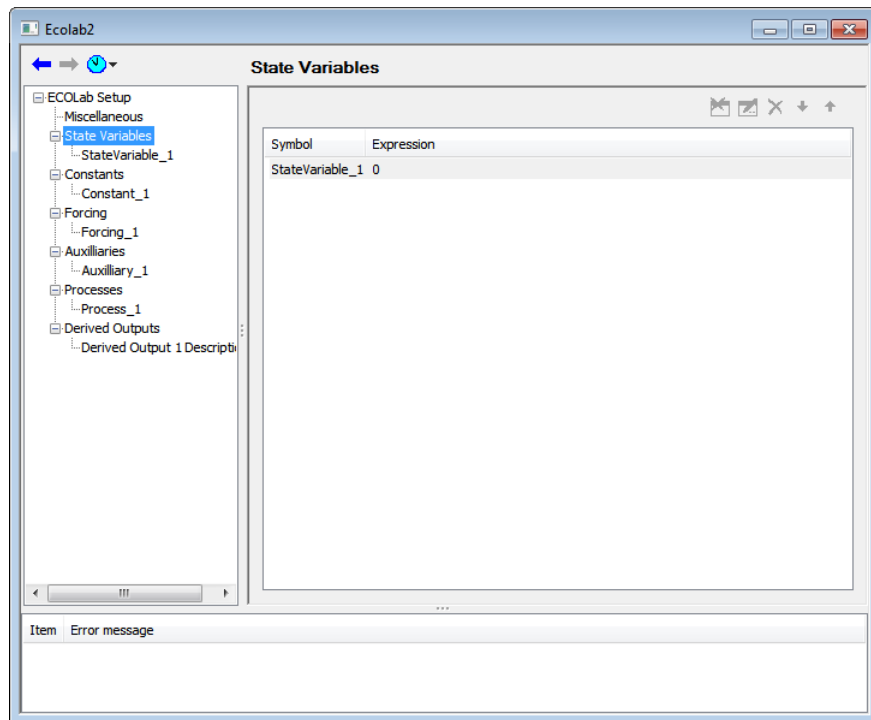


Figure 4.6 State Variables - overview

This overview lists all defined state variables and their expression. You can navigate to the details page by a double-click or the 'edit-' icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon. If the current selected state variable is not referenced the 'delete' icon is active and the item can be removed. The up/down arrows can be used to change the item and thus computation order.





### 4.3.1 State variables - details

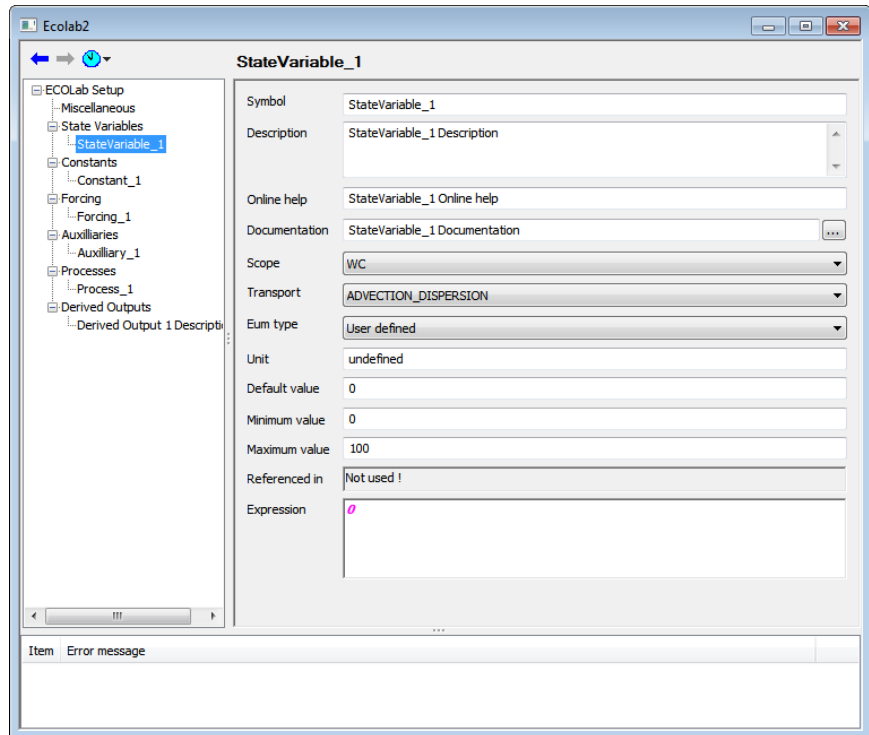


Figure 4.7 State Variables - properties

This page shows the properties of a state variable. A description of each is listed below:

#### Symbol

This is the *IDENTIFIER* used to address the concentration/value represented by the current state variable in the remaining parts of the template. For practical reasons identifiers should be kept short. The user can also use any valid identifiers. DHI suggests to follow the Identifier naming scheme in Section 5.1.2.

#### Description

This is short text used as annotation in the results and the setup interface.

#### Online help

The content of this field will be displayed as 'mouse-over' tooltip in the setup.

#### Documentation

A link or textual description for a more detailed description.

### Scope

Definition of the *SCOPE* describing the principal vertical domain of the current item.

### Transport

State variables can be transported with the water flow or be stationary (not moving). Not moving state variables usually represent elements like benthic biomass etc.

### EUM type

Specify the EUM used for output of this item. If one predefined EUM type is selected the appropriate unit is automatically chosen. If you need an not available output type set to 'undefined' and specify an output unit manually.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Unit

Specify the output unit.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Default value

This value will be used as default initialisation/boundary value in the setup.

### Minimum value

Minimum allowed value for this item as input data. Depending on the hydraulic engine a mass balance error is noted when the item value would fall below this value. Consult the manual (AD-transport) of the selected hydrodynamic mode for details. Depending on the debug level an undershooting will be noted in the log file.

### Maximum value

Maximum allowed value for this item as input data. Depending on the hydraulic engine a mass balance error is noted when the item value would grow above this value. Consult the manual (AD-transport) of the selected hydrodynamic mode for details. Depending on the debug level an overshooting will be noted in the log file

### Referenced in

Here all MIKE ECO Lab items are listed that refer to the current element. The listed items work as hyperlinks.

### Expression

This expression is the differential equation describing the change rate of an state variable. It is the right-hand side of the general equations:

$$\frac{dSymbol}{dt} = process1 + process2 + \dots$$



Only process and numeric values are allowed. See Section 5.6 for more details.

## 4.4 Constants

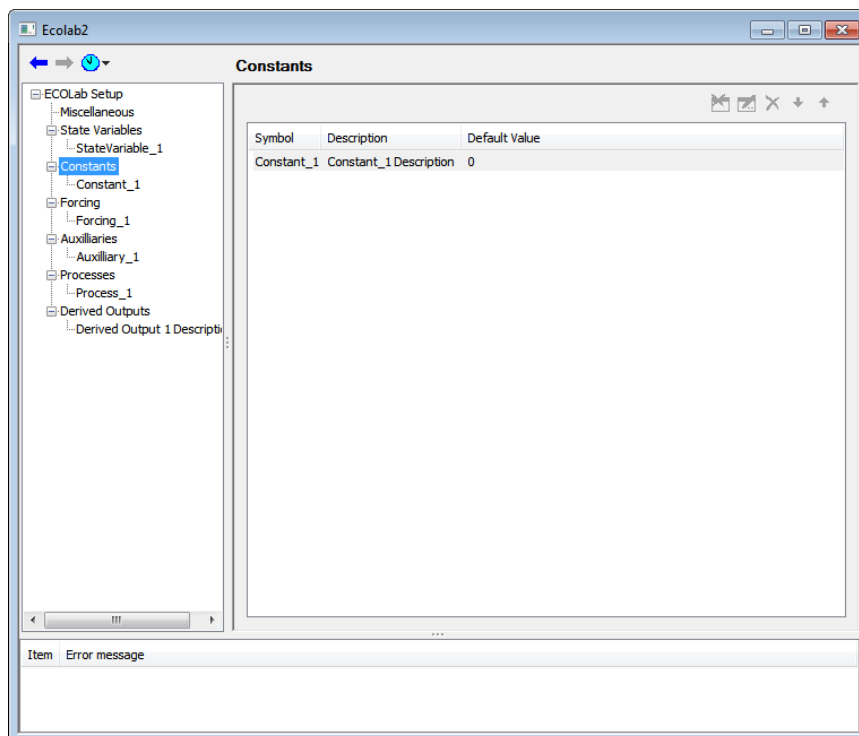


Figure 4.8 Constants overview

This overview lists all defined constants, their description and the default value. You can navigate to the details page by a double-click or the 'edit'-icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon. If the current selected state variable is not referenced the 'delete' icon is active and the item can be removed. The up/down arrows can be used to change the item order.

## 4.4.1 Constants - details

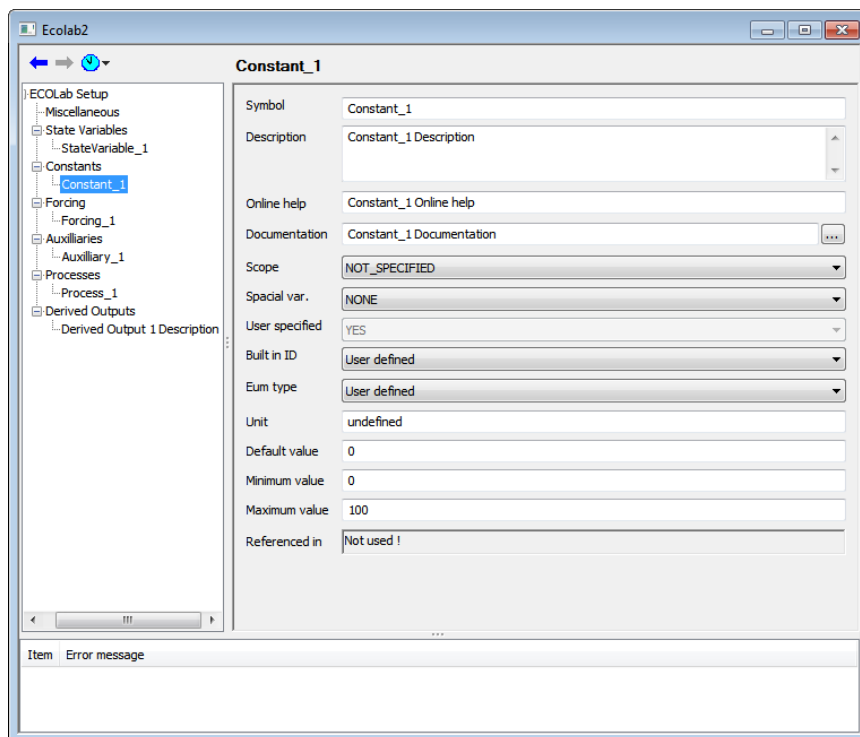


Figure 4.9 Constants - details

### Symbol

This is the *IDENTIFIER* used to address the concentration/value represented by the current constant in the remaining parts of the template. For practical reasons identifiers should be kept short. The user can also use any valid identifiers. DHI suggests to follow the Identifier naming scheme in Section 5.1.2.

### Description

This is short text used as annotation in the results and the setup interface.

### Online help

The content of this field will be displayed as 'mouse-over' tooltip in the setup.

### Documentation

A link or textual description for a more detailed description.

### Scope

Definition of the *SCOPE*, describing the principal vertical domain of the current item.



### Spatial variation

Definition of the spatial representation of the current item. MIKE ECO Lab items can have a global validity when the same value is present globally the model domain. Otherwise the item can vary horizontally (2D) or may be different in every grid point horizontally and vertical (3D).

### User defined

'YES' if user defined or 'NO' if a built-in constant is selected.

### Built\_in ID

Use a pre-defined constants (cf. Section 5.7) or set the 'user defined' and specify all other properties. If a built-in definition is selected other properties will be set automatically.

### EUM type

Specify the EUM used for output of this item. If a predefined EUM type is selected, the appropriate unit is automatically chosen. If you need a not available output unit, set to 'undefined' and specify an output unit manually.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Unit

Specify the unit in case of a manual/undefined EUM type.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Default value

This value will be used as default initialisation/boundary value in the setup.

### Minimum value

Minimum allowed value for this item as input data. Depending on the debug level an undershooting will be noted in the log file.

### Maximum value

Maximum allowed value for this item as input data. Depending on the debug level an overshooting will be noted in the log file.

### Referenced in

Here all MIKE ECO Lab items are listed that refer to the current element. The listed items work as hyperlinks.

## 4.5 Forcings

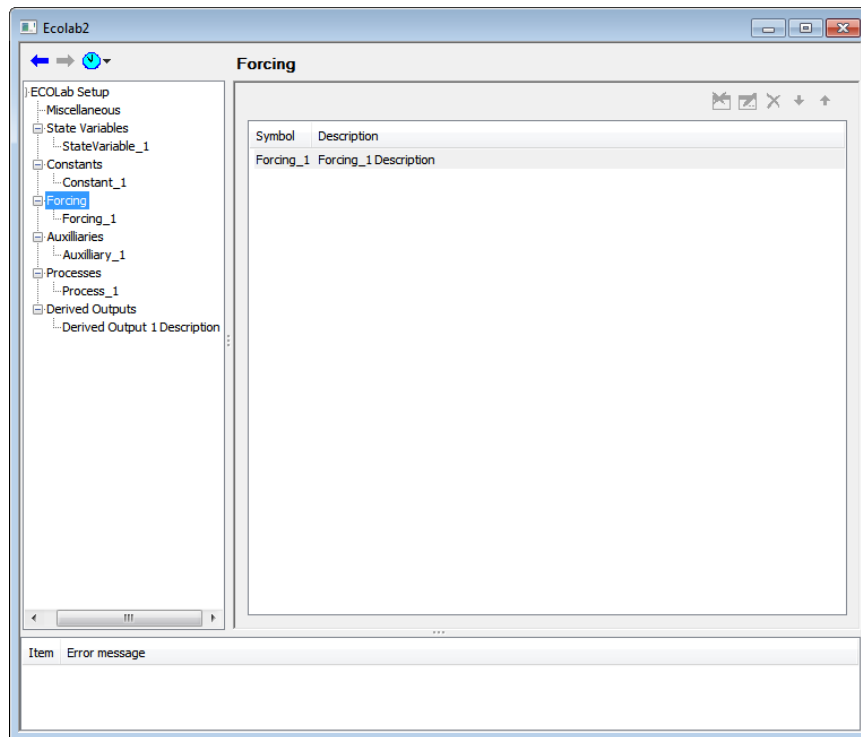


Figure 4.10 Forcings - overview

This overview lists all defined forcings and their description. You can navigate to the details page by a double-click or the 'edit'-icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon. If the current selected state variable is not referenced the 'delete' icon is active and the item can be removed. The up/down arrows can be used to change the item order.



### 4.5.1 Forcings - details

The screenshot shows the 'Ecolab2' application window. On the left is a tree view under 'Ecolab Setup' containing folders like 'Miscellaneous', 'State Variables', 'Constants', 'Forcing', 'Auxiliaries', 'Processes', and 'Derived Outputs'. The 'Forcing' folder is expanded, showing 'Forcing\_1'. The main area is titled 'Forcing\_1' and contains the following fields:

- Symbol: Forcing\_1
- Description: Forcing\_1 Description
- Online help: Forcing\_1 Online help
- Documentation: Forcing\_1 Documentation
- Scope: NOT\_SPECIFIED
- Spatial var.: NONE
- User specified: YES
- Built in ID: User defined
- Eum type: User defined
- Unit: undefined
- Default value: 0
- Minimum value: 0
- Maximum value: 100
- Referenced in: Not used !

At the bottom of the window is a table with columns 'Item' and 'Error message'.

Figure 4.11 Forcings - details

#### Symbol

This is the *IDENTIFIER* used to address the concentration/value represented by the current forcing in the remaining parts of the template. For practical reasons identifiers should be kept short. The user can also use any valid identifiers. DHI suggests to follow the Identifier naming scheme in Section 5.1.2.

#### Description

This is short text used as annotation in the results and the setup interface.

#### Online help

The content of this field will be displayed as 'mouse-over' tooltip in the setup.

#### Documentation

A link or textual description for a more detailed description.

#### Scope

Definition of the *SCOPE*, describing the principal vertical domain of the current item.

### Spatial variation

Definition of the spatial representation of the current item. MIKE ECO Lab items can have a global validity when the same value is present globally the model domain. Otherwise the item can vary horizontally (2D) or may be different in every grid point horizontally and vertical (3D).

### User defined

'YES' if user defined or 'NO' if a built-in constant is selected.

### Built\_in ID

Use a pre-defined constants (cf. Section 5.8) or set the 'user defined' and specify all other properties. If a built-in definition is selected other properties will be set automatically.

### EUM type

Specify the EUM used for output of this item. If one predefined EUM type is selected the appropriate unit is automatically chosen. If you need an not available output type set to 'undefined' and specify an output unit manually.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Unit

Specify the unit.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Default value

This value will be used as default initialisation/boundary value in the setup.

### Minimum value

Minimum allowed value for this item as input data. Depending on the debug level an undershooting will be noted in the log file.

### Maximum value

Maximum allowed value for this item as input data. Depending on the debug level an overshooting will be noted in the log file.

### Referenced in

Here all the MIKE ECO Lab items are listed that refer to the current element. The listed items work as hyperlinks.





## 4.6 Auxiliaries

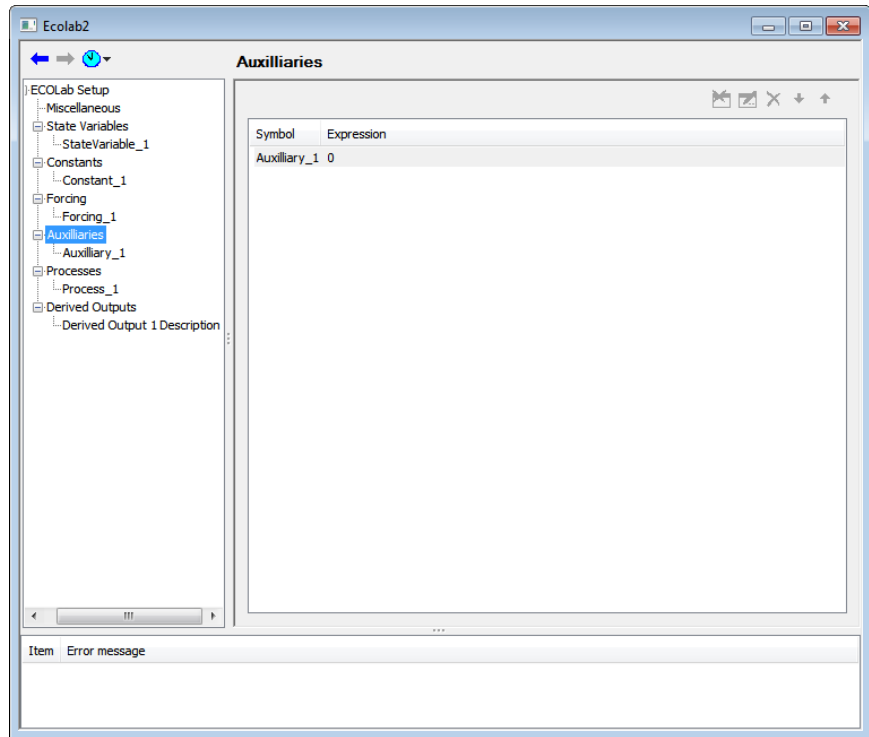


Figure 4.12 Auxiliaries - overview

This overview lists all defined auxiliary variables and their expression. You can navigate to the details page by a double-click or the 'edit-' icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon. If the current selected state variable is not referenced the 'delete' icon is active and the item can be removed. The up/down arrows can be used to change the item order.

## 4.6.1 Auxiliaries - details

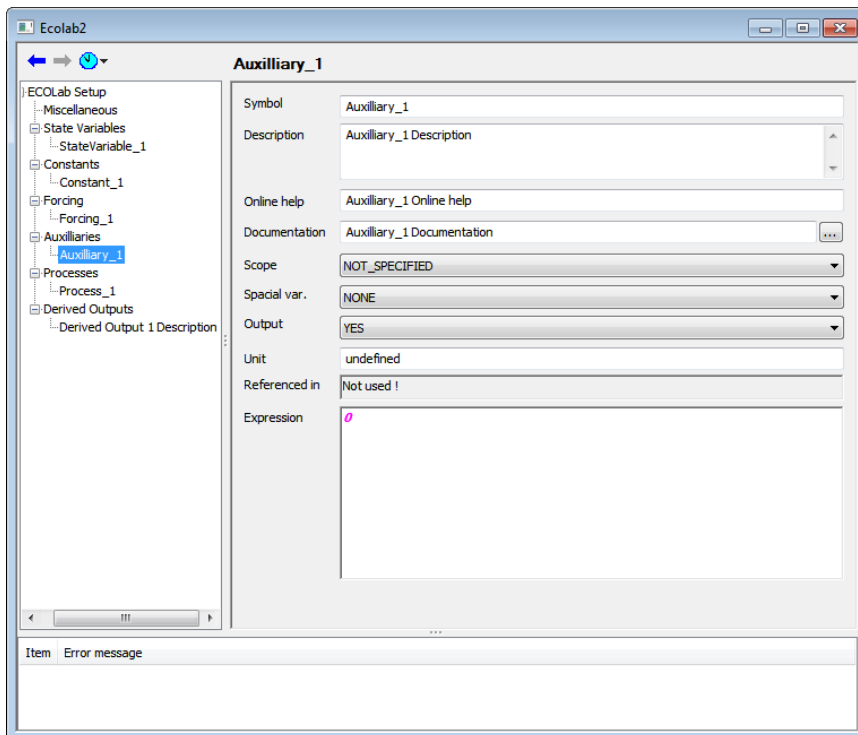


Figure 4.13 Auxiliaries - details

### Symbol

This is the *IDENTIFIER* used to address the concentration/value represented by the current auxiliary in the remaining parts of the template. For practical reasons identifiers should be kept short. The user can also use any valid identifiers. DHI suggests to follow the Identifier naming scheme in Section 5.1.2.

### Description

This is short text used as annotation in the results and the setup interface.

### Online help

The content of this field will be displayed as 'mouse-over' tooltip in the setup.

### Documentation

A link or textual description for a more detailed description.

### Scope

Definition of the *SCOPE*, describing the principal vertical domain of the current item.



### Spatial variation

Definition of the spatial representation of the current item. MIKE ECO Lab items can have a global validity when the same value is present globally the model domain. Otherwise the item can vary horizontally (2D) or may be different in every grid point horizontally and vertical (3D).

### Output

If set to 'YES' this item will be available in the list of output items. If set to 'NO' the item will be not be seen/available as output (usefully to reduce complexity for the enduser).

### Unit

Specify the unit.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Referenced in

Here all MIKE ECO Lab items are listed that refer to the current element. The listed items work as hyperlinks.

### Expression

This is the *MIKE ECO Lab expression* (cf. Section 5.6) represented by the current item.

## 4.7 Processes

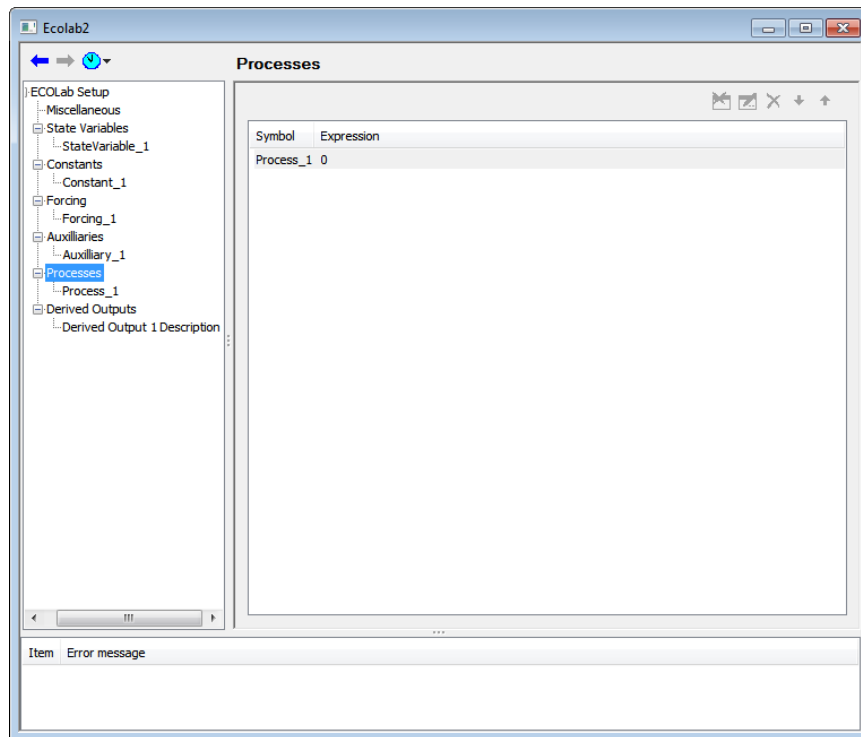


Figure 4.14 Processes - overview

This overview lists all the defined processes and their expressions. You can navigate to the details page by a double-click or the 'edit-' icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon. If the current selected state variable is not referenced the 'delete' icon is active and the item can be removed. The up/down arrows can be used to change the item order.



### 4.7.1 Processes - details

Figure 4.15 Processes - details

#### Symbol

This is the *IDENTIFIER* used to address the concentration/value represented by the current process in the remaining parts of the template. For practical reasons identifiers should be kept short. The user can also use any valid identifiers. DHI suggests to follow the Identifier naming scheme in Section 5.1.2.

#### Description

This is short text used as annotation in the results and the setup interface.

#### Online help

The content of this field will be displayed as 'mouse-over' tooltip in the setup.

#### Documentation

A link or textual description for a more detailed description.

#### Scope

Definition of the *SCOPE*, describing the principal vertical domain of the current item.

### Spatial variation

Definition of the spatial representation of the current item. MIKE ECO Lab items can have a global validity when the same value is present globally the model domain. Otherwise the item can vary horizontally (2D) or may be different in every grid point horizontally and vertical (3D)

### Output

If set to 'YES' this item will be available in the list of output items. If set to 'NO' the item is not available as output (usefully to reduce complexity for the enduser).

### Process type

The process type can be 'Transformation', meaning that the process transforms a variable in the current vertical layer. If 'Settling' or 'Buoyancy' is selected the process represents a transport from the current vertical layer into upper/lower layers. See Section 5.3.5 for more details.

### Unit

Specify the unit.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Referenced in

Here all MIKE ECO Lab items are listed that refer to the current element. The listed items work as hyperlinks.

### Expression

This is the MIKE ECO Lab expression (see Section 5.6) represented by the current item.



## 4.8 Derived Outputs

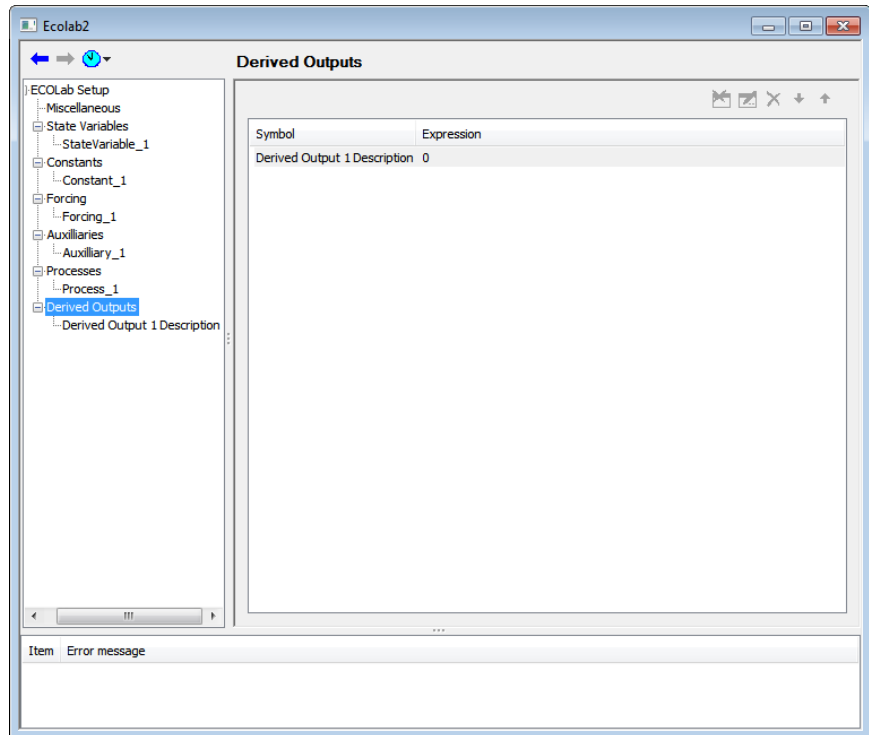


Figure 4.16 Derived Outputs - overview

This overview lists all defined derived outputs and their expressions. You can navigate to the details page by a double-click or the 'edit'-icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon. If the current selected state variable is not referenced the 'delete' icon is active and the item can be removed. The up/down arrows can be used to change the item order.

## 4.8.1 Derived outputs - details

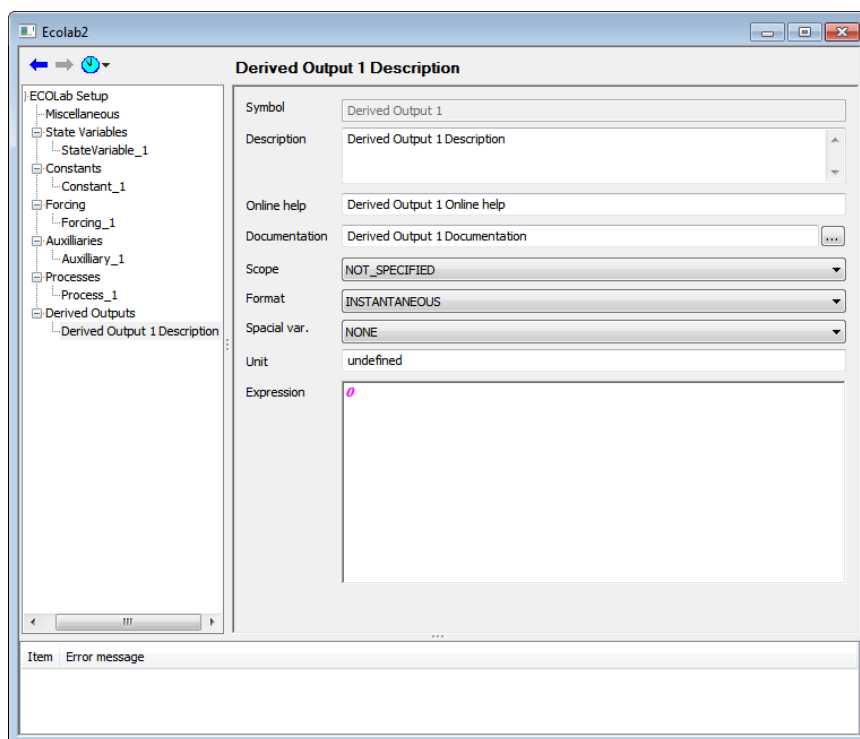


Figure 4.17 Derived Outputs - details

### Symbol

This is the *IDENTIFIER* used to address the concentration/value represented by the current derived output in the remaining parts of the template. For practical reasons identifiers should be kept short. The user can also use any valid identifiers. DHI suggests to follow the Identifier naming scheme in Section 5.1.2.

### Description

This is short text used as annotation in the results and the setup interface.

### Online help

The content of this field will be displayed as 'mouse-over' tooltip in the setup.

### Documentation

A link or textual description for a more detailed description.

### Scope

Definition of the *SCOPE* describing the principal vertical domain of the current item.





### Spatial variation

Definition of the spatial representation of the current item. MIKE ECO Lab items can have a global validity when the same value is present globally the model domain. Otherwise the item can vary horizontally (2D) or may be different in every grid point horizontally and vertical (3D).

### Unit

If set to 'YES' this item will be available in the list of output items. If set to 'NO' the item is not available as output (usefully to reduce complexity for the enduser).

### Expression

This is the MIKE ECO Lab expression (cf. Section 5.6) represented by the current item.

## 4.9 Particle Classes

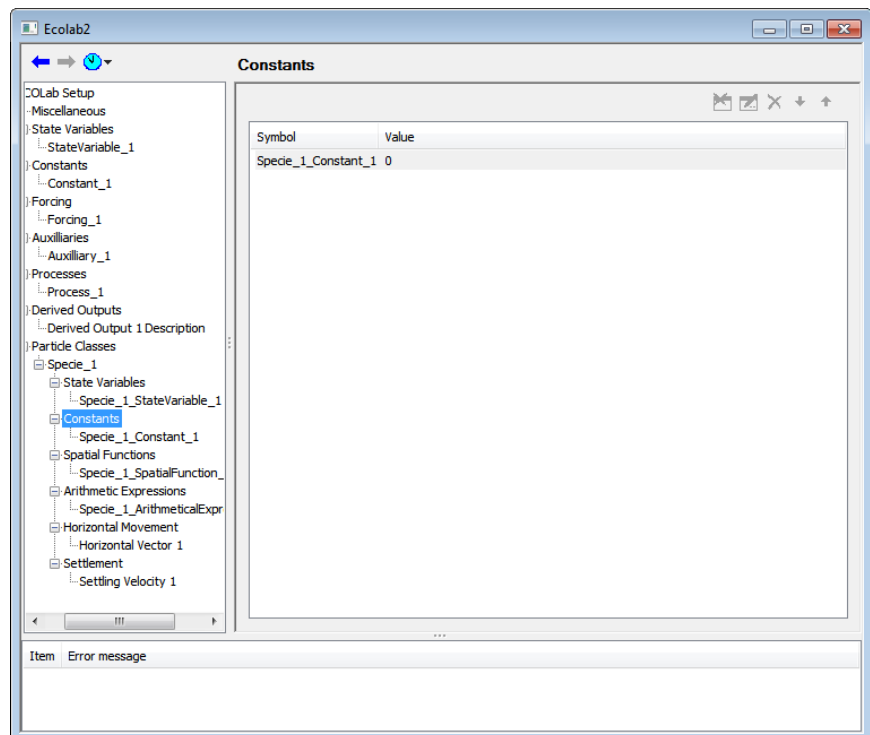


Figure 4.18 Particle Classes - overview

This overview lists all the defined particle classes. You can navigate to the details page by a double-click or the 'edit-' icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon or removed using the 'delete'. The up/down arrows can be used to change the item order.

## 4.9.1 Particle classes - details

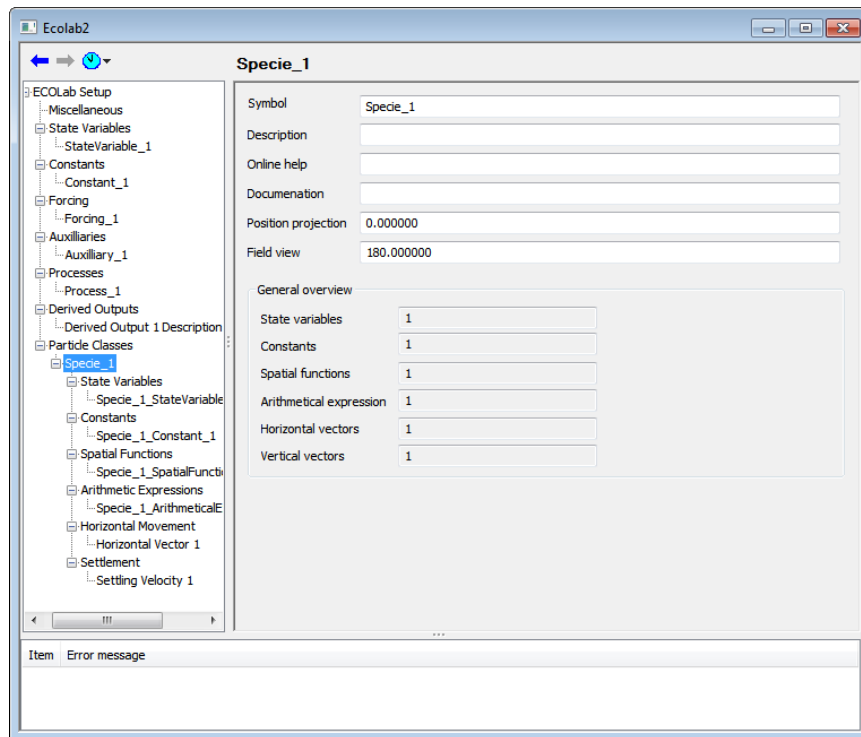


Figure 4.19 Particle Classes - details

This overview page lists some general aspects like the number of class state variables, constants etc.

### Position projection

This property determines how far (in time) the particle position is projected into the future. The resulting position will be stored as particle property (cf. Section 5.10.3) and can be used to e.g. modify movement when a particle is in danger to collide with the land borders.

### Field of view

Determines the opening (in degree) of the perception in view direction. This entry is just relevant if search functions (cf. Section 5.10.8) are used to sense other particles. A field of view of e.g. 90° will enable a particle to detect other particles within a sector of  $\pm 90^\circ$  in viewing direction.



## 4.9.2 Particle classes state variables

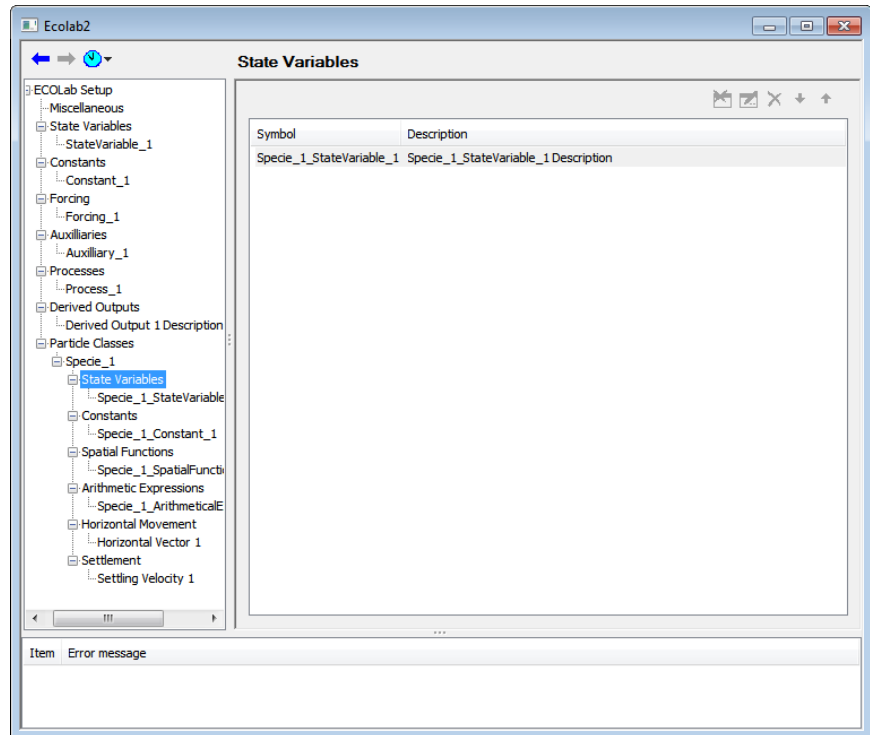


Figure 4.20 Particle Classes State Variables - overview

This overview lists all the defined state variables of the current selected particle class. You can navigate to the details page by a double-click or the 'edit'-icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon. If the current selected state variable is not referenced the 'delete' icon is active and the item can be removed. The up/down arrows can be used to change the item order.

## State Variables - details

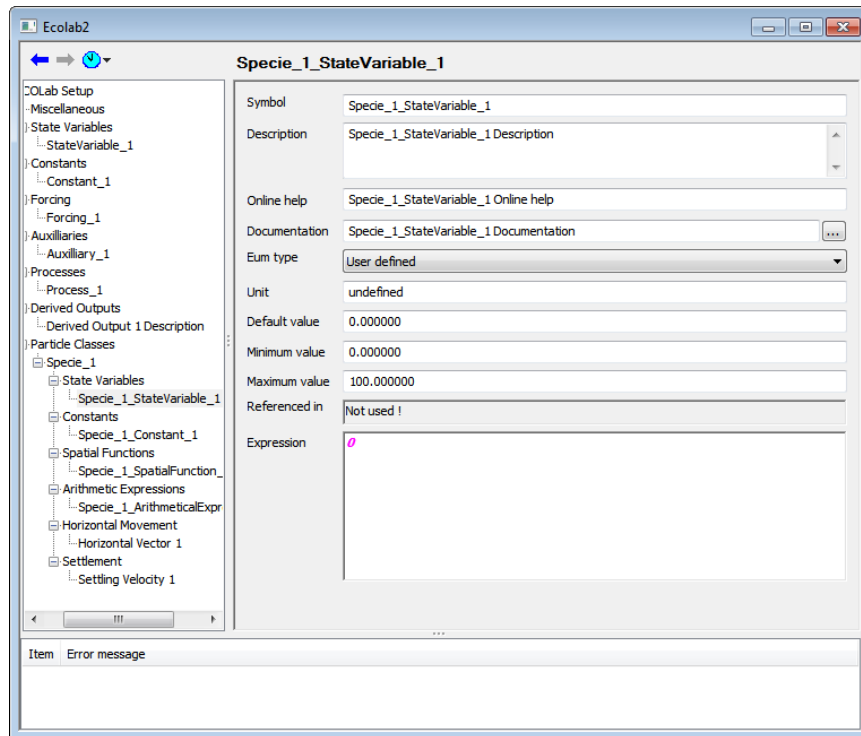


Figure 4.21 State Variables - details

### Symbol

This is the *IDENTIFIER* used to address the concentration/value represented by the current particle state variable in the remaining parts of the template. Every individual from the current particle class will have an independent set of state variables with individual values.

### Description

This is short text used as annotation in the results and the setup interface.

### Online help

The content of this field will be displayed as 'mouse-over' tooltip in the setup.

### Documentation

A link or textual description for a more detailed description.

### EUM type

Specify the EUM used for output of this item. If one predefined EUM type is selected the appropriate unit is automatically chosen. If you need an not available output type set to 'undefined' and specify an output unit manually.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Unit

Specify the output unit.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Default value

This value will be used as default initialisation/boundary value in the setup.

### Minimum value

Minimum allowed value for this item as input data.

### Maximum value

Maximum allowed value for this item as input data.

### Referenced in

Here all MIKE ECO Lab items are listed that refer to the current element. The listed items work as hyperlinks.

### Expression

This expression is the differential equation describing the change rate of an state variable. It is the right-hand side of the general equations:

$$\frac{dSymbol}{dt} = process1 + process2 + \dots$$

Only arithmetic expressions of the same particle class and values are allowed. See Section 5.6 for more details.

### 4.9.3 Particle classes constants

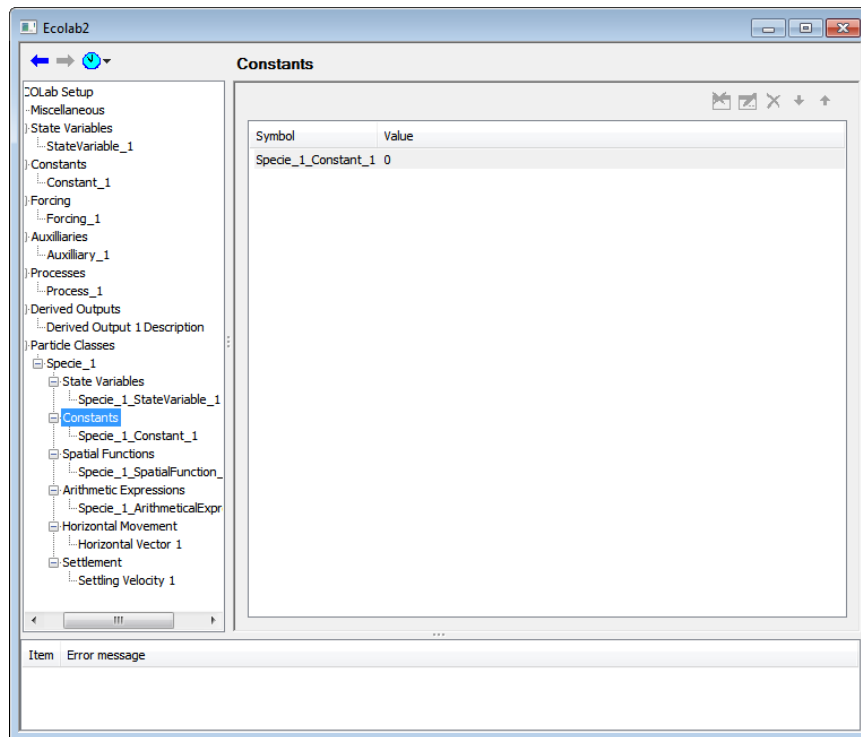


Figure 4.22 Particle Classes Constants - overview

This overview lists all the defined constants, their description and the default value. You can navigate to the details page by a double-click or the 'edit-' icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon. If the current selected state variable is not referenced the 'delete' icon is active and the item can be removed. The up/down arrows can be used to change the item order.



## Particle Classes Constants - details

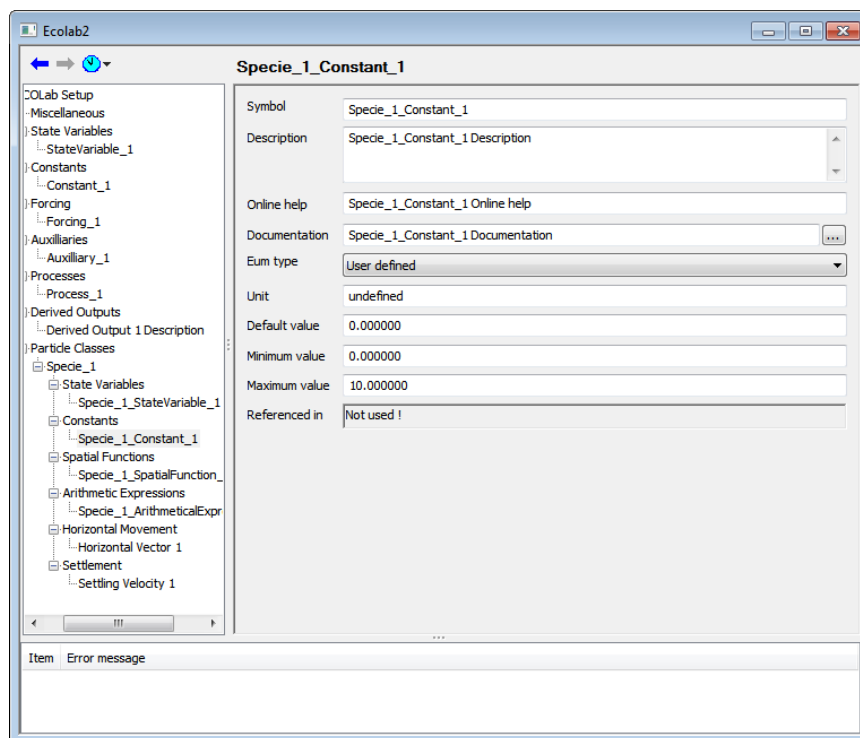


Figure 4.23 Particle Classes Constants - details

### Symbol

This is the *IDENTIFIER* used to address the concentration/value represented by the current constant in the remaining parts of the template. For practical reasons identifiers should be kept short. The user can also use any valid identifiers. DHI suggests to follow the Identifier naming scheme in Section 5.1.2.

### Description

This is short text used as annotation in the results and the setup interface.

### Online help

The content of this field will be displayed as 'mouse-over' tooltip in the setup.

### Documentation

A link or textual description for a more detailed description.

### EUM type

Specify the EUM used for output of this item. If one predefined EUM type is selected the appropriate unit is automatically chosen. If you need an not available output type set to 'undefined' and specify an output unit manually.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Unit

Specify the unit.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Default value

This value will be used as default initialisation/boundary value in the setup.

### Minimum value

Minimum allowed value for this item as input data.

### Maximum value

Maximum allowed value for this item as input data.

### Referenced in

Here all MIKE ECO Lab items are listed that refer to the current element. The listed items work as hyperlinks.



There are no predefined/built-in particle constants. All particle constants have to be user defined.





#### 4.9.4 Restricted area search

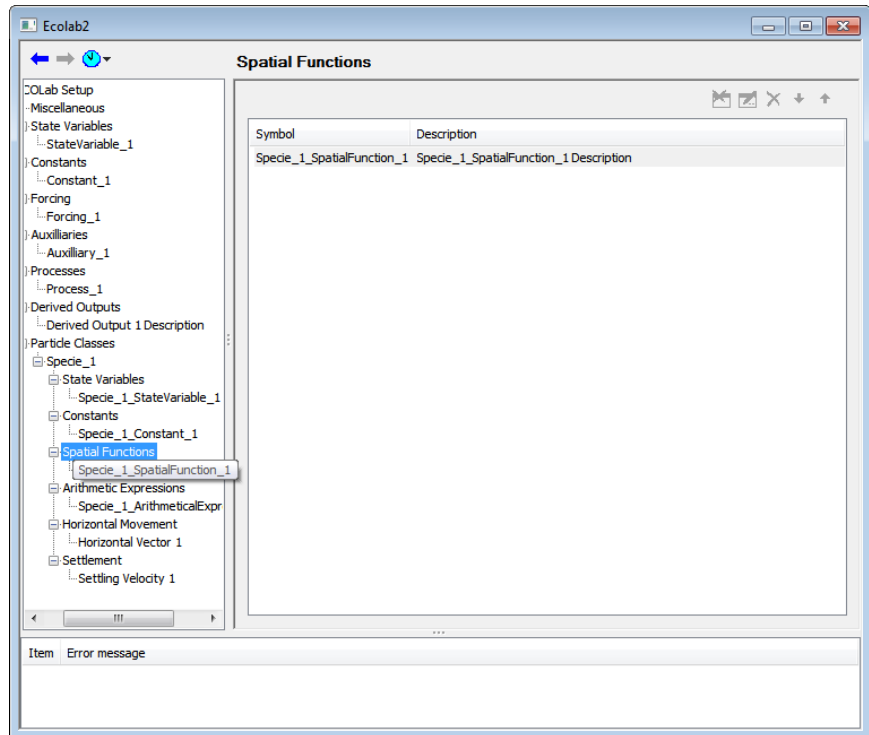


Figure 4.24 Spatial Functions - overview

This overview lists all the defined restricted area search functions. You can navigate to the details page by a double-click or the 'edit'-icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon. If the current selected state variable is not referenced the 'delete' icon is active and the item can be removed. The up/down arrows can be used to change the item order.

## Restricted area search - details

The screenshot shows the Ecolab2 software interface. On the left is a tree view of the model structure. The main panel on the right displays the details for the selected item, 'Specie\_1\_SpatialFunction\_1'.

**Tree View Structure:**

- Lab Setup
  - Miscellaneous
  - State Variables
    - StateVariable\_1
  - Constants
    - Constant\_1
  - Forcing
    - Forcing\_1
  - Auxiliaries
    - Auxiliary\_1
  - Processes
    - Process\_1
  - Derived Outputs
    - Derived Output 1 Description
  - Particle Classes
    - Specie\_1
      - State Variables
        - Specie\_1\_StateVariable\_1
      - Constants
        - Specie\_1\_Constant\_1
      - Spatial Functions
        - Specie\_1\_SpatialFunction\_1
      - Arithmetic Expressions
        - Specie\_1\_ArithmeticalExpres
      - Horizontal Movement
        - Horizontal Vector 1
      - Settlement
        - Settling Velocity 1

**Specie\_1\_SpatialFunction\_1 Details:**

Symbol	Specie_1_SpatialFunction_1
Description	Specie_1_SpatialFunction_1 Description
Online help	Specie_1_SpatialFunction_1 Online help
Documentation	Specie_1_SpatialFunction_1 Documentation
Purpose	MAGNITUDE
Framework	LAGRANGE
Based on variable	
Based on specie	
Dimension	3
Range/radius	0.000000
Applied math	AVERAGE
Look	DIRECTION_HIGHEST_VALUE
Referenced in	Not used !

At the bottom of the window, there is a section for 'Item Error message'.

Figure 4.25 Spatial Functions - details

For a description please see Section 5.10.8.



### 4.9.5 Arithmetic expressions

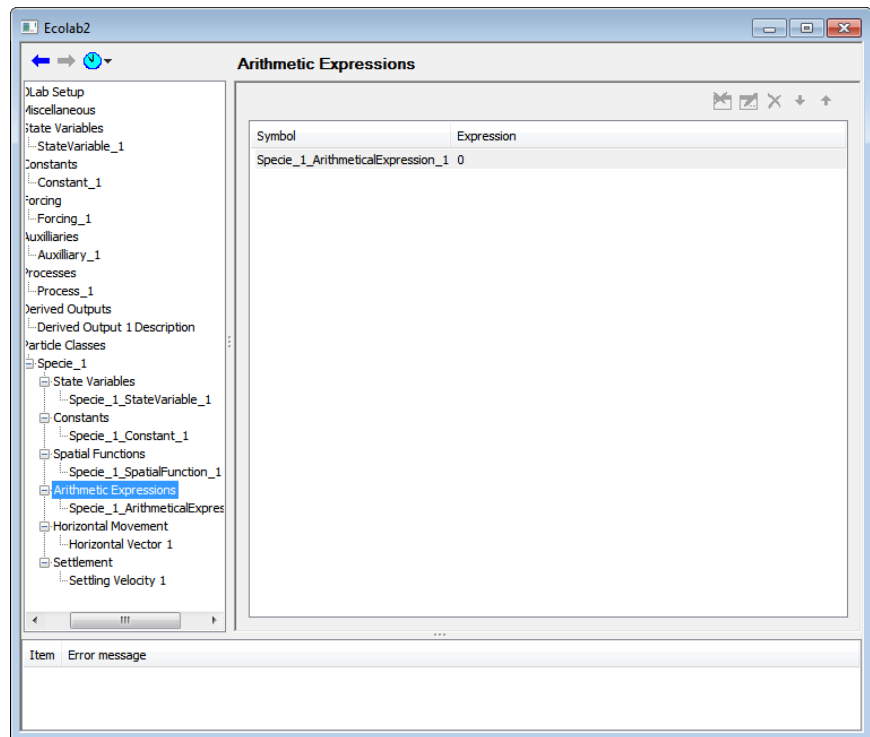


Figure 4.26 Arithmetic Expressions - overview

This overview lists all the arithmetic expressions of the current particle class. You can navigate to the details page by a double-click or the 'edit'-icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon. If the current selected state variable is not referenced the 'delete' icon is active and the item can be removed. The up/down arrows can be used to change the item order.

## Arithmetic Expressions - details

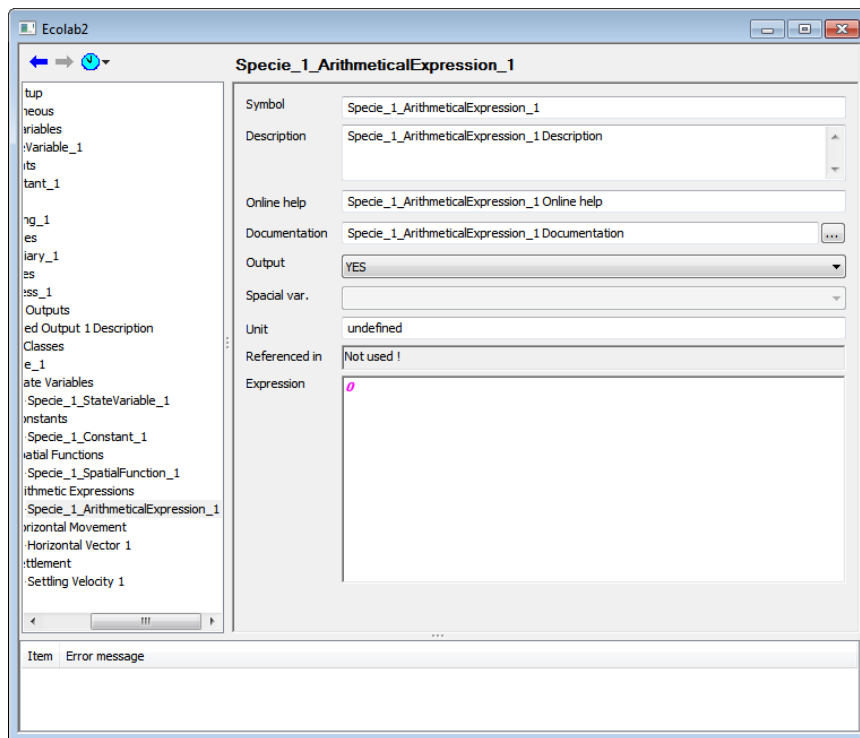


Figure 4.27 Arithmetic Expressions - details

### Symbol

This is the *IDENTIFIER* used to address the concentration/value represented by the current arithmetic expression in the remaining parts of the template. For practical reasons identifiers should be kept short. The user can also use any valid identifiers. DHI suggests to follow the Identifier naming scheme in Section 5.1.2.

### Description

This is short text used as annotation in the results and the setup interface.

### Online help

The content of this field will be displayed as 'mouse-over' tooltip in the setup.

### Documentation

A link or textual description for a more detailed description.

### Output

If set to 'YES' this item will be available in the list of output items. If set to 'NO' the item is not available as output (usefully to reduce complexity for the enduser).



### Spatial variation

This property can not be edited.

### Unit

Specify the unit.



Ensure that the expressions inside the template are compatible with the chosen representation.

### Referenced in

Here all MIKE ECO Lab items are listed that refer to the current element. The listed items work as hyperlinks.

### Expression

This is the MIKE ECO Lab expression (cf. Section 5.6) represented by the current item

## 4.9.6 Horizontal movement vectors

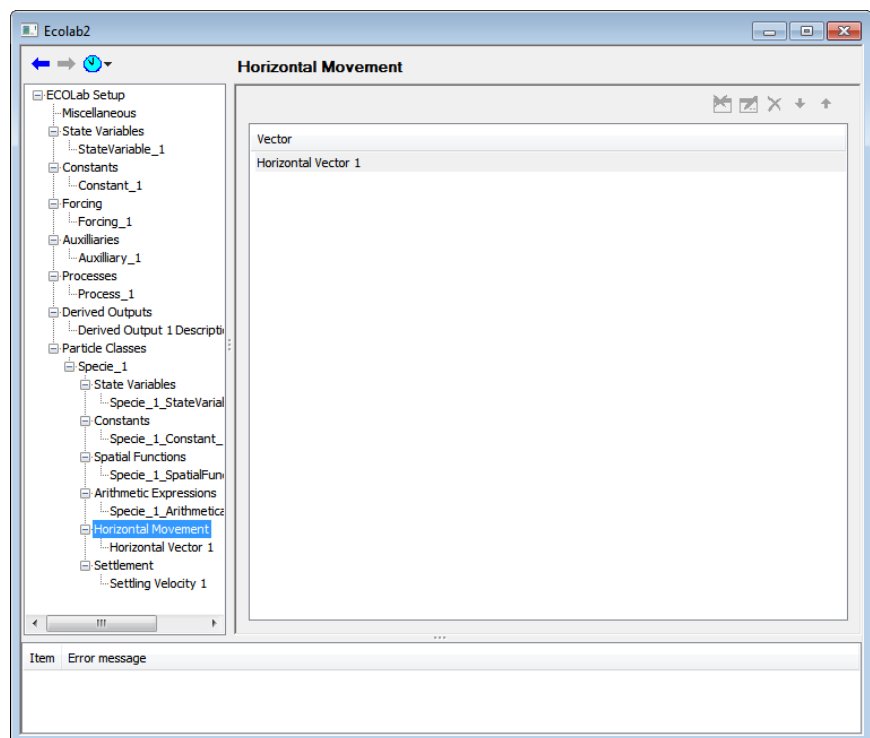


Figure 4.28 Horizontal Movement Vectors - overview

This overview lists all the defined horizontal movement vectors defined for the current particle class. Each particle class can have up to 5 independent horizontal movement vectors that are internally summed up to a final movement

vecotr. You can navigate to the details page by a double-click or the 'edit-' icon. New elements of the current type can be inserted to the end of the list by the 'new'-item icon. If the current selected state variable is not referenced the 'delete' icon is active and the item can be removed. The up/down arrows can be used to change the item order.

## Horizontal Movement Vector - details

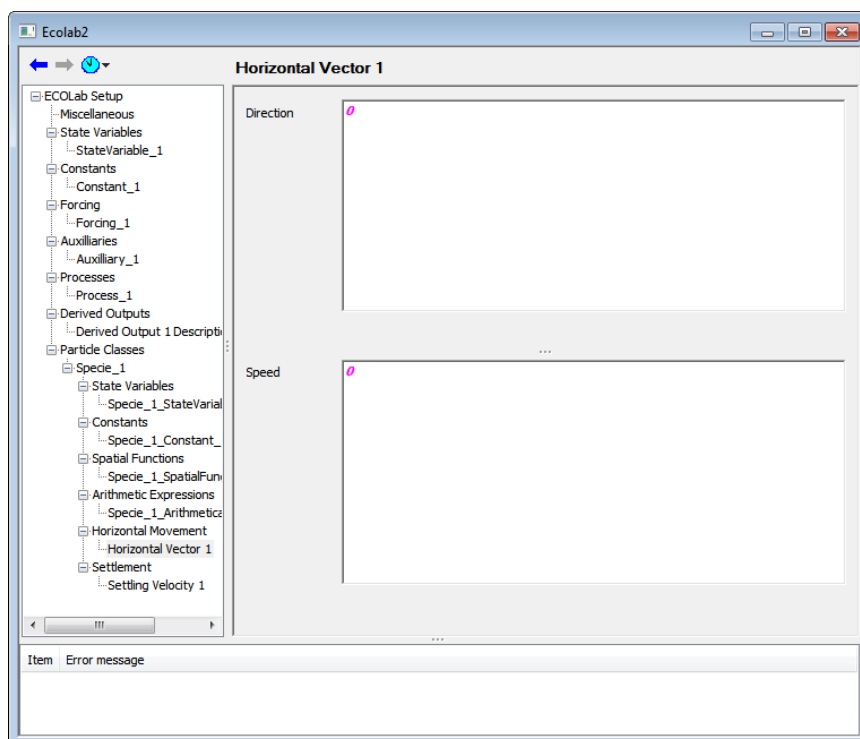


Figure 4.29 Horizontal Movement Vector - details

### Direction

This is an MIKE ECO Lab expression defining the direction–component [degree] of the current movement vector.

### Speed

This is an MIKE ECO Lab expression defining the speed –component [m/s] of the current movement vector.



### 4.9.7 Downward movement

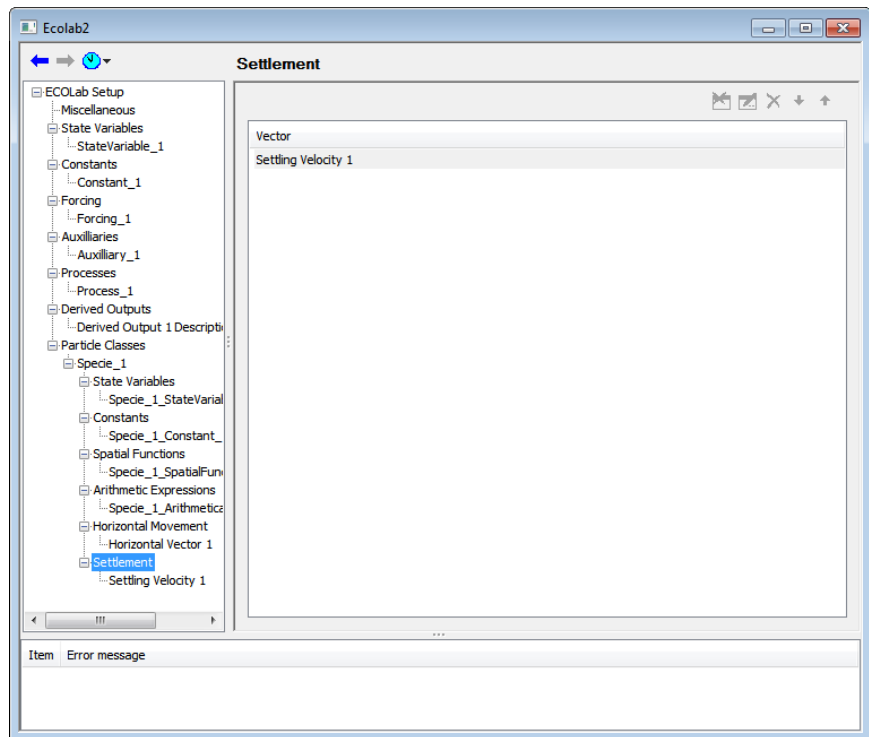


Figure 4.30 Settlement - overview

## Downward movement - details

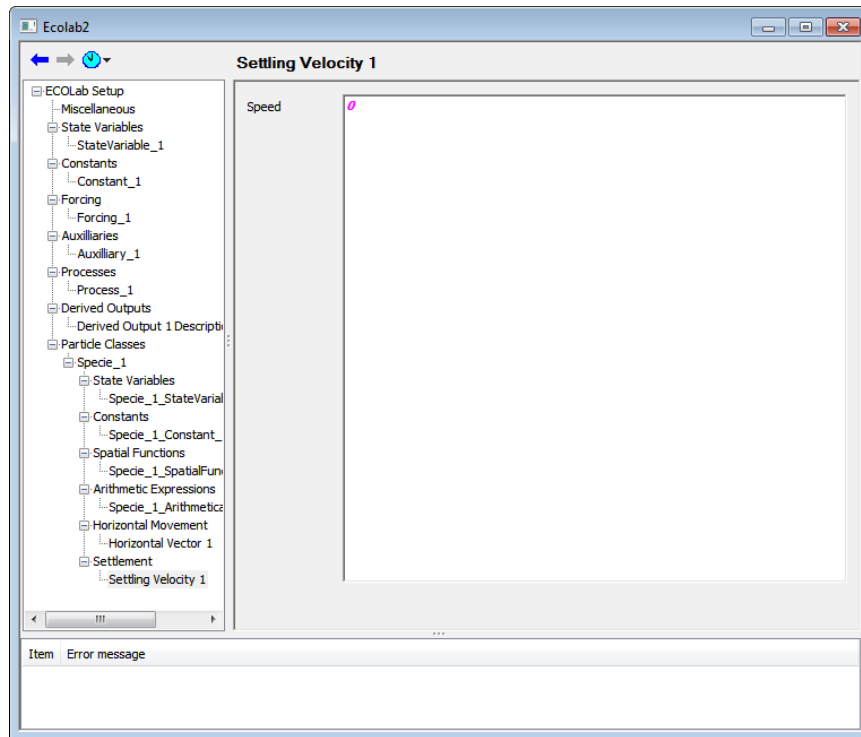


Figure 4.31 Settlement - details

### Speed

This is an MIKE ECO Lab expression defining the settling speed [m/s] of the current movement vector.



A positive settling speed will cause a downward movement and a negative settling speed indicates an upward movement. This is contrary to the flow direction of the vertical flow where a negative flow points downwards whereas a positive flow points upwards!





## 5 Reference Manual

### 5.1 Identifier

Identifiers are used to address the information/numeric value represented by a MIKE ECO Lab element. An identifier has to start with a letter followed by further letters, numbers or an underscore. Spaces are not allowed.

#### 5.1.1 Examples

##### Valid identifiers:

'DO', 'Level\_1', 'temp', 'inorg\_Nitrate'

##### Invalid identifiers::

'inorg.Nitrate'	'.' not allowed
'dissolved oxygen'	space not allowed
'1Level'	Identifier does not start with a letter

#### 5.1.2 Identifier naming scheme

Also any valid identifiers can be used it is general advised to keep identifiers short (less than 8 characters) and descriptive. A proper chosen identifier can help to document a template. DHI suggests the identifiers in Table 5.1 for some general water quality items.

Table 5.1 Identifiers

Identifier	Used for
Temp	Temperature in degree
Kelvin	Absolute temperature in Kelvin
DO	Dissolved oxygen
CHL	Chlorophyll
DC	Detritus, organic carbon
DN	Detritus, organic nitrogen
DP	Detritus, organic phosphorus
PC	Phytoplankton carbon

Table 5.1 Identifiers

Identifier	Used for
PN	Phytoplankton nitrogen
PP	Phytoplankton phosphorus
ZC	Zooplankton carbon
IN	Inorganic nitrogen
IP	Inorganic phosphorus
BC	Benthic / Macroalgae carbon
BN	Benthic / Macroalgae nitrogen
BP	Benthic / Macroalgae phosphorus
SOD	Sediment oxygen demand
NH4	Ammonia
NO3	Nitrate
NO2	Nitrite
PO4	Phosphate

You can build new identifiers for multiply item in a category by adding a underscore and the item class, e.g. if you have two algae, green algae and diatoms, you can specify CHL\_GREEN and CHL\_DIATOM for the chlorophyll of the green respectively diatom algae.

## 5.2 Structure/Elements of a MIKE ECO Lab Template

According to the philosophy of MIKE ECO Lab, ecological modelling is split in two parts. The so called 'MIKE ECO Lab Template' contains description and definition of the ecological model in a generic way. Such a template includes all definitions for variables, forcing, equations and movement rules but does not contain definitions of the specific environment or specific values for calibrations or applied forcing.

The second part of an ecological model formulated in MIKE ECO Lab refers to the actual application of this model, the so called model setup. It specifies the environment (spatial extent, hydrodynamic conditions etc.) and specific values used for calibration and input etc.

A template itself cannot be executed or produce any outputs alone. It always needs to be applied within a model setup to run and produce any output for a specific environment and set of input data.

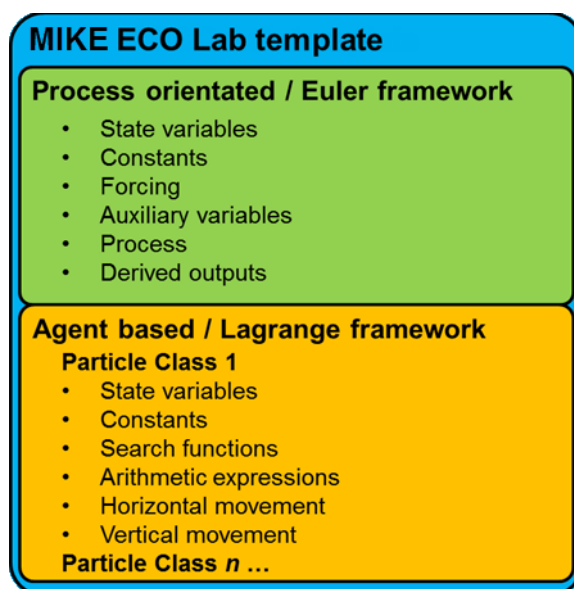


Figure 5.1 Frameworks in MIKE ECO Lab template

Each template can contain elements from two different frameworks. The first so called 'Eulerian framework' is used to describe process orientated state variables but also environmental factors like flow conditions and temperature. The second 'Lagrangian framework' is used to describe elements for agent based modelling.

### 5.2.1 Eulerian framework

The Eulerian framework is used to describe process orientated state variables and environmental factors like flow conditions and temperature.

Table 5.2 describe the elements in the Eulerian framework:

Table 5.2 Elements in Eulerian framework

Element	Usage
State variables	Changing variables describing the state of the modelled system, usually representing the most important information
Constants	Temporal constant but potential spatial varying factors like growth rates, stoichiometric relations, half saturation constants etc.
Forcings	Both temporal and spatial varying external factors like flow conditions, light etc.
Auxiliary variables	Intermediate calculations, additional outputs

Table 5.2 Elements in Eulerian framework

Element	Usage
Processes	Describing the change of a state variable over time
Derived outputs	Additional output calculations

## 5.2.2 Lagrangian framework

The Lagrangian framework is used to describe agents in an agent based model. All necessary information on a specific agent type is structured in a so called particle class. This includes their internal states (i.e. body mass etc.) and their behaviour and movement definitions. A biological species may be described by its own particle class but it is also common to describe different development stages of the same species with different particle classes when their behaviour/characteristics are different.

Table 5.3 describe the elements in the Lagrangian framework:

Table 5.3 Elements in Lagrangian framework

Element	Usage
State variables	Internal variables like body mass, sex etc. that vary individually for each agent
Constants	Constants that are valid for each agent of a specific class, e.g. growth rates, max swimming speed etc.
Searching Functions	Long-range perception of Eulerian or other Lagrangian elements
Arithmetic expressions	Intermediate and state variable process calculations
Horizontal movement	Movement in the X-Y plane
Vertical movement	Vertical movement description

## 5.3 Process Orientated Modelling with MIKE ECO Lab

*Definition: "Models describing the fate of masses/concentrations due to associated processes".*



Process orientated or bio-geochemical modelling is an established modelling technique. Usually the change of a mass or concentration variable over time is described by ordinary differential equation, in short ODE:

$$\frac{dA}{dt} = P_1 + P_2 - P_3 \dots \quad (5.1)$$

A term ( $P_1$ ,  $P_2$ ,  $P_3$ ) occurring in the right-hand side of such an ODE are commonly called "Process" as it usually describes the contribution of an individual process or reactions  $P_x$  for the variable A, e.g. **the change** of A due to growth and respiration process. The dynamics for A follow from the interplay of all its associated process.

Process orientated modelling is widely applied, e.g. to model concentrations like oxygen levels, pollution or biomass. It actually forms the backbone of MIKE ECO Lab.

Any changing element the modeller is interested in must be declared as a so called "State Variable". The change of such a state variable must be computed by associated process. Such a process itself can be formulated as an expression consisting of elements like state variables (representing the actual value in the current computational point), constants or external factors like ambient temperature, mathematical functions and operators or intermediate calculations (called auxiliary variables, may be seen as user declared functions).

MIKE ECO Lab offers the following elements for process orientated modelling:

1. State variables
2. Constants
3. Forcings
4. Auxiliary variables
5. Processes
6. Derived outputs

### 5.3.1 State variables

State variables represent the values that change over time. Usually state variables represent the target variables (e.g. oxygen concentrations, pollutants etc.) of a model. Only state variables are preserved during the time steps of a

model run and any changing element must be formulated as state variable. Each state variable is characterised by a number of properties. :

Table 5.4 Description of state variables

Property	Description
Symbol	Defines the name used in expressions to refer to the actual value of the state variable
Description	A short textual description
Online help	Displayed as mouse-over help in the setup
Documentation	Possible link to detailed external documentation (pdf-file, etc.)
Scope	Definition of the scope, the vertical positioning inside the water column (cf. Section 5.4)
Transport	Definition if the variable is subject to advection/dispersion transport or stationary
EUM type	EUM type definition
Unit	Textual information
Default value	Value used for initialisation
Minimum value	Allowed minimum
Maximum value	Allowed maximum
Expression	<p>Right hand-side of the ODE describing the change of the state variable over time</p> $\frac{dA}{dt} = P_1 + P_2 - P_3 \dots$



Note:

As terms of the expression for state variables only Processes (cf. Section 5.3.5) and numeric constants are allowed!

### 5.3.2 Constants

Constants represent arguments in MIKE ECO Lab expressions that are constant in time but may change in space. Typical examples are rate coefficients,



stoichiometric relations or half saturation concentrations. Each constant is characterised by a number of properties.:

Table 5.5 Description of constants

Property	Description
Symbol	Defines the name used in expressions to refer to the actual value of the constant
Description	A short textual description
Online help	Displayed as mouse-over help in the setup
Documentation	Possible link to detailed external documentation (pdf-file, etc.)
Scope	Not applicable ( <i>NOT SPECIFIED</i> )
Spatial variation	Definition if the constant is globally defined or may vary spatially
User defined	If 'NO' a built-in constant is used
Built.-in ID	Identification of a valid built-in constant (cf. Section 5.7)
EUM type	EUM type definition
Unit	Textual information
Default value	Value used for initialisation
Minimum value	Allowed minimum
Maximum value	Allowed maximum

### 5.3.3 Forcings

Forcings represent arguments in MIKE ECO Lab expressions that may vary both in time and space. Usually forcing represent external factors. Typical examples are temperature, salinity or habitat properties. Each forcing is characterised by a number of properties.

Table 5.6 Description of forcing

Property	Description
Symbol	Defines the name used in expressions to refer to the actual value of the forcing
Description	A short textual description
Online help	Displayed as mouse-over help in the setup

Table 5.6 Description of forcing

Property	Description
Documentation	Possible link to detailed external documentation (pdf-file, etc.)
Scope	Not applicable ( <i>NOT SPECIFIED</i> )
Spatial variation	Definition if the forcing is globally defined or may vary spatially
User defined	If 'NO' a built-in constant is used
Built-in ID	Identification of a valid built-in forcing (cf. Section 5.8)
EUM type	EUM type definition
Unit	Textual information
Default value	Value used for initialisation
Minimum value	Allowed minimum
Maximum value	Allowed maximum

### 5.3.4 Auxiliary variables

Auxiliary variables represent intermediate calculations or additional results. MIKE ECO Lab does not allow user declared functions (i.e. as rate limiting function) but auxiliary variables can be used as such. Each auxiliary variable is characterised by a number of properties.

Table 5.7 Description of auxiliary variable

Property	Description
Symbol	Defines the name used in expressions to refer to the actual value of the auxiliary variable
Description	A short textual description
Online help	Displayed as mouse-over help in the setup
Documentation	Possible link to detailed external documentation (pdf-file, etc.)
Scope	Not applicable ( <i>NOT SPECIFIED</i> )
Spatial variation	Definition if the auxiliary variable is globally defined or may vary spatially
Output	If 'YES' the variable can be used as output item, 'NO' removes the variable from the list of output items





Table 5.7 Description of auxiliary variable

Property	Description
Unit	Textual information
Expression	MIKE ECO Lab expression to calculate value

### 5.3.5 Processes

Processes declare the individual process allowed as terms in a state variable expression.

Table 5.8 Description of process

Property	Description
Symbol	Defines the name used in expressions to refer to the actual value of the process
Description	A short textual description
Online help	Displayed as mouse-over help in the setup
Documentation	Possible link to detailed external documentation (pdf-file, etc.)
Scope	Not applicable ( <i>NOT SPECIFIED</i> )
Spatial variation	Definition if the process is globally defined or may vary spatially
Output	If 'YES' the variable can be used as output item, 'NO' removes the variable from the list of output items
Process type	Defines if a variable is "transformed" in the current computational point or of the process describes a "settling" or "buoyancy" process, transferring matter between vertical layers. See Section 3.1 'Special handling of settling process' in the MIKE ECO Lab Scientific Documentation.
Unit	Textual information
Expression	MIKE ECO Lab expression to calculate value



**Note:**

The change rates described by a process must be expressed as change **per day**, regardless of the time step used in the model setup. The actual time integration of the ODE is automatically performed based on the model time step and requires no explicit time scaling by the user!

### 5.3.6 Derived outputs

Derived outputs can be used to create output information based on the state of all state variables after the time integration. Thus an auxiliary variable and a derived output with the same expression (e.g. summing up total nitrogen) will have different values as the data of the auxiliary variable is based on the information/values of a state variable at the begin of the current time step and that of a derived output on the information at the end of the current step. As derived outputs can not be used in other MIKE ECO Lab expressions no symbol name can be declared.

Table 5.9 Description of derived output

Property	Description
Symbol	Not applicable ( <i>Derived output N</i> )
Description	A short textual description
Online help	Displayed as mouse-over help in the setup
Documentation	Possible link to detailed external documentation (pdf-file, etc.)
Scope	Not applicable ( <i>NOT SPECIFIED</i> )
Spatial variation	Definition if the output is globally defined or may vary spatially
Unit	Textual information
Expression	MIKE ECO Lab expression to calculate value

## 5.4 Scopes

Some MIKE ECO Lab items like concentration state variables, constants, etc. have a property called 'Scope'. Such a scope describes the vertical position of that item in the water column. A scope is mainly intent to organise a template and items with non-overlapping scopes can not interact with each other (e.g. an item with scope='WS' is only defined at the water surface and can not directly interact with an other item defined in the sediment(scope='SED') because they are separated by the water column (items with scope='WC')).

The concept of the scope is most important in multi-layered model setups, i.e. MIKE 3. The following scopes are defined:

Table 5.10    Scopes

Scope	Stands for	Meaning
WS	Water Surface	Just defined in the surface water
WC	Water Column	Defined in the whole water column
WB	Water Bed	Defined as the bed layer between the water column and the sediment
SED	Sediment	Only defined in the sediment layer

The following scheme outlines the defined scopes:

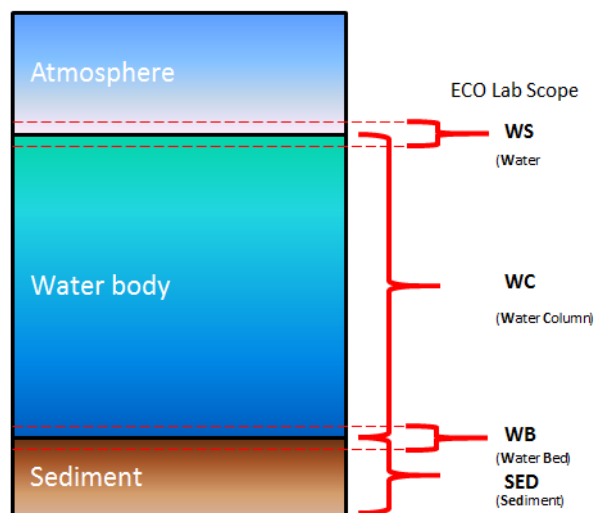


Figure 5.2    MIKE ECO Lab Scopes

## 5.5 Spatial Variabilities

The spatial variability defines the spatial variability of a MIKE ECO Lab item. There are three different spatial variabilities defined:

Table 5.11 Spatial variability

Spatial variability	Explanation
None	Globally defined, same value in all grid points
Horizontal	2D, possibly different values in each horizontal grid point but without variation in the water column
Horizontal and vertical	3D, possibly different values in each horizontal grid point and vertically throughout the water column

The attribute is introduced to avoid wasting computer memory and redundant calculations, so for instance a globally defined auxiliary variable only allocates memory for one number and is only calculated once per time step.

## 5.6 Expressions

Expressions must be specified for state variables, auxiliary variables, processes, derived outputs and ABM state variables, arithmetic expressions and movement vectors. An expression is a sequence of arguments and operators. A 'IF-THEN-ELSE' control structure is also available.

The way expressions are formulated and computed in MIKE ECO Lab is quite similar to a normal spreadsheet calculation: the calculation is essential a 'one-liner' and must result in a single value. Valid arguments are numeric constants, already defined symbols (variables, constants etc.) and function calls.

Valid operators are the standard mathematical operators "+", "-", "\*", and "/".

Function calls can have between zero and up to 6 arguments and result in a single numeric value. Function arguments are separated by commas by default; for simplicity and compatibility to other environments a colon may also be used.

### 5.6.1 Control structures

The MIKE ECO Lab expression syntax offers simple IF-THEN-ELSE control structures. Actually two forms are supported:

- A simple in-line IF-THEN-ELSE control structure  
**IF** (relational\_condition, *Then\_expression*, *Else\_expression*)



- A full specified form  
**IF** (relational\_condition) **THEN** *Then\_expression* **ELSE** *Else\_expression*

It is possible to use other control structures as "then" or "else" expressions, i.e. it is possible to nest various if-then-else constructs to generate complex control structures. Please note that, depending on the result of the relational condition, either the *Then\_expression* or the *Else\_expression* is calculated, i.e. the other is skipped. This may have side effects (random number sequence, direct manipulations etc.).

## Relational\_condition

### Simple comparison

The result of the relational condition determines which branch of the if-then-else control structure is calculated. A relational condition may be a simple comparison:

Expression *OPERATOR* expression

### Valid operators

Operator	Description
==	Equals
!=	Not equal
>	Larger
<	Smaller
>=	Larger or equal
<=	Smaller or equal

Expressions can be all valid MIKE ECO Lab expression, i.e. variables, function calls or control structures.

### Complex Relational conditions

Since version 2017 MIKE ECO Lab supports complex relational conditions, i.e. the logical combination of various simple conditions by logical AND, OR and NOT operators. The MIKE ECO Lab syntax accepts both default written "AND"/ "OR"/ "NOT" or the "C"-style operators "&&"/ "||"/ "!". Please note that the C-style NOT operator ("!") can not be used in combination with the AUTOCAL tool. The evaluation order of multiple conditions is determined by the operator precedence and follows the standard mathematical definition.

Logic operator		Description	Precedence
Default	"C"-Style		
NOT	!	Logical NOT, negation	1
AND	&&	Logical AND	2
OR		Logical OR	3



To force a certain precedence, parentheses may be used.

Examples	Explanation
IF (A > 0, 1, 0)	Simple conditional expression resulting in either 1 or 0, dependent if the variable A is larger than zero or not.
IF ( !(A > 0), 1, 0)	Similar as above but negating the expression, i.e. the result will only be 1 if A is not larger than zero.
IF (A > 0 AND B > 0, 1, 0)	Complex conditional expression, becoming 1 only if both A and B are larger than 0.
IF ((A>0) AND (B>0), 1, 0)	Same as above but using parentheses.
IF (A>0 AND B>0 OR C>0, 1, 0)	Complex expression. As the 'AND' operator has a higher precedence than the "OR" the two first conditions will be evaluated first, i.e. the overall result will be 1 if both A and B are larger than zero or if C is larger than zero, i.e. the expression is equivalent to "IF( (A>0 AND B>0) OR (C>0), 1, 0)".
IF (A>0 && (B>0    C>0), 1, 0)	Complex expression similar to the last using "C" style operators. The parentheses around B and C ensure that these will be valuated first i.e. the overall result will only be 1 if A and either B or C are larger than zero.

## 5.7 Built-in Constants

Table 5.12 lists the most common built-in constants available in MIKE ECO Lab. The MIKE ECO Lab editor may list additional elements but these are not of general interest.

Table 5.12 Built-in constants

Name	Scope	Spatial	Unit	Default	Min	Max
LATITUDE	Global	-	Degrees	55.00	-90.00	90.00
LONGITUDE	Global	-	Degrees	12.00	-180.00	180.00
BED_LEVE	Global	2D	m	0.00	-12000.00	8000.00
AD_DT	Global	-	Seconds	30.00	1.00	86400.00
WQ_DT	Global	-	Seconds	30.00	1.00	86400.00



Table 5.12 Built-in constants

Name	Scope	Spatial	Unit	Default	Min	Max
GRIDSPACING	Global	3D	m	0.00	0.00	1.00E+10
MIKE11_NODE_- TYPE	Global	3D		0.00	0.00	4.00
MIKE_SHE_SUP- PLIED_CONSTANT	WC	3D	auto	0.00	- 1.0E+08	1.0E+10

## 5.8 Built-in Forcings

Table 5.13 lists the most common built-in forcings available in MIKE ECO Lab. The MIKE ECO Lab editor may list additional elements but these are not of general interest.

Table 5.13 Built-in forcings

Name	Scope	Spatial	Unit	Default	Min	Max
WATER_LAY- ER_HEIGHT	WC	3D	m	2.00	0.00	8000.00
SALINITY	WC	3D	psu	0.00	0.00	350.00
TEMPERATURE	WC	3D	degrees C	10.00	0.00	100.00
WIND_VELOCITY	WS	2D	m/s	2.00	0.00	100.00
WIND_DIRECTION	WS	2D	degrees	0.00	0.00	360.00
WATER_SUR- FACE_LEVE	Global	2D	m	0.00	0.00	200.00
WATER_DEPTH	WC	2D	m	8.00	0.00	12000.00
HORIZONTAL_CUR- RENT_SPEED	WC	3D	m/s	0.20	0.00	10.00
HORIZONTAL_CUR- RENT_DIRECTION	WC	3D	degrees	0.00	0.00	360.00
VERTICAL_CUR- RENT_SPEED	WC	3D	m/s	0.00	0.00	10.00
WATER_SUR- FACE_SLOPE_IN_- FLOW_DIRECTION	Global	2D	m/m	0.00	-10.00	10.00
BED_AREA_OF_GRI DCEL	Global	2D	m <sup>2</sup>	1.00E+0 5	0.00	1.00E+10

Table 5.13 Built-in forcings

Name	Scope	Spatial	Unit	Default	Min	Max
FLOOD- ED_AREA_OF_GRID CEL	Global	2D	m2	1.00E+0 5	0.00	1.00E+10
SUR- FACE_AREA_OF_G RIDCEL	Global	2D	m2	1.00E+0 5	0.00	1.00E+10
VOLUME_OF_GRID- CEL	Global	3D	m3	2.00E+0 5	0.00	1.00E+10
HORISON- TAL_GRID_CELL_- FLUX	Global	3D	m3/s	0.00	0.00	1.00E+10
VERTICAL_GRID_- CELL_FLUX	Global	3D	m3/s	0.00	0.00	1.00E+10
WATER_DENSITY	Global	3D	ton/m3	1.00	0.00	20.00
CHEZY_NO	Global	2D	m <sup>1/2</sup> /s	0.00	0.00	1.00E+10
WATER_SUR- FACE_WIDTH	WS	2D	m	0.00	0.00	1.00E+10
HEAT_FLUX_WA- TER_ATMOSPHERE	Global	2D	W/m2	0.00	0.00	1.00E+10
HORIZONTAL_ED- DY_VISCOSITY	WC	3D	m2/s	0.00	0.00	1.00E+08
MIKE_SHE_SUP- PLIED_FORCING	WC	3D	auto	0.00	1.00E+0 8	1.00E+10
VERTICAL_DISPER- SION	WC	3D	m2/s	0.10	0.00	1000.00
HORIZONTAL_DIS- PERSION	WC	3D	m2/s	0.10	0.00	1000.00
TURBULENT_KI- NETIC_ENERGY	WC	3D	m2/s2	0.00	0.00	1000.00
DISSIPATION_TUR- BULENT_KINET- IC_ENERGY	WC	3D	m2/s3	0.00	0.00	1000.00
BED_SHEAR_STRE SS	WB	2D	Pa	0.00	0.00	1000.00





Note: Depending on the hydraulic engines and enabled features not all constant/forcing can be supplied automatically. For example vertical current information is only available in a 3D model, a 2D or 1D engine can not provide this information. Similar temperature and density will be just transferred from the hydrodynamic calculations when the according information is enabled, e.g. when a heat balance is switched on or density variation is considered in the HD calculation. If a built-in constant/forcing is utilised in a template the user must therefore check if the hydrodynamic setup can provide the data. In this case the 'type' element listed in the setup interface will read 'built-in'. If the engine can not provide data, the type will switch to 'user defined' and the necessary data can be specified as with any other user defined constant/forcing.

## 5.9 Built-in Functions

MIKE ECO Lab provides a set of so-called built-in functions. These are mathematical functions like sinus/cosinus or special domain specific functions that can be used inside MIKE ECO Lab expressions. Below you will find a list of the available, built-in functions:

- Standard mathematical functions
- Trigonometric functions
- Date/time functions
- Random number functions
- Biological/aquatic domain functions
- Special ABM functions
- Special functions to load/save reference values
- Averaging functions for concentration variables
- Table functions
- pH functions

If not otherwise noted all functions are defined in **R** (floating point numbers) and all arguments are in **R** as well (MIKE ECO Lab internally uses a double precision floating point format). For each function you will find a short description, the needed arguments, a formal definition and possibly one or more examples.

### 5.9.1 Standard mathematical functions

#### EXP

**Description:**

Compute the exponential of an argument

**Arguments:**

x

**Definition:**

$$EXP(x) = e^x \quad (5.2)$$

**Example(s):**

$$EXP(2.302585093) = 10.0$$

## ABS

**Description:**

Compute the absolute, positive value of an argument

**Arguments:**

x

**Definition:**

$$ABS(x) = |x| \quad (5.3)$$

**Example(s):**

$$ABS(1.3) = 1.3$$

$$ABS(-1.3) = 1.3$$

## SIGN

**Description:**

Computes the sign of the argument

**Arguments:**

x

$$\text{Definition: } SIGN(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

## LN

**Description:**

Compute the natural logarithm of an argument

**Arguments:**

x

**Definition:**

$$LN(x) = \ln(x) \quad (5.4)$$

**Example(s):**

LN(10.0) = 2.30259...

**LOG10****Description:**

Compute the decadic logarithm of an argument

**Arguments:**

x

**Definition:**

$$LOG10(x) = \frac{\ln(x)}{\ln(10)} \quad (5.5)$$

**Example(s):**

LOG10 (1000) = 3.0

**MAX****Description:**

Returns the largest of two arguments

**Arguments:**

x, y

**Definition:**

$$MAX(x, y) = \begin{cases} \text{if } (x \geq y), x \\ \text{if } (x < y), y \end{cases} \quad (5.6)$$

**Example(s):**

MAX(1.0, 2.0) = 2.0

**MIN****Description:**

Returns the smallest of two arguments

**Arguments:**

x, y

**Definition:**

$$MIN(x, y) = \begin{cases} \text{if } (x \leq y), x \\ \text{if } (x > y), y \end{cases} \quad (5.7)$$

**Example(s):**

MIN(1.0, 2.0) = 1.0



## POW

**Description:**

Compute first argument raised the power of the second argument

**Arguments:**

x, y

**Definition:**

$$POW(x, y) = x^y \quad (5.8)$$

**Example(s):**

$POW(2.0, 3.0) = 8.0$

$POW(16, 0.5) = 4.0$

## SQR

**Description:**

Compute the square of an argument

**Arguments:**

x

**Definition:**

$$SQR(x) = x \cdot x \quad (5.9)$$

**Example(s):**

$SQR(2.0) = 4.0$

## SQRT

**Description:**

Compute the square root of an argument

**Arguments:**

x

**Definition:**

$$SQRT(x) = \sqrt{x} \quad (5.10)$$

**Example(s):**

$SQRT(4.0) = 2.0$

$SQRT(-1) = -1.\#IND$



## FLOOR

**Description:**

Compute the largest integer value less than or equal to an argument

**Arguments:**

x

**Definition:**

$$FLOOR(x) = \lfloor x \rfloor \quad (5.11)$$

**Example(s):**

FLOOR(2.4) = 2.0

FLOOR(-2.4) = -3.0

FLOOR(x + 0.5) : round x to the nearest integer value

## CEIL

**Description:**

Compute the smallest integer value greater than or equal to an argument

**Arguments:**

x

**Definition:**

$$CEIL(x) = \lceil x \rceil \quad (5.12)$$

## MOD

**Description:**

Compute the floating point remainder of the division of two arguments

**Arguments:**

x, y

**Definition:**

$$MOD(x, y) = \begin{cases} 0,0 & \text{if } (y = 0) \\ x - i \cdot y & \text{if } (i \in \mathbb{N}, |x - i \cdot y| < |y|) \end{cases} \quad (5.13)$$

**Example(s):**

MOD(10.0, 3.0) = 1.0

MOD(10.0, -5.5) = 4.5

## PI

**Description:**

Return the numeric constant Pi

**Arguments:**

none

**Definition:**

$$PI() = \pi \quad (5.14)$$

## 5.9.2 Trigonometric functions

By default trigonometric functions expect the angle unit degree. This default angle unit was set when introducing the ABM modelling capabilities in MIKE ECO Lab<sup>(1)</sup> to handle angle information (forcing/constant, expression) using the same format. The angle unit can be changed to radians manually using the miscellaneous settings.



Old templates without a miscellaneous information section will cause the default angle unit to be set to 'radians'!

The user can use the functions RAD2DEG and DEG2RAD to manually convert arguments from radians to degree and back.

### RAD2DEG

**Description:**

Convert the argument from radians to degree

**Arguments:**

angle in radians

**Definition:**

$$RAD2DEG(x) = x \cdot \frac{180}{\pi} \quad (5.15)$$

**Example(s):**

$$RAD2DEG(1.5708) = 90.0$$

### DEG2RAD

**Description:**

Convert the argument from degree to radians

---

<sup>1</sup> MIKE by DHI Release 2012

**Arguments:**

angle in degree

**Definition:**

$$DEG2RAD(x) = x \cdot \frac{\pi}{180} \quad (5.16)$$

**Example(s):**

$$DEG2RAD(45) = 0.7854$$

**COS****Description:**

Compute the cosine of an argument

**Arguments:**

x, angle in degree (default, may be set to radians)

**Definition:**

$$COS(x) = \cos x \quad (5.17)$$

**Example(s):**

$$COS(180) = -1.0$$

**SIN****Description:**

Compute the sine of an argument

**Arguments:**

x, angle in degree (default, may be set to radians)

**Definition:**

$$SIN(x) = \sin x \quad (5.18)$$

**Example(s):**

$$SIN(90.0) = 1.0$$

**TAN****Description:**

Compute the tangent of an argument

**Arguments:**

x, in degree (default, may be set to radians)

**Definition:**

$$TAN(x) = \tan x \quad (5.19)$$

**Example(s):**

TAN(45) = 1.0

**COSH****Description:**

Compute the hyperbolic cosine of an argument

**Arguments:**

x, in degree (default, may be set to radians)

**Definition:**

$$COSH(x) = \cosh x \quad (5.20)$$

**Example(s):**

COSH(0.5) = 1.12763...

**SINH****Description:**

Compute the hyperbolic sine of an argument

**Arguments:**

x, in degree (default, may be set to radians)

**Definition:**

$$SINH(x) = \sinh x \quad (5.21)$$

**Example(s):**

SINH(0.5) = 0.521095...

**TANH****Description:**

Compute the hyperbolic tangent of an argument

**Arguments:**

x, in degree (default, may be set to radians)

**Definition:**

$$TANH(x) = \tanh x$$

**Example(s):**

TANH(0.5) = 0.462117...





## ARCCOS

**Description:**

Compute arccosine, the inverse of the cosine function in degree (default, may be set to radians)

**Arguments:**

$x, 0 \leq x \leq \pi$

**Definition:**

$$ARCCOS(x) = \cos^{-1} x \quad (5.22)$$

**Example(s):**

$ARCCOS(-1) = 180$

## ARCSIN

**Description:**

Compute arcsine, the inverse of the sine function in degree (default, may be set to radians)

**Arguments:**

$x, -\frac{\pi}{2} \leq x \leq \frac{\pi}{2}$

**Definition:**

$$ARCSIN(x) = \sin^{-1} x \quad (5.23)$$

**Example(s):**

$ARCSIN(1) = 90$

## ARCTAN, ARCTAN2

**Description:**

Compute arctangent of the argument  $x$ , ARCTAN2 of  $y/x$ , in degree (default, may be set to radians)

**Arguments:**

ARCTAN:  $x$

ARCTAN2:  $x, y$

**Definition:**

$$ARCTAN(x) = \arctan(x) \quad (5.24)$$

$$ARCTAN2(x, y) = \arctan\left(\frac{y}{x}\right)$$

**Example(s):** $\text{ARCTAN}(0.5) = 26.5650$  $\text{ARCTAN2}(0.5, 5.0) = 84.2894$  $\text{ARCTAN2}(1, 3) = 71.5650$ 

### 5.9.3 Date/time functions

MIKE ECO Lab provides a number of functions to handle date/time calculations. These functions can be used to e.g. calculate the relative day length, time of sunrise/sunset based on the model location. An other subset of the built-in functions/reserved words allows to query the current simulation time.

#### DAYNUMBER

**Description:**

Calculate the sequential number of the day in a year on the a given date

**Arguments:**

Year, Month, Day of month

**Definition:**

*See description*

**Example(s):** $\text{DAYNUMBER}(2010, 6, 1) = 152$  (no leap year) $\text{DAYNUMBER}(2008, 6, 1) = 153$  (2008 was a leap year!)

#### RELATIVE\_DAYLENGTH

**Description:**

Compute the relative length of the daytime for a given location and date. The value is about 0.5 at equinox (around 20 March and 23 September) when day and night have the same length. The min/max values are around the solstices (21 June, 22 December).

**Arguments:**

Year, Month, Day, Latitude in degree

**Definition:**

day = *Daynumber*(year, month, day)

$$\phi = -\tan\left(\text{Latitude} \cdot \frac{\pi}{180}\right) \cdot \tan\left(23.45 \cdot \frac{\pi}{180} \cdot \sin\left(2 \cdot \pi \cdot \frac{284 + \text{day}}{365}\right)\right) \quad (5.25)$$

$$\text{Relative\_Daylength} = \begin{cases} 1, & \text{if } \phi \leq -1 \text{ polar day, sun shines 24h/day} \\ 0, & \phi \geq 1 \text{ sun does not rise at all} \\ \frac{\cos^{-1}\phi}{\pi} & \text{normal sunrise/sunset} \end{cases}$$

**Example(s):**

RELATIVE\_DAYLENGTH( 2010, 3, 20, 55.67) =0.4931  
(spring equinox)

RELATIVE\_DAYLENGTH( 2010, 6, 21, 55.67) =0.72892  
(summer solstice)

RELATIVE\_DAYLENGTH( 2010, 9, 23, 55.67) =0.49148  
(autumn equinox)

RELATIVE\_DAYLENGTH( 2010, 12, 22, 55.67) =0.27115  
(winter solstice)

## SUNRISE

**Description:**

Compute the approximate time of the sunrise (in hours from midnight local time) for a given location and date.

**Arguments:**

Year, Month, Day, Latitude in degree, fi (time displacement)

fi: time correction giving the displacement of the solar radiation maximum from 12:00

**Definition:**

$$\begin{aligned} \text{Sunhours} &= \text{RELATIVE\_DAYLENGTH}(\text{Year}, \text{Month}, \text{Day}, \text{Latitude}) \\ \text{Sunrise} &= (12:00 + \text{fi}) - 0,5 \cdot 24 \cdot \text{Sunhours} \end{aligned} \quad (5.26)$$

**Example(s):**

SUNRISE(2010, 3, 22, 55.67, 1) = 7.0

(Sunrise 2010/03/22 in Copenhagen approx. 7:00 local time)

## SUNSET

**Description:**

Compute the approximate time of sunset (in hours from midnight local time) for a given location and date

**Arguments:**

Year, Month, Day, Latitude in degree, fi (time displacement)

fi: time correction giving the displacement of the solar radiation maximum from 12:00

**Definition:**

$$\begin{aligned}\text{Sunhours} &= \text{RELATIVE\_DAYLENGTH}(\text{Year}, \text{Month}, \text{Day}, \text{Latitude}) \\ \text{Sunset} &= \text{SUNRISE}(\text{Year}, \text{Month}, \text{Day}, \text{Latitude}, \text{fi}) + \text{Sunhours}\end{aligned}\quad (5.27)$$

**Example(s):**

SUNSET (2010, 3, 22, 55.67, 1) = 19.0

(Sunset 2010/3/22 in Copenhagen approx. 19:00 local time)

## MOONPHASE

**Description:**

This function calculates the current moon phase. The result is a number between 0.0-1.0. The value of 0.0 indicates a full moon that transits to new moon (0.5) and then to 1.0 (next full moon).

**Arguments:**

Year, Month, Day, Hour, Minute, Second

**Definition:**

The moon phase is calculated as the fractional remainder of the date difference between the given date and the 31 December 2009 20:12:36 (full moon) divided by the average length of the synodic month (29.530588 days). One earth day corresponds to about  $\sim 0.034$  moon phase, i.e. to determine if there is full moon a given date  $\pm 12$  h you can use the equation like:

$$\text{IF}(\text{ABS}(\text{MOONPHASE}(\text{YEAR}, \text{MONTH}, \text{DAY}, \text{HOUR}, \text{MINUTE}, \text{SECOND}) - 0.5) < (0.5 - 0.034/2), 0, 1)$$

Or analogue for new moon:

$$\text{IF}(\text{ABS}(\text{MOONPHASE}(\text{YEAR}, \text{MONTH}, \text{DAY}, \text{HOUR}, \text{MINUTE}, \text{SECOND}) - 0.5) < 0.034/2, 1, 0)$$



To determine the moon quarter:

$\text{Moon} = \text{MOONPHASE}(\text{YEAR}, \text{MONTH}, \text{DAY}, \text{HOUR}, \text{MINUTE}, \text{SECOND})$

$\text{ceil}(\text{IF} (\text{Moon} > 0.5, \text{Moon}-0.5, \text{Moon}+0.5) * 4)$



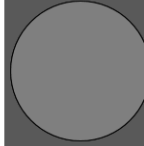
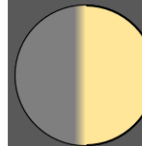
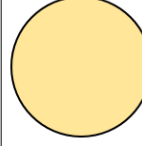
Full Moon	Third Quarter	New Moon	First Quarter	Full Moon
				
0.00	0.25	0.50	0.75	1.00

Figure 5.3 Example of moonphase coding

**Example(s):**

$\text{MOONPHASE} (2009, 12, 31, 20, 12, 36) = 0.0$  COND = 10

## SECOND

**Description:**

Return the number of seconds in current simulation time minute

**Arguments:**

none

**Example(s):**

SECOND = 10 (assumed simulation date 2010/03/22 12:35:10)

## MINUTE

**Description:**

Return the minutes of the current simulation time hour

**Arguments:**

none

**Example(s):**

MINUTE =35 (assumed simulation date 2010/03/22 12:35:10)

## HOUR

**Description:**

Return the current hour of the day for the simulation time

**Arguments:**

none

**Example(s):**

HOUR = 12 (assumed simulation date 2010/03/22 12:35:10)

## DAY

**Description:**

Return the current day of a month for the simulation time

**Arguments:**

none

**Example(s):**

DAY = 22 (assumed simulation date 2010/03/22 12:35:10)

## MONTH

**Description:**

Return the current month for the simulation time

**Arguments:**

none

**Example(s):**

MONTH = 3 (assumed simulation date 2010/03/22 12:35:10)

## YEAR

**Description:**

Return the current year for the simulation time

**Arguments:**

none

**Example(s):**

YEAR = 2010 (assumed simulation date 2010/03/22 12:35:10)

### 5.9.4 Random number functions

MIKE ECO Lab provides a couple of functions to derive pseudo- random numbers. By default the random generator will be seeded in a way to generate a different sequence of random numbers each run. For debugging purpose a defined seed (different from zero) will result in the same sequence of random numbers each time the model is executed.



The RANDSEED parameter is located in the 'Miscellaneous'– parameter properties of a template.

## RAND

**Description:**

Returns a uniform distributed random number in the interval [0..1]

**Arguments:**

none

**Usage:**

Generate a uniform random number between 0-1

**Note:**

This is the same call as 'U\_RANDOM' and available for compatibility with older MIKE ECO Lab templates. New developments should only use the 'U\_RANDOM' call!

## U\_RANDOM

**Description:**

Returns a uniform distributed random number in the interval [0..1]

**Arguments:**

none

**Usage:**

Generate a uniform random number between 0-1

## U\_RANDOM2

**Description:**

Returns a uniform distributed random number in the given interval

**Arguments:**

min, max

**Definition:**

$$U\_RAND2(min, max) = min + (max - min) \cdot U\_RAND ( ) \quad (5.28)$$

**Usage:**

Generate a uniform random number in a given interval

## N\_RANDOM

**Description:**

Returns a random number from a standard normal distribution with mean=0.0 and variance = 1.0

**Arguments:**

none

**Usage:**

Generate a normal distributed random factor, e.g. speed change



## N RAND2

**Description:**

Returns a random number from a normal distribution with specified mean and variance.

**Arguments:**

mean, variance

**Definition:**

$$N\_RAND2(\text{mean}, \text{variance}) = \text{mean} + \sqrt{\text{variance}} \cdot N\_RAND ( ) \quad (5.29)$$

**Usage:**

Generate a normal distributed random, e.g. average length increase.

## E RAND

**Description:**

Returns an exponentially distributed, positive random deviate of unit mean

**Arguments:**

none

**Definition:**

$$E_{RAND} = -\ln(U\_RAND ( )) \quad (5.30)$$

**Usage:**

Waiting time between independent Poisson-events

## P RAND

**Description:**

Returns a random deviate (integer number) drawn from a Poisson distribution of given mean.

**Arguments:**

mean

## G RAND

**Description:**

Returns a deviate from a gamma distribution of integer order

**Arguments:**

order





## B RAND

**Description:**

Returns a deviate (integer number) from a binomial distribution for an elementary probability P and N trails.

**Arguments:**

probability P, N trails.

### 5.9.5 Biological/aquatic domain functions

MIKE ECO Lab provides a couple of built-in biological and aquatic domain specific functions that can be used in MIKE ECO Lab. These functions include temperature models, functions to compute oxygen saturation concentrations and more.

## OXYGENSATURATION

**Description:**

Compute an oxygen saturation concentration for a given salinity and temperature.

**Arguments:**

salinity, temperature

**Definition:**

$$\text{conc\_DO}(\text{salinity}, \text{temp}) = 14,65 - 0,0841 \cdot \text{salinity} + \text{temp} \cdot (0,00256 \cdot \text{salinity} - 0,41022 + \text{temp} \cdot (0,007991 - 0,0000374 \cdot \text{salinity} - 0,000077774 \cdot \text{temp})) \quad (5.31)$$

**Example(s):**

OXYGENSATURATION (32, 20) = 7.48829

## OXYGENSATURATION\_WEISS

**Description:**

Compute an oxygen saturation concentration for a given salinity and temperature based on the equation of Weiss (1970) 'Solubility of Nitrogen, Oxygen and Argon in Water and Sea Water' Deep-Sea Research 17:721-735

**Arguments:**

salinity, temperature

**Definition:**

$$\begin{aligned}
 conc_{DO}(salinity, temp) &= \frac{e^a}{0,69997} \\
 t &= \frac{temp + 273,15}{100} \\
 e &= \frac{249,6339}{t} \\
 d &= 143,3483 \cdot \log(t) \\
 c &= -21,8493 \cdot t \\
 b &= salinity \cdot (-0,033096 + 0,014259 \cdot t - (0,0017 \cdot t^2)) \\
 a &= -173,4192 + b + c + d + e
 \end{aligned} \tag{5.32}$$

## ARRHENIUS5

**Description:**

Returns the temperature dependency based on a simple Arrhenius expression with a reference temperature of 5°C

**Arguments:**

theta (temperature coefficient), temperature

**Definition:**

$$ARRHENIUS5(theta, temp) = theta^{temp - 5,0} \tag{5.33}$$

**Example(s):**

ARRHENIUS5( 1.06,20.0)=2.3965

## ARRHENIUS20

**Description:**

Returns the temperature dependency based on a simple Arrhenius expression with a reference temperature of 20°C

**Arguments:**

theta (temperature coefficient), temperature

**Definition:**

$$ARRHENIUS20(theta, temp) = theta^{temp - 20,0} \tag{5.34}$$

**Example(s):**

ARRHENIUS20( 1.06,20.0)=1



## LASSITER

### Description:

Compute a temperature corrected (growth) rate according to the Lassiter model. The Lassiter temperature model is an optimum temperature model.

### Arguments:

Lassiter Constant  $\alpha$ , rate  $k_m$ , optimum temperature  $T_m$ , lethal temperature  $T_L$ , actual temperature  $T$

### Definition:

The Lassiter-Model is defined as:

$$LASSITER(\alpha, k_m, T_m, T_L, T) = \quad (5.35)$$

$$\begin{cases} k_m \cdot \exp(\alpha \cdot (T - T_m)) \cdot \left( \frac{T_L - T}{T_L - T_m} \right)^{\alpha \cdot (T_L - T_m)}, & T \leq T_L \\ 0, & T > T_L \end{cases}$$

In this model  $T$  is the current,  $T_L$  the lethal,  $T_m$  the optimal temperature,  $k_m$  the maximal (growth) rate and  $\alpha$  a specific constant.

The constants  $\alpha$  and  $k_m$  may be fitted. If the growth rates  $k_1$  and  $k_2$  at two different temperatures  $T_1$  and  $T_2$  are known, Lassiter gives the following equations (you need to know/estimate  $T_L$  and  $T_m$ ):

$$\alpha = \frac{\ln(k_1 - k_2)}{(T_1 - T_2) + (T_L - T_m) \cdot \ln\left(\frac{T_L - T_1}{T_L - T_2}\right)} \quad (5.36)$$

$$k_m = k_1 \cdot \exp(-\alpha \cdot (T_1 - T_m)) \cdot \frac{(T_L - T_1)^{-\alpha \cdot (T_L - T_m)}}{(T_L - T_m)}$$

## MICHAELIS\_MENTEN1

### Description:

Compute a simple limitation factor based on a half saturation constant

**Arguments:**

concentration, half saturation

**Definition:**

$$\text{MICHAELIS\_MENTEN1} = \frac{\text{concentration}}{(\text{concentration} + \text{half\_saturation})} \quad (5.37)$$

**Example(s):**

MICHAELIS\_MENTEN1 (10, 3) = 0.76931

## MICHAELIS\_MENTEN2

**Description:**

Compute a simple limitation factor based on a half saturation constant

**Arguments:**

concentration, half saturation

**Definition:**

$$\text{MICHAELIS\_MENTEN2} = \frac{\text{concentration}^2}{(\text{concentration}^2 + \text{half\_saturation})} \quad (5.38)$$

**Example(s):**

MICHAELIS\_MENTEN2 (10, 3) = 0.97087

## REVERSE\_MICHAELIS\_MENTEN

**Description:**

Compute a simple inhibiting factor

**Arguments:**

concentration

**Definition:**

$$\text{REVERSE\_MICHAELIS\_MENTEN} = \frac{1}{(1 + \text{concentration})} \quad (5.39)$$

**Example(s):**

REVERSE\_MICHAELIS\_MENTEN(10) = 0.0909..

## LAMBERT\_BEER\_1

**Description:**

This function estimates the light penetration in water column based on Lambert Beer expression. The light intensity in the top of each grid layer is returned

**Arguments:**

Surface Light, Layer Height, Light Extinction Coefficient

**Definition:**

$$Lambert_{Beer1} = light \cdot e^{-eta \cdot Z_{top}} \quad (5.40)$$

**LAMBERT\_BEER\_2****Description:**

This function estimates the light penetration in water column based on Lambert Beer expression. The light intensity in the bottom of each grid layer is returned

**Arguments:**

Surface Light, Layer Height, Light Extinction Coefficient

**Definition:**

$$Lambert_{Beer2} = light \cdot e^{-eta \cdot Z_{bottom}} \quad (5.41)$$

**LAMBERT\_BEER\_AVR****Description:**

This function computes the average light intensity at a given depth by integrating over the complete layer thickness. Please note that this function does not work as the other two built in light functions, i.e. it does not automatically take the attenuation form above layers into account. However, you can easily combine it with the LAMBERT\_BEER\_1 function to do so.

**Arguments:**Light intensity I<sub>0</sub> at top of layer, layer height z, attenuation coefficient λ**Definition:**

$$I_{avr} = \frac{\int_0^z I_0 \cdot e^{-\lambda \cdot x} dx}{\int_0^z 1 dx} = \frac{I_0 (1 - e^{-\lambda \cdot z})}{\lambda \cdot z} \quad (5.42)$$

**Example(s):**

LAMBERT\_BEER\_AVR (1000, 10, 0.1) = 632.121

**SECCHI\_DEPTH****Description:**

Returns the Secchi depth at a horizontal point as function of a vertically varying light extinction (1/m). The dampening percentage is included, so that the user can specify own definitions of Secchi depth. Secchi depth is the depth at

which the light intensity has been dampened by a given percentage relative to the surface light intensity. Note that at shallow water the Secchi depth will not describe the transparency of the water because the function can not return values higher than the water depth. An alternative is to look at the extinction coefficient itself. An often used definition of Secchi depth uses a dampening percentage of 80.



The variable referring to this function must be specified with spatial variation 'Horizontal and Vertical' in order for the function to use the vertical varying extinction coefficient. The calculated Secchi depth is returned in the top layer in output of multi-layered model systems, the values in the layers below can be ignored.

**Arguments:**

Light Extinction Coefficient, Dampening Percentage

## GET\_WATER\_DENSITY

**Description:**

Compute the density of water of a given salinity and temperature according to the Unesco technical papers in marine science #44 ("Algorithms for computation of fundamental properties of seawater", N.P. Fofonoff, R.C. Millard 1983).

**Arguments:**

Salinity [psu], Temperature [deg.C],

**Definition:**

See reference

**Example(s):**

GET\_WATER\_DENSITY

## GET\_DEPTH\_PRESSURE

**Description:**

This function calculates the absolute pressure in bars at a given depth. It conforms to the calculations according to the Unesco technical papers in marine science #44 ("Algorithms for computation of fundamental properties of seawater", N.P. Fofonoff, R.C. Millard 1983), albeit the assumed latitude is constant 45.716 deg. north. It returns the absolute pressure, thus the sum of the relative hydrostatic pressure plus one bar atmospheric pressure.

**Arguments:**

Depth [m], Salinity [psu], Temperature [deg.C],

**Definition:**

See reference

**Example(s):**

GET\_DEPTH\_PRESSURE (0, 35, 20) = 1.0 // at surface



GET\_DEPTH\_PRESSURE (100, 35, 20) = 11.054 // at 100m depth

### 5.9.6 Special ABM functions

Agent/Individual based models may require some special functions to query the individuals position or the result of some vector additions.

#### GETX

**Description:**

Compute the X-coordinate of a given polar vector

**Arguments:**

speed, direction in degree

**Definition:**

$$GETX(speed, direction) = speed \cdot \cos\left(direction \cdot \frac{\pi}{180}\right) \quad (5.43)$$

**Example(s):**

GETX( 1.0, 45) = 0.707107

#### GETY

**Description:**

Compute the Y-coordinate of a given polar vector

**Arguments:**

speed, direction in degree

**Definition:**

$$GETY(speed, direction) = speed \cdot \sin\left(direction \cdot \frac{\pi}{180}\right) \quad (5.44)$$

**Example(s):**

GETY( 1.0, 45) = 0.707107

#### GETDISTANCE<sup>(1)</sup>

**Description:**

Compute the Euclidian distance between two points in space

---

<sup>1</sup> When the projection is in longitude/latitude the GETDISTANCE function expects coordinates in degree longitude/latitude and ignores the z-positions. The distance calculations are based on the Haversine distance calculation on a spheroid earth model with an assumed earth radius  $r = 6371.009$  km. Typical accuracy of the calculated distances is between 0.3-0.5%.

**Arguments:**

$x1, y2, z1, x2, y2, z2$

**Definition:**

$$GETDISTANCE(x1, y1, z1, x2, y2, z2) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2} \quad (5.45)$$

**Example(s):**

$GETDISTANCE(1, 1, 0, 10, 1, 0) = 9.0$

## GETANGLE<sup>(1)</sup>

**Description:**

Compute the angle in degree aligned to true north between two 2D points

**Arguments:**

$x1, y1, x2, y2$

**Definition:**

$$GETANGLE(x1, y1, x2, y2) = \tan^{-1}\left(\frac{x2 - x1}{y2 - y1}\right) \quad (5.46)$$

**Example(s):**

$GETANGLE(0, 0, 1, 1) = 45$

$GETANGLE(0, 0, 1, 0) = 90$

$GETANGLE(0, 0, -1, 0.5) = 296.565$

## RESULTINGSPEED

**Description:**

Compute the speed component of a vector addition of two polar vectors

---

<sup>1</sup> When the projection is in longitude/latitude the GETANGLE function expects coordinates in degree longitude/latitude and ignores the z-positions. The distance calculations are based on the Haversine distance calculation on a spheroid earth model with an assumed earth radius  $r = 6371.009$  km. Typical accuracy of the calculated distances is between 0.3-0.5%



**Arguments:**

speed1, direction1, speed2, direction2

**Definition:**

$$\begin{aligned}
 x1 &= \text{speed1} \cdot \cos(\text{direction1}) \\
 y1 &= \text{speed1} \cdot \sin(\text{direction1}) \\
 x2 &= \text{speed2} \cdot \cos(\text{direction2}) \\
 y2 &= \text{speed2} \cdot \sin(\text{direction2}) \\
 \text{RESULTINGSPEED} &= \sqrt{(x2 - x1)^2 + (y2 - y1)^2}
 \end{aligned}
 \tag{5.47}$$

## RESULTINGDIRECTION

**Description:**

Compute the direction component of a vector addition of two polar vectors

**Arguments:**

speed1, direction1, speed2, direction2

**Definition:**

$$\begin{aligned}
 x1 &= \text{speed1} \cdot \cos(\text{direction1}) \\
 y1 &= \text{speed1} \cdot \sin(\text{direction1}) \\
 x2 &= \text{speed2} \cdot \cos(\text{direction2}) \\
 y2 &= \text{speed2} \cdot \sin(\text{direction2}) \\
 \text{RESULTINGDIRECTION} &= \text{GETANGLE}(x2 + x1, y2 + y1)
 \end{aligned}
 \tag{5.48}$$

## ANGLEBETWEEN

**Description:**

Compute the angle in degrees spanned by two polar vectors

**Arguments:**

speed1, direction1, speed2, direction2

**Definition:**

$$\begin{aligned}
 x1 &= \text{speed1} \cdot \cos(\text{direction1}) \\
 y1 &= \text{speed1} \cdot \sin(\text{direction1}) \\
 x2 &= \text{speed2} \cdot \cos(\text{direction2}) \\
 y2 &= \text{speed2} \cdot \sin(\text{direction2}) \\
 \text{ANGLEBETWEEN} &= \text{GETANGLE}(x2 - x1, y2 - y1)
 \end{aligned}
 \tag{5.49}$$

## DISTANCE\_TO<sup>(1)</sup>

### Description:

Compute the distance for the current active particle/individual to a specified location

### Arguments:

x,y,z

### Definition:

$$DISTANCE\_TO = GETDISTANCE([XPOS], [YPOS], [ZPOS], x, y, z) \quad (5.50)$$

## INSIDE

### Description:

Returns '1' when the current active particle/individual is within the rectangle/box defined by the lower left/upper right coordinates. When z1=z2 only a 2D check is done and the z-coordinates are ignored.

### Arguments:

lower left x,1,y1,z1, upper right x2,y2,z2; when z1=z2 only a 2D check is done and the z-coordinates are ignored.

### Definition:

$$INSIDE = \begin{cases} 1,0 & \text{if } [XPOS, YPOS, ZPOS] \text{ inside} \\ & [(x1, y1, z1) \text{ to } (x2, y2, z2)] \\ else & 0,0 \end{cases} \quad (5.51)$$

## TESTHITSTATE

### Description:

This function can be used to check if a certain hitstate or combination thereof is set

### Arguments:

hitstate, flag to test

### Definition:

$$TESTHITSTATE = \begin{cases} 1,0 & \text{if at least one flag is set in the hitstate} \\ else & 0.0 \end{cases} \quad (5.52)$$

<sup>1</sup> When the projection is in longitude/latitude the DISTANCE\_TO function expects coordinates in degree longitude/latitude and ignores the z-positions. The distance calculations are based on the Haversine distance calculation on a spheroid earth model with an assumed earth radius r = 6371.009 km. Typical accuracy of the calculated distances is between 0.3-0.5%



## TERMINATE

**Description:**

Send a termination request to the hydraulic engine

**Arguments:**

none

## SET\_RADIUS

**Description:**

This function can be used to set the search radius of a specific RASF for the current ABM class. See 'Sensing functions – perception of environment/particles' and 'Search Radius'

**Arguments:**

RASF-Nr (species specific, i.e. 1, 2... etc.), new radius

**Example(s):**

SET\_RADIUS (1, 200)

## SET\_PROJECTION

**Description:**

This function can be used to set the position projection time for the current individual. See 'Movement projection'

**Arguments:**

New projection time in sec

**Example(s):**

SET\_PROJECTION (300)

## SET\_FOV

**Description:**

This function can be used to set the "Field of view" for the current individual. See also Sensing functions – perception of environment/particles' and 'Search Radius'

**Arguments:**

New opening angle in degrees of the FOV

**Example(s):**

SET\_FOV (45)

### 5.9.7 Special functions to load/save reference values

MIKE ECO Lab itself is not able to address/query information outside the current computational point. This means that you cannot query a variable value

from another location (e.g. to be used as a reference value). For this reason a small internal storage memory has been introduced. This functionality can be utilised to save and store any information during a time step and retrieve it in later time steps. To ensure consistency it is only possible to retrieve data from previous time steps, i.e. it takes one time step before a "saved" value will be the result of a "load" call. You can use the "peek" call to inspect the current value saved to the memory slot. If the same memory location is used to store different values at different spatial locations in the model domain during the same time step it is impossible to predict for the user which value will be saved. Thus save calls should only be used selective.

On simulation start-up all memory is initialised to zero. If a storage location outside the defined number of storage memory is specified the functions will return a value of '-1'.



Please note that there is just a limited memory size (100 numbers), thus this feature should by no means be misused to store intermediate calculation results or history data!

## SAVE

### Description:

Store a value into a storage location. Please note that the new value will be active in the next time step when a "LOAD" function is called (you can use the "PEEK" function to query the current value)

### Arguments:

storage place, value

### Example(s):

SAVE(1, 0.5)

saves the value '0.5' into the storage location ; on success it will return the save value i.e. '0.5'.

SAVE(2, DO)

saves the current (location) value of a state variable DO into the location 2.

SAVE (1000, 1)

try to save the value '1' to memory location 1000; this does not exists and the return value will be '-1'

## LOAD

### Description:

Retrieve a value from a storage location. Please note that a "LOAD" value results in the last saved value from previous simulation time, i.e. it cannot give you value saved in the current time step. See the PEEK function.

### Arguments:

storage place

**Example(s):**

LOAD(1)

Load the value saved to storage location 1

LOAD(1000)

Load a value from a not existing location, result will be '-1'.

**PEEK****Description:**

Retrieve the last saved value from a storage location for the current time step. Contrary to a LOAD function the PEEK function will result in the last saved value, i.e. it is possible to inspect the last saved value of the current time step.

**Arguments:**

storage place

**Example(s):**

PEEK(1)

Load the value saved to storage location 1, also including the latest change for the current time step.

PEEK(1000)

Load a value from a not existing location, result will be '-1'.

**5.9.8 Averaging functions for concentration variables**

The following functions can be used to calculate some kind of averaging.



Please note that these functions will only work correctly with concentration variables and should not be applied to Lagrange/ABM variables!

**AVERAGE\_WATER\_COLUMN****Description:**

Returns the average value in the water column of a specified variable with vertical variation. Can be used for estimating primary production per  $\text{m}^2$  if the average water column primary production per  $\text{m}^3$  is multiplied with water depth.



**IMPORTANT:** The variable referring to this function must be specified with spatial variation 'Horizontal and Vertical' in order for the function to find the average the vertical varying variable. The calculated average of the water column is returned in the top layer in output of multi-layered model systems, the values in the layers below can be ignored.

**Arguments:**

state variable

## SMOOTHING\_AVERAGE

**Description:**

Description: Returns a smooth value  $S$  of a variable that has been weighted against both latest value of the variable  $Y$  and historical values of the variable. Can be used in ecology when a variable does have immediate effect on other variables. For instance can long periods of anoxia kill benthic organisms, but maybe not the peaks of low oxygen concentrations. Smoothing average is suitable for describing this phenomena. It has a similar function as moving average, but it has the advantage that it uses much less memory than the moving average function.

**Arguments:**

variable, weight of latest value  $\alpha$  of the variable

**Definition:**

$$S_t = \alpha \cdot \text{Variable}(t) + (1 - \alpha) \cdot S_{t-1} \quad (5.53)$$

## MOVING\_AVERAGE

**Description:**

Returns a moving average of a variable. The average is an average of the  $x$  latest values of the variable. Can be used in ecology when a variable does have immediate effect on other variables. For instance can long periods of anoxia kill benthic organisms, but maybe not the peaks of low oxygen concentrations. Moving average is suitable for describing this phenomena. It has a similar function as smoothing average, but be aware that it is very heavy on memory to use this function.

**Arguments:**

variable,  $x$  number of timesteps to average

### 5.9.9 Table functions

The following functions can be used to map a value to a short table and retrieve the value of an index position in such a table. Due to restrictions in MIKE ECO Lab the tables can only hold 5 elements but nesting tables is also possible. The functions can be used to do some linear interpolation etc.

## TABLE\_INDEX

**Description:**

The TABLE\_INDEX function computes the index, i.e. the relative position of a given value inside a table. If the value is smaller than the 1st table value the result will be "0", if it is larger than the 5th value the result will be "5". Otherwise the result will be relative position within the table, i.e. if the value is exactly between the 1st and the 2nd value the result will be 1.5.

**Arguments:**

Value, TableValue\_1, TableValue\_2, TableValue\_3, TableValue\_4, TableValue\_5

**Definition:****Example(s):**

TABLE\_INDEX (25, 0, 10, 20, 30, 40) = 3.5

TABLE\_INDEX (25, 0, 10, 25, 35, 40) = 3.0

TABLE\_INDEX (25, 0, 10, 20, 35, 40) = 3.3

TABLE\_INDEX (250, 0, 10, 20, 35, 40) = 5

TABLE\_INDEX (-1, 0, 10, 20, 35, 40) = 0

## TABLE\_VALUE

**Description:**

The TABLE\_VALUE function computes the linear interpolated value of an index inside a table. If the index is smaller than 1 or larger 5 the result will be "0". Otherwise the result will be interpolated value between the next table entries, i.e. if the index of 1.5 will return a value exactly between the 1st and the 2nd table value.

**Arguments:**

Index, TableValue\_1, TableValue\_2, TableValue\_3, TableValue\_4, TableValue\_5

**Definition:****Example(s):**

TABLE\_INDEX (3.5, 0, 10, 20, 30, 40) = 25

### 5.9.10 pH functions

MIKE ECO Lab provides a series of functions to calculate pH and related chemical properties of seawater. Several of the functions are adapted from the CO2SYS Program<sup>(1)</sup> and are based on alkalinity and the solution chemistry of carbon dioxide. The parameters for the equations depend on salinity, temperature and hydrostatic pressure; the latter is derived from water depth. The effects of pressure are small and noticeable effects only occur for deep waters (> 100 m). In most cases the hydrostatic pressure can safely be neglected by setting the water depth to 0 m. To include pressure effects the

---

<sup>1</sup> Lewis E.R.; Wallace D.W.R. (1998): Program Developed for CO2 System Calculations. CDIAAC. doi:10.15485/1464255

water depth can be specified with the built-in forcing LAYER\_CENTER\_DEPTH. The alkalinity of seawater depends on the concentration of other species than carbon; the concentrations of some species are estimated from salinity, but the concentrations of total phosphorus and total silicon are specified as input. If the concentrations of phosphorus and silicon are not known, they can be assumed to be zero; the calculation of pH will still work, though the result may be less accurate. Note that some concentrations are in the standard ECO Lab concentration unit mg/l, whereas others are in the unit  $\mu\text{mol/kg SW}$ , micromole per kilogram seawater (this is approximately the same as  $\mu\text{mol/m}^3$ ).

The following functions are currently available:

- SOLVE\_PH  
solve pH system, i.e. calculate pH value
- CONC\_CO2  
calculate CO<sub>2</sub> concentration
- CONC\_CO3  
calculate CO<sub>3</sub><sup>2-</sup> concentration
- CONC\_HCO3  
calculate HCO<sub>3</sub><sup>-</sup> concentration
- CONC\_OH  
calculate OH<sup>-</sup> concentration
- CONC\_H\_FREE  
calculate free H<sup>+</sup> concentration
- FUGACITY\_CO2  
calculate fugacity of CO<sub>2</sub>
- PP\_CO2  
calculate partial pressure of CO<sub>2</sub>
- FLUX\_CO2  
calculate CO<sub>2</sub> flux between atmosphere and water surface

## Options

There are several settings that affect the calculations of the functions. These settings are defined for each MIKE ECO Lab Template in "MIKE ECO Lab Setup" -> "Miscellaneous" -> "pH Settings". Figure 5.4 shows the settings menu with the default settings. The setting "Option selection" allows to choose between predefined selections of the basic settings. Choosing "User defined" in this setting allows the user to choose between any combination of the basic settings. The four basic settings "Set of constants", "K HSO<sub>4</sub>", "[B]T Value" and "pH scale" correspond to the settings in the CO<sub>2</sub>SYS Program. The "pH scale" setting defines the pH scale for input and output values of pH. The other three settings affect specifics of the calculations, such as the estimates of equilibrium constants; see the scientific documentation for details. The conversions between pH scales are not static, but depends on salinity, temperature and pressure and are also affected by the choice of other set-





tings. For seawater models, the default settings are most often suitable. For freshwater models, it is recommended to use the "Freshwater option" in "Option selection".

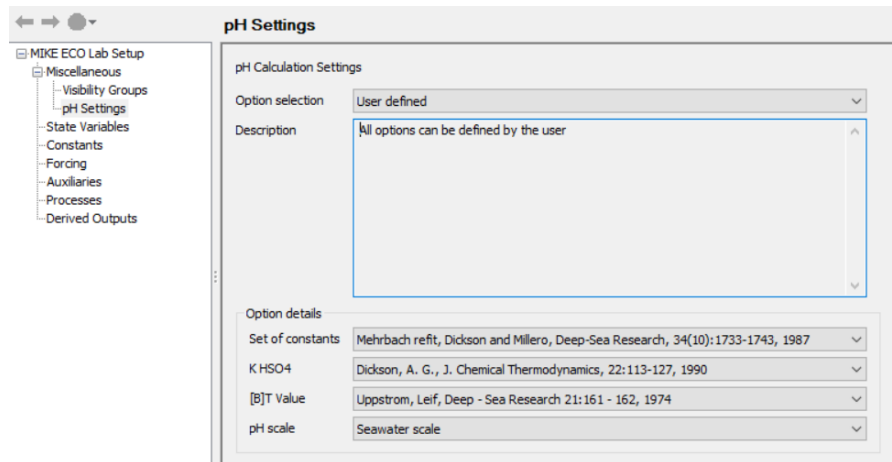


Figure 5.4 The settings for the pH calculation functions with default values.

## SOLVE\_PH

### Description:

This function is based on the CO2SYS Program and calculates pH from alkalinity and total dissolved inorganic carbon (DIC). The pH value is the result of very fast equilibrium reactions. The parameters for the system are estimated from salinity, temperature and water depth. The total concentrations of phosphorus and silicon are needed for the alkalinity expression.

### Arguments:

Alk: Alkalinity [ $\mu\text{mol/kg SW}$ ]  
 DIC: Total dissolved inorganic carbon [ $\text{mg/l}$ ]  
 Salinity: Salinity [psu]  
 TempC: Temperature [ $^{\circ}\text{C}$ ]  
 Depth: Water depth [m]  
 TP: Total phosphorus concentration [ $\text{mg/l}$ ]  
 TSi: Total silicon concentration [ $\text{mg/l}$ ]

The input arguments Alk, DIC, Salinity and TempC are essential for calculating the pH value and their values must be specified in the function call. The arguments Depth, TP, TSi have are not mandatory for calculating pH and, if their values are not known, a reasonable approximation of pH can still be obtained by setting their value to zero.

### Definition:

SOLVE\_PH (Alk, DIC, Salinity, Temperature, Depth, TP, TSi)

An iterative Newton-Rapson solver is used to solve the equation system and find the pH value. See the scientific manual for a detailed description.

### Example(s):

For the default settings

SOLVE\_PH (2200, 25.223, 35.0, 20.0, 0.0, 0.0, 0.0) = 7.81976

### Validation example:

To compare the result to the CO2SYS Program the SOLVE\_PH function is called for the input values in Table 5.14 and using the default settings:

SOLVE"\_" PH(2400,30,35,10,5,0.1,1.5)=7.39795.

The corresponding input data for the CO2SYS Program are listed in Table 5.14. Both MIKE ECO Lab and the CO2SYS Program take alkalinity, salinity and temperature in the same unit. MIKE ECO Lab takes the depth as input, whereas the CO2SYS Program takes the hydrostatic pressure. For the given salinity and temperature, a depth of 5 m corresponds to a hydrostatic pressure of 5.0349 dbar. The total concentrations of carbon, phosphorus and silicon are given in mg/l in MIKE ECO Lab and in  $\mu\text{mol}/(\text{kg-SW})$  in CO2SYS. The conversion from mass to molar concentration requires the molar mass of the respective substance and the solution density. The latter depends on salinity and temperature. For more details of converting the input arguments see the scientific documentation. Entering the corresponding input data for the CO2SYS Program into the CO2SYS excel spreadsheet and using the default settings gives the pH value 7.39795. This is the same result as the MIKE ECO Lab function.

Table 5.14 Corresponding input values for MIKE ECO Lab function and the CO2SYS Program

	MIKE ECO Lab		CO2SYS	
Alkalinity (TA in CO2SYS)	2400	$\mu\text{mol}/\text{kg-sw}$	2400	$\mu\text{mol}/\text{kg-SW}$
Carbon (TCO2 in CO2SYS)	30	$\mu\text{mol}/\text{l}$	2432.1579	$\mu\text{mol}/\text{kg-SW}$
Salinity	35	psu	35	psu
Temperature	10	$^{\circ}\text{C}$	10	$^{\circ}\text{C}$
Depth (Pressure in CO2SYS)	5	m	5.0349	dbar
Phosphorus	0.1	mg/l	3.1438	$\mu\text{mol}/\text{kg-SW}$
Silicon	1.5	mg/l	52.0076	$\mu\text{mol}/\text{kg-SW}$

## CONC\_CO2

### Description:

This function calculates the concentration of dissolved  $\text{CO}_2$  as carbon mass



per volume in the unit mg/l. The function is based on the CO2SYS Program and use equilibrium reactions for carbon dioxide dissolving in water to calculate the CO<sub>2</sub> concentration from DIC and pH.

**Arguments:**

DIC: Total dissolved inorganic carbon [mg/l]

pH: pH value

Salinity: Salinity [psu]

TempC: Temperature [°C]

Depth: Water depth [m]

**Definition:**

CONC\_CO2 (DIC, pH, Salinity, TempC, Depth)

See the scientific documentation for details of the calculation.

**Example(s):**

For the default settings

CONC\_CO2 (30, 7, 35, 10, 0) = 2.62152

## CONC\_CO3

**Description:**

This function calculates the concentration of dissolved CO<sub>3</sub> as carbon mass per volume in the unit mg/l. The function is based on the CO2SYS Program and use equilibrium reactions for carbon dioxide dissolving in water to calculate the CO<sub>3</sub> concentration from DIC and pH.

**Arguments:**

DIC: Total dissolved inorganic carbon [mg/l]

pH: pH value

Salinity: Salinity [psu]

TempC: Temperature [°C]

Depth: Water depth [m]

**Definition:**

CONC\_CO3 (DIC, pH, Salinity, TempC, Depth)

See the scientific documentation for details of the calculation.

**Example(s):**

For the default settings

CONC\_CO3 (30, 7, 35, 10, 0) = 0.170408

## CONC\_HCO3

**Description:**

This function calculates the concentration of dissolved HCO<sub>3</sub> as carbon mass per volume in the unit mg/l. The function is based on the CO2SYS Program



and use equilibrium reactions for carbon dioxide dissolving in water to calculate the  $\text{HCO}_3$  concentration from DIC and pH.

**Arguments:**

DIC: Total dissolved inorganic carbon [mg/l]

pH: pH value

Salinity: Salinity [psu]

TempC: Temperature [°C]

Depth: Water depth [m]

**Definition:**

CONC\_HCO3 (DIC, pH, Salinity, TempC, Depth)

See the scientific documentation for details of the calculation.

**Example(s):**

For the default settings

CONC\_HCO3 (30, 7, 35, 10, 0) = 27.2081

## CONC\_OH

**Description:**

This function calculates the concentration of dissolved hydroxide in the unit  $\mu\text{mol/kg-SW}$ . The calculations follow the CO2SYS Program and use the ionization equilibrium for water to calculate the  $\text{OH}^-$  concentration from pH. Note that for the GEOSECS option in the "Set of Constants" setting, the concentration of  $\text{OH}^-$  is always assumed to be zero.

**Arguments:**

pH: pH value

Salinity: Salinity [psu]

TempC: Temperature [°C]

Depth: Water depth [m]

**Definition:**

CONC\_OH(pH, Salinity, TempC, Depth)

See the scientific documentation for details of the calculation.

**Example(s):**

For the default settings

CONC\_OH(7, 35, 10, 0) = 0.146538

## CONC\_H\_FREE

**Description:**

This function calculates the concentration of dissolved free hydrogen ions in  $\mu\text{mol/kg-SW}$  from pH. The calculations are adapted from the CO2SYS Program and relies on the definition of pH. The input values for salinity, temperature and water depth are used for conversion between pH scales.

**Arguments:**

pH: pH value  
Salinity: Salinity [psu]  
TempC: Temperature [°C]  
Depth: Water depth [m]

**Definition:**

CONC\_H\_FREE(pH, Salinity, TempC, Depth)

See the scientific documentation for details of the calculation.

**Example(s):**

For the default settings  
CONC\_H\_FREE(7, 35, 10, 0) = 0.0849116

## FUGACITY\_CO2

**Description:**

This function calculates the fugacity of CO<sub>2</sub> in Pa. The fugacity is used to describe the partial pressure of a real gas compared to an ideal gas (i.e. it describes the "apparent" partial pressure of a real gas). The function is based on the CO2SYS Program and use equilibrium reactions for carbon dioxide dissolving in water to calculate CO<sub>2</sub> fugacity from DIC and pH.

**Arguments:**

DIC: Total dissolved inorganic carbon [mg/l]  
pH: pH value  
Salinity: Salinity [psu]  
TempC: Temperature [°C]  
Depth: Water depth [m]

**Definition:**

FUGACITY\_CO2(DIC, pH, Salinity, TempC, Depth)

See the scientific documentation for details of the calculation.

**Example(s):**

For the default settings  
FUGACITY\_CO2 (25.223, 7.81977, 35.0, 20.0, 0) = 68.8812

## PP\_CO2

**Description:**

This function calculates the partial pressure of CO<sub>2</sub> in Pa. The function is based on the CO2SYS Program and uses the relation between fugacity and partial pressure as well as the calculations for CO<sub>2</sub> fugacity.

**Arguments:**

DIC: Total dissolved inorganic carbon [mg/l]  
pH: pH value



Salinity: Salinity [psu]  
TempC: Temperature [°C]  
Depth: Water depth [m]

**Definition:**

PP\_CO2(DIC, pH, Salinity, TempC, Depth)

See the scientific documentation for details on the calculation.

**Example(s):**

For the default settings  
PP\_CO2(30, 7, 35, 10, 0) = 492.67

## FLUX\_CO2

**Description:**

This function calculates the carbon dioxide transfer rate  $F_{CO_2}$  across the water-air interface in [g C/m<sup>2</sup>/d]. The function is based on <sup>(1)</sup> and can be used to estimate the CO<sub>2</sub> exchange with the atmosphere on a regional to global scale. It is valid for temperatures between -2°C and 40°C and wind speeds of 3-15 m/s.

Please note that the **sea-air flux** is calculated, i.e. positive numbers indicate an uptake into the water body and negative numbers a degassing! In scientific literature you often find the air-sea flux, i.e. there may be a difference in the sign of the expression value.

**Arguments:**

pCO2air: CO2 partial pressure in air [Pa] (~ 400 ppm == 101325\*400/1e6 ~ 40.53)  
pCO2water: CO2 partial pressure in water [Pa]  
Salinity: Salinity [psu]  
TempC: Temperature [°C]  
Windspeed10m: Average wind speed 10m above sea surface [m/s]

Note: Use the partial pressure function (PP\_CO2) to get the partial pressure of CO2 in water.

**Definition:**

$FLUX\_CO2(pCO2_{air}, pCO2_{water}, Salinity, TempC, Windspeed10m) = k_{CO_2} * K_0 * (pCO2_{air} - pCO2_{water})$

$k_{CO_2}$ : Carbon dioxide transfer velocity

$K_0$ : Solubility of carbon dioxide

See the scientific documentation for further details on the calculation of the transfer velocity  $k_{CO_2}$  and  $K_0$ .

---

1 Wanninkhof, R. 2014. Relationship between wind speed and gas exchange over the ocean revisited. *Limnology and Oceanography Methods*. 12(6): 351-362

**Example(s):**

FLUX\_CO2 (101325\*400e-6, 67.2408,35, 20.0, 4) = -1.00757 [g C /m<sup>2</sup>/d]

## 5.10 Agent Based Modelling with MIKE ECO Lab

Definition: 'Models describing the (autonomic) behaviour and states of agents, objects or individuals'.

Agent based modelling or models, short ABM (or synonymously 'Individual based modelling', IBM) is a relatively recent development. As Grimm and Railsback[1] point out, 'classical theoretical ecology...usually ignores individuals and their adaptive behaviour'. Ecosystems are commonly seen from a process based view, following the fate of masses or concentrations in the system. This process orientated modelling is done by describing the flow between different components and the associated process. It is a well-established and proven method and usually the first choice if you want to simulate dissolved substances e.g. oxygen concentrations, BOD levels, pollutants or phyto- or zooplankton distributions on a larger scale. Many phenomena, however, cannot be described satisfactory using such process orientated models.

For example, it is well known that plankton organism (even if they by definition are subject to passive transport by the water currents) may show an explicit diel or diurnal<sup>(1)</sup> vertical migration through the water column. In the ocean this migration can span over several 100 m in the vertical. By feeding at the surface and defecating large pellets in deeper zones plankton organism can greatly influence the transport of organic matter to deeper layers. If the flow patterns are different throughout the water column such movement will also affect the species distribution pattern. ABMs can be used to describe and investigate such patterns by reproducing the observed movement of individuals (agents) and the resulting changes for the system.

The MIKE ECO Lab ABM module allows you to formulate agent based models within the MIKE FM series of hydrodynamic environments of the MIKE framework easily. It allows to combine the normal, process orientated MIKE ECO Lab framework with a Lagrangian particle movement model.

It is assumed that the reader is familiar with 'classic' process orientated modelling in MIKE ECO Lab in the following text. In doubt result the other MIKE ECO Lab manuals.

### 5.10.1 Structure of an ABM in MIKE ECO Lab

Each ABM in MIKE ECO Lab consists of elements from the Eulerian framework ('classic' process orientated components like dissolved substances,

---

<sup>1</sup> Many plankton organisms migrate to feed in the euphotic zone near the surface and rest in the mesopelagic zone (or the hypolimnion in freshwater lakes) during a day cycle.

global constants, forcing like hydraulic parameters etc.) and definitions for the individual particle class in the Lagrangian framework (internal variables, constants, equations, movement definitions).

### Eulerian framework

- State variables
- Constants
- Forcing
- Auxiliary variables (Eulerian variables)
- Process definitions (Eulerian variables)

### Lagrangian framework

- Individual particle class
  - Class state variables
  - Class constants
  - Spatial functions
  - Arithmetical expressions
  - Movement definitions

## 5.10.2 Inside a particle class

Any individual/agent formulated using the ABM MIKE ECO Lab must belong to a so called particle class. The following chapter gives an overview on the single elements of such a particle class. It defines the general properties on an individual. Although it declares default values for e.g. state variables or constants, each individual/agent has its individual set of variables and thus values between individuals can vary within the simulation.



Please note that state variables, particle class, constants and forcing definition inside an ABM template correspond to entities, state variables and input data in the ODD-protocol (for Overview, Design concepts and Details) proposed by Grimm et al. [2], [3].

In general, DHI strongly encourages to utilise the ODD protocol to document and communicate an ABM model.

## State variables

State variables are the only elements of a MIKE ECO Lab calculation that are preserved in time. Any other element (i.e. arithmetic expression) is calculated on per time-step base. State variables represent elements of an individual that can change due to some associated process. Body mass/length but also nutrition state, etc. belongs to this. The change of the state variable is computed according to the given differential state variable equation.





A State variable has the following properties:

Table 5.15 State variable properties

Property	Description
Symbol	Defines the name to be used in expressions to refer to the actual value of the state variable
Description	A short textual description
Online help	Displayed as mouse-over help in the setup
Documentation	Possibility to write a text/hyperlink with detailed descriptions
EUM type	Data type for result processes
Unit	Textual information
Minimum value	Allowed minimum
Maximum value	Allowed maximum
Expression	<p>Right-hand side of the ODE describing the change of the state variable over time:</p> $\frac{dSymbol}{dt} = process1 + process2$

## Constants

Constants represent information that influences the internal calculations but does not change during the simulation time for each individual. Typical constants would be some thresholds, growth rates but also stoichiometric relationships.

The following properties are available for Constants:

Table 5.16 Constants properties

Property	Description
Symbol	Defines the name to be used in expressions to refer to the actual value of the state variable
Description	A short textual description
Online help	Displayed as mouse-over help in the setup
Documentation	Possibility to write a text/hyperlink with detailed descriptions

Table 5.16 Constants properties

Property	Description
EUM type	Data type for result processes
Unit	Textual information
Default value	Value used for initialisation, if not specified differently
Minimum value	Allowed minimum
Maximum value	Allowed maximum

## Restricted area search functions

Restricted area search functions represent the remote sensing abilities of an agent. Restricted area search functions can be used to query magnitudes and directions of remote concentration gradients around an individual and relations to other neighbouring individuals (distances, directions, etc.). See Section 5.10.8 for details how to specify a Restricted area search function.

The following properties are available for the Restricted area search functions:

Table 5.17 Restricted area search function properties

Property	Description
Symbol	Defines the name to be used in expressions to refer to the actual value of the state variable
Description	A short textual description
Online help	Displayed as mouse-over help in the setup
Documentation	Possibility to write a text/hyperlink with detailed descriptions
Owner	Lists the species declaring the particular restricted area search function
Framework	'Euler' stands for information based on the process orientated template part whereas 'Lagrange' stands for information based on other agents
Purpose	Declares the general purpose, i.e. if a direction or magnitude is to be sensed. Depending on the selected framework various selections are possible
Consider	In case of a function defined in the Lagrangian framework, the 'targeted' particle class has to be selected. Instead of a specific particle class also 'any particle' can be selected.



Table 5.17 Restricted area search function properties

Property	Description
Function	Defines the general functionality, i.e. if the function should target the minimum/maximum value, a gradient or absolute data
Based on variable	Declares the target state variable. Please note that only 2D variables should be selected for the 'Euler' framework. To select a particle property in the Lagrangian framework, select 'NONE' and pick a property below. If 'Any particle' is to be considered, no variable can be selected.
Based on property	Select a specific particle property in case of the Lagrangian framework.
Dimension	Defined if the restricted area search function works in 2D or 3D
Range/radius	Declares the search radius around the particle

## Arithmetic expressions

Arithmetic expressions correspond to the auxiliary variables and process definitions of a process orientated MIKE ECO Lab model. Arithmetical expression can be used to define process changing internal state variables or to compute other internal functions. Any behaviour or calculation must be coded using such arithmetical expressions. Arithmetical expressions have to confer the syntax and semantic of MIKE ECO Lab expressions (cf. Section 5.6).

The following properties are available for an Arithmetic expression:

Table 5.18 Arithmetic expression properties

Property	Description
Symbol	Defines the name to be used in expressions to refer to the actual value of the state variable
Description	A short textual description
Online help	Displayed as mouse-over help in the setup
Documentation	Possibility to write a text/hyperlink with detailed descriptions
Output	Specifies if the symbol is available as output item
Spatial variation	Declares the spatial variation

Table 5.18 Arithmetic expression properties

Property	Description
Unit	Textual description of unit
Expression	Expression to be calculated

## Horizontal movement

The horizontal movement defines up to 5 polar vectors as the direction and speed (in degree north and m/s) a particle indent to move in the X-Y plane. The direction and speed equations have to follow the same syntax and semantic as any other MIKE ECO Lab expression (cf. Section 5.6). Internally all defined horizontal vectors are summed up to a final movement vector. See Section 5.10.5 for details.

The following properties are available for the Horizontal movement:

Table 5.19 Horizontal movement properties

Property	Description
Direction	Expression to calculate the direction (degree North)
Speed	Expression for the speed in m/s



Contributions from a random walk process governed by the dispersion properties may be added when moving an individual depending on the settings of the hydraulic model setup. Such a random walk process is intrinsic part of the movement routines and should normally not be specified explicitly in the MIKE ECO Lab ABM description. For further details, please refer to the MIKE 21 & MIKE 3 Flow Model FM Particle Tracking Module User Guide.

## Vertical movement

Similar to the horizontal movement an expression must be given defining the vertical movement for an individual. See Section 5.10.5 for details.

The following properties are available for the Vertical movement:

Table 5.20 Vertical movement properties

Property	Description
Speed	Expression for the vertical movement in m/s



### 5.10.3 Individual properties

Each individual has a set of properties (cf. Table 5.21) describing some characteristics, e.g. its age or position in space. These properties cannot be directly modified using some MIKE ECO Lab expressions. The update of the underlying information is done automatically when a particle moves or ages.

Properties can be used as an argument inside an expression. The general way to refer to a property is to specify its name in squared brackets:

[Property]

Only age and the current movement vector are directly available as output items for particle tracks. If you plan to include other properties in a track output you have to add an arithmetical expression for each, containing just the property value.

Table 5.21 Individual properties

Property	Description
AGE	Age in seconds since release of the particle
HDIR	Last used horizontal movement direction [degree], 1 <sup>st</sup> movement vector only
HSPEED	Last used horizontal movement speed [m/s], 1 <sup>st</sup> movement vector only
VSPEED	Last used vertical speed [m/s]
FINAL_HDIR	Last used horizontal movement direction [degree], sum of all horizontal movement vectors
FINAL_HSPEED	Last used horizontal movement speed [m/s], sum of all horizontal movement vectors
FINAL_VSPEED	Last used vertical speed [m/s], same as VSPEED
XPOS	Current X position (model projection) [m]
YPOS	Current Y position [m]
ZPOS	Current Z position [m]
ABOVE_GROUND	Distance above the ground [m]
BELOW_SURFACE	Distance below the surface [m]
HITSTATE	Spatial status of an individual, see below
PROJ_XPOS	X position projected into the future based on current speed and direction (see 5.10.6)
PROJ_YPOS	Projected Y position

Table 5.21 Individual properties

Property	Description
PROJ_ZPOS	Projected Z position
PROJ_HITSTATE	Projected hitstate for projected position
PREV_XPOS	Last X Position
PREV_YPOS	Last Y position
PREV_ZPOS	Last Z position
PREV_HITSTATE	Last hitstate

### Examples:

The following expression stub will check if the individual is within a certain range below the surface:

```
'IF ( [BELOW_SURFACE] <= threshold) THEN ... ELSE ...'
```

The following expression will compute the individual's age in days since release:

```
'[AGE] / 86400'
```

## Movement information

### HDIR, HSPEED, VSPEED

These properties hold information from the last realized 1<sup>st</sup> horizontal movement vector. If more than one movement vectors are defined, the first movement vector should describe the "main" movement cause, i.e. the decision controlled movement whereas additional vectors should ideally define the influence of minor or pure physical cues, i.e. random process, wind influence etc. However, additional vectors can of course also describe alternative decision controlled movement, e.g. different/alternative decision controlled movement.

The example below sketches two functional equivalent movement descriptions, where two movement strategies/alternatives and a random component are considered. Option A is realised using three movement vectors where each one corresponds to a single movement clue. Option B is functional equivalent but sums all decision controlled movement strategies in the 1<sup>st</sup> movement vector.

**Example:**

Vector	Option A	Option B
1	IF (Strategie==1, Movement1, 0)	IF (Strategie==1, Movement1, IF (Strategie==2, Movement2, 0))
2	IF (Strategie==2, Movement2, 0)	RandomMovement
3	RandomMovement	
...		

**FINAL\_HDIR, FINAL\_HSPEED, FINAL\_VPSEED**

These properties hold information from the last realised movement, that means it represents the sum of all movement vectors and includes contributions from the dispersion random walk (cf. Section 5.10.5).

**HITSTATE – Spatial status of an individual**

To facilitate handling of certain states each individual has a so called 'HITSTATE'. This 'HITSTATE' indicates if the particle hits (meaning is touching) the water surface, the bottom or any borders of the environment (model domain). It is possible that some of the events occur simultaneously (e.g. an individual in a dry element will be simultaneously 'at the surface' and 'at the bottom') and such configurations will be reflected by the HITSTATE.

Technically, the 'HITSTATE' is a binary coded list of the single flags. This means that you can sum up the individual basic values for any events, if two or more occur simultaneously.

Table 5.22 HITSTATE conditions and basic values

HITSTATE	Condition	Basic value
No hit event	Fully suspended in the water column	0
Internal boundary	Hitting an internal boundary	1
External boundary	Hitting an external boundary/leaving the spatial model domain	2
Water surface	Individual at the surface	4
Bed	Individual on the bed	8
Land	Hitting a land boundary	16



There is a built-in function in MIKE ECO Lab to test conditions of the HITSTATE flag. This built-in function is called 'TESTHITSTATE (input\_hitstate,

test\_value)', and the result will be '1' if any of the test value flags is present in the input\_hitstate, otherwise it will be zero. See section 5.9.6 for more details.

**Example:**

```
IF (TESTHITSTATE( [HITSTATE], 4+8)==1) THEN ... ELSE ...
```

The property 'Hitstate' is tested if the individual touches either the surface or the bed.

## 5.10.4 Spatial representation

Regardless of the spatial representation in the (gridded) hydraulic environment any individual/particle has a true 3D representation. The coordinates represent the internal model projection and have a unit on 1 m unless the projection is in longitude/latitude. For longitude/latitude projections the x- and y coordinates represent the position in decimal degrees.

Every particle knows the information about the current position ([XPOS], [YPOS], [ZPOS]) and the position from the previous time step ([PREV\_XPOS], [PREV\_YPOS], [PREV\_ZPOS]). Additionally, an individual may project its future position ([PROJ\_XPOS], [PROJ\_YPOS], [PROJ\_ZPOS]) based on the projection time horizon (cf. Section 5.10.6).

## 5.10.5 Movement

### Horizontal movement

Every individual must have at least one horizontal movement vector and has always one settling speed definition. A horizontal movement vector defines the direction [angle in degree from true north] and the speed [m/s] that a particle wants to move in the X-Y plane. Every individual can specify up to 5 independent horizontal movement vectors that are internally summed up to define the overall horizontal movement.

The vector sum is done by converting the horizontal vectors into their u- and v- (Cartesian) representation and summing up the individual components. The resulting vector is mapped back to 0-360°. By having several movement vectors it is easy to specify an individual's own movement and contributions from the flow.

### Downward / vertical movement

Contrary to the horizontal movement, there is just one settling velocity definition, specifying the vertical movement [m/s].



Although the coordinate system points upward, that means zero marks the still water level and the water depth are negative numbers, the vertical speed is a settling speed. So a positive settling velocity will cause a particle to sink





to the bottom whereas negative values will cause a particle to rise to the surface.



**IMPORTANT:** If an individual reaches the bed it will be considered as sedimented and stop any movement (horizontally and vertically). Depending on the specifications/settings in the hydrodynamic model setup it might be resuspended if a resuspension criterion is met.

To bypass this intrinsic handling of sedimented particles by hydrodynamic model it is possible to set a flag in the model setup to prevent the sedimentation handling. In this case all sedimentation/ resuspension criteria and associated process must be handled inside the MIKE ECO Lab ABM template.

## Dispersion / random walk

The realised movement, the distance a particle has moved, may contain contributions from the random walk due to the dispersion settings. Please refer to the MIKE 21 & MIKE 3 FLOW MODEL FM Particle Tracking Module User Guide for further details on how to specify the dispersion parameters.

### 5.10.6 Movement projection

It is often needed to project an individual's position into the future. This ability is essential to prevent collisions with borders/land etc. MIKE ECO Lab ABM supports this by allowing you to specify a 'position projection' timespan. This time span (in [sec]) is used to project an individual's position based on the current movement. The 'Hitstate' for the projected position will also be updated and can be used to change e.g. swimming speed and direction.

The projection time can be set in the general class settings for every species.

### 5.10.7 Special built-in ABM functions

There are several built-in functions for ABM modelling in MIKE ECO Lab, cf. Table 5.23.

**Table 5.23** Built-in functions for ABM modelling in MIKE ECO Lab

Function	Description
GETX (speed, direction)	Returns the X- (u-) component of the given polar vector (speed, direction)
GETY (speed, direction)	Returns the Y- (v-) component of the given polar vector (speed, direction)
GETDISTANCE (X1,Y1,Z1, X2,Y2,Z2)	Returns the Euclidian distance between the given two points

Table 5.23 Built-in functions for ABM modelling in MIKE ECO Lab

Function	Description
GETANGLE (X1,Y1, X2,Y2)	Returns the angle in 0-360° from (X1,Y1) to (X2,Y2)
RESULTINGSPEED (speed1, direction1, speed2, direction2)	Returns the speed component of the vector sum of the two polar vectors
RESULTINGDIRECTION (speed1, direction1, speed2, direction2)	Returns the direction component of the vector sum of the two polar vectors
ANGLEBETWEEN (speed1, direction1, speed2, direction2)	Returns the angle spanned between the two vectors
TESTHITSTATE (value, testvalue)	Test if at least one test flag is present, cf. Sections 5.9.6 and 5.10.3
DISTANCE_TO (x,y,z)	Compute the distance of the current individual to the given position
INSIDE (X1,Y1,Z1, X2,Y2,Z2)	Returns 1 if the current individual is within the rectangle defined by the lower left (X1,Y1,Z1) and upper right (X2,Y2,Z2) corner. NOTE: If Z1=Z2, the z-coordinates are ignored and just the 2D coordinates are checked.
SET_RADIUS	This function can be used to set the search radius of a specific RASF for the current ABM class
SET_PROJECTION	This function can be used to set the position projection time for the current individual.
SET_FOV	This function can be used to set the "Field of view" for the current individual.



Please note: when the projection is in longitude/latitude the GETDISTANCE, GETANGLE and DISTANCE\_TO functions expect coordinates in degree longitude/latitude and ignore the z-positions. The distance calculations are based on the Haversine distance calculation on a spheroid earth model with an assumed earth radius  $r = 6371.009$  km. Typical accuracy of the calculated distances is between 0.3-0.5%

### 5.10.8 Sensing functions – perception of environment/particles

Sensing the environment is an essential ability for an individual, either to find suitable conditions and environmental cues or to interact with other individuals.



Sensing can be differentiated into **local perception**, which is gathering information in the direct surrounding environment (conditions like temperature, concentration etc. of the current location<sup>(1)</sup>). In MIKE ECO Lab agents can directly address this local information declared in the process orientated part of the template.

**‘Long range perception’**, information within a certain perception distance must be ‘sensed’ with an appropriate function. In MIKE ECO Lab ABM such remote sensing is done via so called ‘Restricted Area Search Functions’ (RASf). Such search functions are computed individually for each agent and can be used to query the direction, magnitude and distance of concentrations or other particles variables. Search functions can, for example, be used to calculate to direction of the highest concentration of food (and its magnitude), the gradient of the bathymetry or flow variables or the distance to the nearest neighbour. Applied to other particles/individuals a search function can also resolve the average swimming direction or speed of all individuals within the perception range.

Each of the Restricted Area Search Functions (RASf) has a set of properties, which are described below.

### Framework

The framework defines if the RASf works on Eulerian or Lagrangian variables. Eulerian variables are concentrations or forcings declared in the process orientated part of template whereas Lagrangian variables refer to particle variables.

Table 5.24 RASf Framework properties

Value	Meaning
Euler	RASf based on concentrations or forcing defined in the process orientated template section
Lagrange	RASf based on values from other particles

---

<sup>1</sup> There is no interpolation of such process orientated information based on the agent's position.

## Purpose

This property declares the general nature of the return value of the RASF. The available purposes depend on the framework. The following values are possible:

Table 5.25 RASF Purpose properties

Value	Supported framework	RASF return value
Direction	Euler and Lagrange	(Absolute) angle pointing to the query item
Magnitude	Euler and Lagrange	Magnitude of the query item
Distance	Euler only	Distance (m) between current position and query item
Particle	Lagrange only	Handle of a particle owning the query item

## Consider

This property is just valid for RASF in the Lagrangian Framework. It declares if particles of a specific class or any other particles should be considered.

Table 5.26 RASF Consider properties

Value	Meaning
Any particle	Particles of any specie
Own specie	Only particles of the own specie
Specific class name	Only particles of a specific specie

It is currently not possible to exclude or restrict subset particles of specific classes for a RASF.



### Function

This property declares what function is applied to the search. It is e.g. possible to select the minimum or maximum value in a RASF. The available functions depend on the selected framework and purpose.

Table 5.27 RASF Function properties

Function	Framework	Purpose	Meaning
Min	Euler	Direction	Direction towards minimum gradient (value divided by distance)
		Magnitude	Value of minimum gradient (value divided by distance)
	Lagrange	Direction	Direction towards the particle with the minimum value
		Magnitude	Value of minimum
		Distance	Distance from the particle with the minimum value
		Particle	Handle for the particle with the minimum value



Table 5.27 RASF Function properties

Function	Framework	Purpose	Meaning
Max	Euler	Direction	Direction towards maximum gradient (value divided by distance)
		Magnitude	Value of maximum gradient (value divided by distance)
	Lagrange	Direction	Direction towards the particle with the maximum value
		Magnitude	Value of maximum
		Distance	Distance from the particle with the maximum value
		Particle	Handle for the particle with the maximum value
Min_Absolute	Euler	Direction	Direction towards absolute minimum value in search range
		Magnitude	Absolute minimum value in search range
Max_Absolute	Euler	Direction	Direction towards absolute maximum value in search range
		Magnitude	Absolute maximum value in search range



Table 5.27 RASF Function properties

Function	Framework	Purpose	Meaning
Average	Lagrange	Direction	Direction towards the averaged position (mass centre) of all valid particles in the search range
		Magnitude	Averaged value of all valid particles in search range
		Distance	Distance to the averaged position (mass centre) of all valid particles in search range
Count	Lagrange	Magnitude	Number of valid particles in the search range
Sum	Lagrange	Magnitude	Sum of the values of all valid particles
Sum_norm	Lagrange	Magnitude	Sum of the values normalised to the distance of all valid particles

### Based on variable / property

With these settings the source/target variable is determined. For a RASF EULER framework only the field variable is active and lists all available target variables. In a LAGRANGE framework all variables of the selected target specie (cf. Consider) are listed. If you want the RASF to be based on a particle property, 'NONE' has to be selected for the target variable. If 'consider' is set to 'Any specie', the variable field will become inactive and the target variable can just be selected from the list of properties.

Table 5.28 Available Variable properties

Value	Meaning
AGE	Particle age
HDIR	Final horizontal direction
HSPEED	Final horizontal speed
VSPEED	Final vertical speed

Table 5.28 Available Variable properties

Value	Meaning
XPOS	X-coordinate
YPOS	Y-coordinate
ZPOS	Z-coordinate
DISTANCE	Distance to the particle

## Search Radius

The search radius defines the spatial extent of the RASF. It is given in m. When the map projection is in longitude/latitude the distance calculations are based on the Haversine distance calculation on a spheroid earth model with an assumed earth radius  $r = 6371.009$  km. Typical accuracy of the calculated distances is between 0.3-0.5%. See the definition of space below.

### Space in Euler framework

A RASF in the Euler framework will work on concentration variables. Please note that the considered values will represent the value at the centre of a mesh element, i.e. there is no interpolation possible. Also the distance to the centre of a mesh element determines if an element will be considered to be in the search range or not; this means that some mesh elements may not be considered by the RASF because their centre is outside the search range! See Figure 5.5 for a sketch of the spatial relations.

### Space Lagrange framework

In the Lagrange framework only particles in the perception distance and the view field are considered. See Figure 5.5 for a sketch of the spatial relations.



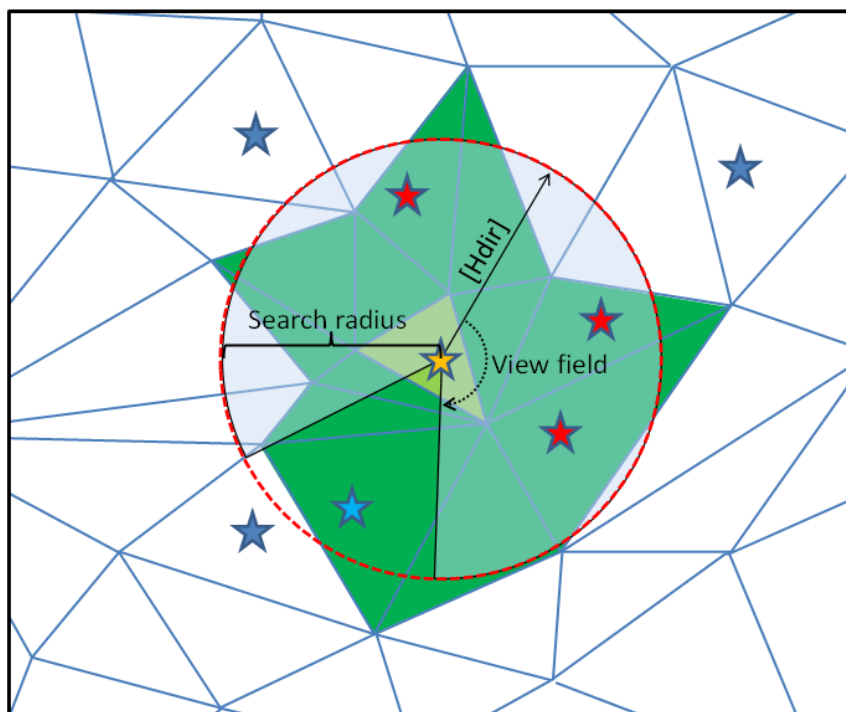


Figure 5.5 Schematic sketch of spatial relations for a RASF. The active yellow particle is in the centre. The local information of concentrations, forcing etc. in this light green cell is directly accessible (local perception). For long range perception a RASF has to be applied. A RASF in the EULER framework with the given search radius would consider only the green mesh elements because their centres are within the search range. For a RASF in a LARGRANGE framework the red particles are the perception range (blue sector) defined by the search range and the view field (based on the last horizontal direction indicated by the arrow). The blue particles are outside the view field or the search range and will thus not be considered in the RASF



In the Lagrange framework the information will always refer to valid particles within the search range without the current active particle itself; this means that the e.g. minimum will refer to the minimum of all valid particles besides the active particle. In this case the active particle could be representing the absolute minimum value but the RASF will refer to next larger particle.



**IMPORTANT:** In the first release version of MIKE ECO Lab/ABM RASF will not work properly if you utilise a parallel OpenMP/OpenMI engine. This temporal restriction and will be removed with later versions.

## Tips & Tricks

A RASF returns a single value, e.g. a direction, distance or a magnitude. Often you want to visualise this information as a vector field plot. The easiest

way to plot such a vector field is to create an output map containing the information on the U- and V-components of the vectors and use the vector plot functionality of the MIKE plotting tools.

RASFs can not return the U- and V-information directly but you can use the built-in functionality of MIKE ECO Lab to prepare the necessary data. It requires that you know at least have a function that returns the direction towards and possibly a second one for the magnitude of a variable (if you are just interested in the direction you can supplement the magnitude with a fixed value!).

If you have two RASFs, one computing the direction towards a value and the other one the magnitude of the gradient, you can convert this polar vector (direction and magnitude) into its U- and V-component. This can be done by creating two arithmetic expressions and use the built-in functions 'GETX' and 'GETY' to derive the individual components. Naturally one would use the 'GETX' to retrieve the U-component and 'GETY' for the V-component.

This will, however, result in a 'wrong' vector field plot as the mathematical coordinate system and the map projection does not match in respect to the directions (map projection: zero=North, clockwise vs. math. projection zero=left and counter-clockwise). A correct transformation can be achieved by simply switching the U- and V-components, i.e. using 'GETX' as V-component and 'GETY' as U-component of the polar vector representing the direction and magnitude of a sensing function. Of course the vector information will be just present when a particle is within an element. Note also that the value will represent the average value of all particles in an element, thus you should just have one particle in a mesh element.

### 5.10.9 Interaction with other elements

#### Interaction with concentration type state variables

Lagrange particles can modify state variables declared in the process orientated template section via so called 'Direct Manipulation' functions. These functions can be used to directly modify Eulerian state variables using basic math functions. A direct manipulation is indicated by a state variable followed by a double colon and the operator.

All direct manipulations encountered during the processing of all particles from all classes will be stored as sequential list and will be processed after all particle calculations in the order of their creation (First in, first out principle). You can use direct manipulation in normal MIKE ECO Lab expressions as well. This is especially useful to reset a variable and summing up the influence of particles. Due to the timing (the results of direct manipulations will be computed as last element of a time step, see scheme below), the new state variable data will be available with a time lag of one time step inside the MIKE ECO Lab template; However the output of the modified variables will be up to date.



Computation order:

1. Euler framework
  - Auxiliary Variables
  - Process
  - State Variables
2. Derived outputs
3. Lagrange framework
  - Particle Class
    - Arithmetic expressions
    - State Variables
4. Direct Manipulations

Table 5.29 Available manipulation functions

Function	Command/Syntax	Explanation
Direct Assign	Var ::= [value]	Set Var to value
Direct Add	Var ::+ [value]	Add value to Var
Direct Minus	Var ::- [value]	Subtract value from Var
Direct Multiplication	Var ::* [value]	Multiply Var by value
Direct Division	Var ::/ [value]	Divide Var by value



Please note that no conversion between masses and concentrations are performed, i.e. if you want to change a concentration you have to calculate the correct amount yourself.



**IMPORTANT:** In the first release the direct manipulation functions will bypass the mass budget calculations. Be careful not to introduce mass balance errors!



**IMPORTANT:** If direct manipulations are used in the Euler framework, i.e. as part of an auxiliary expression or a process equation, special care must be taken when using the direct assign operator "::-". Using the direct assign operator in a form of "{StateVariable\_A} ::= {StateVariable\_A}...", i.e. the state variable occurs on both sides of the operation, the AD transport of this variable is reset and it appears to be stationary. In such a situation the direct assignment should be moved into a derived output.

## Interaction with other individuals

Agents may interact with or create/remove other agents during a simulation. These interactions are realised by a set of special commands.

Table 5.30 Special interaction commands

Command	Purpose
Create	Dynamically create new particles of a given class with specified absolute values
Remove	Remove (kill) the current particle
Split	Split the active particle into new particle(s) and transfer some amount of its state variables to the new created particles
Kill	Kill ('eat') a second particle and transfer some amount of its state variables to the active particle
Transfer	Transfer state variables between the active and a second particle

The general syntax of such an interaction consists of the command keyword followed by a pair curly bracket containing a target description and optional command parameters. A target description is a pair of normal bracket enclosing a species name (the target specie) and a target ID parameter. Depending on the command one or more modification specification can follow.

**Interaction** = 'Command{ (Specie, ID) ,modification }'

### The Target ID parameter

The target ID parameter is used to specify either a particle handle of an existing particle when issuing a **Kill** or **Transfer** command. Such a handle must be obtained by an appropriate RASF for interaction between two particles. For the **Create** or **Split** command the value of the Target ID parameter determines the number of dynamically created particles.

The command **Remove** does not require any other parameter or modification specifications and any value for the Target ID will be ignored (but you have to provide a number, preferably '-1').

### Modification specification

A modification specification consists of a pair or normal brackets encompassing a list of two state variable identifiers and four modification values describing how much from each state variable is transferred to the other one and how much it will lose.



modification = '(Variable A, Variable B,  $cp_{ab}$ ,  $cp_{ba}$ ,  $red_a$ ,  $red_b$ )'

The following scheme describes the performed calculations for the **Kill**, **Split** and **Transfer** command to compute the new values:

$$A_{new} = (1 - red_a) \cdot A_{old} + cp_{ba} \cdot B_{old}$$

$$B_{new} = (1 - red_b) \cdot B_{old} + cp_{ab} \cdot A_{old}$$

The modification values  $cp_{ab}$ ,  $cp_{ba}$ ,  $red_a$  and  $red_b$  represent relative proportions of exiting variable values.



Please note: To prevent mass errors the parameters  $red_a$  ('reduce a') should be related to the sum of all (in respect of involved particles) copied  $cp_{ab}$  ('copy from a to b'):

$$red_a = \sum cp_{ab} \quad (5.54)$$

## Explanation of the commands

### The Create command

The modification specification for the **Create** command differs from other modification specification schemes. Only the first and third parameter will be evaluated (but all parameters have to be present) and specifies the **absolute amount** for the new created state variable.

$$A_{new} = red_a$$

**Example:**

```
CREATE { (Drifter, 10), (A, A, 0, 0, 10, 0), (B, B, 0, 0, 5, 0) }
```

Create ten new particles of the class 'Drifter' and set the state variables A and B to the absolute values A=10 and B=5.

### The Remove command

The **Remove** command is the simplest interaction command. It does not have any modification specifications.

**Example:**

```
REMOVE { (Drifter, -1) }
```

Remove (kill) the current active particle (must be of the class 'Drifter') from the simulation.



### The Split command

The **Split** command creates a number of new particles and transfers state variable content to the new created particles. The main difference to the create command is that the transferred amount represents a relative proportion of the existing state variables.

#### Example:

```
SPLIT { (Drifter, 4), (A, A, 0.2, 0, 0.8, 0) }
```

Split the current particle into four new particles of the particle class 'Drifter' and assign each new variable 'A' 20% of the value of 'A' of the current particle. Finally to keep mass balance the exiting variable 'A' is reduce about 80%. Note that the original particle will be still alive after the split!

### The Kill command

The **Kill** command needs a handle to a specific particle. Normally such a handle will be provided by a RASF with the 'Particle' purpose. When issuing a kill command on a specific particle state variable data from the target particle can be transferred to the active particle. The target particle will be removed from the simulation.

#### Example:

```
KILL { (Drifter, Handle), (A, B, 0, 1, 0, 1) }
```

Remove the particle from class 'Drifter' identified by the 'Handle' and transfer 100% of its state variable 'B' to the variable 'A' of the active particle.

To keep mass balance, remove 100% of B afterwards.

### The Transfer command

The **Transfer** command can be used to issue a transfer of state variable data between the current particle and a particle identified by a specific handle. Normally such a handle will be provided by a RASF with the 'Particle' purpose. Different to the 'Kill' command the target particle will not be removed from the simulation.

#### Example:

```
Transfer{ (Drifter, Handle), (A, B, 0.5, 0.2, 0.5, 0.2) }
```

Transfer data between the current particle and the particle from class 'Drifter' identified by the 'Handle'. The current particle will transfer 50% of its variable A to variable B of the target particle. To keep mass balance it will lose 50%. At the same time 20% of the variable B of the target specie will be transferred to the current particle variable A. Again to keep mass balance, the target particle will lose 20% of B.



### Special variable property [ALL]

To address all existing state variables in one modification specification the special variable property '[ALL]' can be used. This will apply the specified modification to all existing state variables of the active particle. This special variable is most usefull when you want to split an existing particle and equally distribute the existing variable.

#### Example:

SPLIT { (Drifter, 4), ([All], [All], 0.2, 0, 0.8, 0) }

Split the current particle into four new and distribute all state variables by transferring 20% of the existing amount and reducing it by 80%. This will apply to ALL exiting variables, not just the variable 'A' as in the above 'Split' example.

### 5.10.10 The XML track format

The position and status of particle tracks can be saved to a result file for visualisation or post processing. The general structure of such an XML file is shown in schematic form below (closing tags partially omitted in the overview); it consists of a DHI data element containing a declaration/ header describing the saved data and a data element for each saved time step.

<?xml version='1.0', encoding='utf-8'?>	XML declaration
<dhidata xmlns='PTM_v3.xsd'>	DHI data element
<DataAttributes>	Data header, see below
<TimeStep nr='[n]'>	Data for time step [n], see below
</dhidata>	Closing data element
<b>Data header element</b>	
<DataAttributes>	
<TimeStepSeconds>	Time step length
<Engine>	Numeric engine name string
<StartTime>	Time stamp for output start
<ExpectedEndTime>	Expected end time for output
<Projection>	Projection for stored X,Y,Z data
<ParticleClassDefintion>	Declaration of particle class(es), see below

</DataAttributes>	Closing header element
<b>Particle class</b>	
<ParticleClassDefintion>	
<ClassID>	Output class ID
<Name>	Name string
<NrVariabels>	Number of saved variables
<var>	Variable declaration element, see below
</ParticleClassDefintion>	Closing Particle declaration
<b>Variable declaration element</b>	
<var>	
<code>	Element (MIKE ECO Lab) Identifier
<Name>	Name string
<EUMType>	EUM type string
<EUMUnit>	EUM unit string
</var>	Closing variable declaration

The data for each time step is saved into an own element. This element will contain the information for each written particle class and each of its active particle.

Data for time step [n]	
<TimeStep nr='[n]'	
<DataTime>	Time stamp
<ParticleClass id='[k]'	Data for output class ID=[k], see below
</TimeStep>	Closing time step

**Particle data:** The data for each active particle is saved an own element inside the time step data. Note that there are two 'formats'; a default, so called 'compressed' and an uncompressed one. In the compressed format all





data for a particle is stored as a comma separated block in a 'CDATA' element whereas an uncompressed file will have individual elements for a variable.

Compressed track:	
<Particle Nr='[m]'	Data for particle <b>[m]</b>
<![CDATA[ a,b,c,...]]>	<u>Values</u> for variables a,b,c...
</Particle>	Closing time step
Uncompressed track:	
<Particle Nr=[m]>	Data for particle <b>[m]</b>
<a> value </a>	Data for variable a=value
<b> value </b>	Data for variable b=value
<c> value </c>	Data for variable c=value
...	
</Particle>	Closing time step

## Compressed vs. uncompressed particle tracks

The information in both formats is the same but a compressed file has significant less memory demand. DHI tools can handle both formats but for post processing using external tools the uncompressed format is more advantageous. However, particle tracks can become large and many 3<sup>rd</sup> party tools can not effectively handle large XML files. In this case you might consider splitting the output into different files (based either on particle numbers or covered time).



**HINT:** When a simulation terminates not correctly, often the last closing data element '</dhidata>' is not written to the file. In this case the file will be considered to be 'not well defined' as a missing closing tag violates the XML definition. To resolve this problem use a plain text editor and add the missing '</dhidata>' element at the end of the file. In some cases you have to delete the last invalid time step data as well.





## 6 Error Messages and Codes

Table 6.1 MIKE ECO Lab error messages and codes

Decimal	Hex	Description
-2147024882	0x8007000e	Out of memory
-2147467261	0x80004003	NULL pointer
-2147024809	0x80070057	Invalid argument
-2147287038	0x80030002	File not found
-2147220991	0x80040201	Illegal command for state
-2147220990	0x80040202	Cannot load process matrix database
-2147220989	0x80040203	Cannot load process matrix database
-2147220736	0x80040300	Error in input file
-2147220735	0x80040301	IDs must be numbered consecutively
-2147220734	0x80040302	Symbol already defined
-2147220733	0x80040303	Invalid scope for state variable
-2147220732	0x80040304	Unknown built-in variable
-2147220731	0x80040305	Unknown built-in function
-2147220730	0x80040306	Unknown built-in function parameter
-2147220729	0x80040307	Illegal combination of scope and spatial variation
-2147220728	0x80040308	Undefined variable
-2147220727	0x80040309	Wrong variable type
-2147220726	0x8004030a	State variable already has an equation
-2147220725	0x8004030b	Error creating IF statement
-2147220724	0x8004030c	Illegal expression element scope
-2147220723	0x8004030d	Internal compiler error
-2147220722	0x8004030e	Wrong number of arguments
-2147220721	0x8004030f	Cannot get procedure address from PMDB



Table 6.1 MIKE ECO Lab error messages and codes

Decimal	Hex	Description
-2147220718	0x80040312	Cannot use 'Horizontal and Vertical' variable in expression for 'Horizontal' variable
-2147220717	0x80040313	Illegal scope for settling process
-2147220716	0x80040314	Illegal scope for sediment water transfer process
-2147220715	0x80040315	Functions with direction 'Down' must have at least 1 parameter
-2147220714	0x80040316	To deep nested IF-expression, maximal 8 nestings allowed!
-2147220480	0x80040400	Virtual machine error
-2147220479	0x80040401	Vector list already defined
-2147220478	0x80040402	No vector list
-2147220477	0x80040403	Vector already defined
-2147220476	0x80040404	Invalid vector list index
-2147220475	0x80040405	Illegal vector length; must be greater than 0
-2147220474	0x80040406	Invalid vector index
-2147220473	0x80040407	Invalid variable id
-2147220472	0x80040408	set_value() can not be used with state variables; use set_initial_value()
-2147220471	0x80040409	set_initial_value() can only be used with state variables; use set_value()
-2147220470	0x8004040a	More PMDB function parameters than VM can handle
-2147220469	0x8004040b	PMDb function null pointer encountered
-2147220468	0x8004040c	Invalid gradient property for Lagrange gradient cal
-2147220466	0x8004040e	Unknown or invalid math function for Lagrange gradient
-2147220465	0x8004040f	Error getting Lagrange value for Lagrange gradient
-2147220464	0x80040410	Invalid species name/unknown species
-2147220226	0x800404fe	Unknown variable



Table 6.1 MIKE ECO Lab error messages and codes

Decimal	Hex	Description
-2147220225	0x800404ff	Unknown state variable
-2147220223	0x80040501	Integration step size smaller than minimum
-2147220222	0x80040502	Too many integration steps
-2147220221	0x80040503	Step size not significant in RKQC
-2147220220	0x80040504	Blow up error
-2147220219	0x80040505	Unknown integration method





## INDEX



<b>A</b>	
ABM	123
ABM random bearing	32
ABM search tree	32
ABOVE_GROUND	129
AGE	129
Agent based modelling (ABM)	123
Angle unit	33
ANGLEBETWEEN	109
ARCCOS	93
ARCSIN	93
ARCTAN, ARCTAN2	93
Arithmetic expressions	127
Auxiliary variables	20, 76
AVERAGE_WATER_COLUMN	113
Averaging functions	85, 113
<b>B</b>	
B_RANDOM	101
Based on property	127
Based on variable	127
Based on variable/property	139
BELOW_SURFACE	129
Bio-geochemical modelling	73
Biological/aquatic domain functions	85, 101
Built_in ID	41, 44
Built-in ABM functions	133
Built-in constants	18, 82
Built-in forcings	83
Built-in functions	85
<b>C</b>	
CEIL	89
Compressed/uncompressed	149
Consider	126, 136
Constants	16, 74, 125
control structures	80
COS	91
COSH	92
Create	144
Create command	145
<b>D</b>	
Date/time functions	85, 94
DAY	98
DAYNUMBER	94
Debug level	32
Debug log	32
DEG2RAD	90
Derived output	26
Derived outputs	78
Direct add	143
Direct assign	143
Direct division	143
Direct minus	143
Direct multiplication	143
Direction	66
Dispersion/random walk	133
DISTANCE_TO	110
Downward/vertical movement	132
<b>E</b>	
E_RANDOM	100
ECO Lab Template	70
ECO Lab version	31
Eulerian framework	71
EXP	85
Expressions	80
<b>F</b>	
Field of view	54
Final movement vector	128
FLOOR	89
Forcings	75
Framework	126, 135
Function	127, 137
<b>G</b>	
G_RANDOM	100
GETANGLE	108
GETDISTANCE	107
GETX	107
GETY	107
Global	18, 19, 22, 25, 27
<b>H</b>	
HDIR	129
HITSTATE	129
Horizontal	18, 19, 22, 25, 27, 80
Horizontal and vertical	18, 20, 25, 27, 80
Horizontal movement	128, 132
HOUR	97
HSPEED	129
<b>I</b>	
IBM	123
Identifier	69





Identifier naming scheme . . . . .	69	Parser version . . . . .	32
Individual based modelling (IBM) . . . .	123	Particle class . . . . .	124
Individual properties . . . . .	129	pH functions . . . . .	85
INSIDE . . . . .	110	PI . . . . .	89
Interaction with other elements . . . . .	142	Polar vectors . . . . .	128
Interaction with other individuals . . . .	144	Position projection . . . . .	54
Interaction with state variables . . . . .	142	POW . . . . .	88
<b>K</b>		PREV_HITSTATE . . . . .	130
Kill . . . . .	144	PREV_XPOS . . . . .	130
Kill command . . . . .	146	PREV_YPOS . . . . .	130
<b>L</b>		PREV_ZPOS . . . . .	130
Lagrangian framework . . . . .	71, 72	Process orientated modelling . . . . .	73
LAMBERT_BEER_1 . . . . .	104	Process type . . . . .	50
LAMBERT_BEER_2 . . . . .	105	Processes . . . . .	23, 77
LN . . . . .	86	PROJ_HITSTATE . . . . .	130
LOAD . . . . .	112	PROJ_XPOS . . . . .	129
Local perception . . . . .	135	PROJ_YPOS . . . . .	129
LOG10 . . . . .	87	PROJ_ZPOS . . . . .	130
Long range perception . . . . .	135	Property . . . . .	129
<b>M</b>		Purpose . . . . .	126, 136
MAX . . . . .	87	<b>R</b>	
MICHAELIS_MENTEN1 . . . . .	103	RAD2DEG . . . . .	90
MICHAELIS_MENTEN2 . . . . .	104	RAND . . . . .	98
MIN . . . . .	87	RAND seed . . . . .	32
MINUTE . . . . .	97	Random number functions . . . . .	85, 98
Miscellaneous . . . . .	31	Random walk process . . . . .	128
Miscellaneous features . . . . .	30	RANDSEED . . . . .	98
MOD . . . . .	89	RELATIVE_DAYLENGTH . . . . .	94
Model setup . . . . .	70	Remove . . . . .	144
Modification specification . . . . .	144	Remove command . . . . .	145
MONTH . . . . .	98	Restricted area search functions . . . .	135
Movement . . . . .	132	RESULTINGDIRECTION . . . . .	109
Movement projection . . . . .	133	RESULTINGSPEED . . . . .	108
MOVING_AVERAGE . . . . .	114	REVERSE_MICHAELIS_MENTEN . . . . .	104
<b>N</b>		<b>S</b>	
N_RANDOM . . . . .	99	SAVE . . . . .	112
N_RANDOM2 . . . . .	100	Scope . . . . .	15
None . . . . .	80	Scopes . . . . .	78
<b>O</b>		Search radius . . . . .	140
Optimise expressions . . . . .	33	SECCHI_DEPTH . . . . .	105
Output . . . . .	47	SECOND . . . . .	97
<b>P</b>		Sediment . . . . .	15
P_RANDOM . . . . .	100	Sensing functions . . . . .	134
		SET_FOV . . . . .	111
		SET_PROJECTION . . . . .	111
		SET_RADIUS . . . . .	111
		SIN . . . . .	91



SINH . . . . . 92  
SMOOTHING\_AVERAGE . . . . . 114  
Space Lagrange framework . . . . . 140  
Spatial representation . . . . . 132  
Spatial variabilities . . . . . 80  
Spatial variation . . . . . 17, 19, 22, 27  
Special ABM functions . . . . . 85, 107  
Special functions to load/save . . . . . 85, 111  
Speed . . . . . 66, 68  
Split . . . . . 144  
Split command . . . . . 146  
SQR . . . . . 88  
SQRT . . . . . 88  
Standard mathematical functions . . . . . 85  
State variables . . . . . 14, 73, 124  
SUNRISE . . . . . 95  
SUNSET . . . . . 96  
Symbol . . . . . 37

## T

Table functions . . . . . 85  
TAN . . . . . 91  
TANH . . . . . 92  
Target ID parameter . . . . . 144  
Template revision . . . . . 32  
TERMINATE . . . . . 111  
TESTHITSTATE . . . . . 110  
Transfer . . . . . 144  
Transfer command . . . . . 146  
Transport . . . . . 38  
Trigonometric functions . . . . . 85, 90

## U

U\_RAND . . . . . 99  
U\_RAND2 . . . . . 99  
Unit types . . . . . 18, 20

## V

Vertical movement . . . . . 128  
VSPEED . . . . . 129

## W

Water bed . . . . . 15  
Water column . . . . . 15  
Water surface . . . . . 15

## X

XML track format . . . . . 147  
XPOS . . . . . 129

## Y

YEAR . . . . . 98  
YPOS . . . . . 129

## Z

ZPOS . . . . . 129