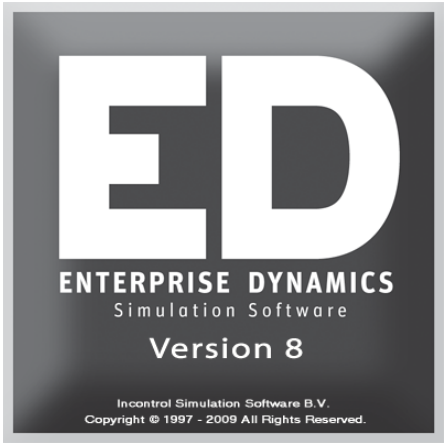
ANNEXES TUTORIAL



Simulation Software / TUTORIAL ANNEXES

Enterprise Dynamics®

Papendorpseweg 77, 3528 BJ Utrecht, The Netherlands
www.IncontrolSim.com

INCONTROL
Simulation Solutions

Annex 1 The menu structure

NB: An option displayed in *italic* is more suitable for the advanced user.

File	*Explanation*
New Model	Closes the current model and opens an empty model.
Open Model…	Closes the current model and opens an existing model.
Merge Model…	*Opens a model and includes this model into the selected atom or into the current model. Consequently, the existing model is not deleted.*
Save Model	Saves the current model under the same file name.
Save Model As…	Saves the current model under a new file name. The previous model remains.
Add Atom to Library…	*Loads an atom and places it at the bottom of the library.*
Save TreeAtom	*Saves the selected atom in the Library Tree to pdir([atoms]) if the workdir is set to pdir([Work]). Else the atom will be saved in Workdir([Atoms\GroupDir]).* An atom can only be saved if it is not a group atom and it is directly under a group atom or under the library. When an atom is successfully saved, a message will be displayed that and where the atom is saved.
Save Atom As…	*Displays a window allowing you to select an atom out of the library. Afterwards, the selected atom can be saved under a new name.*
Import	*This option allows you to import a 2D/VR (VR=Virtual Reality) icon or a VR sound. The icon or sound is added to the lists with icons and sounds that the user can allocate to an atom.*
Print 2D Layout…	Prints the 2D window on the standard printer.
Print Setup	Allow you to define the settings of the standard printer.
Preferences	This option displays a number of tabs to define the standard settings of Enterprise Dynamics.
Startup Script	*Allow you to modify the Start-up Script. This script is performed each time Enterprise Dynamics is started. The modifications have to be defined in 4DScript, the programming language of Enterprise Dynamics.*
Exit	Shuts down Enterprise Dynamics.

Model	*Explanation*
Create	Shows the library tree and the model layout window. You can build your own model by dragging atoms into the model layout.
Layout window	Shows the model layout window. Atoms can be created in this window either by dragging atoms from the library or by

	using the Taskbar.
Sublayout window	*Shows another layout window, but one hierarchical layer lower, e.g. the contents of a composition container.*
Model Tree	Shows the model tree. This 'model tree' gives a hierarchical overview of the atoms displayed in the model.
Library Tree	This option shows all the atoms in the library.

Simulate	***Explanation***
Run Control	Displays and activates the Run Control Window which is used to start, stop and modify the speed of a simulation run.
Clock	Displays the clock window.
Run	Starts the simulation run.
Stop	Stops the simulation run.
Stop and Reset	Stops the simulation run and resets the model.
History	Allows you to collect data during a single run of the model which allows you to afterwards generate graphs and reports from that run from the menu Results|Graphs. The option 'general history' needs to be checked. It is linked to the simulation via the run control window. When graphs and reports of a specific atom are requested, the history of this atom has to be maintained. The easiest way to achieve this is to select the option 'All on'. However, this results in the history collection of all atoms, which can lead to huge data files!

Results	***Explanation***
Summary Report	The Summary Report displays an overview of the basic statistics relating to all atoms present in the model, **based on a single run**. NB: For more detailed reports, the Report Atom can be dragged out into the model.
Graphs	Shows various graphs of atoms such as queues, histograms and pie charts, **based on a single run**. These graphs can only be created if the History option relating to the atom in question is switched on.

Experimentation	***Explanation***
Experiment Wizard	Displays the Experiment Wizard in which you can define and run your experiment. At first it guides you in defining your experiment settings such as the run length and a warm-up period and secondly the output variables (performance metrics PFM's). After defining the experiment you can choose to start the experiment.
Analyze Experiment Results	Allow you to display the report of an experiment.

Tools	*Explanation*
Atom Editor	*With the Atom Editor, you can adjust the functionality as well as the appearance of an atom.* *This very effective tool allows you to alter the behaviour of existing atoms and to create your own atoms. The programming language 4DScript has to be used here.*
4DScript Interact	*A window in which 4DScript can be entered. Direct execution of the command follows.*
Text Editor	*A simple text editor, the functionality of which can be compared to MS Notepad.*
Debugger…	*The debugger allows you to analyze your 4DScript code step-by-step while it is being executed.*
CAD Import wizard	Add on tool CAD Import wizard.
GUI Builder…	GUI is short for Graphical User Interface: it allows the user to create his own input fields.
Scenario Manager	The scenario manager assist you in running multiple simulation runs after each other. In this way you can start a scenario, go home, and find the results in the morning.
View Atom Labels	This option displays all labels of the selected atom (and of all atoms contained in that atom). Labels are variables and attributes the user can allocate to an atom.
Autofit	The Autofit function analyses a data set and searches for the best fit probability distribution.

Display	*Explanation*
2D Model Layout	Opens the 2D modelling window.
2D Model View	Opens the 2D visualization window. *Warning!* In this window, you cannot add atoms to the model or reposition existing ones. For this you must use the Model Layout window.
2D Model Subview	*Opens the same window as the 2D Model View option, but here, only the contents of the selected atom is shown.*
2D Background Color…	This option allows you to change the background color of your active 2D modelling window.
3D Model View	Opens the 3D visualization window.
3D Model Subview	*Opens the same window as the 3D Model View option, but here, only the contents of the selected atom is shown.*
3D Background Color	*Opens a color selection window. This option allows you to define the background color for the 3D window. This color will also be used in the VR window.*

Search	*Explanation*
Search Text or Atom	Opens the Search window.
TreeAtom in 2D Model view	Makes the AnimAtom the TreeAtom.
AnimAtom in Treeview	Opens the treeview and makes the TreeAtom the AnimAtom.
Mother of TreeAtom in Treeview	Makes the TreeAtom the mother of the selected TreeAtom.

Window	*Explanation*
Close all windows	Closes all open windows.
4DScript Overview	Shows an overview of all 4DScript expressions together with an explanation of their syntax. You can also open this window by pressing F2.
Error Monitor	*Opens a window displaying errors encountered in 4DScript.*
Tracer	*Opens the Tracer window. You can enter 4DScript expressions in this window.*
Layers	*You can create models on various layers allowing you to lock certain layers or to hide them. This can be very useful in large models where atoms are piled on top of each other.*
Resources Manager	*Opens a window in which all available atom icons are displayed.*
Graph Window	*Opens the most recent graph. It is not possible to produce new graphs in this window.*

Help Menu	*Explanation*
Help Overview	Gives you access to the complete manuals, consisting of the 3 following menu sub-units.
Quickstart	These documents cover and explain some of the new features of Enterprise Dynamics.
Tutorials	Includes various tutorials to teach you how to work with Enterprise Dynamics.
Add-ins	*These add-in help files contain some useful information of some the additional packages you can obtain of Enterprise Dynamics.*
Example Wizard	Opens the Example wizard. The Example Wizard contains some of the Example models that are included in ED.
About Enterprise Dynamics	Displays information regarding the version in use and about Incontrol Enterprise Dynamics.

Speed buttons on the Main Toolbar

There are several speed buttons available in the main toolbar. Some of these buttons are for standard File actions such as saving your model. Others can be used to show a special window such as the Model Layout and the Run Control. Most of the buttons are to quickly insert an atom in the Model Layout. There are also buttons for tools such as the Autofit tool or a tool to view the Labels of an atom.

File actions	***Explanation***
	Create a new model.
	Open model an existing model.
	Save the model.
	Save library. Asks for a library name and creates an .lbr file (in Pdir([Libs]) if workdir is pdir([Work]), else in WorkDir([Libs])) of the selected library atom(tree). This includes structuring using group atoms, creation of progress bar,
	Automatically creates a preregister file for all functions used in the library. Can be used to prevent errors when loading a library.
	Print the current view of the Model Layout.

Window actions	***Explanation***
	Show Model Layout and Library Tree.
	Show Model Layout in which you can build your model.
	Show the 2D model view. In this view you can only change the atom settings and not insert new atoms.
	Show the 3D model view.
	Show layer window. Create Layers and set the current Layer in the model. And change the settings of each layer, i.e., set whether atoms are visible, atoms are resizable, atoms are selectable or set if atoms can be deleted.
	Show the Library Tree.
	Show the Model Tree.
	*Show the Atom Editor.*
	Show the Run Control.
	Show the Clock.
	*Show the GUI Builder.*
	Show the 4DScript Interact window.
4D	Show the 4DScript overview.
	Show the text file editor.
	Show the Summary Report.
	Show the Graph window.
	Show the Help.

Insert atoms	*Explanation*
Basic modelling	
	Insert a **Source** atom in the Model.
	Insert a **Queue** atom in the Model.
	Insert a **Server** atom in the Model.
	Insert a **Sink** atom in the Model.
	Insert a **Container** atom in the Model.
	Insert a **Node** atom in the Model.
Processes	
	Insert an **Assembler** atom in the Model.
	Insert an **Unpack** atom in the Model.
	Insert a **Splitter** atom in the Model.
	Insert a **Multiserver** atom in the Model.
Product transformation	
	Insert a **Single transform** atom in the Model.
	Insert a **Multiple transform** atom in the Model.
Storage	
	Insert a **Warehouse** atom in the Model.
Transport	
	Insert an **Accumulating conveyor** atom in the Model.
	Insert a **Non-Accumulating conveyor** atom in the Model.
	Insert a **Transporter** atom in the Model.
	Insert a **Dispatcher** atom in the Model.
	Insert a **Destinator** atom in the Model.
	Insert a **Portal Crane** atom in the Model.
	Insert a **Robot** atom in the Model.
Network *Atoms necessary to build a Network for an Advanced Transporter or Operator.*	
	Insert a **Network node** atom in the Model.
	Insert a **Node Manipulator** atom in the Model.
	Insert a **Network Controller** atom in the Model.
Operators	
	Insert an **Operator** atom in the Model.
	Insert a **Team** atom in the Model.
	Insert a **Call Operator** atom in the Model.
	Insert a **Free Operator** atom in the Model.
Time	
	Insert an **Arrival List** atom in the Model.
	Insert a **User Events** atom in the Model.

Tools	
	Insert a **Composition Container** atom in the Model.
	Insert an **Empirical Distribution** atom in the Model.
Availability	
	Insert an **Availability Control** atom in the Model.
	Insert a **MTBF MTTR availability** atom in the Model.
	Insert a **Time schedule availability** atom in the Model.
Flow control	
	Insert a **Lock** atom in the Model.
	Insert an **Unlock** atom in the Model.
	Insert a **Condition Control** atom in the Model.
	Insert a **Notify Router** atom in the Model.
Visualization	
	Insert a **Text Box** atom in the Model. Insert Static text in the 2D Model
	Insert a **Bitmap Box** atom in the Model. Insert an Image in the Model
Results	
	Insert a **Monitor** atom in the Model.
	Insert a **Status indicator** atom in the Model.
	Insert a **Status monitor** atom in the Model.
	Insert a **Status histogram** atom in the Model.
	Insert a **Status monitor stacked bar** atom in the Model.
	Insert a generic **Circle diagram** atom in the Model.
	Insert a generic **Histogram** atom in the Model.
	Insert a **Scatterplot** atom in the Model.
Data	
	Insert a **Table** atom in the Model.
Data atoms to establish a DDE connection with other applications	
	Insert a **Word** atom in the Model.
	Insert an **Excel** atom in the Model.
	Insert a **Database** atom in the Model.

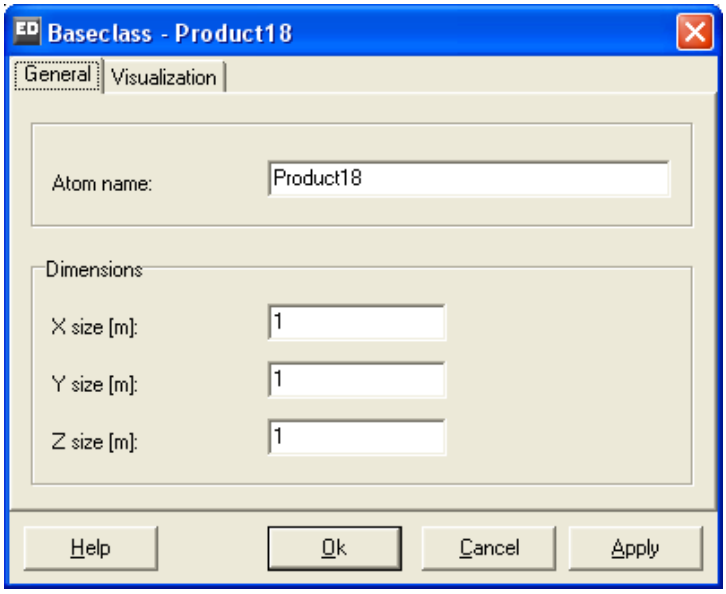
Tools	***Explanation***
	Show the Autofit tool.
	Show the Label window containing a table with the Labels of the atom selected in the Model Layout and of the atoms contained in the selected atom.
	*Show a window that you can use to add attributes and corresponding access functions. This form is meant for the expert user.*

Debugging tools	***Explanation***
	Show the Tracer Window.
	Show the Error Monitor.
	Show the Watches.
	Show the Debugger.

Search tools	***Explanation***
	Find the selected tree atom in Model Layout.
	Find the atom selected in the model layout, i.e., the AnimAtom, in the model tree.
	Find the mother atom of the selected tree atom.
	Tool to find general text, code or atom in specified area such as the library or the Model.

Annex 2 Description of a few atoms

1 THE PRODUCT ATOM



Picture 1-1: The Product Atom

The Product Atom is used to model the physical flows in Enterprise Dynamics. These flows can consist of products, goods, documents or persons. The following atom settings can be defined:

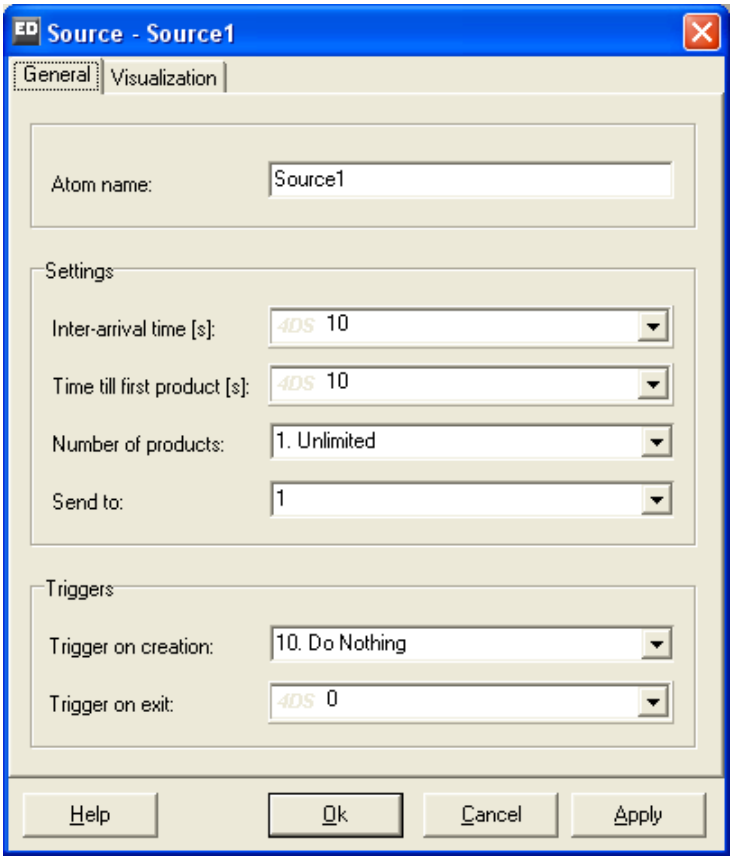
General Tab:

- *Atom name*
  The name given to the atom.
- *Size X*
  The size of the atom in the x direction (length in meters).
- *Size Y*
  The size of the atom in the y direction (width in meters).
- *Size Z*
  The size of the atom in the z direction (height in meters).

General visualization:

- *2D Icon*
  The symbol used to represent the Product atom in the 2D window.
- *Show 2D Icon*
  Gives you the possibility to display the 2D Icon. If the option is checked (standard setting), the icon is displayed.
- *3D Icon*
  The symbol used to represent the Product atom in the 3D window.
- *Color*
  The color given to the atom.

2 THE SOURCE ATOM



Picture 2-1: The Source Atom

The Source Atom allows atoms, mostly products, to enter the model at a specified rate and therefore functions as a product or customer generator. This atom is often the first element of a model.

The following settings can be adjusted:

General Tab:

- *Atom name*
  The name given to the atom.
- *Inter-arrival time*
  Time between 2 product atom arrivals. This time is measured in seconds and can be constant, but also defined by a probability distribution. Click on the right triangle to display the pull down menu featuring a number of possible probability distributions with parameters and values.
- *Time till first product*
  Arrival time of the first product. After this first arrival the probability distribution from the inter-arrival time applies.

Default value is 10 seconds; if you want all the products have the same inter-arrival time, choose the same expression as used in the inter-arrival time.

- *Number of products*
  With this option it is possible to limit the entry of products to your model. There are two options:
  1. Unlimited (default)
  2. Generate maximum 100 products

  With option two you can choose your preferred number of arrivals. This is very similar to the more general Lock atom.
- *Send to*
  Displays the number of the output channel through which other atoms (mostly products) leave this atom. A number between 1 and the total number of the atom's output channels has to be displayed here. If the result is 0, sending never takes place. If an atom is blocked because the input channels of the receiving channels are closed, the 'Send to' statement is re-evaluated only when the situation changes and sending is allowed.

  In the Send to option, the user can thus enter a figure or select one of the following pre-defined options. In these options, the **bold** items (shown in blue on the screen) can be altered by the user:

  *1: Specific channel: always send to channel **1**.*
  The Product Atom will always be sent to a defined output channel.

  2: An open channel (First channel first): search, starting from the first channel, and send to the first open channel found.
  The Product Atom is sent to the first open channel that Enterprise Dynamics finds. The search starts from the first output channel, then to number two and so on.

  3: *An open channel (Last channel first): search, starting from the last channel, and send to the first open channel found.*
  The product is sent to the first open channel Enterprise Dynamics comes across, starting from last channel to the one before and so forth.

  4: *A random open channel: choose a random channel from all the open output channels.*
  Enterprise Dynamics selects a random channel from all open channels. With long simulation runs, it results in equal utilizations of e.g. a group of servers.

  5: *By percentage: **90**% of products go to channel **1**, the remaining percentage go to channel **2**.*
  A definite percentage of the products is sent to a specific channel and the rest to another channel. The user can define the channels and the percentage.

  6: *By atom name: if the atom name of the **1st** atom in the queue matches **AtomName** then send to channel **1** else **2**.*
  The atoms are forwarded on the basis of their names. If the name corresponds to the name the user entered, the products are sent to channel **1** and otherwise to channel **2**. The user can adjust the channel numbers and the atom names.

  7: *By label value (direct): the channel number is written directly on the label named **LabelName** of the **1st** atom in the queue. If the label value is 0 then send to channel **1**.*
  The atoms are forwarded on the basis of a label value. The user has defined a

name for the label. The value of the variable corresponds to the output channel's value. If the value is 0, a pre-defined exit is used. Note that searching for a label not present on the atom results in the value 0 as well.

8: *By label value (conditional): if the value on the label named **LabelName** of the **1st** atom in the queue is **<** the value **1** then send to channel **1** else **2**.*
Here too, the value of a specified label determines the choice of the output channel. If the value of the atom is lower than 1, the atom is sent to channel **1**, otherwise to channel **2**. All values and comparisons (lower than, higher than, equal to) can be edited.

9: *By label text: if the text on the label named **LabelName** of the **1st** atom in the queue matches **text** then send to channel **1** else **2**.*
When the value of a defined label is equal to a specific text, the atom is sent to channel 1, otherwise to channel 2. The text and the channel numbers are editable.

10: *Conditional statement: If **1** is **>** than **0** then send to channel **1** else send to channel **2**.*
If a specific value is higher than another value, the atom is sent to channel **1**, otherwise to channel **2**. The comparisons and channel numbers can be edited.

11: *By icon name: if the icon name of the **1st** atom in the queue matches **IconName** then send to channel **1** else **2**.*
If the name of the atom icon corresponds to a defined name, the atom is sent to channel **1**, otherwise to channel **2**. The icon names and channel numbers can be specified.

12: *By icon number: if the icon number of the **1st** atom in the queue is **=** the value **1** then send to channel **1** else **2**.*
If the atom's icon number **is equal to 1**, the atom is sent to channel **1**, otherwise to channel **2**. The comparisons and channel numbers can be defined.

13: *Round robin: all output channels are used in rotation. If channel is closed, then wait till open.*
All output channels are used consecutively. If a channel is not open, Enterprise Dynamics waits until it becomes open.

14: *Lowest queue: Send to the channel connected to the atom with the lowest queue.*
The atom is sent to the output channel with the shortest queue. In the case of equal queues, the output channel with the lowest number is chosen.

15: *Largest queue: Send to the channel connected to the atom with the largest queue.*
The atom is sent to the output channel with the longest queue. In the case of equal queues, the output channel with the highest number is chosen.

16: *Lookup table: Send to the channel specified in row **1** column **2** of global table named **table1**.*
Sends the atom to the channel defined in row **1** and column **2** of a table. The row and column numbers, as well as the table name, can be specified.
Note that the table must be present in the model as a separate atom!

17: *Round robin if available: all output channels are used in rotation if channel is available. If channel is closed, then next available channel is chosen.*
All channels are used consecutively, but when the channel required is closed, the next available channel is selected.

18: *Matching icon number or empty: Sends to a queue containing products of same icon. If no icons match, then sends to first empty queue starting with last output channel.*
Forwards atoms so that they always arrive in a queue containing atoms with the same icon number. If a queue containing atoms with the same icon number is not found, ED searches for the first empty queue, starting from the queue connected to the last output channel.

19: *Lowest queue of next two atoms: Sends to the output channel connected to the lowest queue, where lowest queue takes into account the next TWO atoms.*
Enterprise Dynamics evaluates the total queue for each atom connected to an output channel, and the atom connected to that atom. It then sends the next atom to the channel connected to the queue with the lowest contents. For example, an atom can be sent to 3 different queues, each of which is followed by a server. This option prevents the products from being sent to a queue where the server is already in action, while the other servers are available.

20: *By user: enter your own 4DScript expression resulting in a value between 1 and the number of channels: 1. You can press the small button for the 4DScript editor.*
The user writes a 4DScript code that results in the output channel. Clicking on the small square button by the text will open the 4DScript editor.

21: *Random channel: randomly choose a channel. If the channel is open then send to it, otherwise choose again when any channel opens.*
Enterprise Dynamics chooses a random channel. If this channel is open, the product is sent to that channel. However, if it is closed, choose again when another channel opens.

- *Trigger on Creation*
The command in this field is performed when an atom enters the model. The user can define their own 4DScript expression, or pick from one of the following options:

1: *Assign label: products are assigned a label named **LabelName** with a value of **1**.*
The products are assigned a label with a specific name and a definite value. The label name and the value can be defined.

2: *Auto Name: a counter is added to the end of each product's name.*
A counter is added to the product's name. The first product is called for example Product1 and the second Product2.

3: *Random icons: products are assigned a random icon number between **2** and **6**.*
A random icon is allocated to each product. The icon number lies between two defined values.

4: *Set Size: product dimensions are set to: X= **50** cm, Y= **40** cm, Z= **30** cm.*
The product's dimensions change according to the entered values.

5: *Random Size: product dimensions are randomly set within the following ranges: X= **50** to **100** cm, Y= **50** to **100** cm, Z= **50** to **100** cm.*
The product's dimensions change according to random values inside defined ranges.

6: *Set Color: products are set to the **colorpurple**.*
A product's color changes into the color defined by the user. Note that in 4DScript, the selected color has to be prefixed by the word "color". So *colorpurple* is the command for purple. Instead of the command *colorpurple*, you can also enter the color number.
7: *Random color: products are assigned a random color.*
The products are given a random color.
8: *Random Size and Color: products are assigned a random color and its dimensions are randomly set within the following ranges: X=**50** to **100** cm, Y= **50** to **100** cm, Z= **50** to **100** cm.*
The products get a random color as well as a random size (within defined ranges).
9: *Outline: display the products as a simple outline, not its icon.*
A product's icon is not visible any more, only its outline is.
10: *Do Nothing.*
Nothing happens. This is the standard setting.

- *Trigger on Exit*
The command in this field is executed when a product is leaving the atom. You can either use your own 4DScript command, or one of the pre-defined expressions. The question marks indicate where the user must enter a value.

The possible pre-defined expressions in the Trigger on Exit field are:
1: *setlabel([**?**],**?**,i).*
With this 4DScript command, a label is added to the atom leaving the Source. The code is: setlabel([**label name**], **value**, i).

Example
To allocate a label "complete" with the value 1 to the product, use the following code: setlabel([complete], 1, i).
The letter i refers to the *involved* atom. This is the atom undergoing the process of leaving the Source. If a label had to be placed on the Source itself, the i could be replaced by a c (of *current*).

Example
Setlabel([cycletime], Uniform(25,45), i) defines a label "cycletime" on the product with a value from a uniform distribution of between 25 and 45 seconds. This result can later be used as cycle time for a server, for instance.

2: *set(Name(i),[**?**]).*
Changes the name of the atom leaving the Source. The '?' must be replaced by the name chosen for the atom.
3: *set(Icon(i),**?**).*
Changes the atom's icon into the icon with the number '?'.
4: *set(Icon(i), IconByName([**?**])).*
Changes the icon into the icon with the name '?'.
5: *set(Color(i), **ColorYellow**).*
Changes the atom's color into the defined color. In Enterprise Dynamics, the colors must be specified either by their number or with the prefix "color" followed by the color in question, for example ColorRed.

6: *setsize(?,?,?,i).*
Changes the atom's dimensions according to the defined sizes (x,y,z).

7: *setloc(?,?,?,i).*
Gives the atom a new location, as defined in the command (x,y,z).

8: *FreeOperators(AtomByName([**Team**], Model), i).*
Allows the Operator atoms to be re-used. Replace 'Team' by the name of the Team atom. This option is only for advanced users.

9: *if(=(?,?),?,?).*
A conditional comparison. For example, entering the following code gives:
If(=(thesis1,thesis2),command1,command2)

If thesis1 and thesis2 are equal, command1 will be executed, otherwise command2. Command2 can also be omitted.

10: *if(=(label([?],i),?),?,?).*
A conditional comparison, in which a label's value is considered.

Example
With the following command, if the label 'Reject' has the value 1, we can color all rejected products red and all approved products green:
if(Label([Reject], i) = 1, Color(i) := ColorRed, Color(i), ColorGreen).

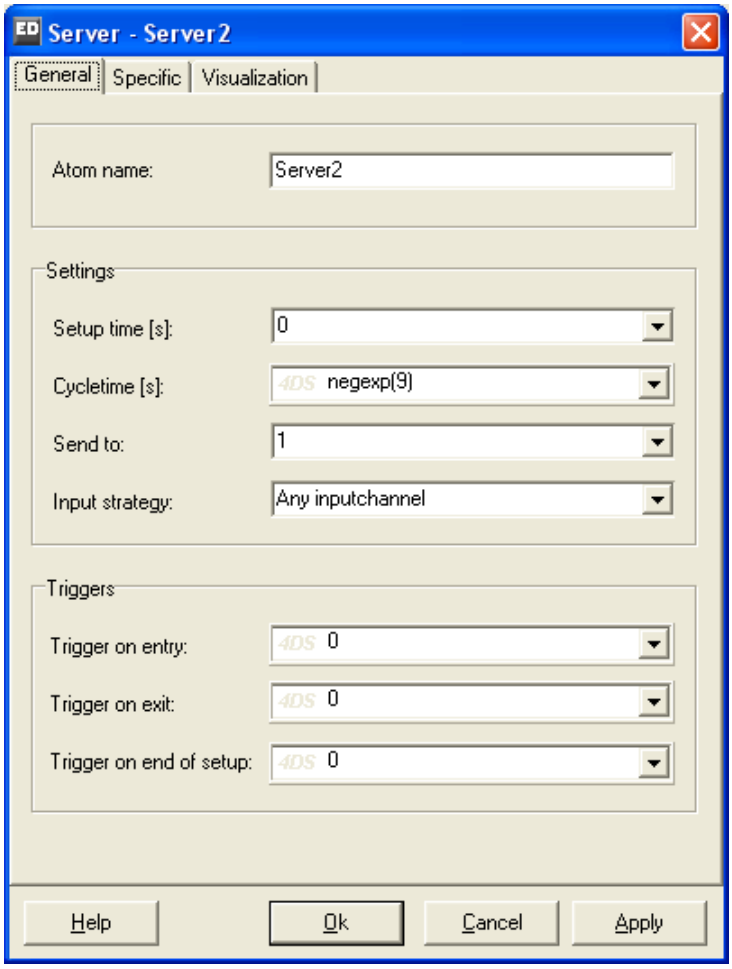
11: *if(CompareText(Name(i),[?]),?,?).*
A conditional comparison, in which the atom's name is used. Its functioning is otherwise the same as option 10.

Vizualization tab:

- *Icon*
  The symbol used to represent the Source atom in the 2D window.

For a more detailed explanation relating to 4DScript commands, we refer you to Annex 3 or the help files included in Enterprise Dynamics.

3 THE SERVER ATOM



Picture 3-1: The Server Atom

The Server is used to model operations taking a certain amount of time such as the processing of a product by a machine or a customer's settlement at a cash desk. As a result, the Server can represent a machine, a counter, an assistant or another type of processing place or device. As well as cycle times, other parameters can be defined such as setup times or the simultaneous processing of several products.

The following values are editable:

General tab:

- *Atom name*
  The name given to the atom.

- *Setup time*
  Time needed before the actual processing starts. For example: cleaning of machines, adjusting settings for new products, etc.

Clicking on the triangle opens a series of options, including one that allows the settings to be defined for each product, or for products of a different type only. In addition to using these options, the user can also create his own 4DScript.

- *Cycletime*
  The time needed to process the product. By clicking on the arrow, a list of pre-defined 4DScript commands appears.
  Important: in the case of a grouped processing of products (batch processing), the cycle time refers to the whole batch and not to each individual product. Further, the processing starts only when the batch is complete.

- *Send to*
  Displays the channel to which the products have to be sent.
  For more details: see the 'Send to' explanation of the Source atom.

- *Input strategy*
  Regulates the access to an atom from previous atoms via their output channels to this specific atom. The input strategy has several roles: it can open one or more channels and it can define the order in which products will be accepted from the available channels.

  You can compare input strategy to the sequencing of traffic lights, where some traffic lights are switched from 'red' to 'green' for one or several minor roads, irrespective of the actual traffic, and where the processing priority of these minor roads is defined.

  Warning: the first three input strategies open all input channels and the last two open one input channel each time!

  1: *Any inputchannel.*
  When activated, this strategy opens all input channels of an atom. If more than one of the atoms that are connected via the input channel can be sent, the atom arriving through lowest number input channel will have priority. While products keep entering through the first channel, the other channels will be blocked.
  2: *Largest queue.*
  When activated, this strategy opens all input channels of an atom. If more than one of the atoms that are connected via the input channel can send, the atom with the longest queue or largest contents will have priority. Note that in the case of several equally long queues, the input channel with the lowest number is chosen.
  3: *Longest waiting.*
  When activated, this strategy opens all input channels of an atom. If more than one of the atoms that are connected via the input channel can send, the atom with the highest average waiting time will have priority. In the case of several atoms with an equal waiting time, the input channel with the lowest number is always chosen. Note that it does not mean that the queues get approximately equally long, as is the case in the previous option.

4: *Round robin.*
This strategy first opens the first input channel and then waits for a product to be sent through this input channel. In the second cycle, it is the turn of the second input channel etc. When the products have run through the last input channel, the procedure is resumed with the first one.
Important remark: this strategy becomes active after the first product! So, in case of three input channels this strategy gives x,2,3, 1,2,3, 1,2,3 where x can be 1,2 or 3!

5: *Channel **1**.*
In this case, you can enter a specific input channel through which all products must enter. If 1 is entered, the products may only enter through input channel 1. Note that this rule does not apply to the first product entering as all channels are open initially.

- *Trigger on Entry*
The command entered in this field is performed when a product enters the Server.
For more details: see the 'Trigger on Exit' explanation of the Source.

- *Trigger on Exit*
The command entered in this field is performed when a product leaves the Server.
For more details: see the 'Trigger on Exit' explanation of the Source.

- *Trigger on end of setup*
A rule that determines what kind of action needs to be executed at the end of the server setup time. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

Specific Tab:

- *Batch (B)*
The batch size can be entered here. The standard setting is 1.

- *Batch rule*
To set up the batches. There are 3 options:

1: *B in, 1 out (the first).*
As soon as the number of products that have entered the Server is equal to the batch size, the product at the front is forwarded to the next atom. The other products disappear.

2: *B in, B out.*
As soon as the number of products that have entered the Server is equal to the batch size, the products are forwarded to the next atom. The Server re-admits products only when all products of the batch have left the Server.

3: *1 in, B out (copies of in).*
Each time a product enters the Server, as many products as defined in the Batch input field leave the atom. The products are all copies of the atom that entered the Server. The Server re-admits a product only when all other products have left the atom.

- *Busy time*
  When this option is checked, the time taken into consideration in the "Mean Time Between Failure" options is only the time that the Server is actually in use, and not the total simulation time.

- *MTTF*
  This abbreviation stands for Mean Time To Failure, that is the average time elapsing between the end of a repair and the beginning of next failure. This average time between two Server failures can be defined in the input field. The time must be entered in seconds.

- *MTTR*
  This abbreviation stands for Mean Time To Repair. The average time needed to fix the Server can be defined in the input field.

- *MCBF*
  Abbreviation for Mean Cycles Between Failure. The average number of cycles between two failures can be entered in the input field. In MCBF, there is not a definite time between two failures but a definite number of batches.
  NB: When both fields MTBF and MCBF are filled in, failures will be generated by both definitions.

- *MTTR for cycles*
  This Mean Time To Repair input field applies to failures defined by MCBF.

- *Trigger on Breakdown*
  The command entered in this field is performed at the start of the Breakdown period of the Server.
  For more details: see the 'Trigger on Exit' explanation of the Source.

- *Trigger on Repair*
  The command entered in this field is performed at the start of the Repair period of the Server. For more details: see the 'Trigger on Exit' explanation of the Source.

Visualization Tab:

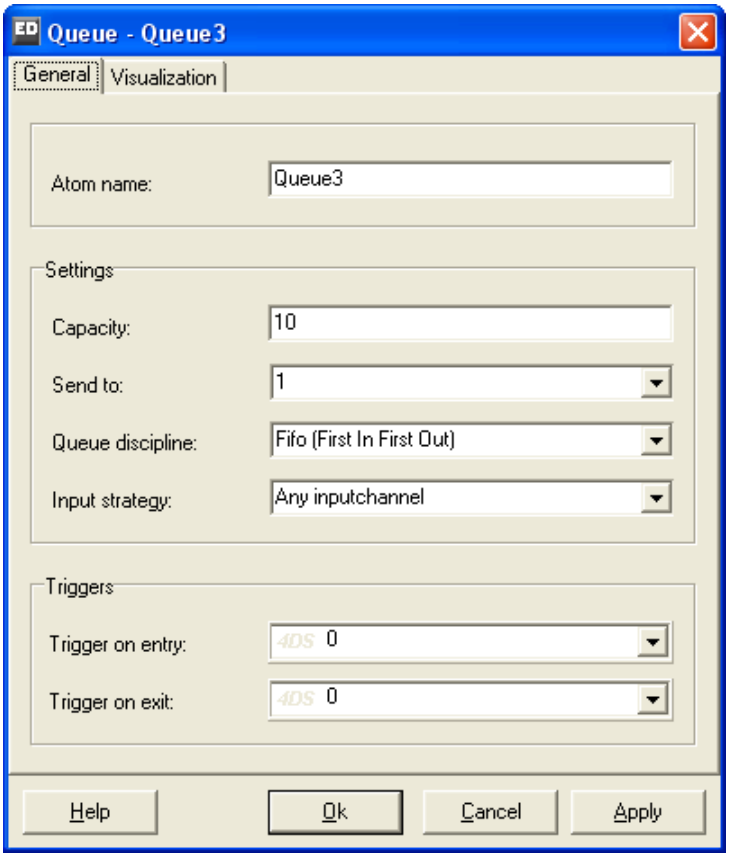
- *Icon*
  The symbol used to represent the Server atom in the 2D window.

- *3D Icon*
  The symbol used to represent the Server atom in the 3D window.

- *Main color*

- *Second color*

4 THE QUEUE ATOM



Picture 4-1: The Queue Atom

When the next atom is occupied, the Queue atom places products in a queue. The following settings can be adjusted:

General tab:

- *Atom name*
  The name given to the atom.

- *Capacity*
  The Queue's capacity. When as many products are present in the queue as defined in the capacity input field, no new products can be placed in the queue.

- *Send to*
  Displays the output channel to which the products have to be sent.
  For more details: see the Send to explanation of the Source atom.

- *Queue discipline*
  The way products are arranged in the queue. The following options are possible:

1: *First in first out.*
The atoms are put in the queue according to their order of entry.
2: *Last in first out.*
The entering atoms are placed at the front of the queue. Consequently, the products leave the queue in reverse of their order of entry.
3: *Random.*
This queue discipline places the incoming products in a random spot in the queue
4: *Sort by **Label** Ascending.*
The products with the lowest value for a specific label are placed at the front of the queue.
Warning!: if the products are not sorted properly, the cause might be a space before or after the label name.
5: *Sort by **Label** Descending.*
The products with the highest value for a specific label are placed at the front of the queue.
Warning!: if the products are not sorted properly, the cause might be a space before or after the label name.
6: *User defined.*
The products are placed in the queue according to a position defined by the user.

- *Input Strategy*
This input window can be used for indicating which input channel is to be used**.**
For more details: see the 'Input Strategy' of the Server atom.

- *Trigger on Entry*
The command entered in this field is performed when a product enters the Queue atom.
For more details: see the 'Trigger on Exit' explanation of the Source atom.

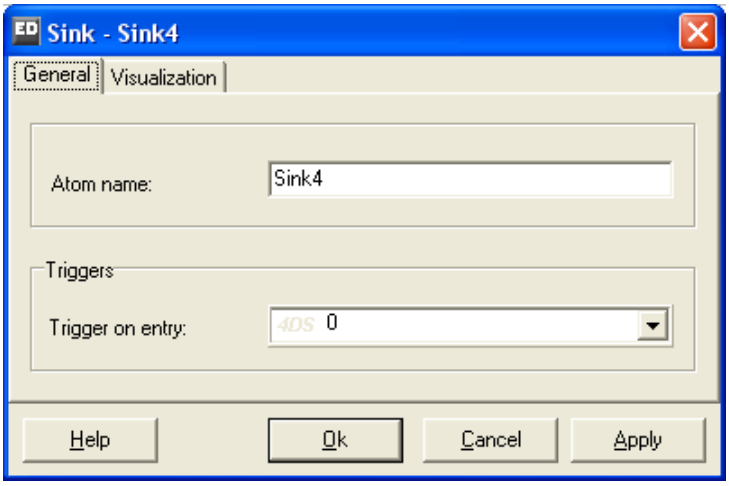
- *Trigger on Exit*
The command entered in this field is performed when a product leaves the Queue atom.
For more details: see the 'Trigger on Exit' explanation of the Source atom.

Visualization tab:

- *Icon*
The symbol used to represent the Queue atom in the 2D window.

- *3D Icon*
The symbol used to represent the Queue atom in the 3D window.

5 THE SINK ATOM



Picture 5-1: The Sink Atom

This atom allows products to leave the model. The following settings can be adjusted:

General Tab:

- *Atom name*
  The name given to the atom.

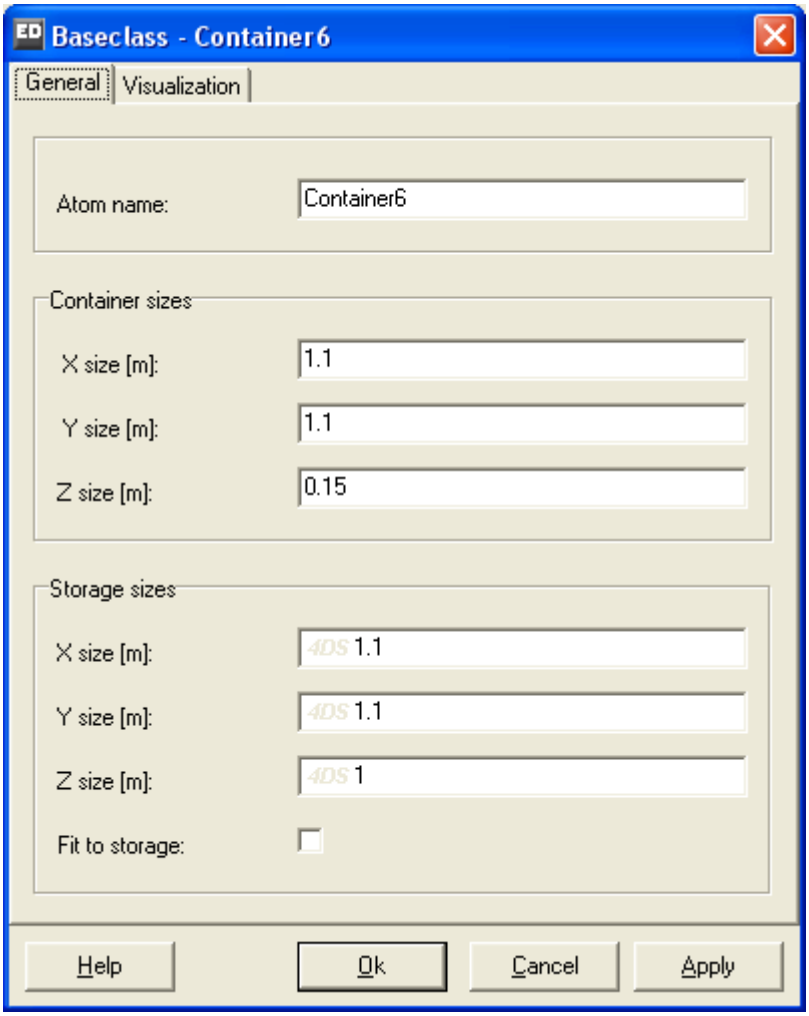
- *Trigger on Entry*
  The command entered in this field is executed as soon as a product enters the Sink.

Visualization Tab:

- *Icon*
  The symbol used to represent the Sink atom in the 2D window.

- 3D Icon
  The 3D icon that is used to represent the Sink atom in the 3D window.

6 THE CONTAINER ATOM



Picture 6-1: The Container Atom

The Container atom is created especially for storing or stacking other atoms, such as boxes or pallets. For the Container atom, a number of standard options, such as special 3D icons, have been designed for improved visualization. Moreover, the size of a product can automatically be adjusted to the size of the Container. In principle, the Container atom, like the Product atom, is placed in the model via a source (or via the Arrival list atom). Using an Assembler atom, products can then be put in the Container.

General Tab:

- *Atom name*
  The name given to the atom.

- *Container X size*
  The size of the container measured in the x direction (length).

- *Container Y size*
  The size of the container measured in the y direction (width).

- *Container Z size*
  The size of the container measured in the z direction (height).

- *Storage X size*
  Lists the size (in the x direction) required for storing an arriving product. Entering more space than actually required by the product results in empty spaces between the products.

- *Storage Y size*
  Lists the size (in the y direction) required for storing an arriving product. Entering more space than actually required by the product results in gaps between the individual products.

- *Storage Z size*
  Lists the size (in the z direction) required for storing an arriving product. Entering more space than actually required by the product (see Product atom) results in gaps between the products.

- *Fit to storage*
  Checking this option adjusts the size of a product such that it exactly matches the pre-defined storage size.

Warning!
There are 3 types of sizes: the size of the product (see Product atom), the size of the container and the storage size. The storage size allows (visual) adjusting of the packing method.

Example:
A pallet, Container with a Container size of 1x1x0.15, will stack a product sized 1x1x1 4 high if the Assembler places 4 products on a pallet.
But when Fit to Storage is selected with a storage size of 0.5x0.5x0.5, it will neatly place 4 boxes on one layer!

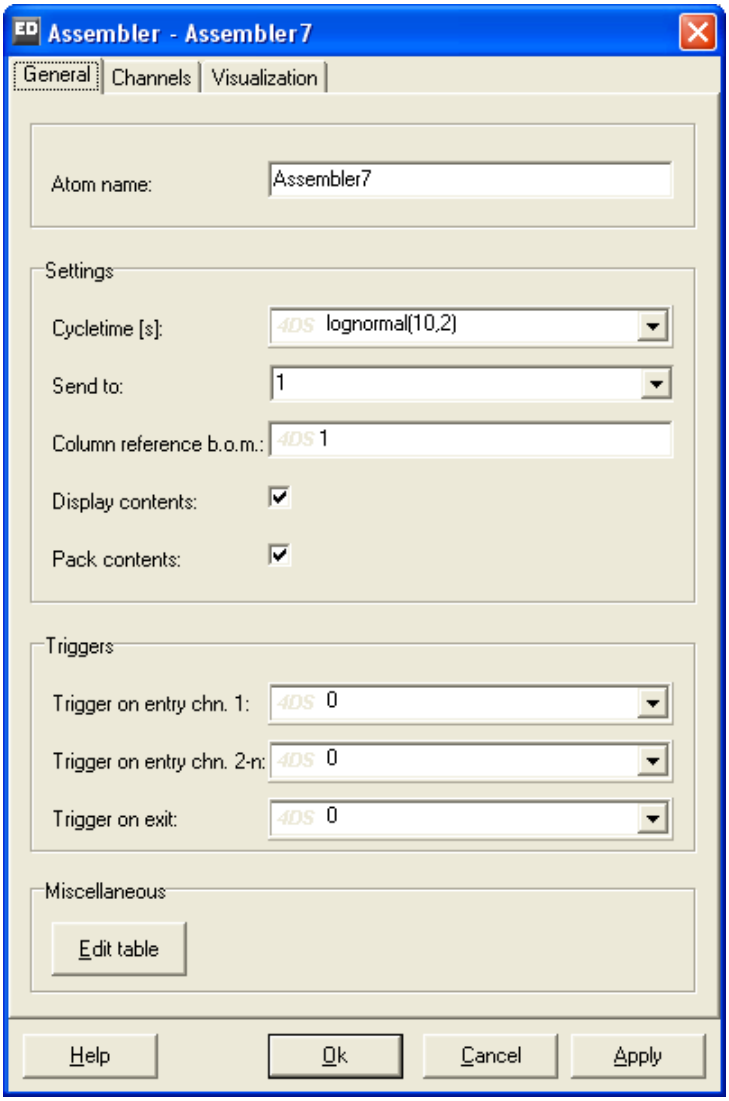
Visualization Tab:

- *Color*
  The color given to the atom.

- *2D Icon*
  The symbol used to represent the Container atom in the 2D window.

- *3D Icon*
  The symbol used to represent the Container atom in the 3D window.

7 THE ASSEMBLER ATOM



Picture 7-1: The Assembler atom

This atom merges atoms from several sources. Atoms placed into another atom can remain stored or be destroyed. Besides simulating real assembly work, this atom is also very useful for packing products in boxes or on pallets, and even for compiling orders.

The pallet, box or order always enters via the first input channel, while the products enter via the other channels. Depending on the settings, these products are destroyed or placed in the atom that has entered via the first input channel.

General tab:

- *Atom name*
  The name given to the atom.

- *Cycletime*
  The time needed to combine the products. This time is measured from the moment when all the atoms required have entered the Assembler, and refers to the whole assembly operation and not to each individual product! Clicking on the arrow opens a number of pre-defined 4DScript expressions.

- *Send to*
  Indicates to which exit channel the products are sent. For more details: see the 'Send to' explanation of the Source atom.

- *Column reference b.o.m.*
  Indicates what column from the b.o.m. will be used. The user can enter a figure, but also a 4DScript command that produces a figure. This field is evaluated when the first atom arrives via input channel 1 and defines which column will be used when the other atoms enter the Assembler.

  Bill of Material
  The Assembler atom also has a bill-of-material (b.o.m.) table, (visible by clicking the "edit Table" button). The b.o.m. lists, by input channel, how many atoms are required for assembling the end product.

  This b.o.m. has a number of rows and columns. The number of columns matches the number of input channels. Although the default setting is 1, the user can define the number of columns himself.

  This means that a separate column can be created for each product type that is assembled. When a product enters via the first input channel, the user defines which column in the b.o.m. is used for the rest of the arriving atoms.

- *Display contents*
  Checking this option displays the contents of the Assembler.

- *Pack contents*
  Checking this option, the atoms (read: products) are placed in the atom that entered via channel 1 (read: main product or Container). If this option is not checked, all atoms that did not enter via channel 1 will be destroyed.

- *Entry Trigger channel 1*
  The command entered in this field is performed when a product enters the Assembler via the first input channel.
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

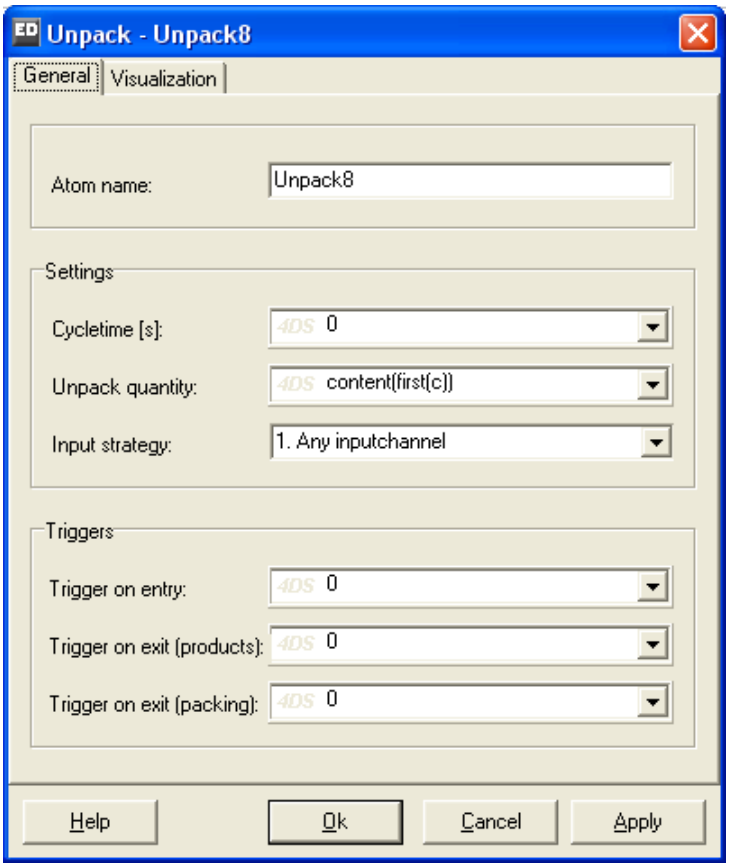
- *Entry Trigger channel 2..n*
  The command entered in this field is executed when a product enters the Assembler via one of the other input channels (i.e. not via channel 1).
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

- *Trigger on exit*
  The command in this field is executed when a product leaves the Assembler.
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

Visualization tab:

- *Icon*
  The symbol that represents the Assembler atom in the 2D window.

8 THE UNPACK ATOM



Picture 8-1: The Unpack atom

The Unpack atom is used for removing products from a Container atom. After unpacking, the Container atom is sent via the second output channel and the products via the first output channel.

General tab:

- *Atom name*
  The name given to the atom.

- *Cycletime*
  The time needed to unpack the container. This time is measured from the moment the container enters the atom and refers to the 'unloading time' per container (and *not* to that of each individual product). Clicking on the arrow will open a number of pre-defined 4DScript expressions.
  For more details: see the 'Cycletime' explanation of the Server atom.

- *Unpack quantity*
  Defines the number of atoms that will be unpacked. When this number is

reached, the container is sent through. The default setting `content(first(c))` ensures that all products are removed from the container. The 3 other pre-defined options are:
duniform(1,10)
label([?],first(c))
if(=(?,?),?,?)

- *Input strategy*
  This field may be used for indicating which input channel should be used.
  For more details: see the 'Input strategy' of the Server atom.

- *Trigger on entry*
  The command entered in this field is performed when a container enters the Unpack atom.
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

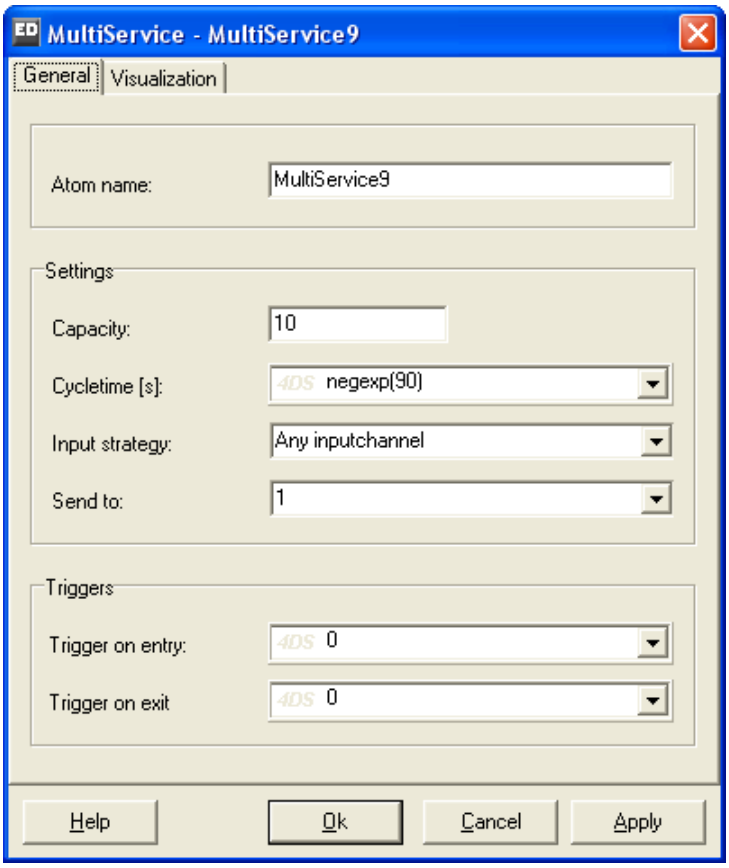
- *Exit trigger products*
  The command entered in this field is performed when a product leaves the Unpack atom.
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

- *Exit trigger packaging*
  The command entered in this field is performed when a container leaves the Unpack atom.
  For more details: see the 'Trigger on Exit' explanation of the Source atom.

Visualization tab:

- *Icon*
  The symbol used to represent the Unpack atom in the 2D window.

9 THE MULTISERVICE ATOM



Picture 9-1: The MultiService atom

The MultiService atom has the same basic functions as several individual Server atoms. Where the Server atom can process only one product at a time, the MultiService atom allows the simultaneous processing of several products. The settings options are not as elaborate as those of the normal server, but all basic functions are available.

General tab:

- *Atom name*
  The name given to the atom.
- *Capacity*
  Lists the number of products that can be processed simultaneously.

For the options below, we refer you to the description of the Server atom:

- *Cycletime*
- *Input strategy*
- *Send to*
- *Trigger on Entry*

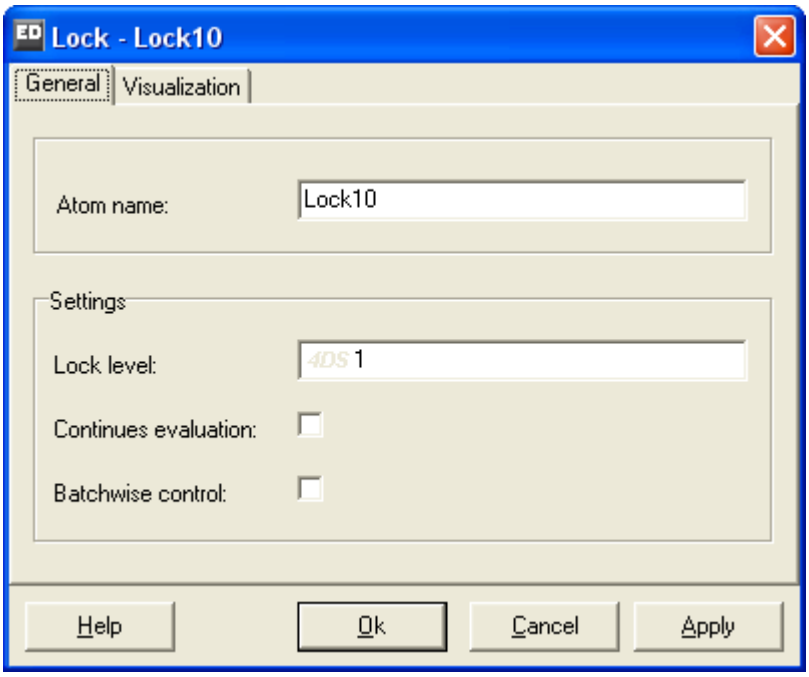
- *Trigger on Exit*

Visualization tab:

- *Icon*
  The symbol used to represent the MultiService atom in the 2D window.

- *2D display*
  The way in which the products are displayed in the atom in 2D:
  1: *Standard display*
     The products in the MultiService atom are not visible, but a text display indicates how many products are present.
  2: *Move left right*
     The products are visible and move from the left to the right.
  3: *Move right left*
     The products are visible and move from the right to the left.
  4: *Level vertical*
     Instead of the products, a colored area is visible which increases in height when more products enter the MultiService atom.
  5: *Level horizontal*
     Instead of the products, a colored area is visible which scrolls to the right when more products enter the MultiService atom.
  6: *Line up content*
     The products are visible and displayed one underneath the other.

10 THE LOCK ATOM



Picture 10-1: The Lock atom

When the Lock atom alone is used, it functions as a gateway that lets only the number of products indicated in the Lock level through.

When the Lock atom is used in combination with the Unlock atom, it serves to control the maximum number of products in a certain part of the model. The amount of work in process in the model delimitated by Lock and Unlock can then be no more than the so-called Lock level.

Due to the connection between the two, the Unlock atom is described below.

General tab:

- *Atom name*
  The name given to the atom.

- *Lock level*
  When the number of atoms between Lock and Unlock reaches the specific level entered in this field, the input of the Lock closes, thereby blocking access. When Unlock is not added, the input closes without reopening.

- *Continues evaluation*
  When this option is checked, the field Lock level is evaluated every time when a product enters the Lock atom, instead of being evaluated only after a model reset.

Only when the field Lock level contains a 4DScript command (and not only a number) may it be necessary to activate this option.

- *Batchwise control*
  When the Lock level is reached, access via the Lock is blocked and reopened only after all products in the channel have disappeared via the Unlock atom. This creates a form of batch processing with the Lock level as batch size.

Visualization tab:

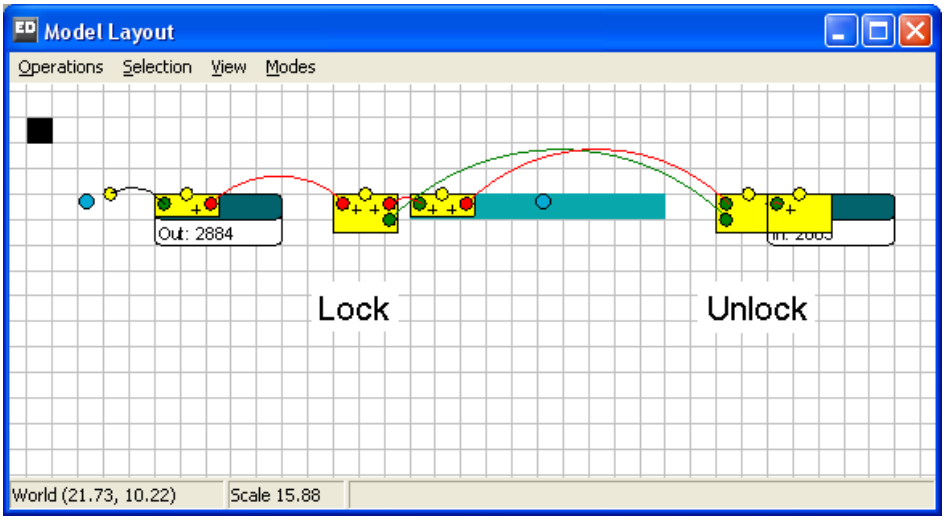
- *Icon*
  The symbol used to represent the Lock atom in the 2D window.

11 THE UNLOCK ATOM

This atom has no settings and is used only in combination with the Lock atom. Each time when a product leaves the Unlock atom, the Lock atom allows a new product to enter the system (unless the “Batchwise control” option is activated); in that case all products must first have passed through the Unlock atom).

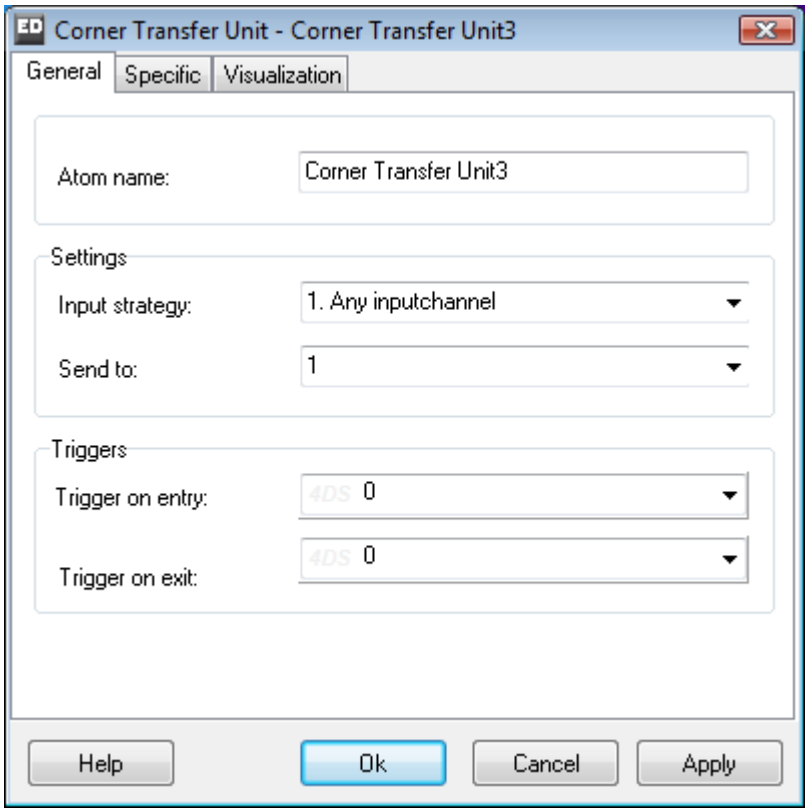
In the example below, the Lock and Unlock atom control the quantity of products on the conveyor. See how both atoms are placed in the product route `in a normal way' and how they are connected via their second output or input channel respectively.

The Unlock atom has no window that needs filling in. A Lock atom can be connected with several Unlock atoms.



Picture 11-1: Model view with a lock and an Unlock atom

12 THE CORNER TRANSFER UNIT ATOM



Picture 12-1: The Corner Transfer Unit atom

A Corner Transfer Unit is an intersection between conveyors of any kind. Unlike the Turntable Unit Atom it does not rotate incoming goods into transport direction before forwarding them to an outgoing conveyor. Products keep their orientation when turning left, right or moving straight ahead. The number of possible changes of transport directions is limited. Products can turn left around a corner of -90 degrees, turn right around a corner of 90 degrees or go straight ahead. Often a turn around a corner is not possible without a transfer time to slow down speed, change the drive mechanism and speed up into the new direction. There is a minimum of one input and a maximum of three inputs. The number of outputs is at least one but has a maximum of three. To control more than one input, an input strategy is required. More than one outputs requires a strategy too.

Note: This atom is part of ED Logistics. ED Logistics edition also has a Turntable Unit atom that rotates incoming goods into transport direction before forwarding them to an outgoing conveyor.

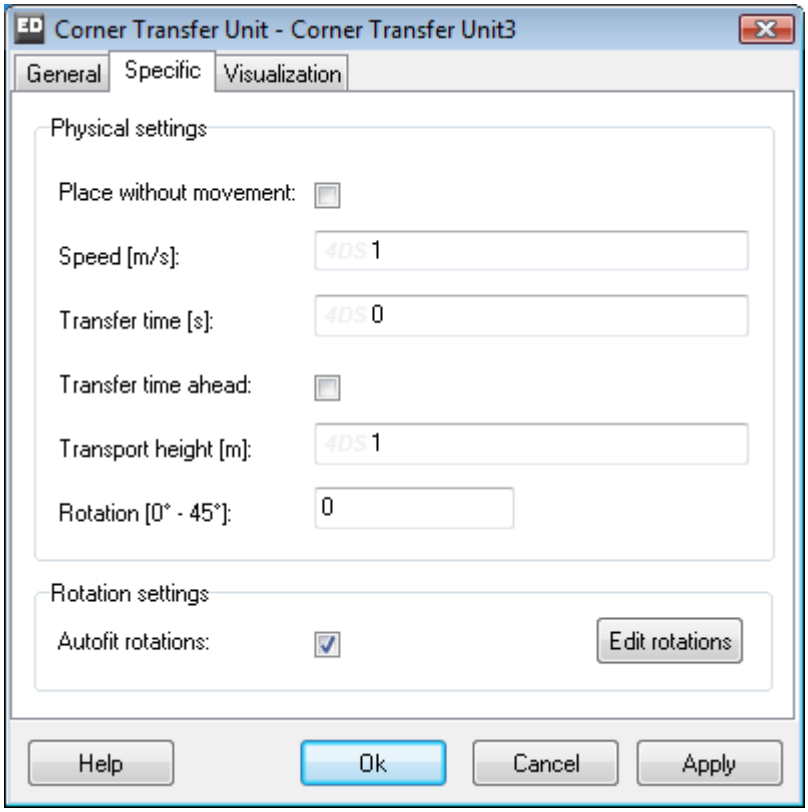
- *Atom name*
  The name given to the atom.

- *Input strategy*
  A rule that determines how Product atoms may be let into the inbound atom of the Corner Transfer Unit. There are a number of rules predefined (see also Input strategy), but you can also create your own rule via a 4DScript expression.

- *Send to*
  A rule that determines to which output channel a Product atom needs to be sent. There are a number of rules predefined (see also Send to), but you can also create your own rule via a 4DScript expression.

- *Trigger on entry*
  A rule that determines what kind of action needs to be executed when a Product atom enters the inbound atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

- *Trigger on exit*
  A rule that determines what kind of action needs to be executed when a Product atom exits the inbound atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.



Picture 12-2: The specific parameters

- *Place without movement*
  If this box is checked the Corner Transfer Unit will create no input movement.

Instead incoming products will be placed immediately in the center of the atom. This functionality may be helpful if the Corner Transfer Units receives goods from an atom of the transportation section like the Portal Crane atom or a Robot atom.

- *Speed [m/s]*
  The speed for the input movement of the products to the center of the Corner Transfer Unit. The speed for the output movement is controlled by the atom connected to the output channels.

- *Transfer time [s]*
  Time to slow down speed, change the drive mechanism and speed up into the new direction when turning around a corner.

- *Transfer time ahead*
  If this box is checked the transfer time is also necessary for straight on going products.

- *Transport height [m]*
  The length of the leg supports of the conveyor (in meters).

- *Rotation [+/- 45°]*
  At default the conveyor is positioned from left to right in your screen. However you can also rotate the conveyor to display the flow of atoms over the conveyor matches reality.

Note: Due to the layout of the Corner Transfer Unit atom rotation is limited +/- 45° because any other rotation can be modeled by turning left or right.

- *Autofit rotations*
  If this box is checked the Corner Transfer Unit tries to detect the rotation of atoms connected to input and output channels OnReset. The Corner Transfer Unit needs to know the rotation of connected atoms to calculate the rotation of passing products.

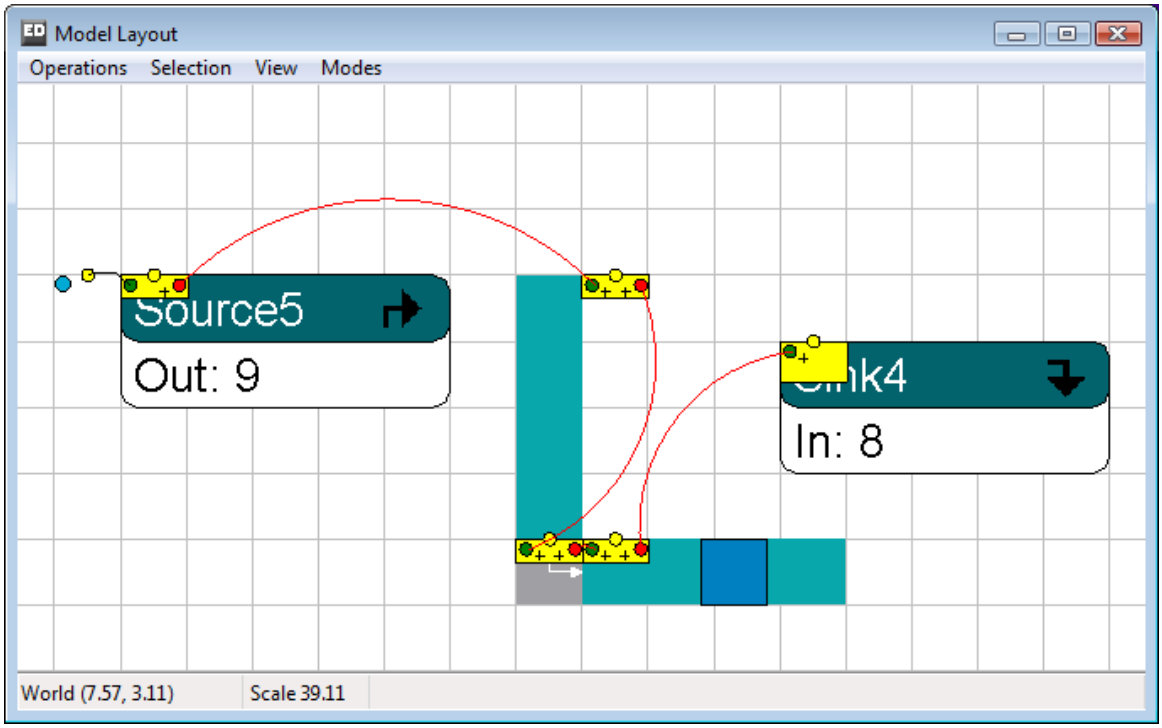
Note: However not all kinds of atoms may be detected by the auto fit functionality. Therefore it can be switched of in order to edit the table of rotations by hand.

- *Edit rotations*
  Click this button to edit the rotation around self of atoms connected to the Corner Transfer Unit. Edit the motion flags for the input and output motion directions. Edit 1 for positive x/y directions and -1 for negative x/y directions.

Note: The number of rows is updated OnReset according to the number of input and output channels. Channels not connected to an atom cause an error message to prevent user from unexpected behavior later on.

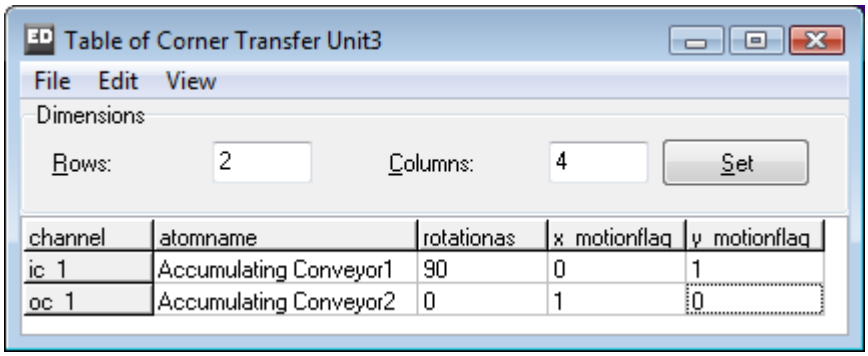
Note: Set only 1 motion flag for each row, either an motion along x-axis or along y-axis.

Example: Connect a Source with an 90° rotated Accumulating Conveyor. The conveyor leads into a Corner Transfer Unit atom. The Products are sent on to another Accumulating Conveyor with a rotation of 0°, before they are destroyed in a Sink.



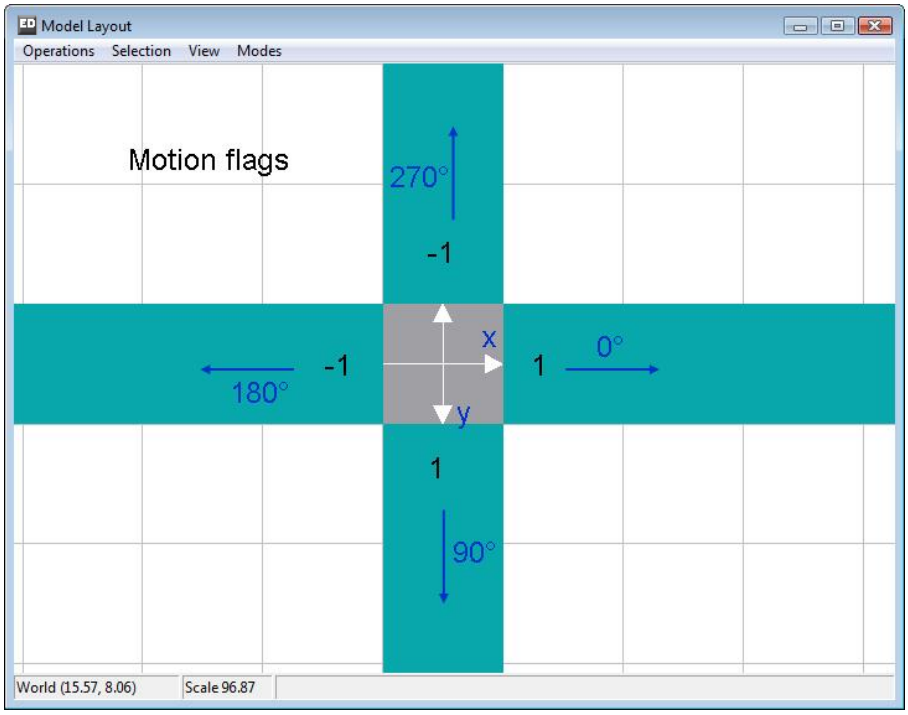
Picture 12-3: 2D Model Layout example

Set x-motion flag to 0 and y-motion flag to 1 for Products coming from an input (with rotation 90° around self) to move in. Set x-motion flag to 1 and y-motion flag to 0° for Products going to an output (with rotation 0° around self) to move out.



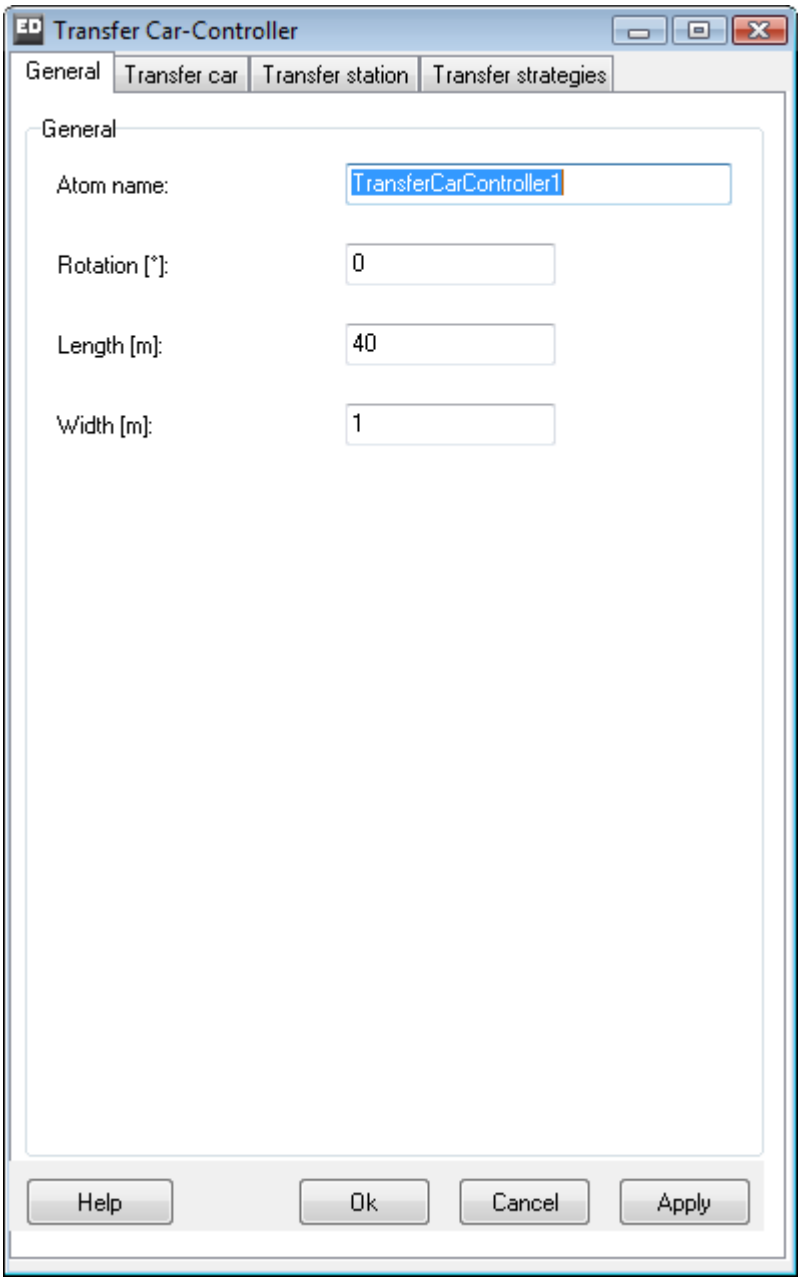
Picture 12-4: The table of rotations settings

If you select the 'Autofit rotations' Checkbox in the GUI of the Corner Transfer Unit atom, the table is filled automatically. The result of the edits is visible in the 2D Model Layout. White arrows show the calculated transport directions.



Picture 12-5: Motion flags

13 THE TRANSFER CAR ATOM



Picture 13-1: The Transfer Car atom

A Transfer Car system is a rail bound single or double track vehicle for fast and reasonable linear transport and/or to combine multiple input and output locations of material flow. Especially for longer distances Transfer Cars are used instead of conveyors for example to connect the inbounds of several warehouse aisles. The Transfer Car system consists of the Car on rails, the connected Stations and a Controller to coordinate the behavior of the system. The Car itself can have one or more storage units to carry more than one Product at the same time for example to swap Products at a particular

station. Products have to be conveyed from the input station to the storage units and after the linear transport movement they have to be conveyed to the output station as well. Sometimes there are altitude differences between input and output station. In this case the Car has to compensate the vertical distance. Each transportation direction has its own behavior regarding acceleration, maximum speed and deceleration. The Stations are located on the left and right side along the rails. A Station can consist of one or more ports. Each port can be separately served by the Car. There are 3 types of ports. Pick-ports are intended to absorb incoming Products. Place-ports are designated to deliver Products to the car. Pick-and-Place-ports can do both. The Controller coordinates interaction between loading, transportation and unloading taking into account all predefined strategy.

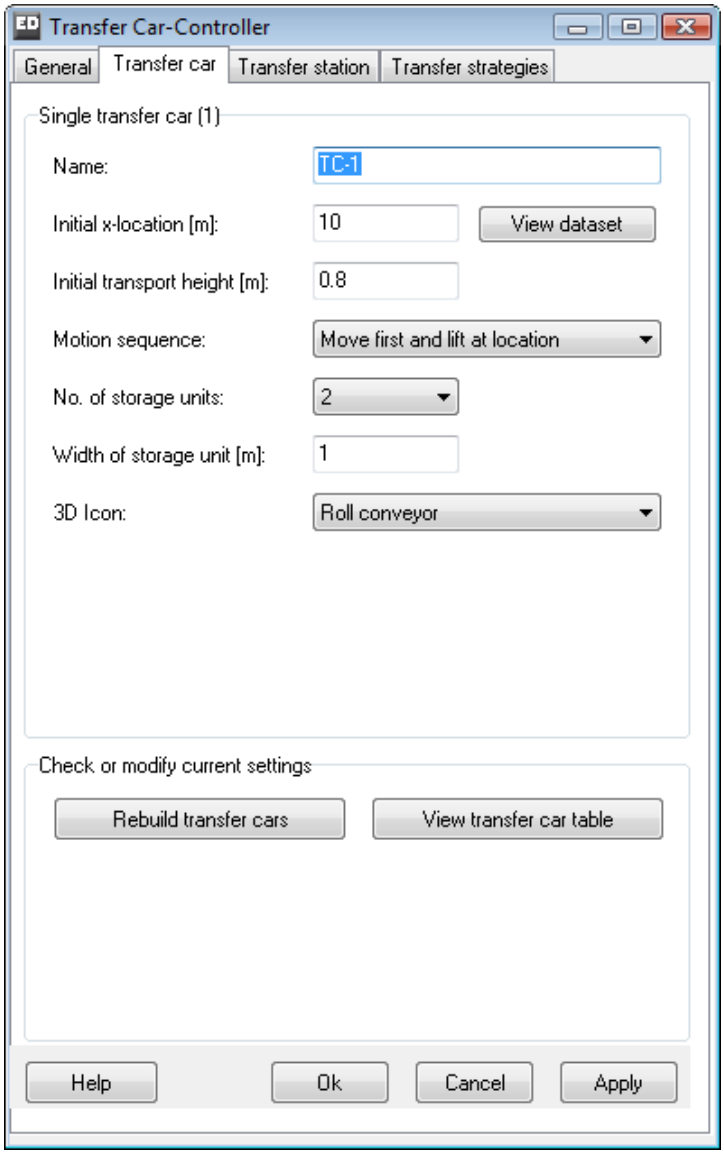
Important: The Transfer Car system is made of several sub atoms not intended to be dragged into the model. Drag the TransferCarController atom into your model, to open the GUI with all the edit fields to build or rebuild a complete Transfer Car system.

- *Atom name*
  The name given to the atom.

- *Rotation [°]*
  At default the Transfer Car system is positioned from left to right in your screen. However you can also rotate the atom to display the flow of atoms over the Transfer Car system matches reality.

- *Length [m]*
  The length of the rail systems (x-axis) to connect all the Stations within the range of the Transfer Car.

- *Width [m]*
  The width of the rail system also defines the y-dimension of the Transfer Car.



Picture 13-2: The Transfer Car parameters

- *Name*
  The name of the Transfer Car.

- *Initial x-location [m]*
  The initial position of the car along the rails on start of simulation.

- *Initial transport height [m]*
  The initial vertical transport height of the car on start of simulation.

- *Motion sequence*
  A rule that determines how the Transfer Car atom coordinates the movements of lifting and driving. There are 2 strategies available: 2D:
  1: Move left right
  The products are visible and move from the left to the right.

2: *Move first and lift at location*
Move and lift together.

- *No. of storage units*
  A Transfer Car can have up to three Storage units in order to move several Products at the same time.

Note: The strategies to control a Transfer Car system are more complex if the Transfer Car has more than one Storage unit, as you have to control the interaction between the several storage units.

- *Width of storage unit [m]*
  The x-dimension of every Storage unit. Defines together with the width [m] of the rail system and the No. of storage units the x-dimension of the Transfer Car.

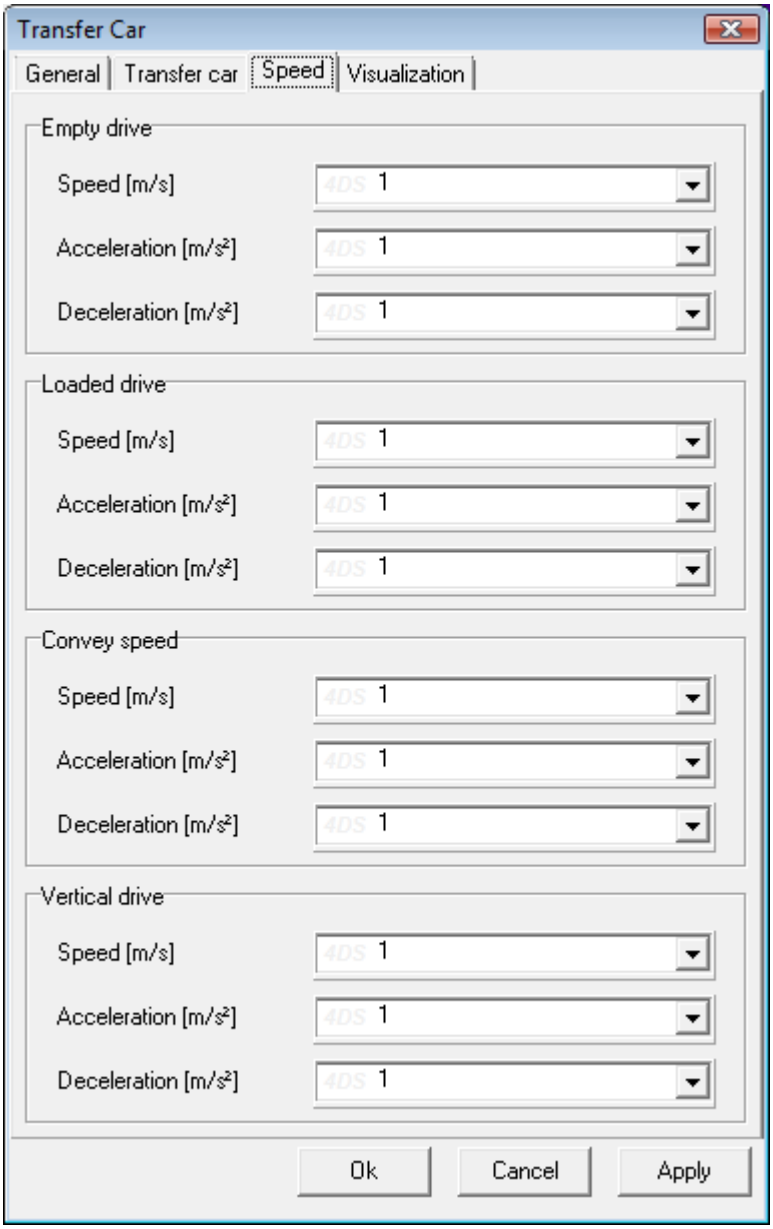
- *3D Icon*
  A 3D icon that can be used to visualize the Transfer Car atom in 3D. Change this icon to switch between simple box, roll conveyor, belt conveyor or chain conveyor.

- *Rebuild transfer cars*
  Click this button to rebuild the Transfer Car and all storage units with the parameters of the edit fields.

- *View transfer car table*
  Click on this button to view the table with the detailed parameters of all Transfer Cars.

Note: Because of the high number of Transfer Car parameters some of them are defined on a separate Transfer Car GUI.

- *View dataset*
  Click on this button to edit detailed parameters of the Transfer Car on a separate Transfer Car GUI.



Picture 13-3: The detailed Transfer Car dataset

The following additional parameters can be changed on page General of the detailed Transfer Car GUI:

- *Description*
  A describing text.

- *Trigger on entry*
  A rule that determines what kind of action needs to be executed when a Product atom enters the Transfer Car atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

- *Trigger on exit*
  A rule that determines what kind of action needs to be executed when a Product

atom exits the Transfer Car atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

The following additional parameters can be changed on page Transfer Car of the detailed Transfer Car GUI:

- *ID*
  The index of the Car.

The following additional parameters can be changed on page Speed of the detailed Transfer Car GUI:

- *Empty Drive*
  Speed [m/s], Acceleration [m/s²] and Deceleration [m/s²] of empty Transfer Car along rails direction.

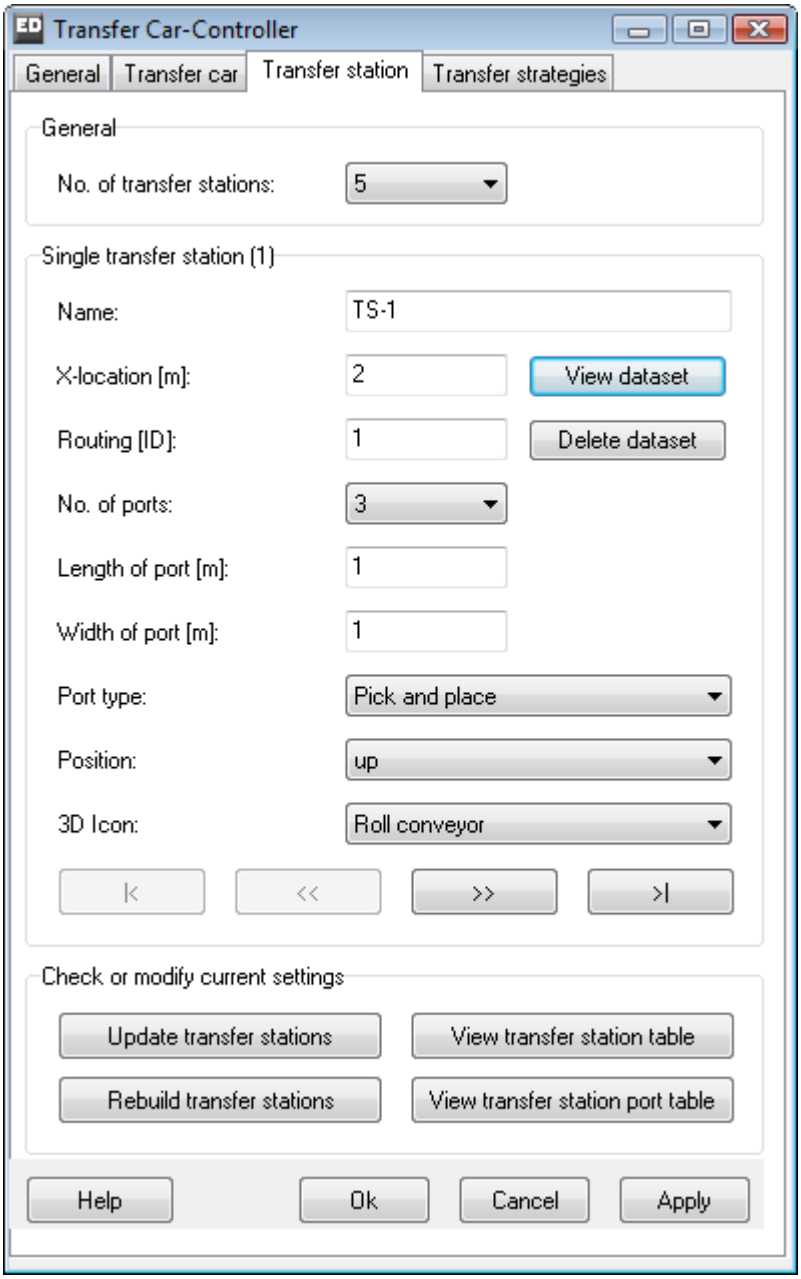
- *Loaded Drive*
  Speed [m/s], Acceleration [m/s²] and Deceleration [m/s²] of loaded Transfer Car along rails direction.

- *Convey Drive*
  Speed [m/s], Acceleration [m/s²] and Deceleration [m/s²] of Transfer Car when loading or unloading goods from or to Stations.

- *Vertical Drive*
  Speed [m/s], Acceleration [m/s²] and Deceleration [m/s²] of Transfer Car when balancing altitude differences between input and output station.

The following additional parameters can be changed on page Visualization of the detailed Transfer Car GUI:

- *Color*
  The color of the Transfer Car.



Picture 13-4: The Transfer Station parameters

- *No. of transfer stations*
  The Transfer Car system can have up to 20 Stations.

Note: Use the arrow buttons to switch between the parameters of every individual Station.

- *Name*
  The name of the Transfer Station.

- *X-location [m]*
  The position of the Station along the rail track.

- *Routing [ID]*
  The index of the Station that is used to define the sequence of transportation inside the Routing plan on page Transfer strategies.

- *No. of ports*
  Each Station can have up to 6 Ports with an individual Port ID from 1 to 6.

- *Length of port [m]*
  The size on the x-axis.

- *Width of port [m]*
  The size on the y-axis.

- *Port type*
  There are 3 types of ports. Pick-Ports are intended to absorb incoming Products. Place-Ports are designated to deliver Products to the car. Pick-and-Place-Ports can do both.

- *Position*
  Select up to place a Station on the left side of the rail track. Select down to place a Station on the right side of the rail track.

- *3D Icon*
  A 3D icon that can be used to visualize the Station atom in 3D. Change this icon to switch between simple box, roll conveyor, belt conveyor or chain conveyor.

- *Delete dataset*
  Click on this button to remove a whole Station with all Ports from the Transfer Car system.

- *Rebuild transfer stations*
  Click this button to update parameters of the Transfer Stations with all Ports according to the parameters of the edit fields.

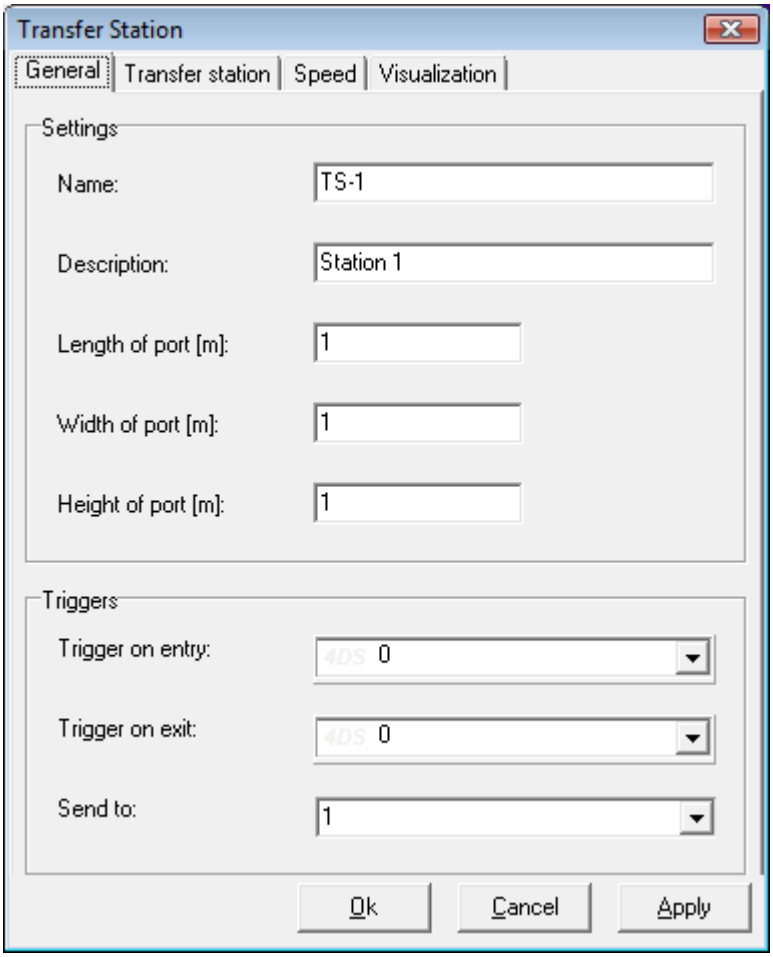
- *Rebuild transfer stations*
  Click this button to rebuild the Transfer Stations with all Ports according to the parameters of the edit fields.

- *View transfer station table*
  Click on this button to view the table with the detailed parameters of all Transfer Stations.

- *View transfer station port table*
  Click on this button to view the table with the detailed parameters of all Transfer Stations and the assigned Ports.

Note: Because of the high number of Station parameters some of them are defined on a separate Transfer Station GUI.

- *View dataset*
  Click on this button to edit detailed parameters of the Station on a separate Transfer Station GUI.



Picture 13-5: The detailed Transfer Station dataset

The following additional parameters can be changed on page General of the detailed Transfer Station GUI:

- *Description*
  A describing text.

- *Height of port [m]*
  The size on the z-axis.

- *Trigger on entry*
  A rule that determines what kind of action needs to be executed when a Product atom enters the Station atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

- *Trigger on exit*
  A rule that determines what kind of action needs to be executed when a Product atom exits the Station atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

- *Send to*
  A rule that determines to which output channel a Product atom needs to be sent. There are a number of rules predefined (see also Send to), but you can also create your own rule via a 4DScript expression.

The following additional parameters can be changed on page Transfer station of the detailed Transfer Station GUI:

- *Station [ID]*
  An information about the index of the Station (read only).

- *Port [ID]*
  An information about the index of the Port (read only).

- *Direction at entry*
  Determines the offset location and the direction of the input movement for incoming Products. There are 7 strategies available:
  1: Product enters from the top.
  2: Product enters from the bottom.
  3: Product enters from the left.
  4: Product enters from the right.
  5: Make entry direction dependent on income channel:
  6: Choose between several entry channels:
  7: By user: Enter your own 4DScript expression:

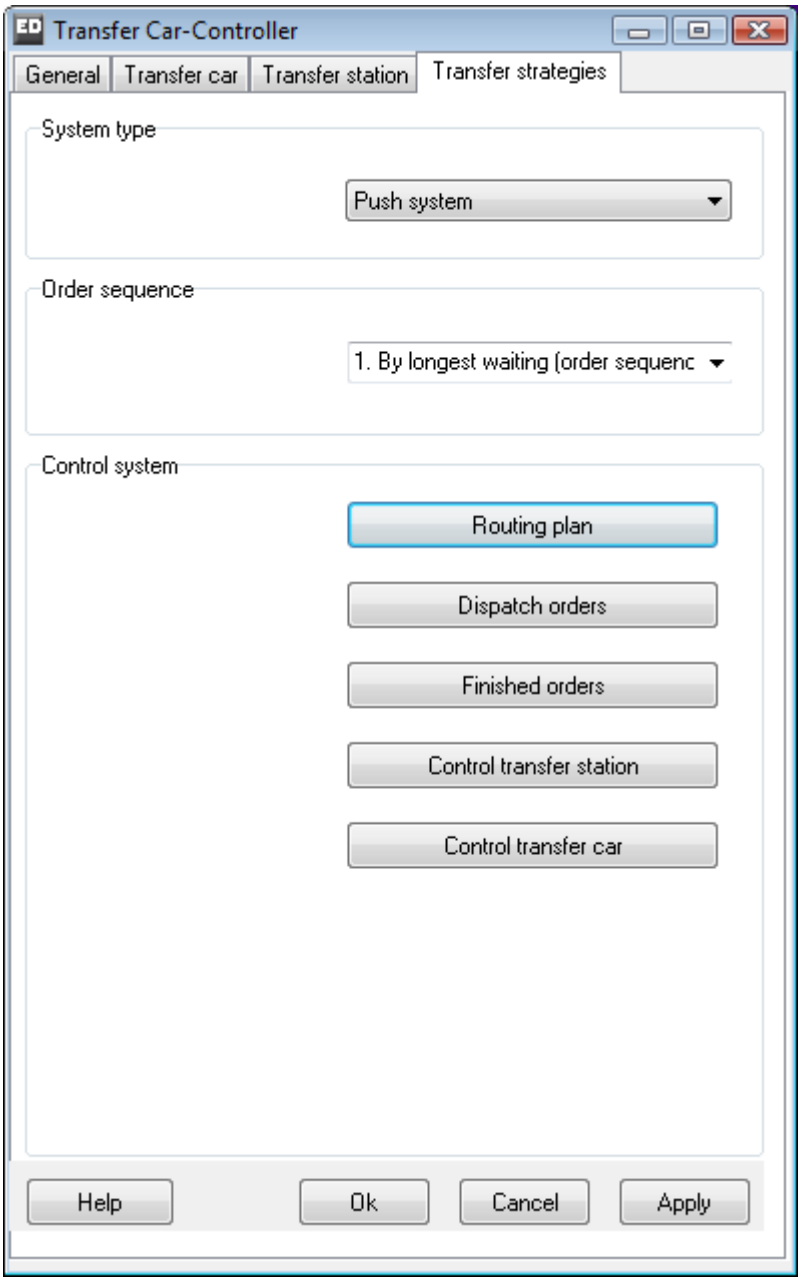
- *Direction at exit*
  Determines the direction of the output movement for outgoing Products. There are 7 strategies available:
  1: Product exits at the top.
  2: Product exits at the bottom.
  3: Product exits at the left.
  4: Product exits at the right.
  5: Make exit direction dependent on income channel:
  6: Choose between several exit channels:
  7: By user: Enter your own 4DScript expression:

The following additional parameters can be changed on page Speed of the detailed Transfer Station GUI:

- *Convey*
  Speed [m/s], Acceleration [m/s²] and Deceleration [m/s²] of Transfer Stations when loading or unloading Products from or to external connections.

The following additional parameters can be changed on page Visualization of the detailed Transfer Station GUI:

- *Color*
  The color of the Transfer Station.

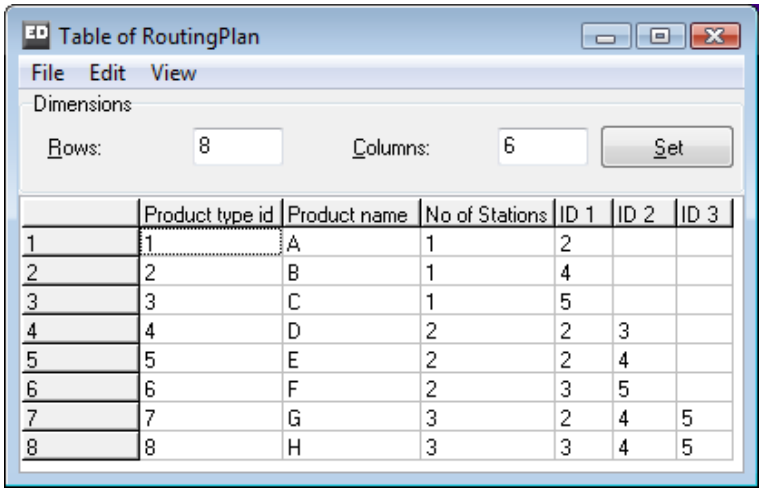


Picture 13-6: The Transfer strategies

- *System type*
  The Transfer Car system can be controlled as Push system or Pull system.

  If you select Push system, the Controller evaluates incoming Product with a Label 'ProductTypeID' to identify the Station (row) inside the Routing plan

table. Be sure that the Routing plan has all information about the sequence of Stations for each product type.



	Product type id	Product name	No of Stations	ID 1	ID 2	ID 3
1	1	A	1	2		
2	2	B	1	4		
3	3	C	1	5		
4	4	D	2	2	3	
5	5	E	2	2	4	
6	6	F	2	3	5	
7	7	G	3	2	4	5
8	8	H	3	3	4	5

Picture 13-7: The Routing Plan (push mode)

If you select Pull system, the Controller is awaiting calls from the Place-Stations, before he is searching for Products ready for transportation. For more information see the 'Transfer Car Pull' Model in the 'Examples' directory of your Enterprise Dynamics program directory.

- *Order sequence*
  Define how the sequence of orders (transportation tasks) is executed by the Transfer Car. Choose a predefined strategy or define your own strategy.

- *Routing plan*
  Click this button to edit the product-specific sequence of Stations.

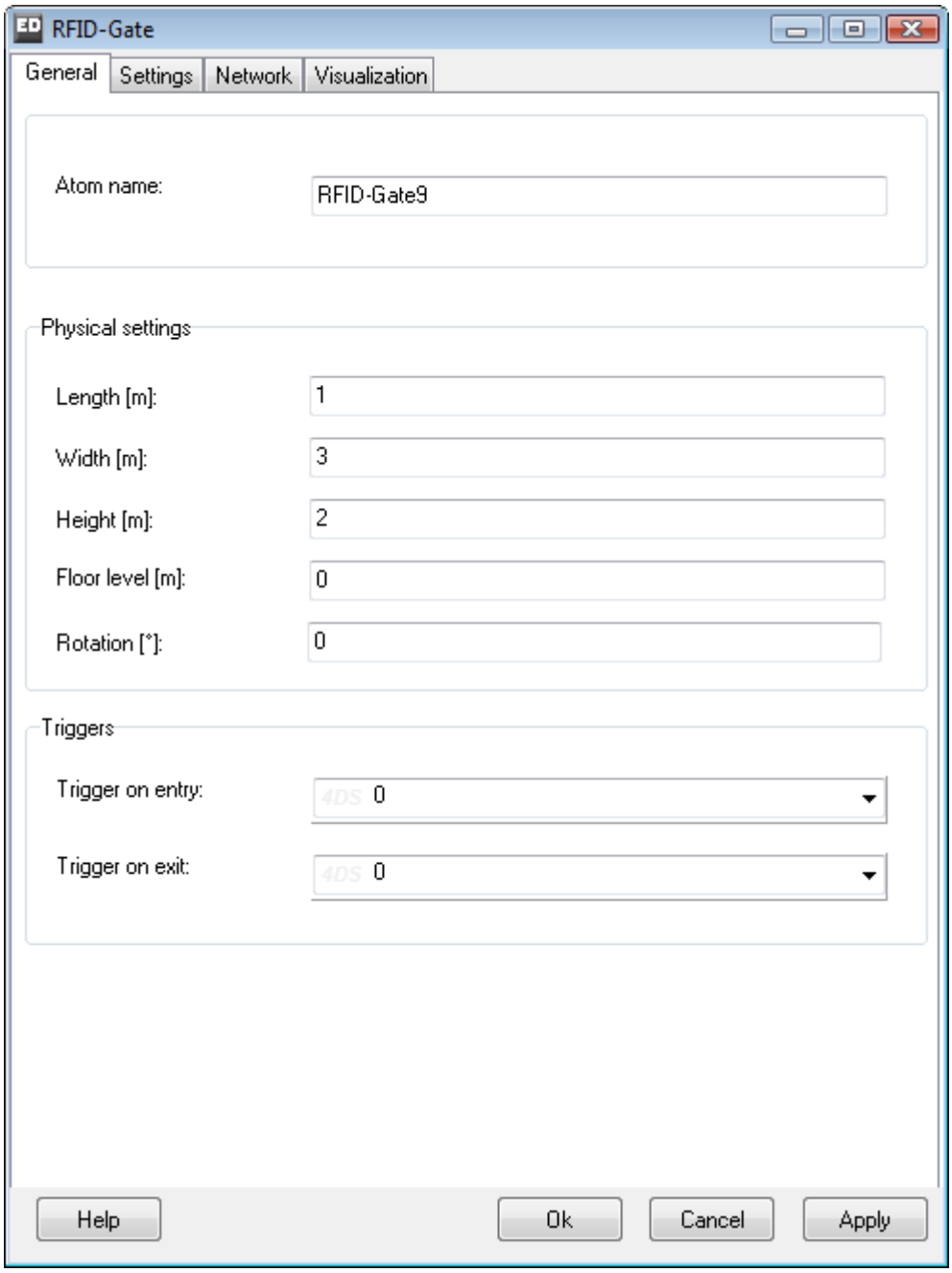
- *Dispatch orders*
  Click this button to view all open orders.

- *Finished orders*
  Click this button to view all finished orders.

- *Control transfer station*
  Click this button to view the table of the Transfer Station Controller.

- *Control transfer car*
  Click this button to view the table of the Transfer Car Controller.

14 THE RFID-GATE ATOM



Picture 14-1: The RFID-Gate atom

This atom can be connected to a network (Network Nodes) and work as a station, where Tags can be manipulated. Tags are treated as Labels. The RFID-Gate atom consists of 3 different working modes; it can read, write and read and write Labels. All 3 modes can be used simultaneously. The user can specify several Labels and operation in the GUI table. When an Advanced Transporter atom passes the RFID-Gate, the values of the Labels are written to the result table. The table can be found in the GUI.

Important: Before an RFID-Gate can be connected to a Network, you have to chose whether the path through the Gate is bidirectional or unidirectional.

- *Atom Name*
  The name of the atom.

- *Length [m]*
  The length of the atom in meters.

- *Width [m]*
  The width of the atom in meters.

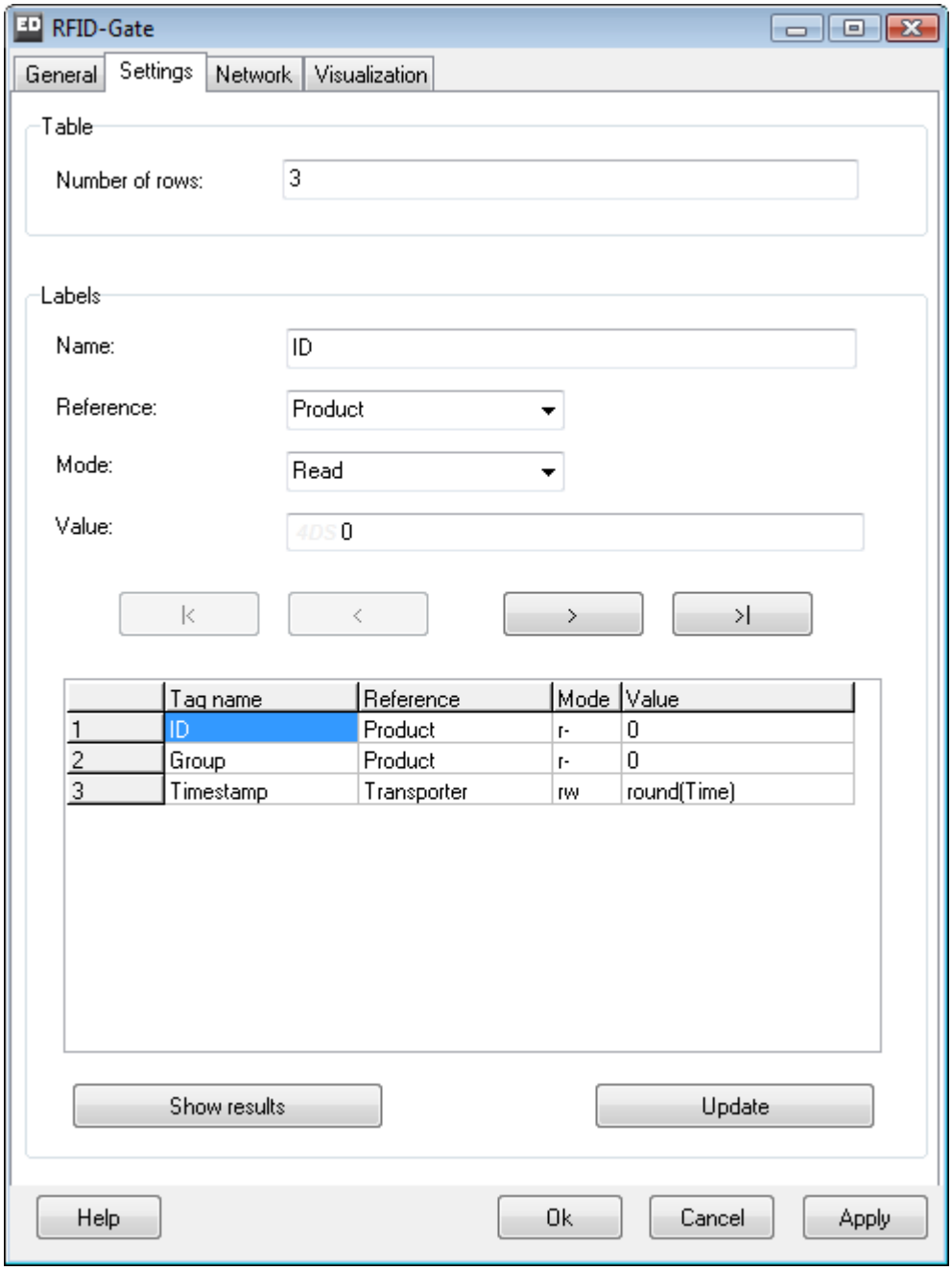
- *Height [m]*
  The height of the atom in meters.

- *Floor level [m]*
  Helpful if the level of transport is not equal to the floor level.

- *Rotation [°]*
  The atom can be rotated into transport direction to match the flow of the connected Network Nodes.

- *Trigger on entry*
  Determines what kind of action needs to be executed when a Advanced Transporter atom enters the RFID-Gate. There are a number of rules predefined (see also Trigger on exit), but you can also create your own rule via a 4DScript expression.

- *Trigger on exit*
  Determines what kind of action needs to be executed when a Advanced Transporter atom exits the RFID-Gate. There are a number of rules predefined (see also Trigger on entry), but you can also create your own rule via a 4DScript expression.



Picture 14-2: The RFID-Gate label settings

- *Number of rows*
  Define the number of operations that needs to be performed in the RFID-Gate. Each operation has an individual row to define the Label of the specific operation.

- *Name*
  The name of the Label (Tag) that you want to define.

- *Reference*
  The operation can refer to the Advanced Transporter or to the transported Product. If the Advanced Transporter carries more than one Product, the desired

operation is performed on all Products inside the Transporter. If the reference is the Transporter, the Label will be read from or written to the Transporter. If the reference is the Product, the Label will be read from or written to all Products carried by the Transporter.

- *Mode*
  The RFID-Gate can work in different modes. There are 3 modes available:
  1: Read mode only,
  2: Write mode only,
  3: Read and write mode
  Means that the Labels are read out and then rewritten.

- *Value*
  If the Label has to be written, you can specify the desired value of the Label here. Expressions are allowed.

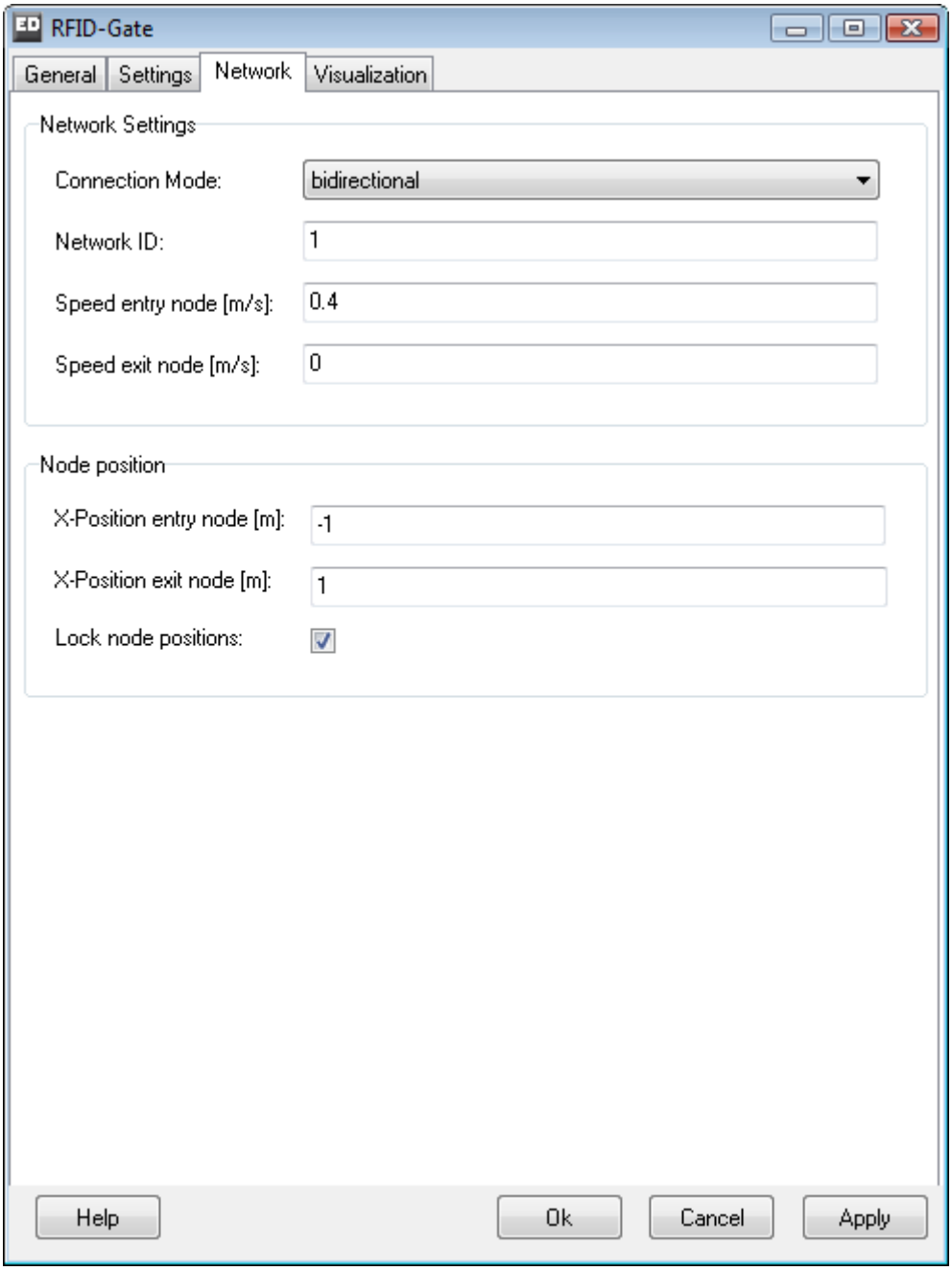
Important: The content of the above field should appear immediately in the GUI table. If this is not the case, choose the 'Reference' or the 'Mode' again.

- *'Forward' and 'Backward' Buttons*
  Click this buttons to switch between the table rows. The marker will go up or down according to the selected action.

- *'Show Results' Button*
  If a Label is read out, its content is written to the result table. Click this button to open the table.

'Update' Button

When the user adds, deletes or edits an operation, this button has to be clicked to apply the changes to the results table. This button cleans the results table from old result values (previous runs) as well.



Picture 14-3: The RFID-Gate network settings

- *Connection Mode*
  Choose whether the RFID-Gate can be passed in two directions (bidirectional) or one way only (unidirectional).

Note: The RFID-Gate will perform its operations if the Advanced Transporter enters through the entry node. No operation will be performed if the Advanced Transporter enters the RFID-Gate from the exit node.

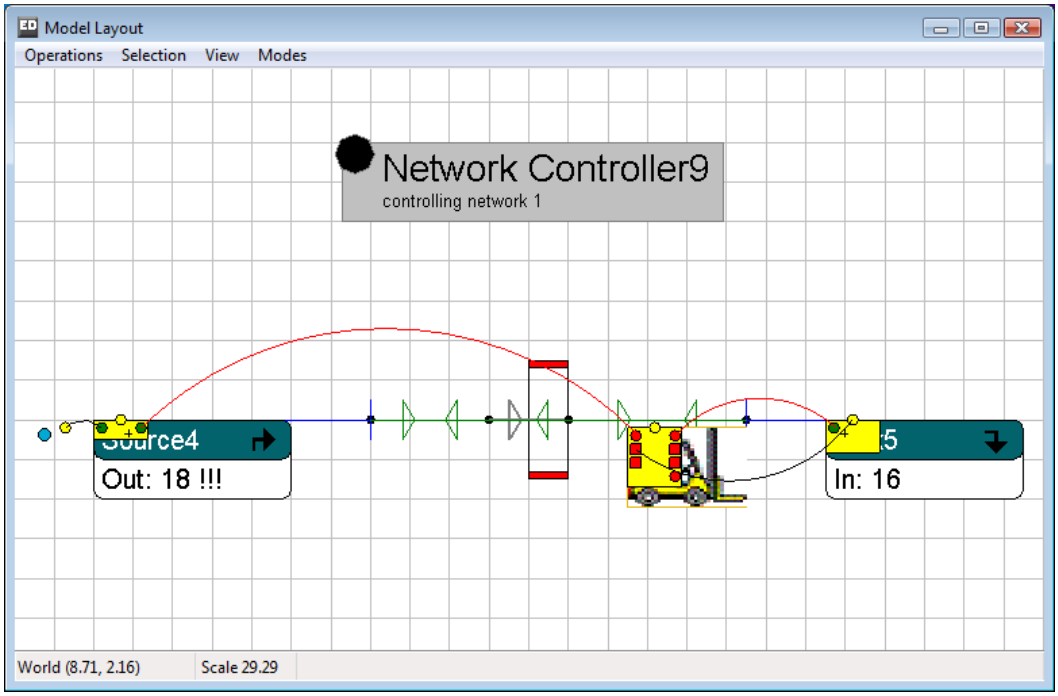
- *Network ID*
  This is the ID of the Network where the RFID-Gate is connected to.

- *Speed entry node [m/s]*
  This is the speed limit for the Advanced Transporter if it enters through the entry node.

- *Speed exit node [m/s]*
  This is the speed limit for the Advanced Transporter if it enters through the exit node.

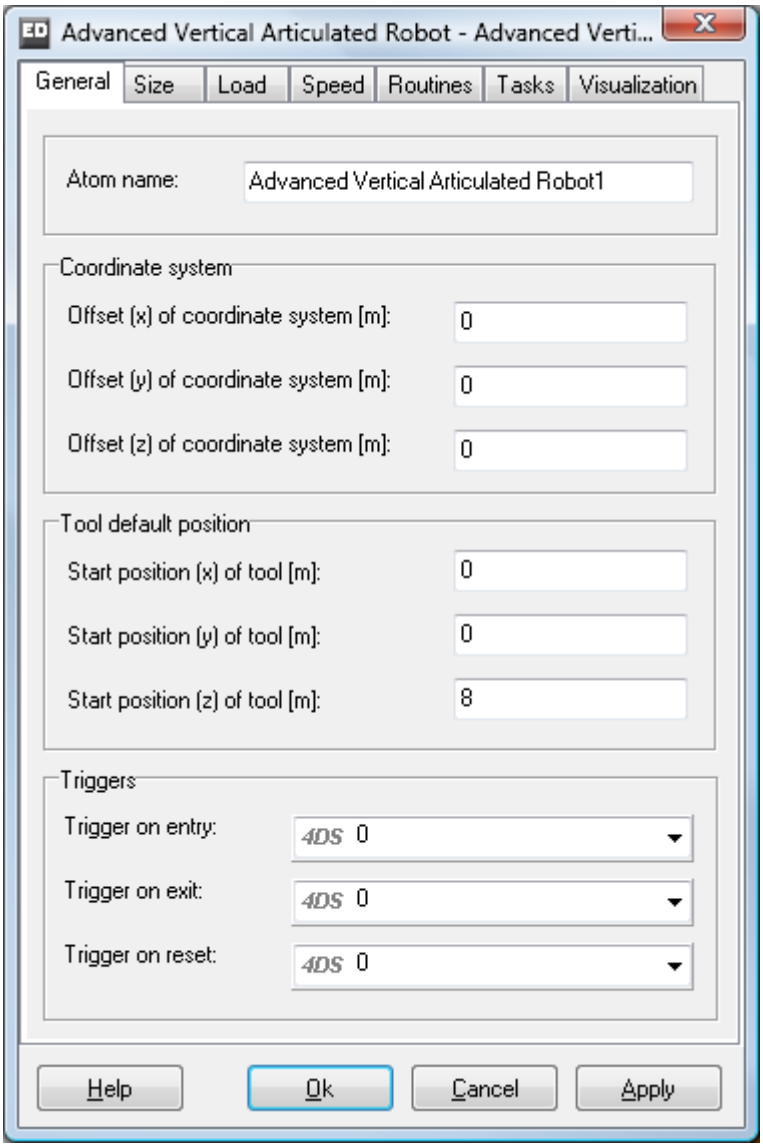
- *X-Position entry node [m/s]*
  This is the position of the entry node on the x-axis. This is measured relative to the RFID-Gate.

- *X-Position exit node [m/s]*
  This is the position of the exit node on the x-axis. This is measured relative to the RFID-Gate.



Picture 14-4: The RFID-Gate 2D Model Layout

15 THE ADVANCED VERTICAL ARTICULATED ROBOT ATOM



Picture 15-1: The Advanced Vertical Articulated Robot atom

An Vertical Articulated Robot is a handling tool for any kind of pick-, place- and motion-tasks. The sphere of action is limited by the length of robot arms. A layout of at least 5 axis are necessary to reach any destination within the sphere of action. Motions are defined based on a world coordinate system. The destinations of the Tool are defined with xyz-coordinates. Any kind of action like loading, unloading, moving, delays are defined within Routines. Those Routines are stored with global references, so they can be used by other Vertical Articulated Robots as well. Every kind of command has its specific parameters like load time or speed to define the behaviour of the robot. There are two ways to define routine motion commands. First one is to move the tool of the robot by hand and then get the coordinates in order to teach the command into the routine.

Second one is the other way around when the robot is assigned to drive to user edited coordinates. If the destination has been reached successfully, the coordinates can be integrated into an motion command. Routines can be assigned to tasks. Routine execution is triggered if there is a task available. Strategies can be defined to decide which routine has to be executed, for what kind of task. If a Routine has been started it executes all commands in sequence before it becomes idle again. Due to high level of danger Advanced Vertical Articulated Robot systems are driven without human interference.

Important: There a two ways to use the Advanced Vertical Articulated Robot. For any kind of pick and place tasks, the robot channels have to be connected to previous and following atoms to make it part of the material flow. Motion without material handling do not need channel connections, but the Routines need to be started by external call.

- *Atom Name*
  The name of the atom.

- *Offset (x/y/z) of coordinate system [m]*
  All motions of Advanced Vertical Articulated Robot are determined based on the origin of its coordinate system. Per default it is located in the rotation center of the robot socket at z-level zero. Edit offsets to shift that point to a more comfortable location, before starting routine definition.

Note: It can be helpful to lock the position of the robot to protect it from unintended dislocation.

- *Start position (x/y/z) of tool [m]*
  Default position of the tool center relating to the coordinate system. The position is set on reset.

Note: Choose a start position that can be reached according to robot geometry and sizes.

- *Trigger on entry*
  Determines what kind of action needs to be executed when a Product atom enters the Advanced Vertical Articulated Robot atom. There are a number of rules predefined (see also Trigger on exit), but you can also create your own rule via a 4DScript expression.

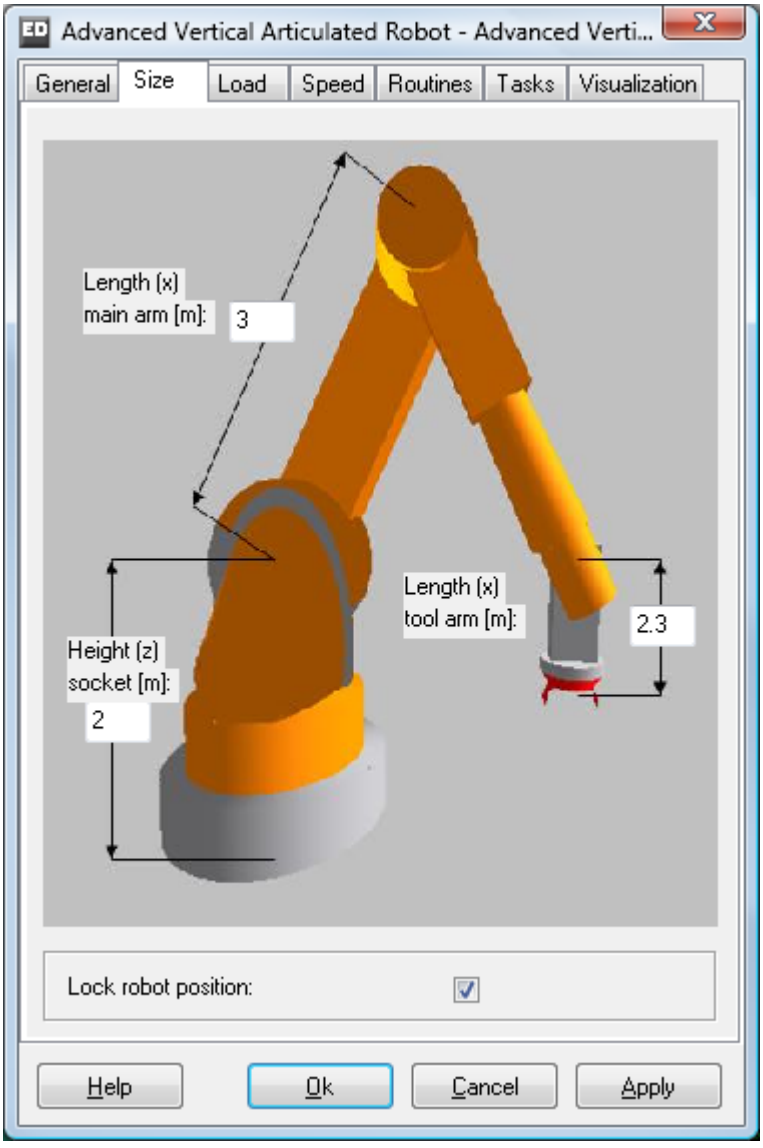
Note: The trigger on entry is executed from the Tool sub-atom, not from the Advanced Vertical Articulated Robot atom itself.

- *Trigger on exit*
  Determines what kind of action needs to be executed when a Product atom exits the Advanced Vertical Articulated Robot atom. There are a number of rules predefined (see also Trigger on entry), but you can also create your own rule via a 4DScript expression.

Note: The trigger on exit is executed from the Tool sub-atom, not from the Advanced Vertical Articulated Robot atom itself.

- *Trigger on reset*
  Determines what kind of action needs to be executed when the Advanced Vertical Articulated Robot atom is reset. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

Note: The trigger on reset is executed from the Tool sub-atom, not from the Advanced Vertical Articulated Robot atom itself.



Picture 15-2: The Advanced Vertical Articulated Robot size parameters

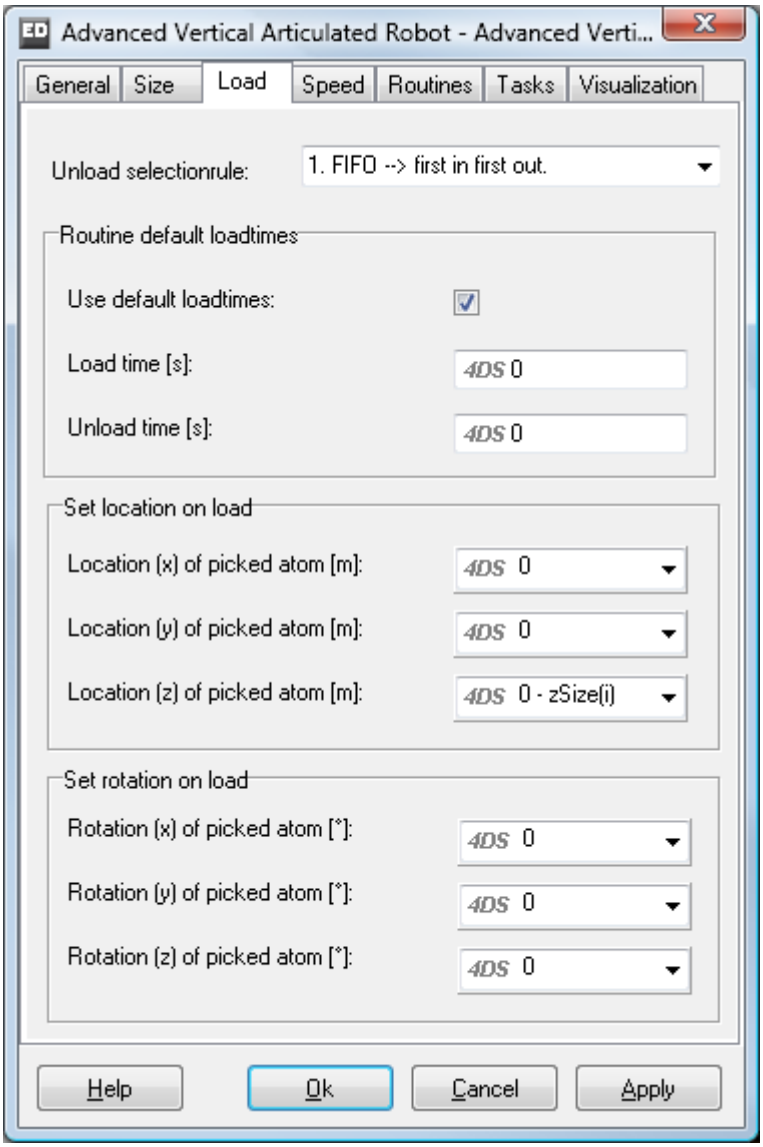
The layout of robots is defined by number/kind of articulated joints and the length of arms. The Advanced Vertical Articulated Robot atom has 5 axis. Starting with a horizontally rotating socket there are two vertical main arms of equal length, followed by another vertical Tool arm that finally can be rotated around its vertical centre.

- *Height (z) of socket [m]*
  The z-location of the first joint of the main arm.

- *Length (x) of main arm [m]*
  Define the main sphere of action.

- *Length (x) of tool arm [m]*
  Determines the sphere of action in detail.

- *Lock robot position*
  If this box is checked the position of the robot is locked to protect it from unintended dislocation.



Picture 15-3: The Advanced Vertical Articulated Robot load parameters

- *Unload selectionrule*
  This option lets unload the picked atoms in a specified sequence. The internal queue of the Advanced Vertical Articulated Robot is sorted in the selected sequence before unloading. Default is first in first out (FIFO). There are 6 strategies available.
  1: FIFO --> first in first out
  2: LIFO --> last in first out
  3: Label minimum (picked atoms)
  4: Label maximum (picked atoms)
  5: Icon index minimum (picked atoms)
  6: Icon index maximum (picked atoms)

Note: Picked atoms are stored in the sub-atom Tool of Advanced Vertical Articulated Robot.

- *Use default load times*
  If this box is checked the default load and unload times are directly assigned to Routine edit fields in order to reduce the number of edit parameters.

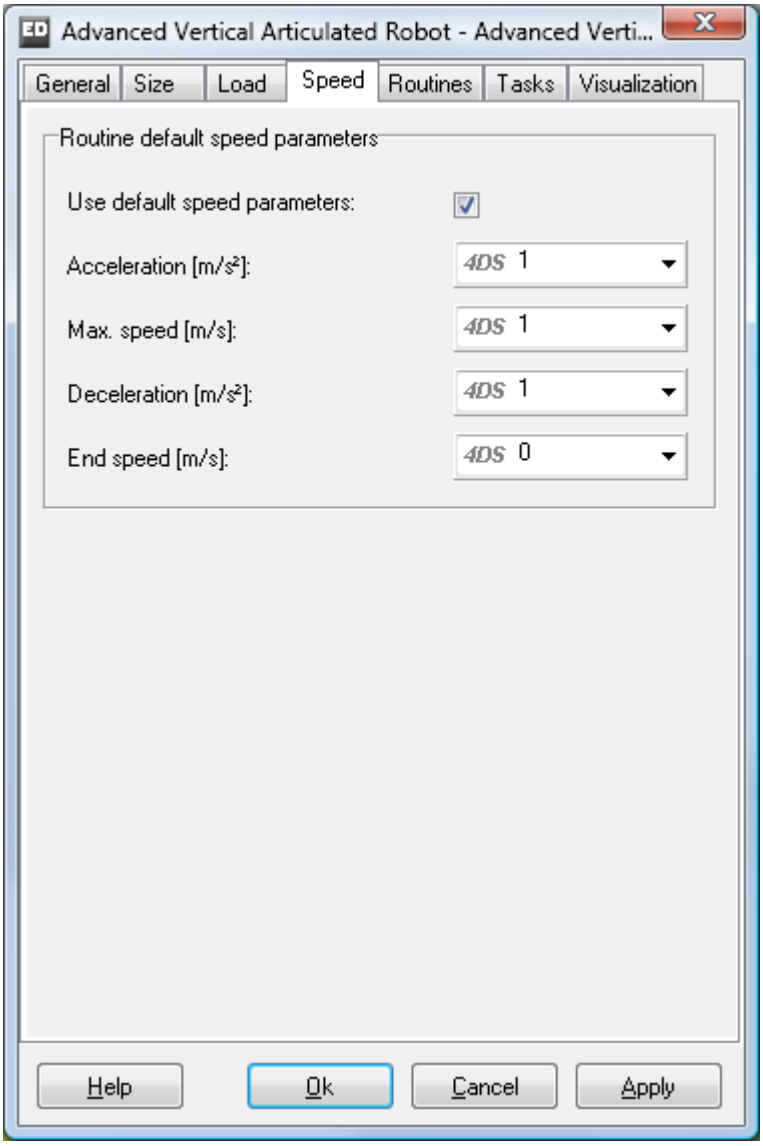
Note: Use this parameter for simplified and fast Routine definition. Uncheck for detailed un- and load times.

- *Load time [s]*
  The default time (s) that is needed to load something to Advanced Vertical Articulated Robot. Assigned to the Routine load command when checkbox use default load times is checked.

- *Unload time [s]*
  The default time (s) that is needed to unload something from Advanced Vertical Articulated Robot. Assigned to the Routine unload command when checkbox use default load times is checked.

- *Location (x/y/z) of picked atom [m]*
  For picked atoms you can define location offsets for each individual x, y, and z.

- *Rotation (x/y/z) of picked atom [m]*
  For picked atoms you can define rotation offsets around each axis individual x, y, and z.



Picture 15-4: The Advanced Vertical Articulated Robot speed parameters

- *Use default speed parameters*
  If this box is checked the default speed parameters are directly assigned to routine edit fields in order to reduce the number of edit parameters.

Note: Use this parameter for simplified and fast Routine definition. Uncheck for detailed speed definitions.

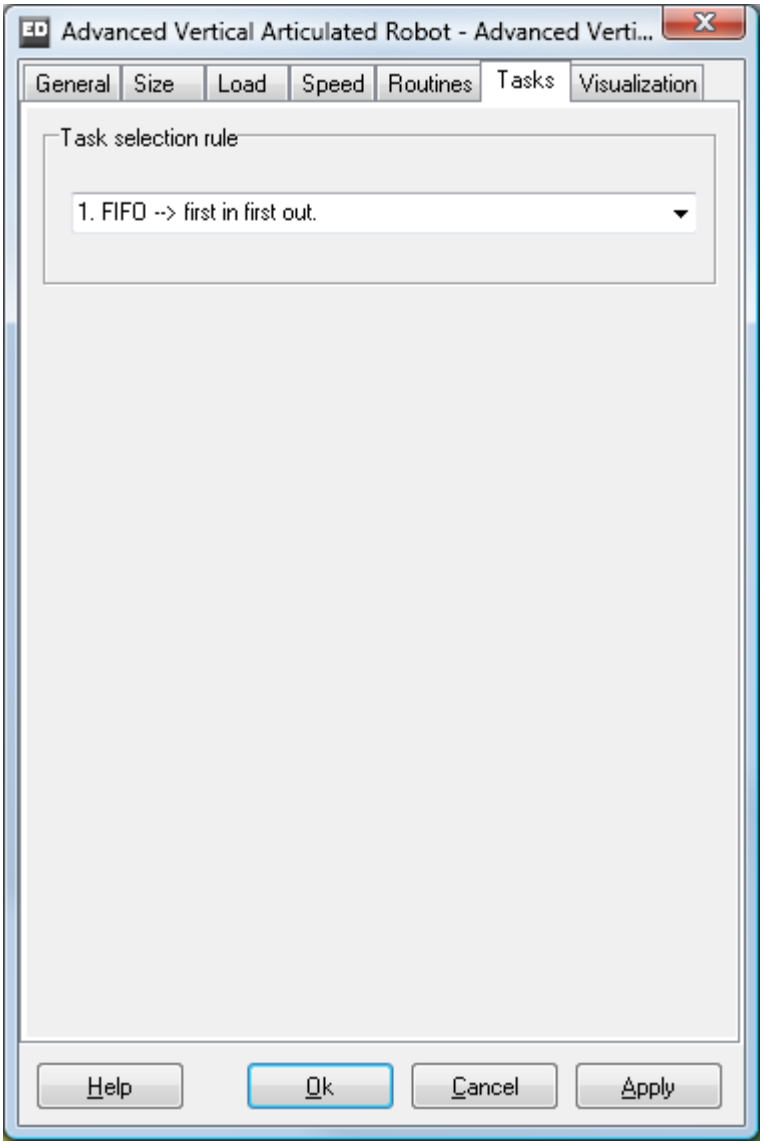
- *Acceleration [m/s²]*
  Acceleration of Tool center in relation to origin of coordinate system.

- *Max. speed [m/s]*
  Maximum speed of Tool center in relation to origin of coordinate system.

- *Deceleration [m/s²]*
  Deceleration of Tool center in relation to origin of coordinate system.

- *End speed [m/s]*
  End speed of Tool center in relation to origin of coordinate system.

Note: The Advanced Vertical Articulated Robot uses the 'MovingTo' command to define the motions between the origin of the coordinate system and the center of the Tool. In 'MovingTo' the acceleration and deceleration are leading. The end speed will be tried to reach, but when this is impossible the end speed is the last speed taking acceleration and deceleration into account.

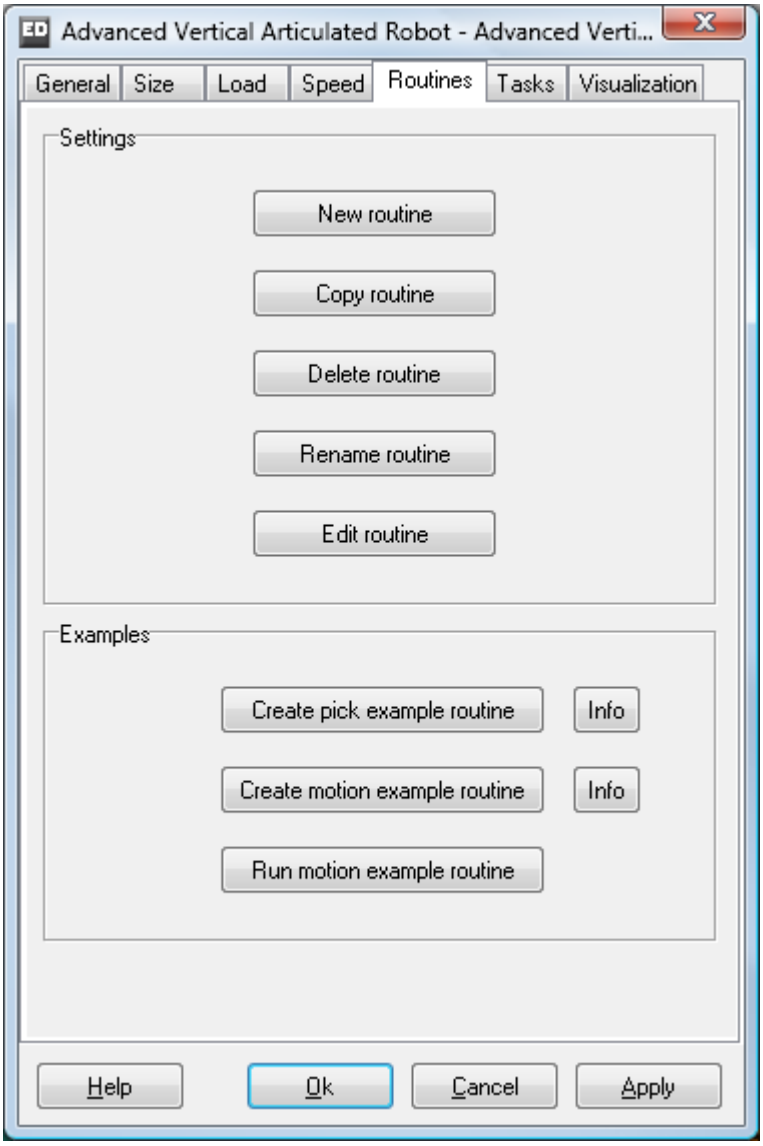


Picture 15-5: The Advanced Vertical Articulated Robot task parameters

Tasks are calls to the Advanced Vertical Articulated Robot to execute a specified Routine. There are two ways to create a Task. Pick and place Task are created every time an atom connected to an input channel of the robot causes an IcReady. Motion Tasks are created if there is an external call created with the 'Advanced_Vertical_Articulated_Robot_create_motion_task' command. If the robot is

busy, the Tasks are stored until the robot is idle again. The Tasks selection rule determines what Tasks has to be done next.

- *Task selection rule*
  This option lets execute the Tasks in a specified sequence. The internal queue of the Advanced Vertical Articulated Robot is sorted in the selected sequence before choosing the next Tasks to execute. Default is first in first out (FIFO). There are 10 strategies available:
  1: FIFO
  First in first out
  2: LIFO
  Last in first out
  3: Content minimum
  Atoms in the container with the smallest content first
  4: Content maximum
  Atoms in the container with the largest content first
  5: IC minimum
  Lowest input channel of robot first
  6: IC Maximum
  Highest input channel of robot first
  7: Label minimum (atom)
  The atom with the lowest value on label first
  8: Label maximum (atom)
  The atom with the highest value on label first
  9: Label minimum (container)
  Atoms in the container with the lowest value on label first
  10: Label maximum (container)
  Atoms in the container with the highest value on label first



Picture 15-6: The Advanced Vertical Articulated Robot routine parameters

Routines are sub atoms of the Advanced Vertical Articulated Robot. Storing all information about repeatable sequences into tables, the behaviour becomes reproducible in order to execute frequently necessary actions like motions, loading, unloading or waiting.

- *New Routine*
  Click this button to create a new Routine. You will be asked to enter a Routine name. If the Routine has been created successfully, you are asked to edit commands to the Routine table by using the edit commands GUI.

- *Copy Routine*
  Click this button to copy a existing Routine with all its commands. Select the sample Routine via atom selector. If selection was successful, you will be asked to enter a different Routine name. If the Routine has been created successfully,

you are asked to edit Commands to the Routine table by using the edit routines GUI.

- *Delete Routine*
  Click this button to delete a existing Routine with all its table-stored Commands.

Note: The global reference to the Routine will be unregistered too. Ensure, that the Routine is not needed by any other Advanced Vertical Articulated Robot.

- *Rename Routine*
  Click this button to rename an existing Routine. You will be asked to enter a new Routine name.

- *Create pick example Routine*
  Click this button to create a Pick Routine example with the name 'Routine_pick_example'. If the Routine has been created successfully, you are asked to edit Commands to the Routine table by using the edit commands GUI.

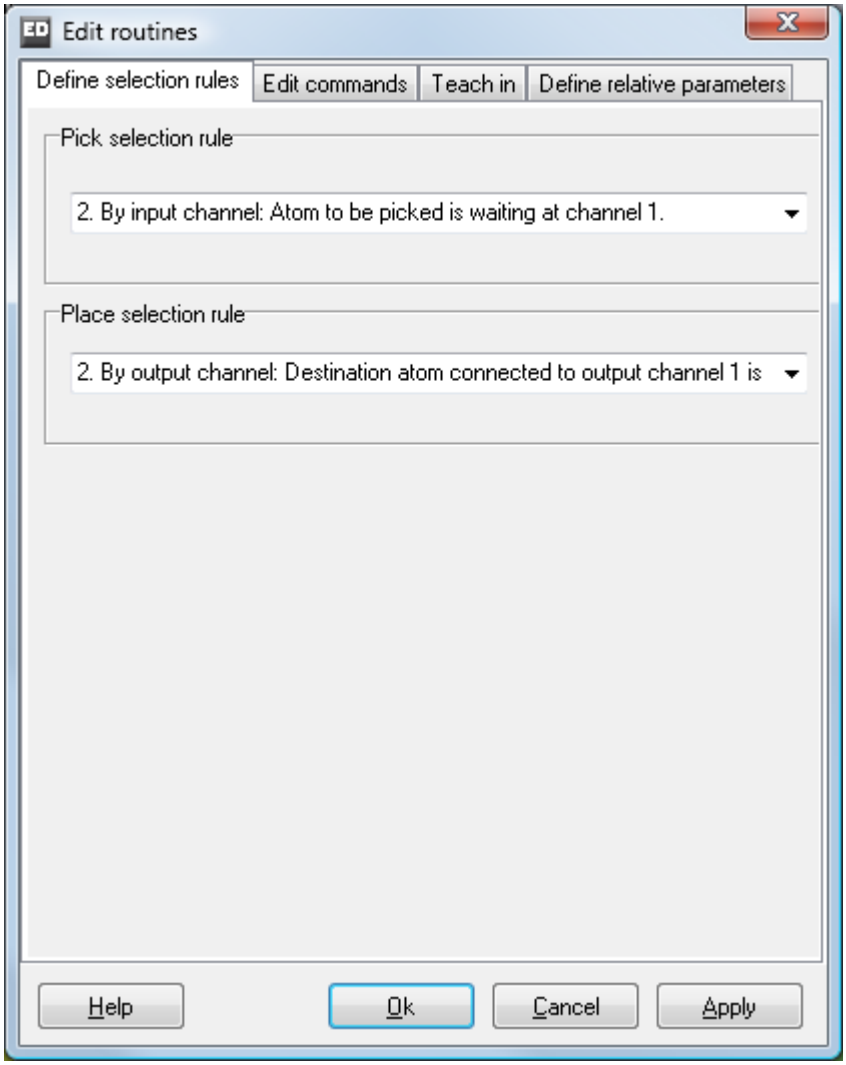
Warning!: Pick example Routine only works if input channel 1 of tool is connected to pick atom and output channel 1 of Tool is connected to place atom.

- *Rename Routine*
  Click this button to rename an existing Routine. You will be asked to enter a new Routine name.

- *Create motion example Routine*
  Click this button to create a Motion Routine example with the name 'Routine_motion_example'. If the Routine has been created successfully, you are asked to edit Commands to the Routine table by using the edit commands GUI.

- *Run motion example Routine*
  Click this button to run the 'Routine_motion_example' Routine.

- *Edit Routine*
  Click this button to select the Routine you want to edit via atom selector. If selection was successfully, you are able to edit Commands to the Routine table by using the Advanced Vertical Articulated Robot Edit Routines GUI.



Picture 15-7: The selection rule definition at the routine editor

Any kind of action like loading, unloading, moving, delays are defined within Routines. The Advanced Vertical Articulated Robot Edit Routines GUI is the tool to apply, insert, copy, delete and edit Routine Commands. It is also possible to define selection rules for the Routines, so the Robot is able to select the Routine that fits best to the current Task. The Teach In functionality allows to create motions in an easy way.

Note: Some of the parameters used for the Edit Routines GUI are defined within the Advanced Vertical Articulated Robot GUI.

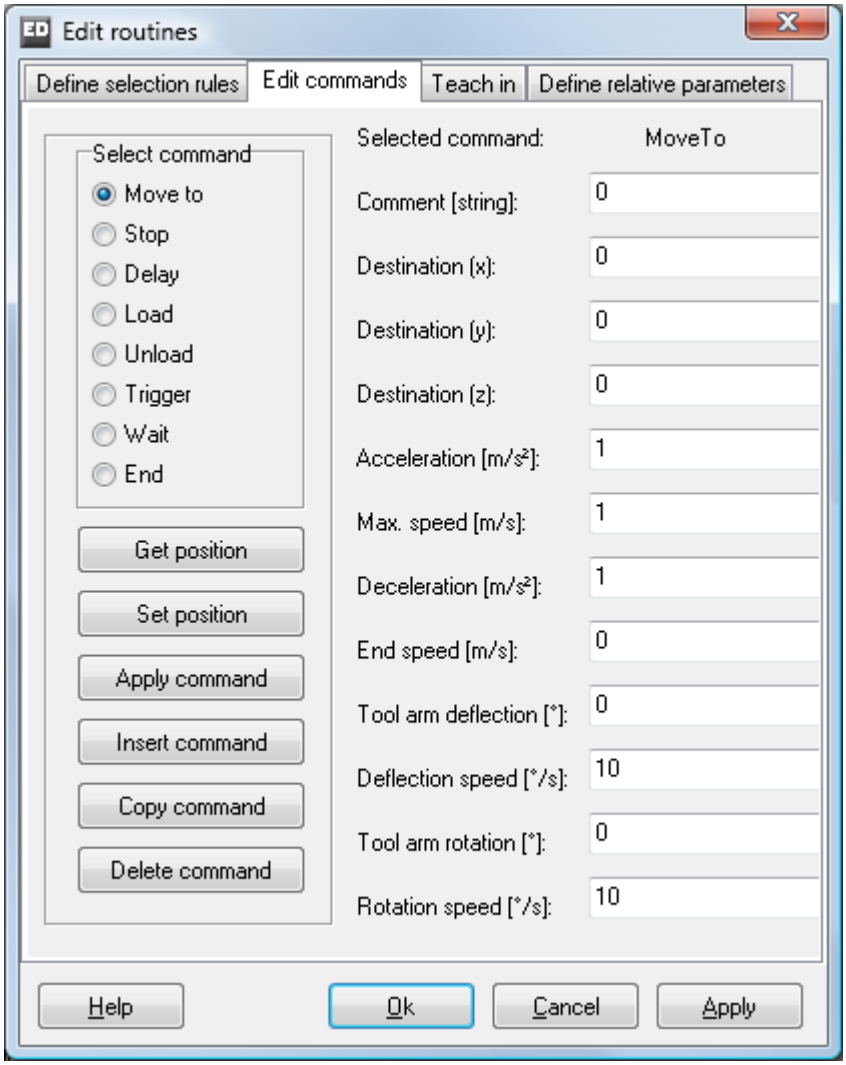
- *Pick selection rule*
  Determines the Pick condition to select the Routine. The Routine is selected, if the selected statement is true. There are 16 strategies available.
  1: No pick restriction or motion routine
  2: By input channel
  3: By pick atom name
  4: By pick icon name
  5: By pick icon number

6: By pick label value (direct)
7: By pick label value (conditional)
8: By pick label text
9: By pick location (direct)
10: By pick location (conditional)
11: By pick size (direct)
12: By pick size (conditional)
13: By pick content (direct)
14: By pick content (conditional)
15: Conditional statement
16: By user

Note: Choose a Pick selection rule if there is more than one Routine available and the selection criteria can be found via input channel connection. The Pick selection rule can be combined with a Place selection rule to define advanced strategies.

- *Place selection rule*
  Determines the Place condition to select the Routine. The Routine is selected, if the selected statement is true. There are 11 strategies available.
  1: No place restriction or motion routine
  2: By output channel
  3: By output channel fail safe
  4: By place location (direct)
  5: By place location (conditional)
  6: By place size (direct)
  7: By place size (conditional)
  8: By place content (direct)
  9: By place content (conditional)
  10: Conditional statement
  11: By user

Note: Choose a Place selection rule if there is more than one Routine available and the selection criteria can be found via output channel connection. The Place selection rule can be combined with a Pick selection rule to define advanced strategies.



Picture 15-8: The Advanced Vertical Articulated Robot command editor

- *Select command*
  Select between the predefined kinds of Commands to define robot behaviour. There are 8 predefined Commands available:
  1: Move to
  Defines any kind of robot motion
  2: Stop
  Stops any kind of robot motion
  3: Delay
  Makes the robot waiting for a specific time
  4: Load
  Executes picking of incoming products
  5: Unload
  Executes placing of outgoing products
  6: Trigger
  Allows the execution of 4DScript code
  7: Wait
  Let the robot wait for an external trigger to continue

8: *End*
Finishes a routine

Choose a Command to edit the necessary parameters. The number and kind of parameters depends on the type of selected Command.

Move to command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Destination (x/y/z)*
  Tool center destination according to the origin of the coordinate system.

- *Acceleration [m/s²]*
  The tool centre acceleration to reach the destination.

- *Max. speed [m/s]*
  The tool centre maximum speed to reach the destination.

- *Deceleration [m/s²]*
  The tool centre deceleration to reach the destination.

- *End speed [m/s]*
  The Tool centre end speed to reach the destination.

- *Tool arm deflection [°]*
  Absolute deflection of the tool arm at the destination.

- *Deflection speed [°/s]*
  The deflection speed to reach the absolute Tool arm deflection.

- *Tool arm rotation [°]*
  Absolute rotation of the Tool arm at the destination.

- *Rotation speed [°/s]*
  The rotation speed to reach the absolute Tool arm rotation.

Note: You are able to edit detailed speed parameters, if the default speed parameters checkboxes is selected at the Advanced Vertical Articulated Robot GUI.

Note: The Advanced Vertical Articulated Robot uses the ‘MovingTo’ Command to define the motions between the origin of the coordinate system and the centre of the Tool. In ‘MovingTo’ the acceleration and deceleration are leading. The end speed will be tried to reach, but when this is impossible the end speed is the last speed taking acceleration and deceleration into account.

Note: You can get the current position of the Tool into the edit fields by click on the Get position button.

Note: You can set the position of the Tool according to the edit fields by click on the Set position button.

Stop command:

- *Comment*
  Comment the Commands to make the Routines more readable.

Delay command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Time [s]*
  The delay before the Routines is continued with the next command (row).

Load command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Time [s]*
  The load time of the atom to be picked.

- *Channel [index]*
  The input channel of the robot, where the atom is waiting.

Note: You are able to edit detailed load parameters, if the default load parameters checkboxes is selected at the Advanced Vertical Articulated Robot GUI.

Unload command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Time [s]*
  The unload time of the atom to be placed.

- *Channel [index]*
  The output channel of the robot, where the atom has to be sent to.

Note: You are able to edit detailed unload parameters, if the default load parameters checkboxes is selected at the Advanced Vertical Articulated Robot GUI.

Trigger command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Trigger [4DScript]*
  The code that has to be executed before the Routine is continued with the next Command. Select between the predefined Commands defined in the kernel function ‘getTriggerEnterExit’ or enter your own 4DScript code.

Wait command:

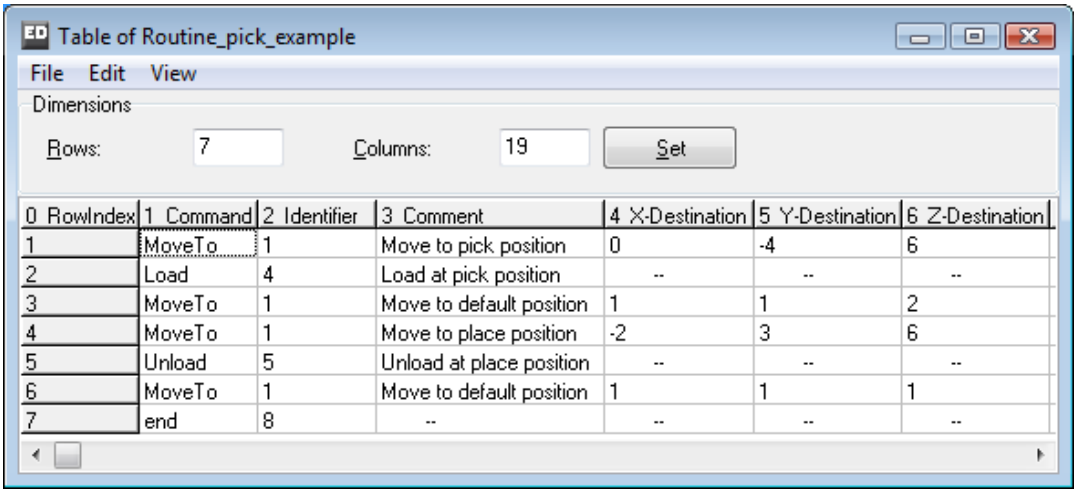
- *Comment*
  Comment the Commands to make the Routines more readable.

- *Awaited message [string]*
  The text message that has to arrive in the OnMessage event handler of the Advanced Vertical Articulated Robot before the Routine is continued with the next Command.

End command:

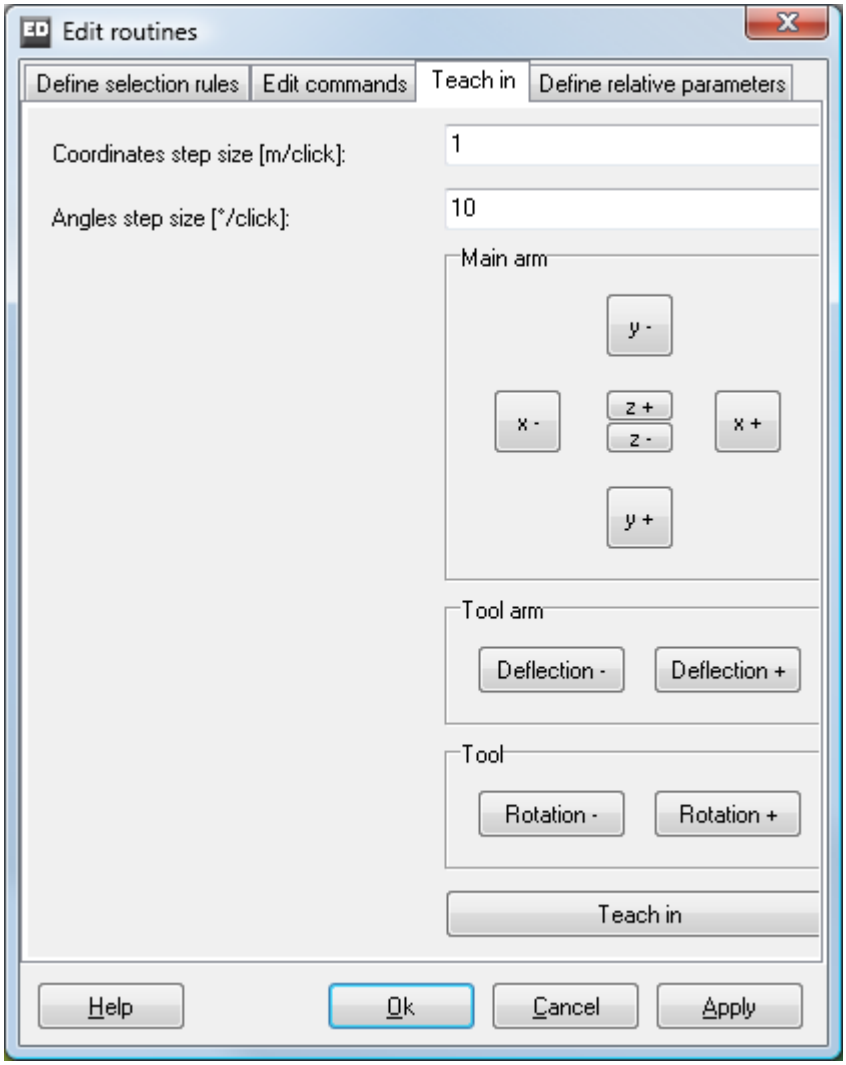
If you select the End Command no parameters can be changed, but another row is added to the Routine in order to set a flag, that that there will be no further Commands. The Robot is set to status Idle and it searches for other Tasks available.

There are four buttons to apply, insert, copy and delete Commands. On click you are asked to edit the row index of the Command that you want to apply, insert, copy or delete.



0 RowIndex	1 Command	2 Identifier	3 Comment	4 X-Destination	5 Y-Destination	6 Z-Destination
1	MoveTo	1	Move to pick position	0	-4	6
2	Load	4	Load at pick position	--	--	--
3	MoveTo	1	Move to default position	1	1	2
4	MoveTo	1	Move to place position	-2	3	6
5	Unload	5	Unload at place position	--	--	--
6	MoveTo	1	Move to default position	1	1	1
7	end	8	--	--	--	--

Picture 15-9: Routine example (pick and place)



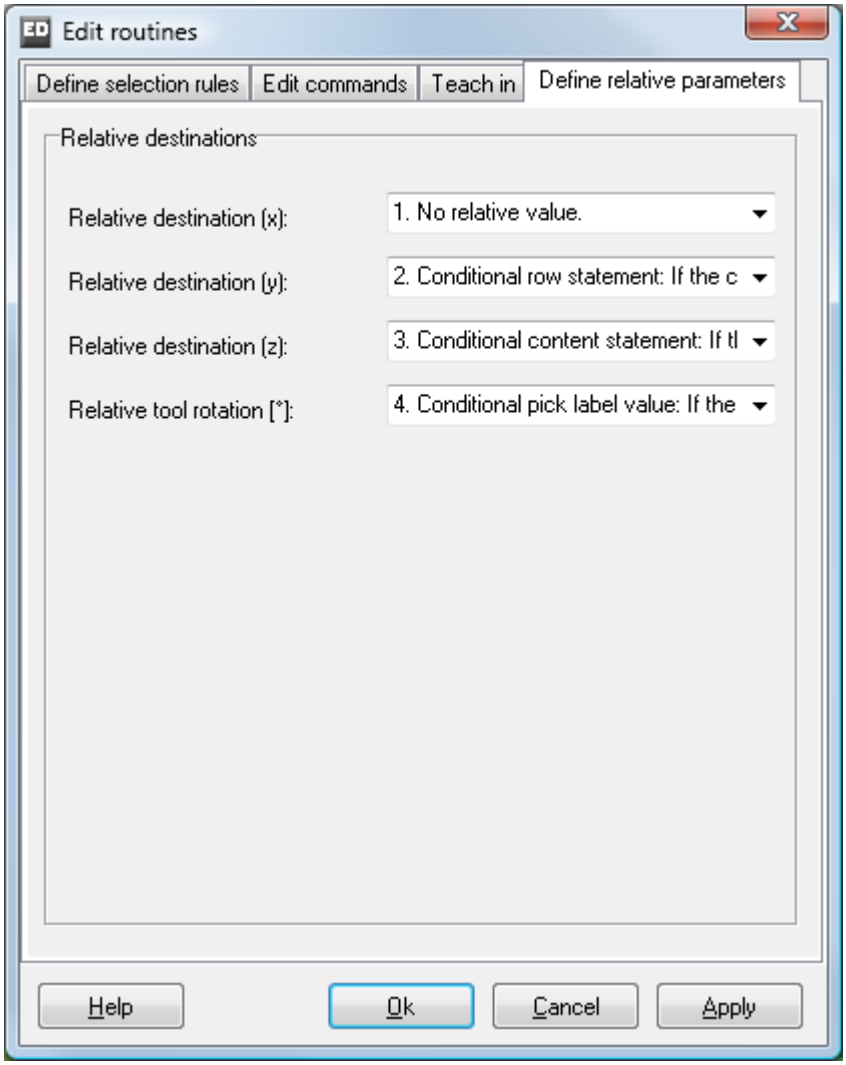
Picture 15-10: Routine teach panel (move to command)

The Teach In functionality allows to create motion commands in an easy way.

- *Coordinates step size [m/click]*
  On click of main arm buttons the tool position will move according to the coordinates step size.

- *Angles step size [°/click]*
  On click of tool arm buttons the tool deflection will rotate according to the angles step size. On click of tool buttons the tool will rotate according to the angles step size.

- *Teach in button*
  On click all motion parameters are updated into the edit fields of the Edit Command page and the destination is stored to the active routine as Move to Command.



Picture 15-11: Relative destinations panel

- *Relative destination (x)*
  The tool relative x destination.

- *Relative destination (y)*
  The tool relative x destination.

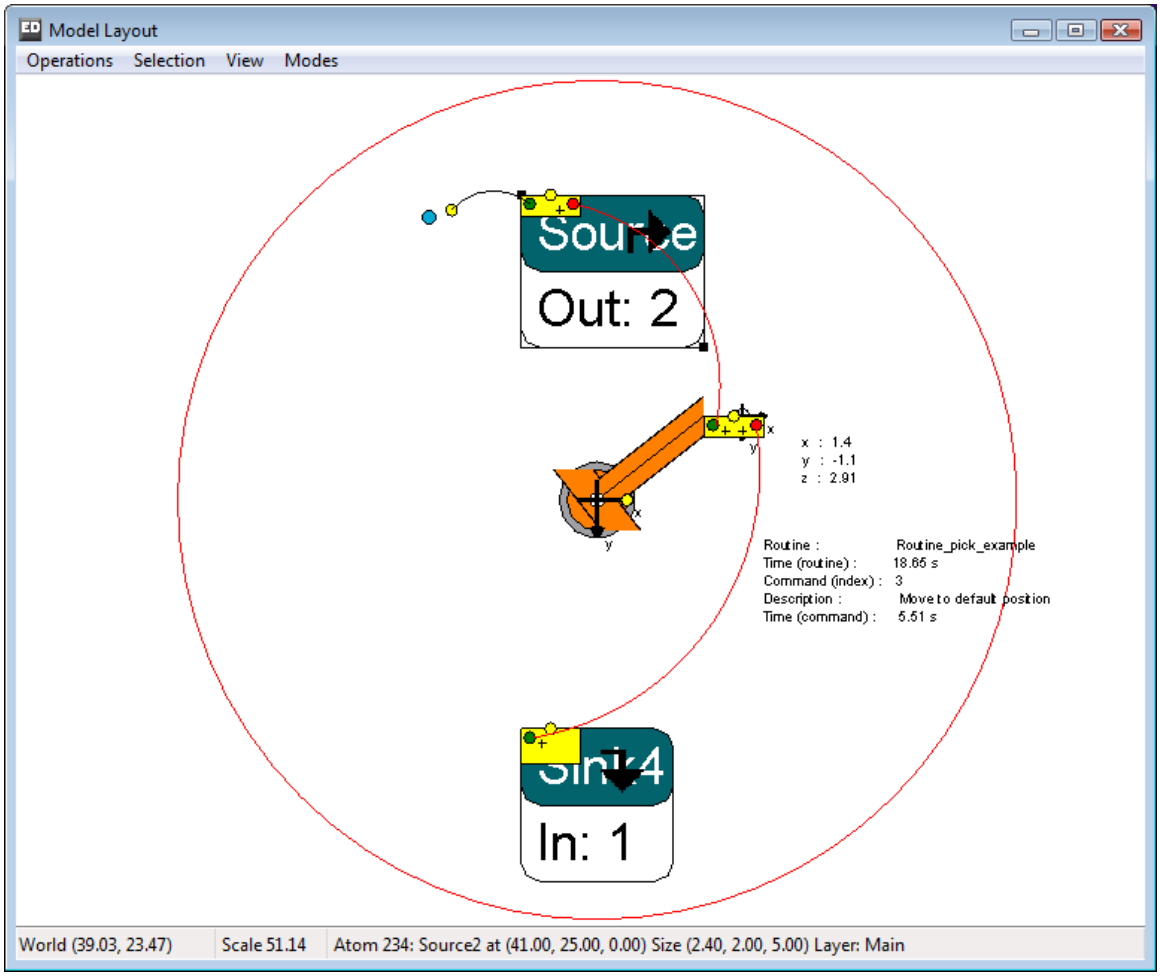
- *Relative destination (z)*
  The tool relative x destination.

- *Relative tool rotation [°]*
  Relative rotation of the tool arm at the destination.

There are 5 predefined relative value definitions available:

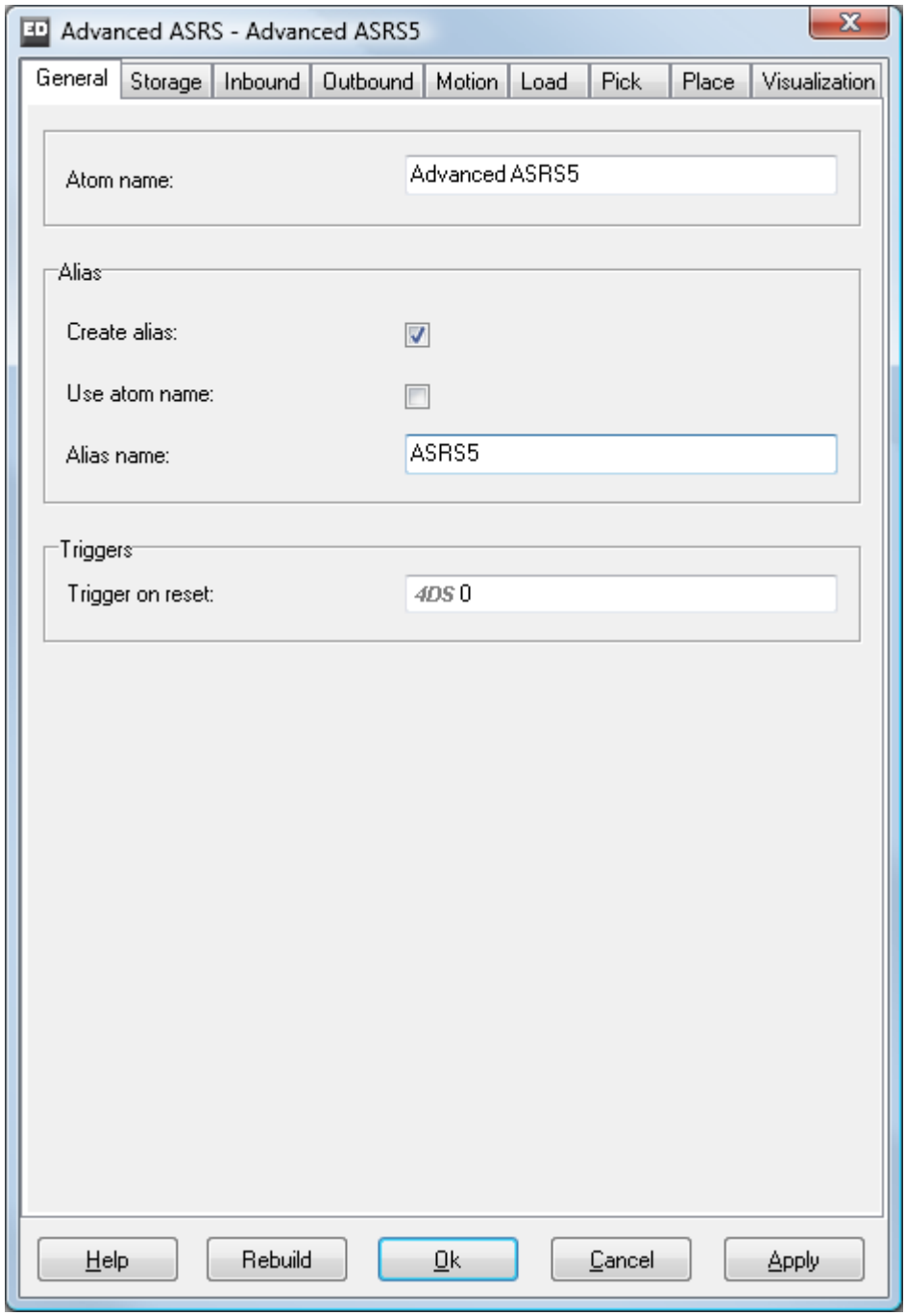
1: No relative value – the parameters from the Routines table are used on any account

2: *Conditional row statement – cases are defined based on the current row index of the active Routine*
3: *Conditional content statement - cases are defined based on the current content of the tool*
4: *Conditional pick label value statement - cases are defined based on Label values of Products waiting at input channel of the tool*
5: *User defined statement – define your own statement using 4DScript*



Picture 15-12: 2D Layout of a running pick example

16 THE ADVANCED ASRS ATOM



Picture 16-1: The Advanced ASRS atom

An ASRS is a rail bound single track vehicle to store and retrieval goods of warehouse racks. Commonly used layout contains one locations for Inbound, one storage and retrieval machine (SRM) with up to 2 High Rise Racks on the sides and one location for Outbound. There are parameters to edit size, position and layout of all elements.

The SRM is made of 3 atoms. The SRM atom itself uses the rail track to move along aisle (x-axis) to the assigned column of the Racks. The Hoist atom is lifting up to the allocated row of the rack (z-axis). The Shuttle atom reaches into the racks to store or retrieve the goods (y-axis). Each axis has its own parameters to control acceleration, maximum speed and deceleration. Detailed load times can be defined for every transfer point inside the system. Due to high level of automation ASRS systems are driven without human interference, but controlled by strategies for Inbound, movement and Outbound of goods.

Important: The Advanced ASRS layout is based on rows and columns of warehouse racks coordinate system. Use rows and columns to define the dimension of Racks. Also use coordinates to define Inbound and Outbound position.

Note: This atom is part of ED Logistics. However ED Logistics also has a single Warehouse atom to model non automated storage systems.

- *Atom Name*
  The name of the atom.

Note: There are 2 ways to access the ASRS's values from another atom. The first possibility is to connect an atom to the ASRS's central channel.

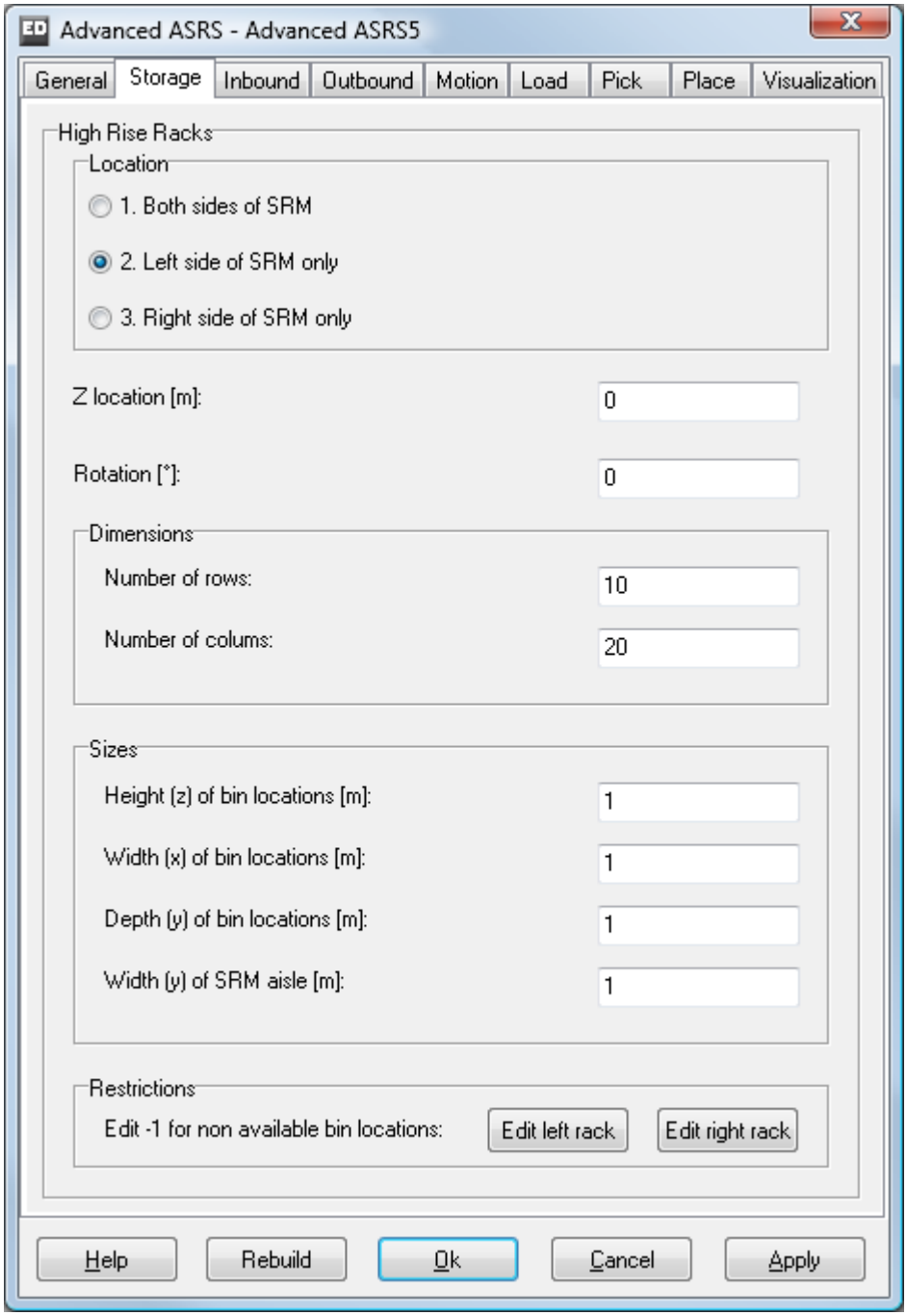
Suppose you connected output channel 1 of an atom XYZ to the ASRS, then you can refer to ASRS from XYZ as follows: Out(1, c).

Since ASRS's are referenced from many atoms this is not a comfortable way to get or set the required data. You can have the ASRS automatically create 4DScript functions (aliases) to be able to access the data directly without any channel connections.

Example:
If you name the ASRS MyASRS, and you check the box Create Alias 3 4DScript functions will be created: MyASRS, SetMyASRS, and RefMyASRS. You can now use these functions to access the table from another atom without connecting channels.

MyASRS is used to obtain a cell value. SetMyASRS is used to set a table value, and RefMyASRS is used to refer to the table atom itself.

- *Trigger on reset*
  A rule that determines what kind of action needs to be executed when the ASRS atom is reset. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.



Picture 16-2: The Advanced ASRS storage parameters

- *Location*
  The layout of the High Rise Racks. Select between left or right High Rise Rack or define High Rise Racks on both sides of the SRM atom.

- *Z location [m]*
  The z-location of the ASRS.

- *Rotation [°]*
  At default the ASRS is positioned from left to right in your screen. However you

can also rotate the atom to display the flow of atoms over the ASRS matches reality.

Dimensions:

- *Nr of rows*
  Number of bin locations in vertical direction (z) in each High Rise Rack.

- Nr of columns
  Number of bin locations in horizontal direction (x) in each High Rise Rack.

Sizes:

- *Height (z) of bin locations (m)*
  The z-size of all bin locations inside each High Rise Rack. Determines (together with nr of rows) the total height of High Rise Racks.

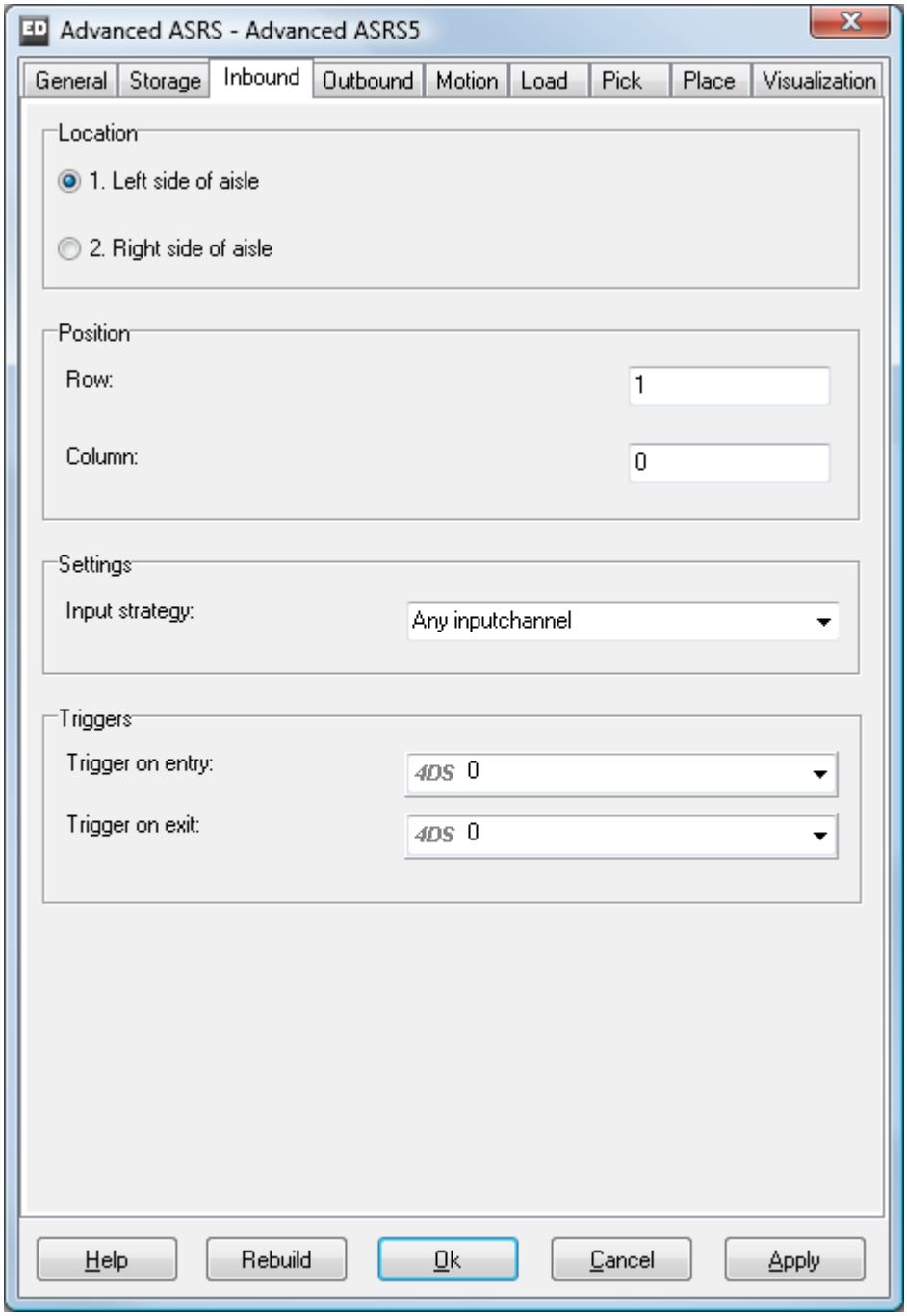
- *Width (x) of bin locations (m)*
  The x-size of all bin locations inside each High Rise Rack. Determines (together with nr of columns) the total length of High Rise Racks.

- *Depth (y) of bin locations (m)*
  The y-size of all bin locations inside each High Rise Rack. Determines the total depth of High Rise Racks.

- *Width (y) of SRM aisle (m)*
  The y-size of the aisle between the High Rise Racks. Determines the space for SRM movement.

- Restrictions
  Due to constructional limitations (pillars, passages, etc.) it is frequently not possible to use all bin locations of the High Rise Racks. Edit -1 into the cells of the High Rise Rack's table to disable place strategies to assign goods to those bin locations.

Note: Capacity of High Rise Racks is calculated without disabled bin locations. Place strategies consider the bowdlerized capacity when calculation the fill level of a High Rise Rack.



Picture 16-3: The Advanced ASRS inbound parameters

- *Location*
  The location of the Inbound atom. Select between left or right side of the aisle.

Position:

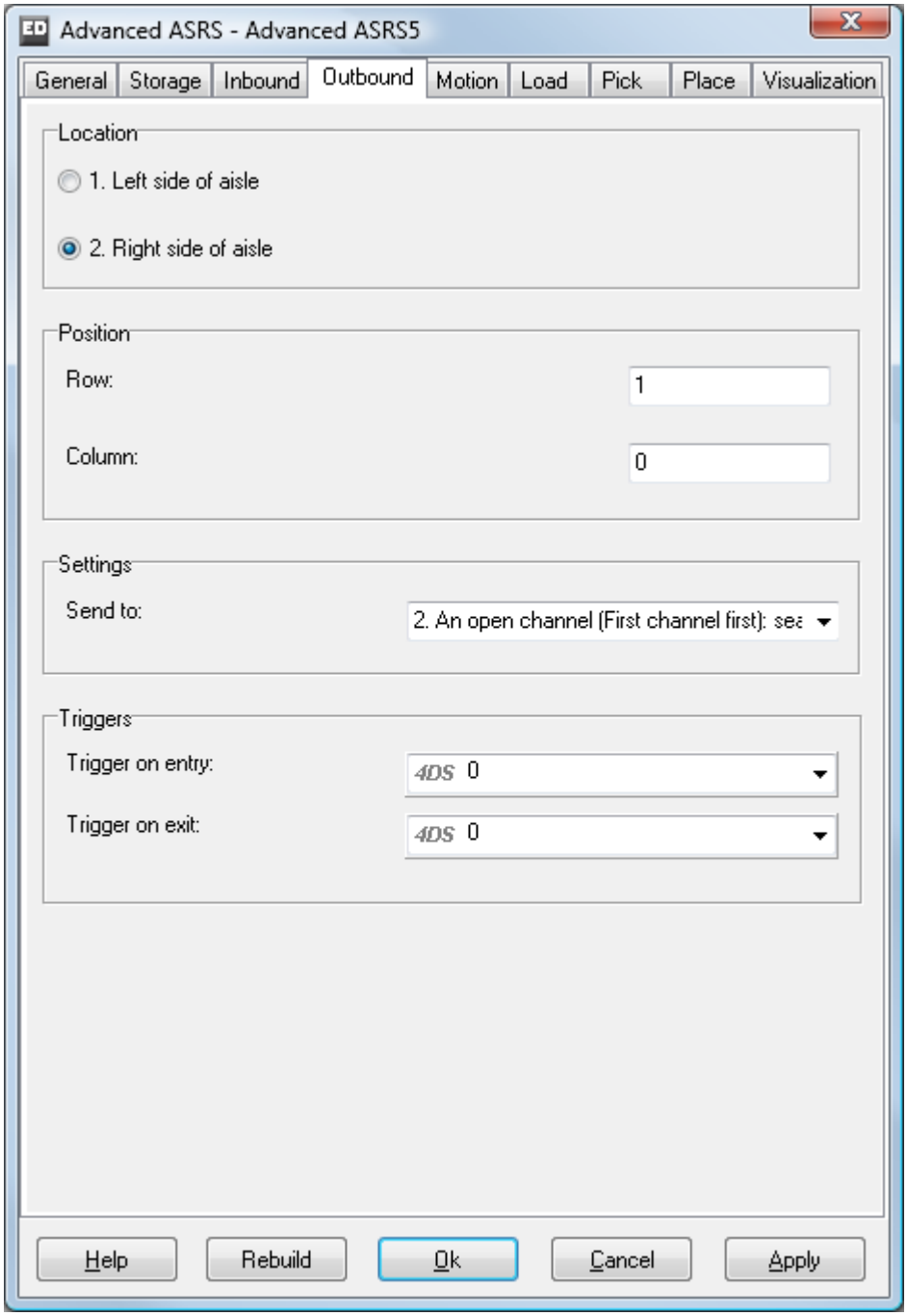
- *Row*
  Locations in vertical direction (z) based on the coordinate system of the High Rise Racks.

- *Column*
  Locations in horizontal direction (x) based on the coordinate system of the High Rise Racks.

- *Input strategy*
  A rule that determines how Product atoms may be let into the Inbound atom of the ASRS. There are a number of rules predefined (see also Input strategy), but you can also create your own rule via a 4DScript expression.

- *Trigger on entry*
  A rule that determines what kind of action needs to be executed when a Product atom enters the Inbound atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

- *Trigger on exit*
  A rule that determines what kind of action needs to be executed when a Product atom exits the Inbound atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.



Picture 16-4: The Advanced ASRS outbound parameters

- *Location*
  The location of the Outbound atom. Select between left or right side of the aisle.

Position:

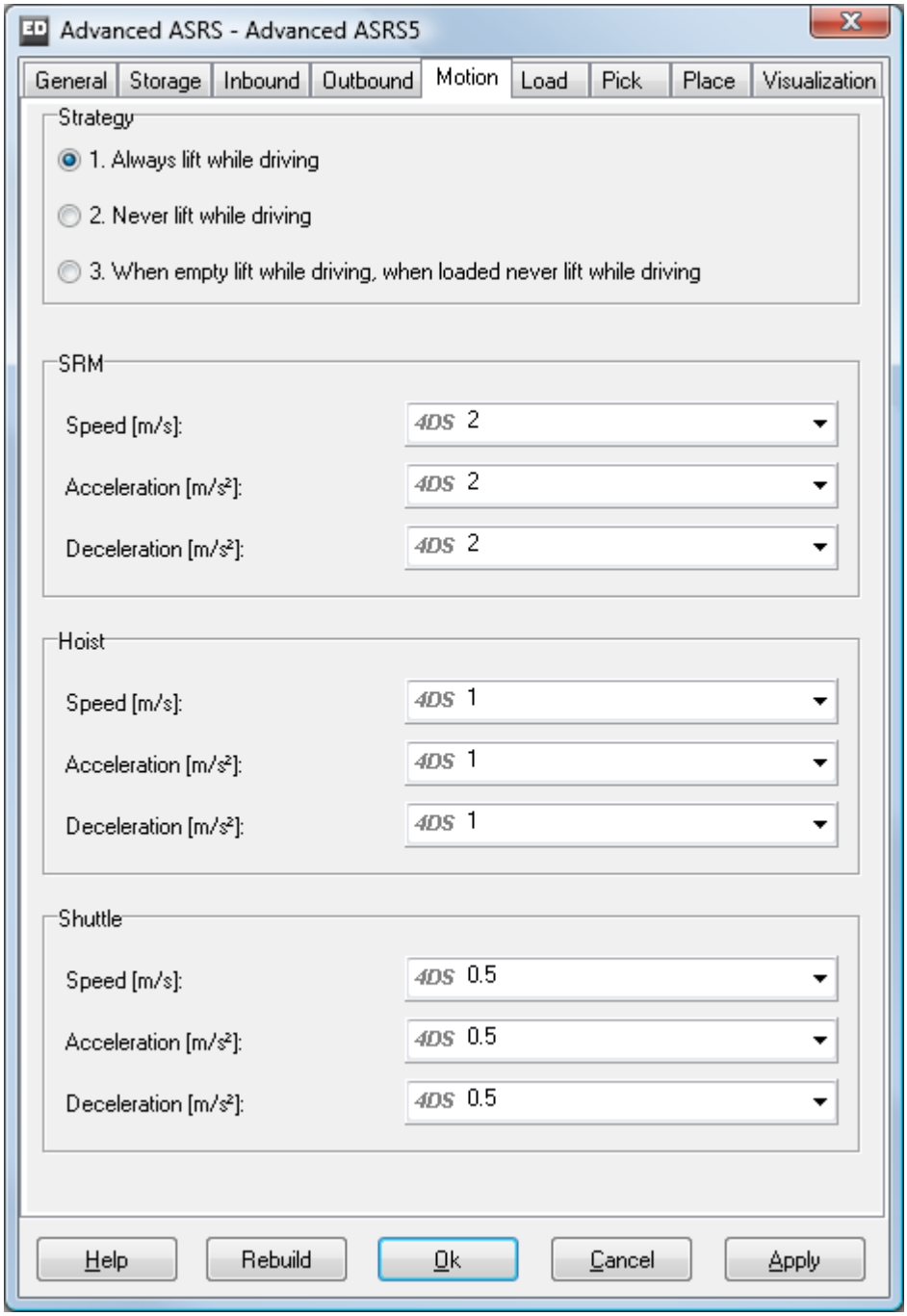
- *Row*
  Locations in vertical direction (z) based on the coordinate system of the High Rise Racks.

- *Column*
  Locations in horizontal direction (x) based on the coordinate system of the High Rise Racks.

- *Send to*
  A rule that determines to which output channel a Product atom needs to be sent. There are a number of rules predefined (see also Send to), but you can also create your own rule via a 4DScript expression.

- *Trigger on entry*
  A rule that determines what kind of action needs to be executed when a Product atom enters the Outbound atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

- *Trigger on exit*
  A rule that determines what kind of action needs to be executed when a Product atom exits the Outbound atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.



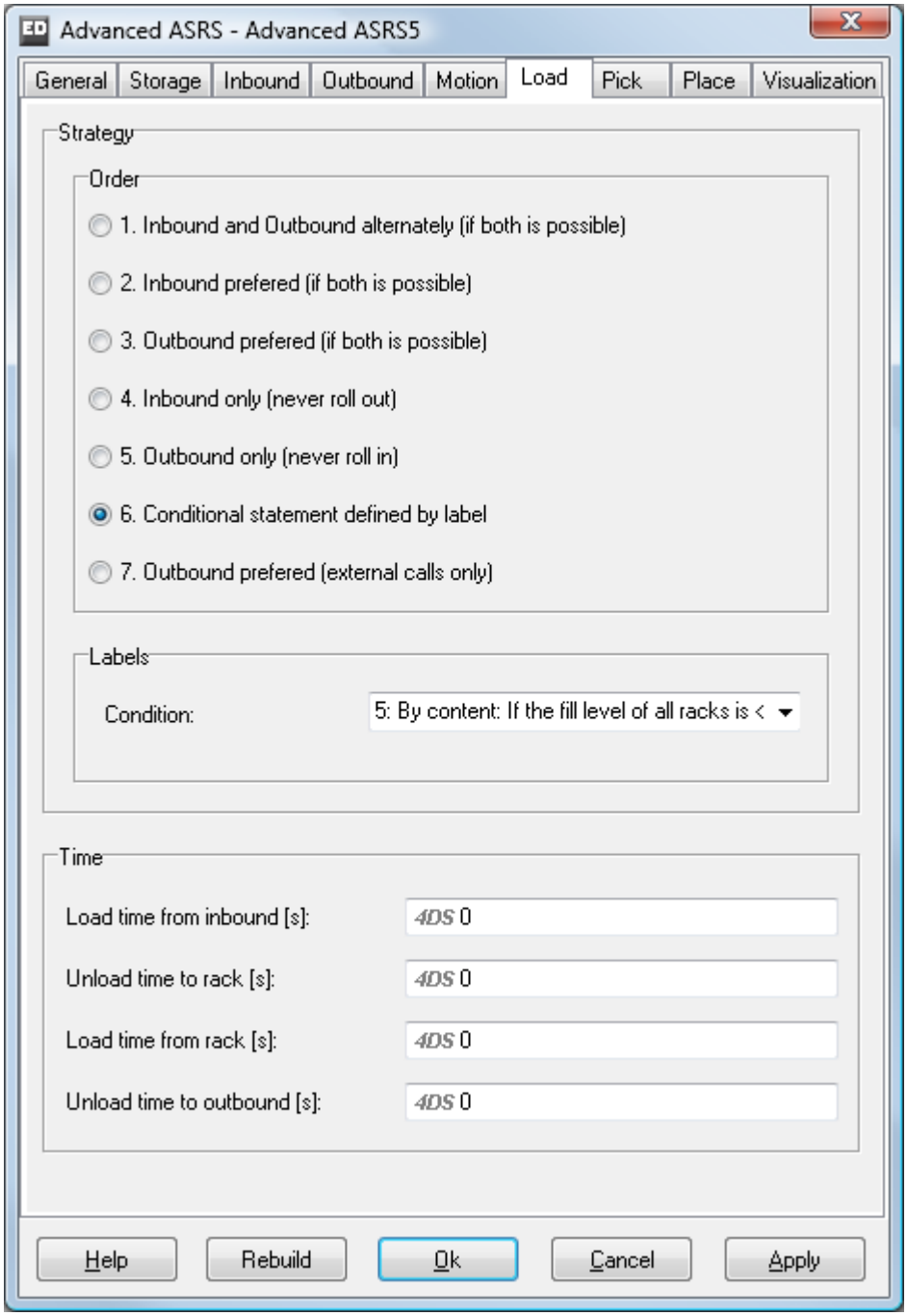
Picture 16-5: The Advanced ASRS motion parameters

- *Motion strategy*
  A rule that determines how the SRM atom is coordinates the movements of lifting and driving. There are 3 strategies available:
  1: Always lift while driving
  2: Never lift while driving
  3: When empty lift while driving, when loaded never lift while driving

- *SRM*
  The speed, the acceleration and the deceleration can be defined for x-direction.

- *Hoist*
  The speed, the acceleration and the deceleration can be defined for z-direction.

- *Shuttle*
  The speed, the acceleration and the deceleration can be defined for y-direction.



Picture 16-6: The Advanced ASRS load parameters

- *Order strategy*
  A rule that determines the interaction between Inbound and Outbound of products. There are 6 strategies available:
  1: Inbound and Outbound alternately (if both is possible)
  2: Inbound preferred (if both is possible)
  3: Outbound preferred (if both is possible)
  4: Inbound only (never roll out)
  5: Outbound only (never roll in)
  6: Conditional statement defined by label

- *Condition*
  Label to be executed when order strategy nr 6 is the active strategy. There are a number of rules predefined, but you can also create your own rule via a 4DScript expression.

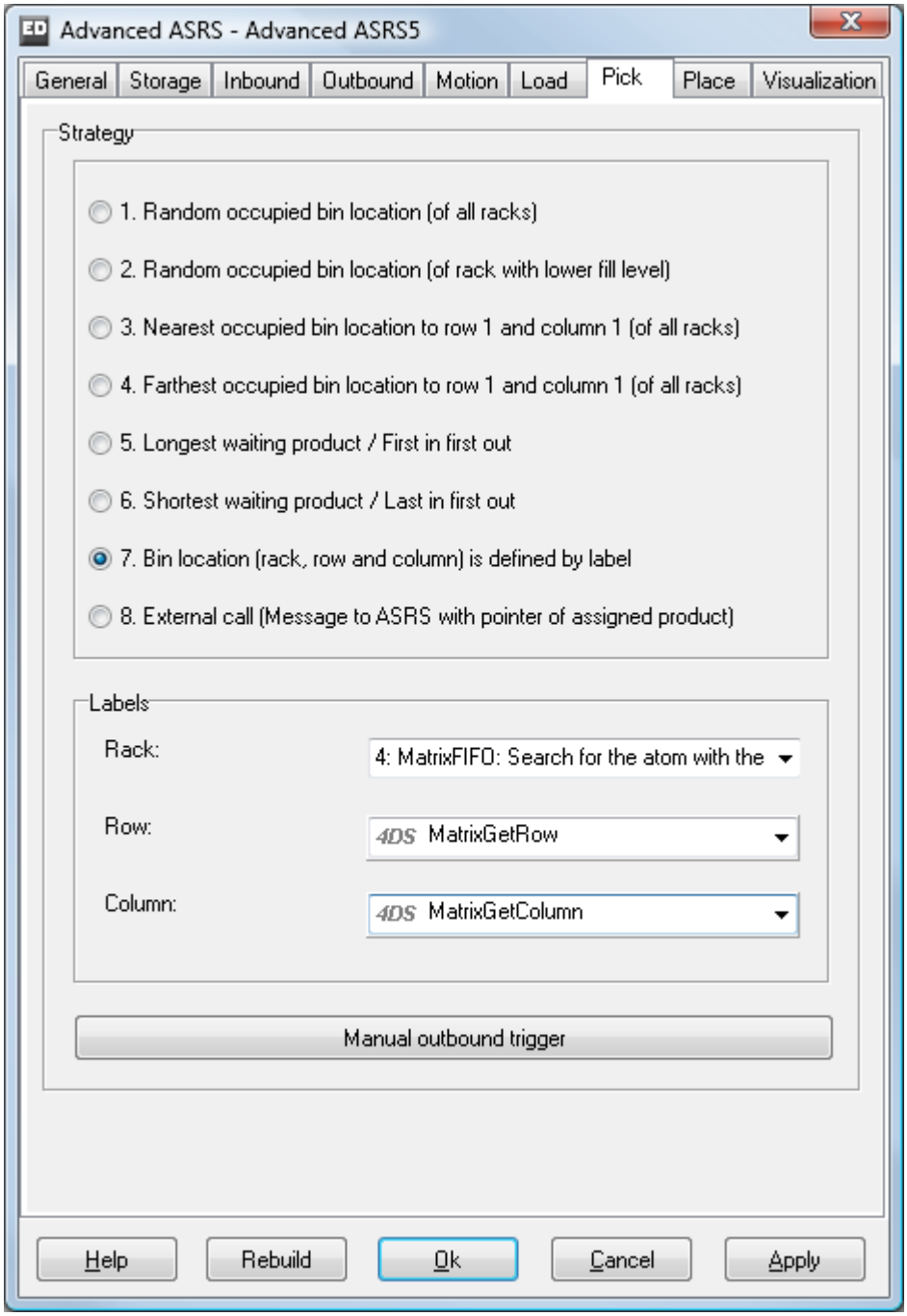
Endurance:

- *Load Time*
  The time (s) that is needed to load something from Inbound to SRM.

- *Unload Time*
  The time (s) that is needed to unload something from SRM to High Rise Racks.

- *Load Time*
  The time (s) that is needed to load something from High Rise Racks to SRM.

- *Unload Time*
  The time (s) that is needed to unload something from SRM to Outbound.



Picture 16-7: The Advanced ASRS pick parameters

- *Pick strategy*
  A rule that determines the Outbound action of ASRS by selecting the Product (and its location) for the next roll out movement. There are 8 strategies available:
  1: Random occupied bin location (of all High Rise Racks)
  2: Random occupied bin location (of High Rise Racks with lower fill level)
  3: Nearest occupied bin location to row 1 and column 1 (of all High Rise Racks)

4: *Farthest occupied bin location to row 1 and column 1 (of all High Rise Racks)*
5: *Longest waiting product / First in first out*
6: *Shortest waiting product / Last in first out*
7: *Bin location (High Rise Rack, row and column) is defined by label*
8: *External call (Message to ASRS with pointer of assigned product)*

Labels:

- *Rack*
  Label to be executed when order strategy nr 7 is the active strategy. There are a number of rules predefined to select the index of the High Rise Rack, but you can also create your own rule via a 4DScript expression.

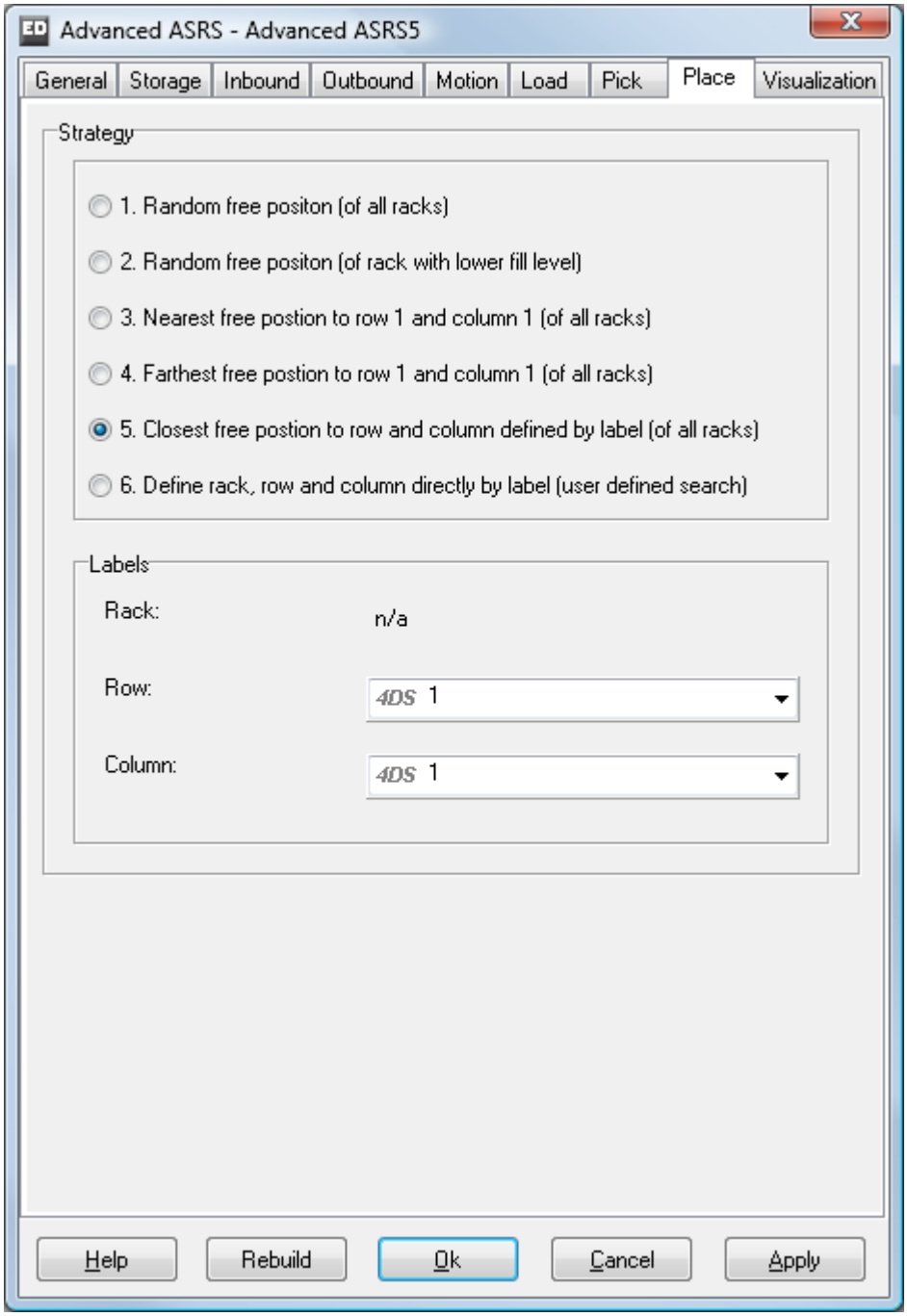
Note: The result of the rule to select a rack has always to be an index between 1 and 2. The left High Rise Rack is always assigned by index 1. The right High Rise Rack is always assigned by index 2. Be sure that the resulting index refers to an existing High Rise Rack.

- *Row*
  Label to be executed when order strategy nr 7 or 8 is the active strategy. There are a number of rules predefined to select the index of the row, but you can also create your own rule via a 4DScript expression.

- *Column*
  Label to be executed when order strategy nr 7 or 8 is the active strategy. There are a number of rules predefined to select the index of the column, but you can also create your own rule via a 4DScript expression.

Note: The result of the rule to select a row or column has always to be > 0 and within the dimensions of your High Rise Racks. Be sure that the resulting index of row and column refers to a non assigned/occupied bin location. Occupied bin locations of the High Rise Rack tables always contain the pointer of the assigned product.

- *Manual outbound trigger*
  Often picking commands are coming from external controller and not from the ASRS itself. When pick strategy nr 8 is the active strategy, ASRS picking is triggered by external messages containing the pointer of the product that has to be rolled out. The button is doing the same as an external controller and sends a message to the ASRS with the pointer of the next product to roll out. Helpful to test ASRS functionality before external controller has been implemented. Furthermore it contains a helpful syntax example for getting started with external product search.



Picture 16-8: The Advanced ASRS place parameters

- *Place strategy*
  A rule that determines the Inbound action of ASRS by selecting the bin location (High Rise Rack, row and column) for incoming products. There are 6 strategies available:
  1: Random free position (of all High Rise Racks)
  2: Random free position inside (of High Rise Rack with lower fill level)
  3: Nearest free position to row 1 and column 1 (of all High Rise Racks)
  4: Farthest free position to row 1 and column 1 (of all High Rise Racks)

5: *Closest free position to row and column defined by label (of all High Rise Racks)*
6: *Define High Rise Rack, row and column directly by label (user defined search)*

Labels:

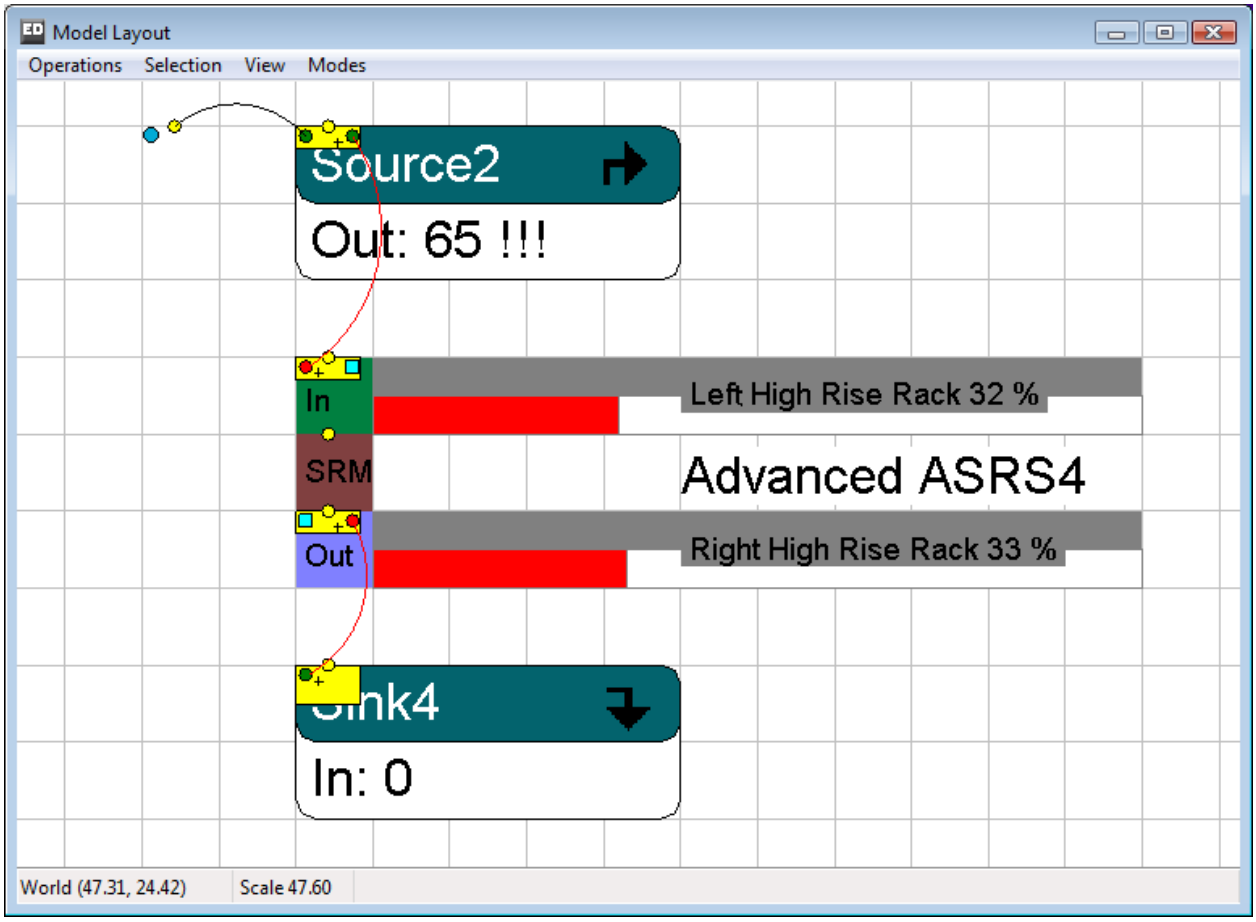
- *Rack*
  Label to be executed when order strategy nr 5 is the active strategy. There are a number of rules predefined to select the index of the High Rise Rack, but you can also create your own rule via a 4DScript expression.

Note: The result of the rule to select a High Rise Rack has always to be an index between 1 and 2. The left High Rise Rack is always assigned by index 1. The right High Rise Rack is always assigned by index 2. Be sure that the resulting index refers to an existing High Rise Rack.

- *Row*
  Label to be executed when order strategy nr 5 or 6 is the active strategy. There are a number of rules predefined to select the index of the row, but you can also create your own rule via a 4DScript expression.

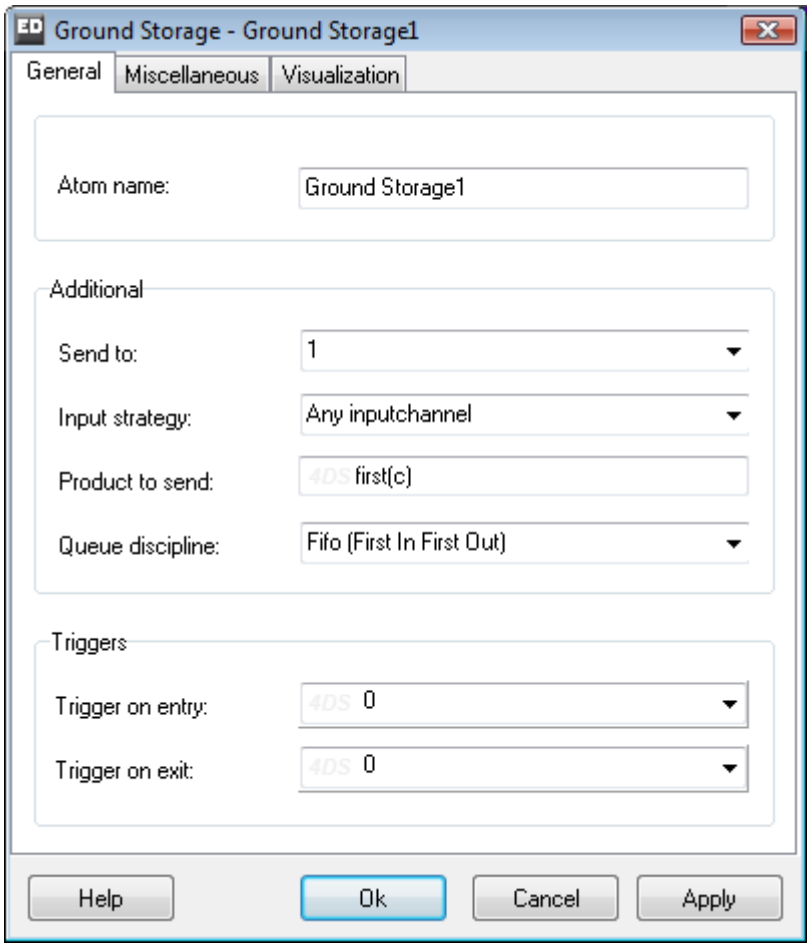
- *Column*
  Label to be executed when order strategy nr 5 or 6 is the active strategy. There are a number of rules predefined to select the index of the column, but you can also create your own rule via a 4DScript expression.

Note: The result of the rule to select a row or column has always to be > 0 and within the dimensions of your High Rise Racks. Be sure that the resulting index of row and column refers to a non assigned/occupied and available (not -1) bin location. Non occupied bin location of the High Rise Rack tables always has the value 0.



Picture 16-9: The Advanced ASRS 2D Model Layout

17 THE GROUND STORAGE ATOM



Picture 17-1: The Ground Storage atom

The Ground Storage atom stores product atoms that enter at a specific location. If the Ground Storage is full its input is closed. Each storing position (a storing position is treated as a cell in the internal table) can contain one product. Specific storing positions can be deactivated to simulate e.g. columns in a building. Just click on the button 'Edit Table' and write '-1' in the cell which you want to deactivate. Storing positions are represented as cells in that table. If the Ground Storage has output channels, products are sent out. You can specify which product to send out.

- *Atom Name*
  The name of the atom.

- *Send to*
  A rule that determines to which output channel a product atom needs to be sent. There are a number of rules predefined (see also Send to), but you can also create your own rule via a 4DScript expression.

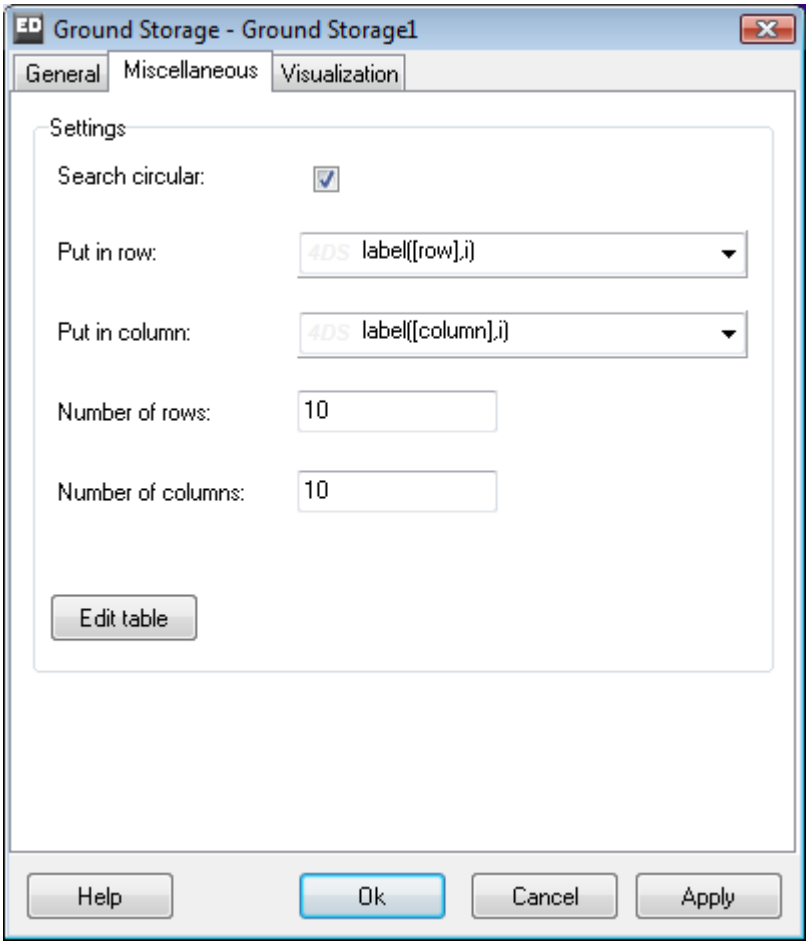
- *Input strategy*
  A rule that determines to which output channel a product atom needs to be sent. There are a number of rules predefined (see also Send to), but you can also create your own rule via a 4DScript expression.

- *Product to send*
  When the Ground Storage is ready to send a new product, this code will be evaluated to see which product should be send. The sequence of the products depends on the Queue Discipline.

- *Queue Discipline*
  When a product enters, this code will specify the way products are ranked in the Ground Storage. This is not related to the place (row and column) of the product in the warehouse.

- *Trigger on entry*
  A rule that determines what kind of action needs to be executed when a product atom enters the Ground Storage atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule by writing a 4DScript expression.

- *Trigger on exit*
  A rule that determines what kind of action needs to be executed when a product atom exits the Ground Storage atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule using 4DScript.



Picture 17-2: The Ground Storage bin location parameters

- *Search circular*
  The collision check can be turned on and off here. When a product enters the atom and the check is on, it is checked if the desired storing place is free. If it is full, it is checked in increasing circles around the desired storing place if a free one can be found. If the check is off and the desired storing place is full, the next free storing place is taken. If you want to have a nicer visualization, you can make the collision check before the product enters the Ground Storage atom. You have to call the function 'GroundStorage__SearchForAFreeStoragePlace(AtomByName([Ground Storage1], Model), i)' on the atom that is connected to the 'Ground Storage'. The function expects the labels 'row' and 'column' on the product (i). They need to be set before you call the function.

- *Important*
  If you use the above function, checking or un-checking the Control 'Search circular' has no effect. This is because the collision check happens in the function and the function is called before the product enters the 'Ground Storage'.

- *Put in row*
  It can be defined in which row an incoming product should be placed with a

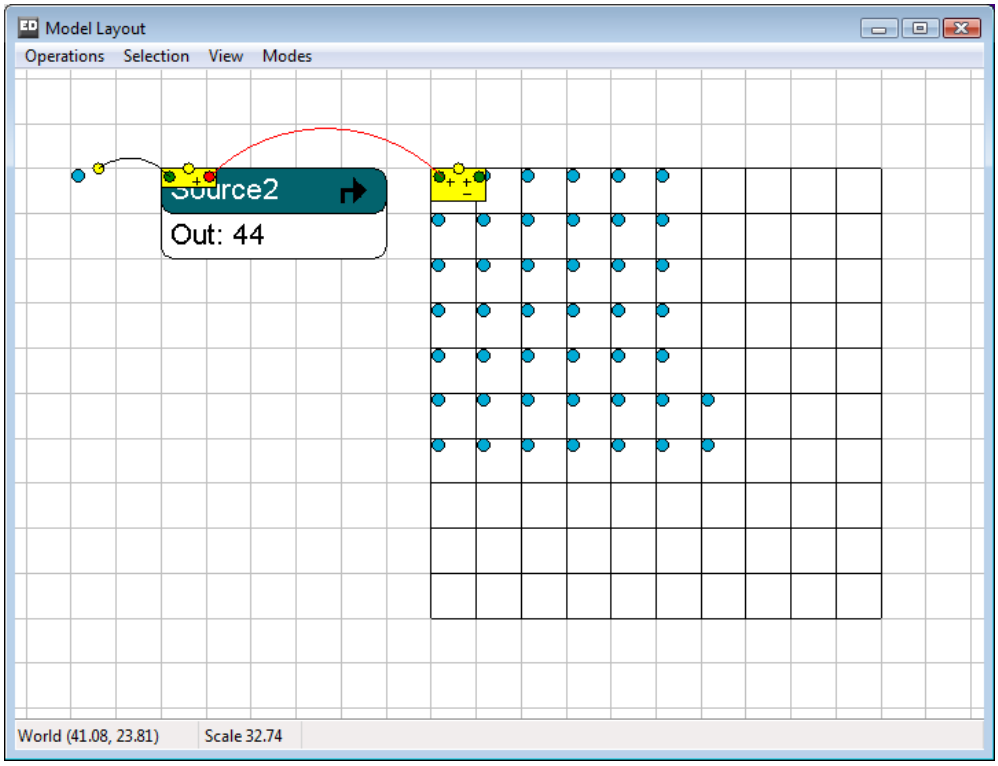
value or an expression here. If this row is completely full, the next row is chosen. If the value is 0, the search for an empty storing place (cell) starts always with row 1.

- *Put in column*
  It can be defined in which column an incoming product should be placed. Values or expressions are allowed here. If this column is completely full, the next column is chosen. If the value is 0, the search for an empty storing place (cell) starts always with column 1.

- *Number of rows*
  This is the number of locations in vertical direction in your Ground Storage.

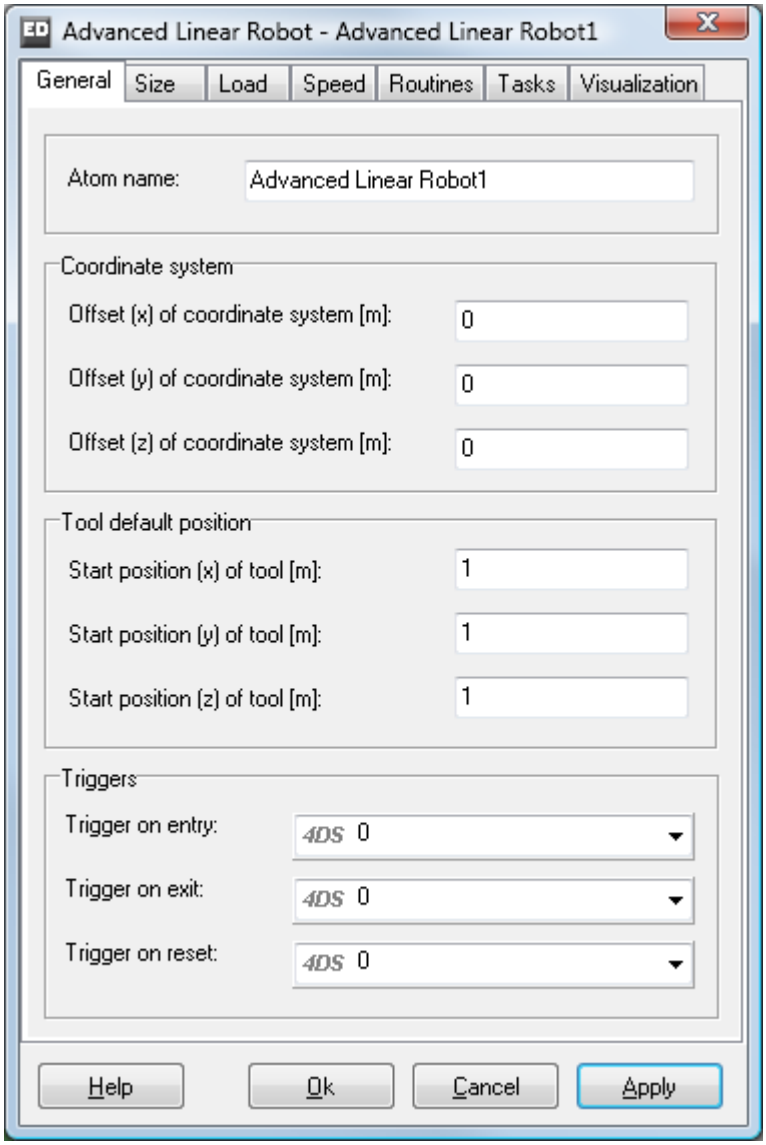
- *Number of columns*
  This is the number of locations in horizontal direction in your Ground Storage.

- *Edit table*
  This will open a window with a table in which you can view the content of the Ground Storage. The ID numbers of the atoms in the Ground Storage are displayed in the table. You can deactivate storing places by writing -1 in the cells.



Picture 17-3: The Ground Storage 2D Model Layout

18 THE ADVANCED LINEAR ROBOT ATOM



Picture 18-1: The Advanced Linear Robot atom

An Linear Robot is a handling tool for any kind of pick-, place- and motion- tasks. The sphere of action is limited by the length, width and height of workspace. Motions are defined based on a world coordinate system. The destinations of the Grabber (tool) are defined with xyz-coordinates. Any kind of action, like loading, unloading, moving and delays are defined within Routines. Those Routines are stored with global references, so they can be used by other Linear Robots as well. Every kind of command has its specific parameters like load time or speed to define the behaviour of the robot. There are two ways to define routine motion commands. First one is to move the tool of the robot by hand and then get the coordinates in order to teach the command into the routine. Second

one is the other way around when the robot is assigned to drive to user edited coordinates. If the destination has been reached successfully, the coordinates can be integrated into a motion command. Routines can be assigned to tasks. Routine execution is triggered if there is a task available. Strategies can be defined to decide which routine has to be executed, for what kind of task. If a Routine has been started it executes all commands in sequence before it becomes idle again. Due to high level of danger Advanced Linear Robot (ALR) systems are driven without human interference.

Important: There a two ways to use the Advanced Linear Robot. For any kind of pick and place tasks, the robot channels have to be connected to previous and following atoms to make it part of the material flow. Motions without material handling do not need channel connections, but the Routines need to be started by external call.

Note: This atom is part of the ED Logistics version of Enterprise Dynamics. However the ED Logistics version of Enterprise Dynamics edition also has a more simplified and therefore faster Robot atom.

- *Atom Name*
  The name of the atom.

- *Offset (x/y/z) of coordinate system [m]*
  All motions of Advanced Linear Robot are determined based on the origin of its coordinate system. Per default it is located in the rotation center of the robot socket at z-level zero. Edit offsets to shift that point to a more comfortable location, before starting routine definition.

Note: It can be helpful to lock the position of the robot to protect it from unintended dislocation.

- *Start position (x/y/z) of tool [m]*
  Default position of the Grabber center relating to the coordinate system. The position is set on reset.

Note: Choose a start position that can be reached according to robot geometry and sizes.

- *Trigger on entry*
  Determines what kind of action needs to be executed when a Product atom enters the Advanced Linear Robot atom. There are a number of rules predefined (see also Trigger on exit), but you can also create your own rule via a 4DScript expression.

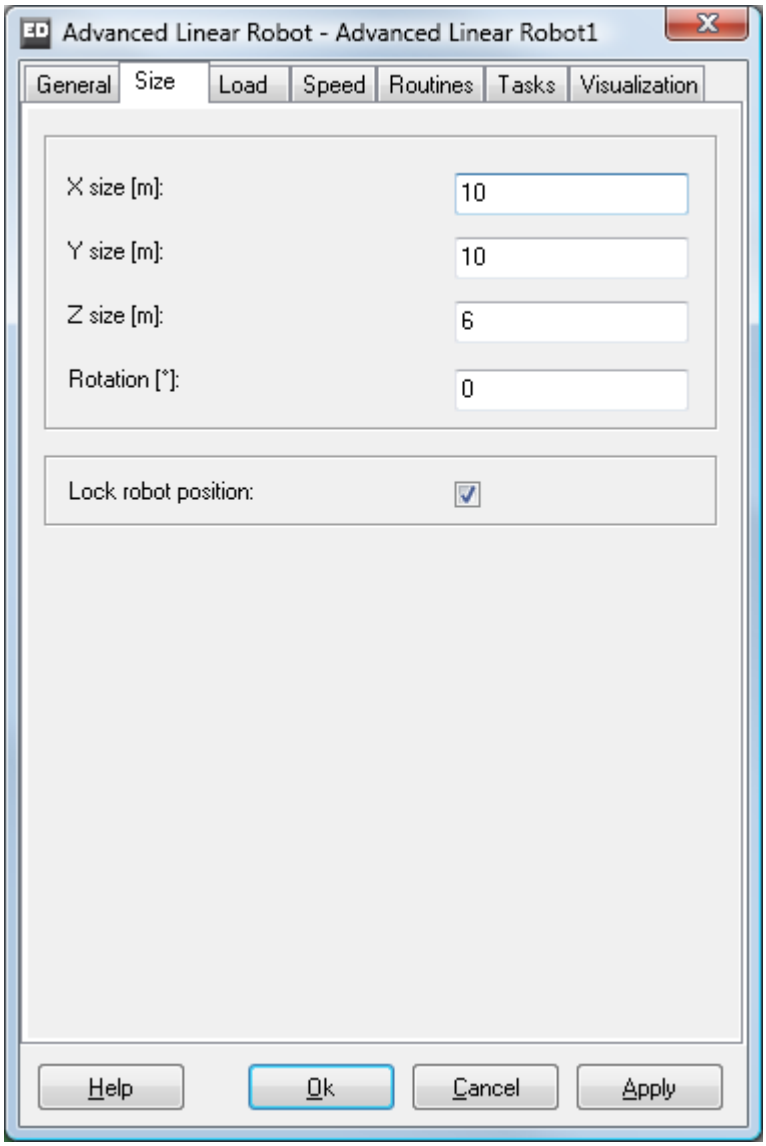
Note: The trigger on entry is executed from the Grabber sub-atom, not from the Advanced Linear Robot atom itself.

- *Trigger on exit*
  Determines what kind of action needs to be executed when a Product atom exits the Advanced Linear Robot atom. There are a number of rules predefined (see also Trigger on entry), but you can also create your own rule via a 4DScript expression.

Note: The trigger on exit is executed from the Grabber sub-atom, not from the Advanced Linear Robot atom itself.

- *Trigger on reset*
  Determines what kind of action needs to be executed when the Advanced Linear Robot atom is reset. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

Note: The trigger on reset is executed from the Grabber sub-atom, not from the Advanced Linear Robot atom itself.



Picture 18-2: The Advanced Linear Robot size parameters

Design of robots is defined by the length of slideways. The Advanced Linear Robot atom has three axis, (x/y/z) and one rotation joint. The Grabber rotates around the vertical axis (z).

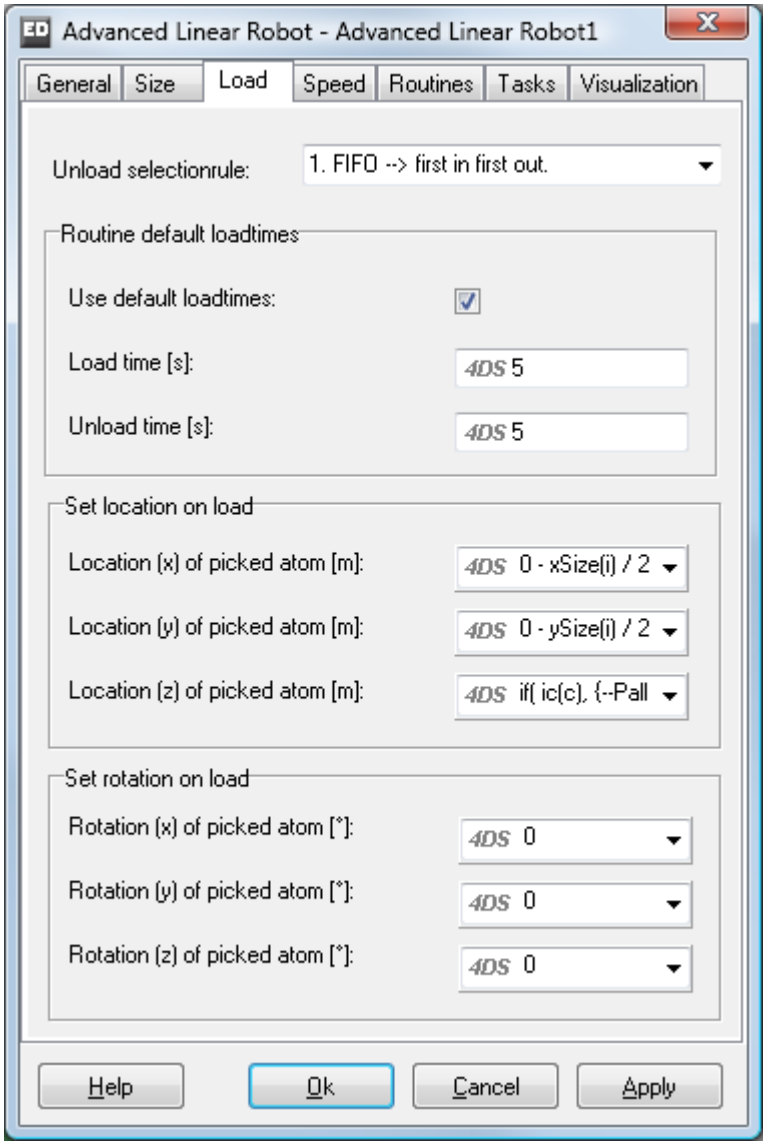
- *X size [m]*
  The length of the main driving slideway (x axis). Also defines the x dimension of the robot.

- *Y size [m]*
  The length of the second driving slideway (y axis). Also defines the y dimension of the robot.

- *Z size [m]*
  The length of the third driving slideway (z axis). Also defines the height of the robot.

- *Rotation [°]*
  At default the Advanced Linear Robot is positioned from left to right on your screen. However you can also rotate the atom to display the flow of atoms over the Advanced Linear Robot matches reality.

- *Lock robot position*
  If this box is checked the position of the robot is locked to protect it from unintended dislocation.



Picture 18-3: The Advanced Linear Robot load parameters

- *Unload selectionrule*
  This option lets unload the picked atoms in a specified sequence. The internal queue of the Advanced Linear Robot is sorted in the selected sequence before unloading. Default is first in first out (FIFO). There are 6 strategies available.

  1: FIFO --> first in first out
  2: LIFO --> last in first out
  3: Label minimum (picked atoms)
  4: Label maximum (picked atoms)
  5: Icon index minimum (picked atoms)
  6: Icon index maximum (picked atoms)

Note: Picked atoms are stored in the sub-atom Grabber of Advanced Linear Robot.

- *Use default load times*
  If this box is checked the default load and unload times are directly assigned to Routine editfields in order to reduce the number of edit parameters.

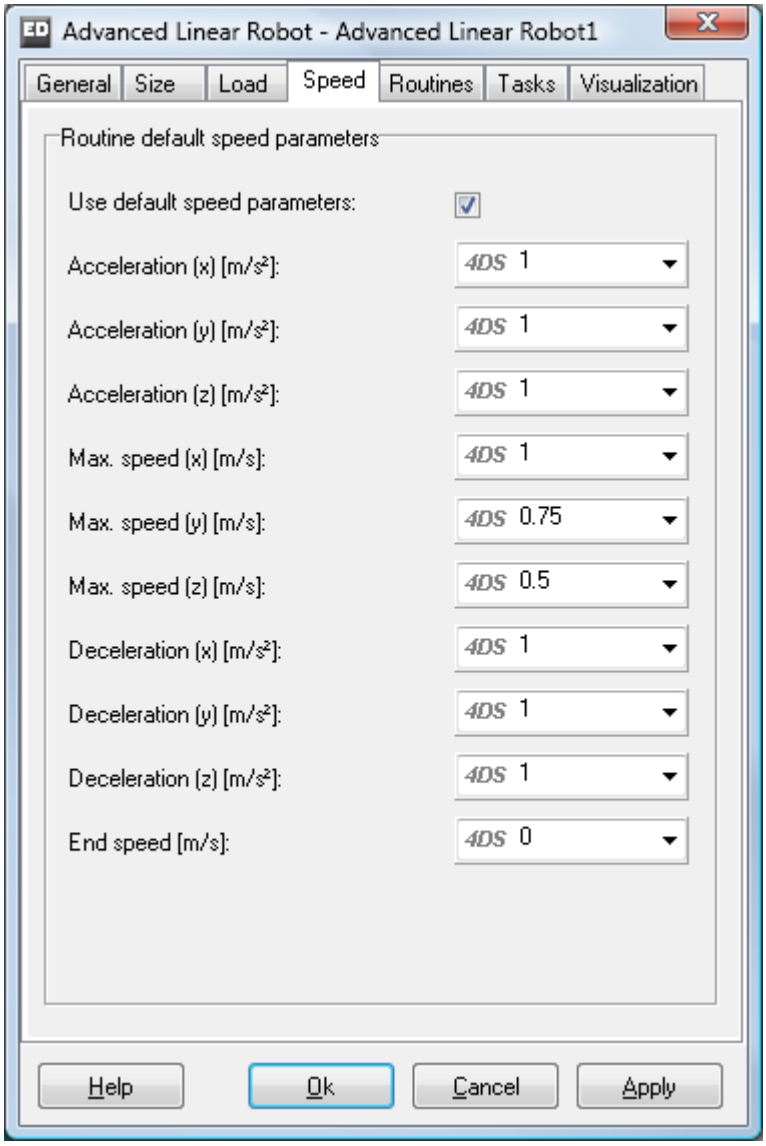
Note: Use this parameter for simplified and fast Routine definition. Uncheck for detailed un-/loadtimes.

- *Load time [s]*
  The default time (s) that is needed to load something to Advanced Linear Robot. Assigned to the Routine load command when checkbox use default load times is checked.

- *Unload time [s]*
  The default time (s) that is needed to unload something from Advanced Linear Robot. Assigned to the Routine unload command when checkbox use default load times is checked.

- *Location (x/y/z) of picked atom [m]*
  For picked atoms you can define location offsets for each individual x, y, and z.

- *Rotation (x/y/z) of picked atom [m]*
  For picked atoms you can define rotation offsets around each axis individual x, y, and z.



Picture 18-4: The Advanced Linear Robot speed parameters

- *Use default speed parameters*
  If this box is checked the default speed parameters are directly assigned to routine editfields in order to reduce the number of edit parameters.

Note: Use this parameter for simplified and fast Routine definition. Uncheck for detailed speed definitions.

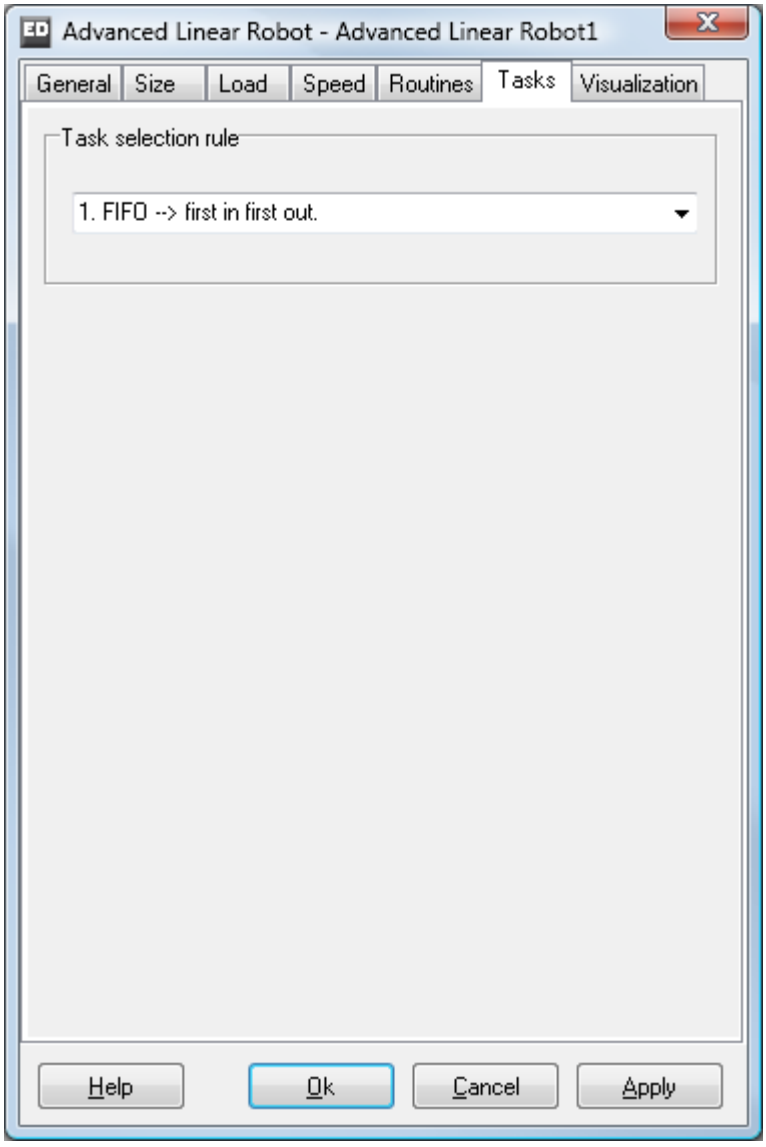
- *Acceleration (x/y/z) [m/s²]*
  Acceleration of Grabber center in relation to origin of coordinate system.

- *Max. speed (x/y/z) [m/s]*
  Maximum speed of Grabber center in relation to origin of coordinate system.

- *Deceleration (x/y/z) [m/s²]*
  Deceleration of Grabber center in relation to origin of coordinate system.

- *End speed [m/s]*
  End speed of Grabber center in relation to origin of coordinate system.

Note: The Advanced Linear Robot uses the ‘MovingTo’ command to define the motions between the origin of the coordinate system and the center of the Grabber. In ‘MovingTo’ the acceleration and deceleration are leading. The end speed will be tried to reach, but when this is impossible the end speed is the last speed taking acceleration and deceleration into account.



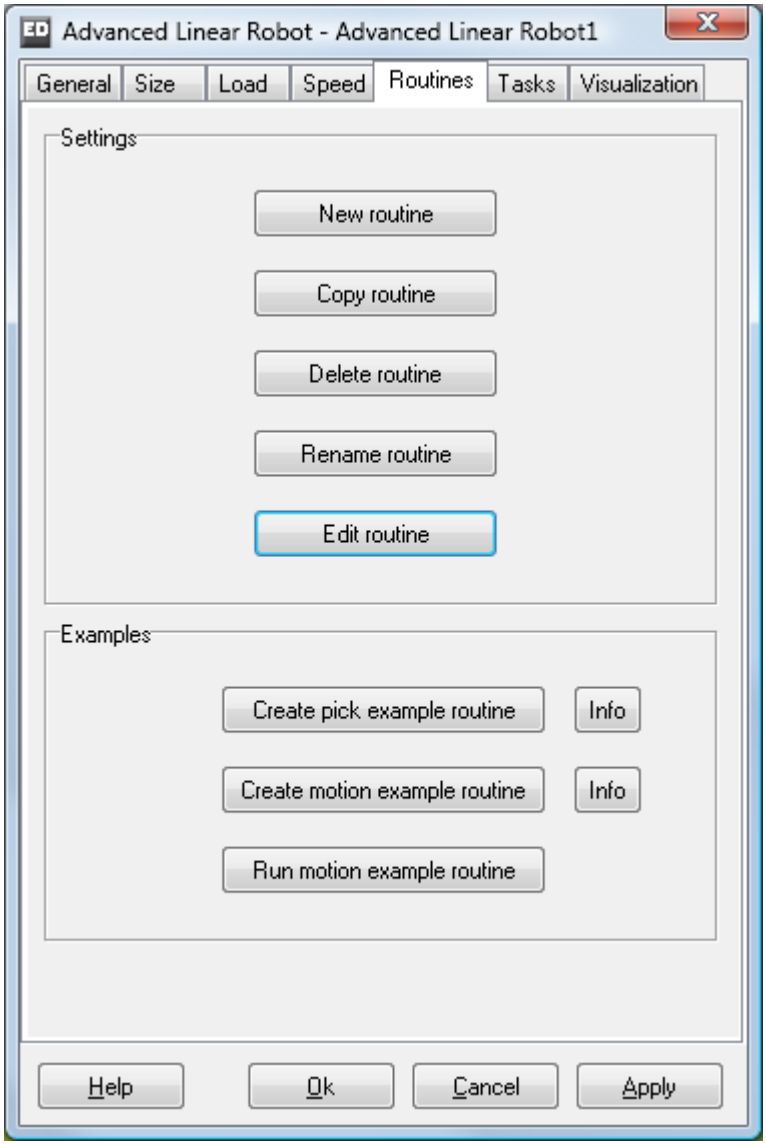
Picture 18-5: The Advanced Linear Robot task parameters

Note: Tasks are calls to the Advanced Linear Robot to execute a specified Routine. There are two ways to create a Task. Pick and place Task (material handling) are created every time an atom connected to an input channel of the robot causes an IcReady. Motion Tasks (welding, painting, etc.) are created if there is an external call created with the ‘Advanced_Linear_Robot_create_motion_task‘ command. If the robot is busy, the Tasks

are stored until the robot is idle again. The Tasks selection rule determines what Tasks has to be done next.

- *Task selection rule*
  This option lets execute the Tasks in a specified sequence. The internal queue of the Advanced Linear Robot is sorted in the selected sequence before choosing the next Tasks to execute. Default is first in first out (FIFO). There are 10 strategies available.

  1: FIFO --> first in first out
  2: LIFO --> last in first out
  3: Content minimum --> atoms in the container with the smallest content first
  4: Content maximum --> atoms in the container with the largest content first
  5: IC minimum --> lowest input channel of robot first
  6: IC Maximum --> highest input channel of robot first
  7: Label minimum (atom) --> the atom with the lowest value on label first
  8: Label maximum (atom) --> the atom with the highest value on label first
  9: Label minimum (container) --> atoms in the container with the lowest value on label first
  10: Label maximum (container) --> atoms in the container with the highest value on label first



Picture 18-6: The Advanced Linear Robot routine parameters

Note: Routines are sub atoms of the Advanced Linear Robot. Storing all information about repeatable sequences into tables, the behaviour becomes reproducible in order to execute frequently necessary actions like motions, loading, unloading or waiting.

- *New Routine*
  Click this button to create a new Routine. You will be asked to enter a Routine name. If the Routine has been created successfully, you are asked to edit commands to the Routine table by using the edit commands GUI.

Note: The global reference to the Routine will be registered too. Be sure, that the Routine is not already defined for any other Advanced Linear Robot.

- *Copy Routine*
  Click this button to copy a existing Routine with all its commands. Select the sample Routine via atom selector. If selection was successful, you will be asked

to enter a different Routine name. If the Routine has been created successfully, you are asked to edit Commands to the Routine table by using the edit routines GUI.

- *Delete Routine*
  Click this button to delete a existing Routine with all its table-stored Commands.

Note: The global reference to the Routine will be unregistered too. Be sure, that the Routine is not needed by any other Advanced Linear Robot.

- *Edit Routine*
  Click this button to select the Routine you want to edit via atom selector. If selection was successfully, you are able to edit Commands to the Routine table by using the Advanced Linear Robot Edit Routines GUI.

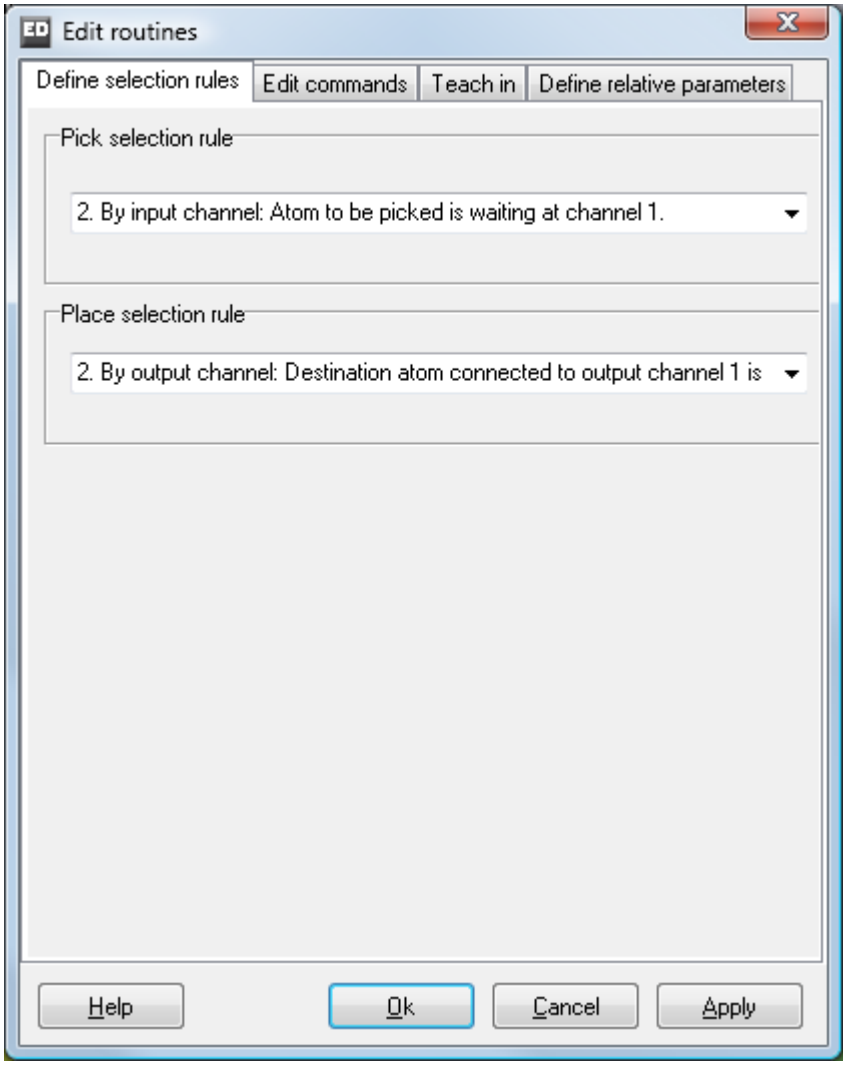
- *Create pick example Routine*
  Click this button to create a Pick Routine example with the name 'Routine_pick_example'. If the Routine has been created successfully, you are asked to edit Commands to the Routine table by using the edit commands GUI.

Warning!: Pick example Routine only works if input channel 1 of tool is connected to pick atom and output channel 1 of sub-atom Grabber is connected to place atom.

- *Create motion example Routine*
  Click this button to create a Motion Routine example with the name 'Routine_motion_example'. If the Routine has been created successfully, you are asked to edit Commands to the Routine table by using the edit commands GUI.

- *Run motion example Routine*
  Click this button to run the 'Routine_motion_example' routine.

- *Edit Routine*
  Click this button to select the Routine you want to edit via atom selector. If selection was successfully, you are able to edit Commands to the Routine table by using the Advanced Linear Robot Edit Routines GUI.



Picture 18-7: The selection rule definition at the routine editor

Any kind of action like loading, unloading, moving, delays are defined within Routines. The Advanced Linear Robot Edit Routines GUI is the tool to apply, insert, copy, delete and edit Routine Commands. It is also possible to define selection rules for the Routines, so the Robot is able to select the Routine that fits best to the current Task. The Teach In functionality allows to create motions in an easy way.

Note: Some of the parameters used for the Edit Routines GUI are defined within the Advanced Linear Robot GUI.

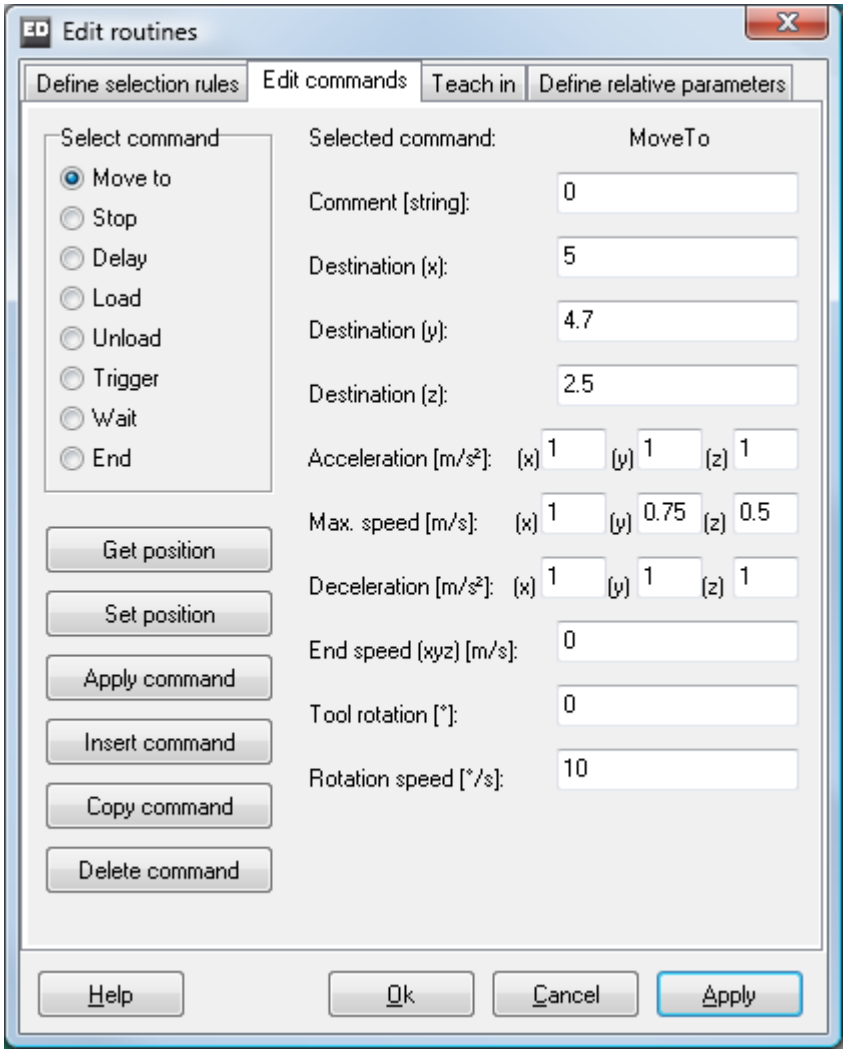
- *Pick selection rule*
  Determines the Pick condition to select the Routine. The Routine is selected, if the selected statement is true. There are 16 strategies available.
  1: No pick restriction or motion routine
  2: By input channel
  3: By pick atom name
  4: By pick icon name
  5: By pick icon number
  6: By pick label value (direct)

7: By pick label value (conditional)
8: By pick label text
9: By pick location (direct)
10: By pick location (conditional)
11: By pick size (direct)
12: By pick size (conditional)
13: By pick content (direct)
14: By pick content (conditional)
15: Conditional statement
16: By user

Note: Choose a Pick selection rule if there is more than one Routine available and the selection criteria can be found via input channel connection. The Pick selection rule can be combined with a Place selection rule to define advanced strategies.

- *Place selection rule*
  Determines the Place condition to select the Routine. The Routine is selected, if the selected statement is true. There are 11 strategies available.
  1: No place restriction or motion routine
  2: By output channel
  3: By output channel fail safe
  4: By place location (direct)
  5: By place location (conditional)
  6: By place size (direct)
  7: By place size (conditional)
  8: By place content (direct)
  9: By place content (conditional)
  10: Conditional statement
  11: By user

Note: Choose a Place selection rule if there is more than one Routine available and the selection criteria can be found via output channel connection. The Place selection rule can be combined with a Pick selection rule to define advanced strategies.



Picture 18-8: The Advanced Linear Robot command editor

- *Select command*
  Select between the predefined kinds of Commands to define robot behavior. There are 8 predefined Commands available:
  1: Move to
  Defines any kind of robot motion
  2: Stop
  Stops any kind of robot motion
  3: Delay
  Makes the robot waiting for a specific time
  4: Load
  Executes picking of incoming products
  5: Unload
  Executes placing of outgoing products
  6: Trigger
  Allows the execution of 4DScript code
  7: Wait
  Let the robot wait for an external trigger to continue

8: *End*
Finishes a routine

Choose a Command to edit the necessary parameters. The number and kind of parameters depends on the type of selected Command.

Move to command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Destination (x/y/z)*
  Tool center destination according to the origin of the coordinate system.

- *Acceleration [m/s²]*
  The tool centre acceleration to reach the destination.

- *Max. speed [m/s]*
  The tool centre maximum speed to reach the destination.

- *Deceleration [m/s²]*
  The tool centre deceleration to reach the destination.

- *End speed [m/s]*
  The Tool centre end speed to reach the destination.

- *Tool arm deflection [°]*
  Absolute deflection of the tool arm at the destination.

- *Deflection speed [°/s]*
  The deflection speed to reach the absolute Tool arm deflection.

- *Tool arm rotation [°]*
  Absolute rotation of the Tool arm at the destination.

- *Rotation speed [°/s]*
  The rotation speed to reach the absolute Tool arm rotation.

Note: You are able to edit detailed speed parameters, if the default speed parameters checkboxes is selected at the Advanced Linear Robot GUI.

Note: The Advanced Linear Robot uses the ‘MovingTo’ Command to define the motions between the origin of the coordinate system and the centre of the Tool. In ‘MovingTo’ the acceleration and deceleration are leading. The end speed will be tried to reach, but when this is impossible the end speed is the last speed taking acceleration and deceleration into account.

Note: You can get the current position of the Tool into the edit fields by click on the Get position button.

Note: You can set the position of the Tool according to the edit fields by click on the Set position button.

Stop command:

- *Comment*
  Comment the Commands to make the Routines more readable.

Delay command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Time [s]*
  The delay before the Routines is continued with the next command (row).

Load command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Time [s]*
  The load time of the atom to be picked.

- *Channel [index]*
  The input channel of the robot, where the atom is waiting.

Note: You are able to edit detailed load parameters, if the default load parameters checkboxes is selected at the Advanced Linear Robot GUI.

Unload command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Time [s]*
  The unload time of the atom to be placed.

- *Channel [index]*
  The output channel of the robot, where the atom has to be sent to.

Note: You are able to edit detailed unload parameters, if the default load parameters checkboxes is selected at the Advanced Linear Robot GUI.

Trigger command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Trigger [4DScript]*
  The code that has to be executed before the Routine is continued with the next Command. Select between the predefined Commands defined in the kernel function ‘getTriggerEnterExit’ or enter your own 4DScript code.

Wait command:

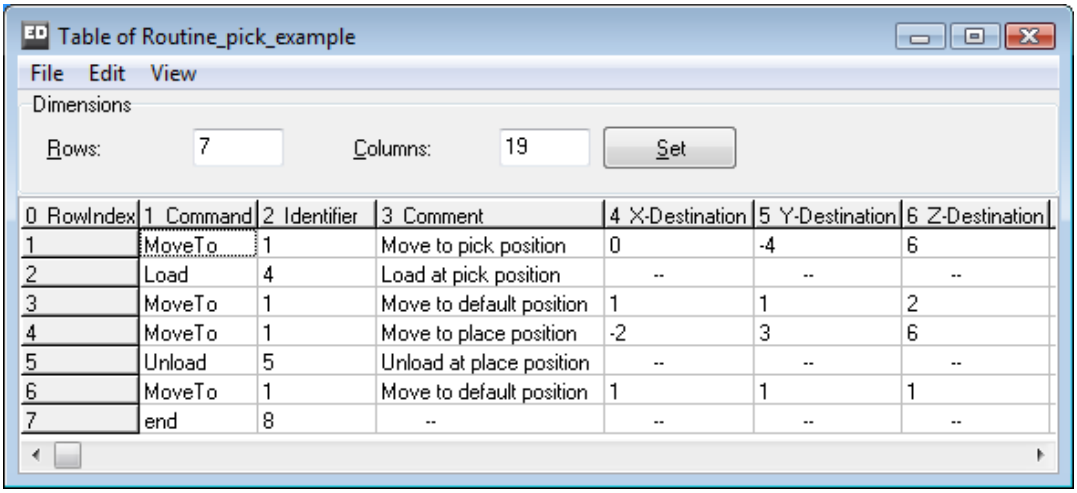
- *Comment*
  Comment the Commands to make the Routines more readable.

- *Awaited message [string]*
  The text message that has to arrive in the OnMessage event handler of the Advanced Linear Robot before the Routine is continued with the next Command.

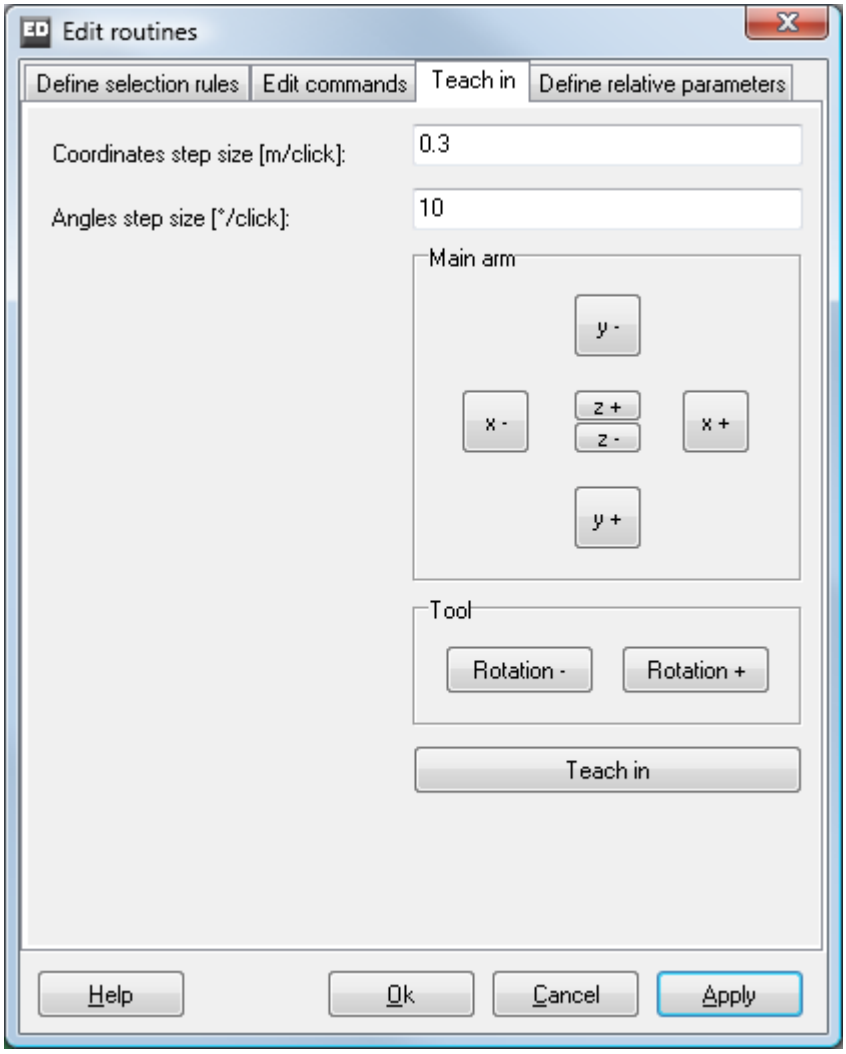
End command:

If you select the End Command no parameters can be changed, but another row is added to the Routine in order to set a flag, that that there will be no further Commands. The Robot is set to status Idle and it searches for other Tasks available.

There are four buttons to apply, insert, copy and delete Commands. On click you are asked to edit the row index of the Command that you want to apply, insert, copy or delete.



Picture 18-9: Routine example (pick and place)



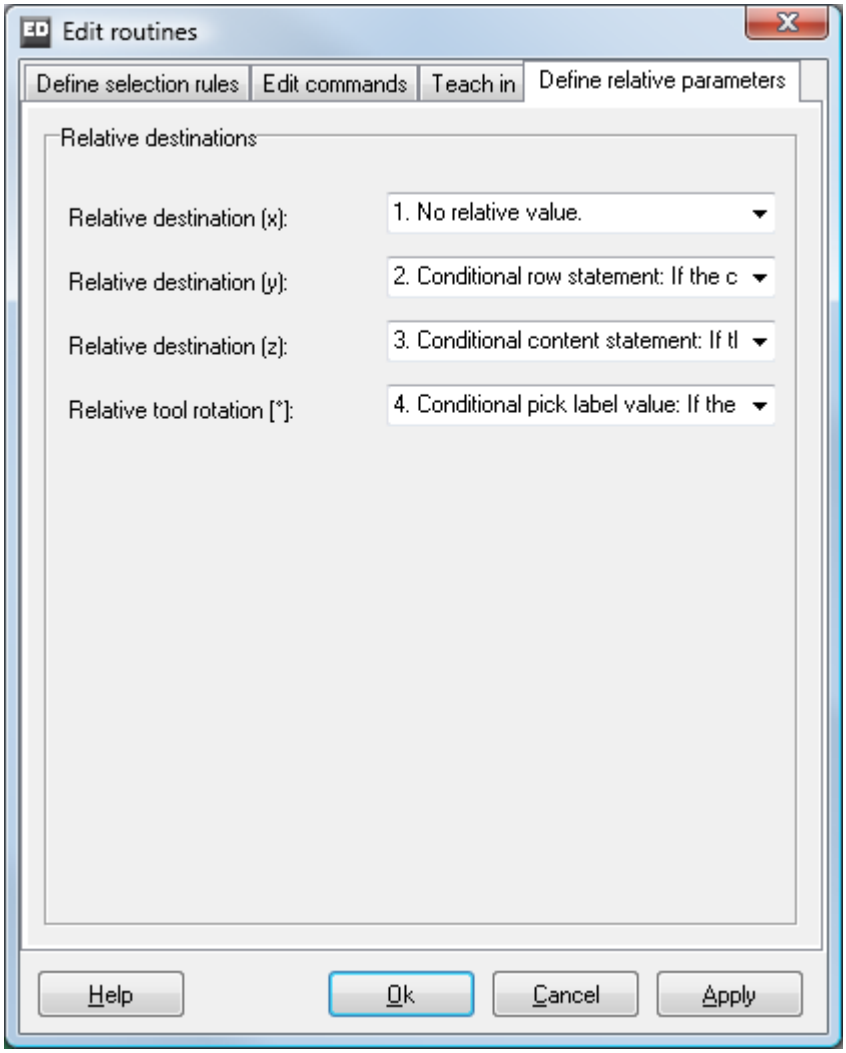
Picture 18-10: Routine teach panel (move to command)

The Teach In functionality allows to create motion commands in an easy way.

- *Coordinates step size [m/click]*
  On click of main arm buttons the tool position will move according to the coordinates step size.

- *Angles step size [°/click]*
  On click of tool arm buttons the tool deflection will rotate according to the angles step size. On click of tool buttons the tool will rotate according to the angles step size.

- *Teach in button*
  On click all motion parameters are updated into the edit fields of the Edit Command page and the destination is stored to the active routine as Move to Command.



Picture 18-11: Relative destinations panel

- *Relative destination (x)*
  The tool (Grabber) relative x destination.

- *Relative destination (y)*
  The tool (Grabber) relative x destination.

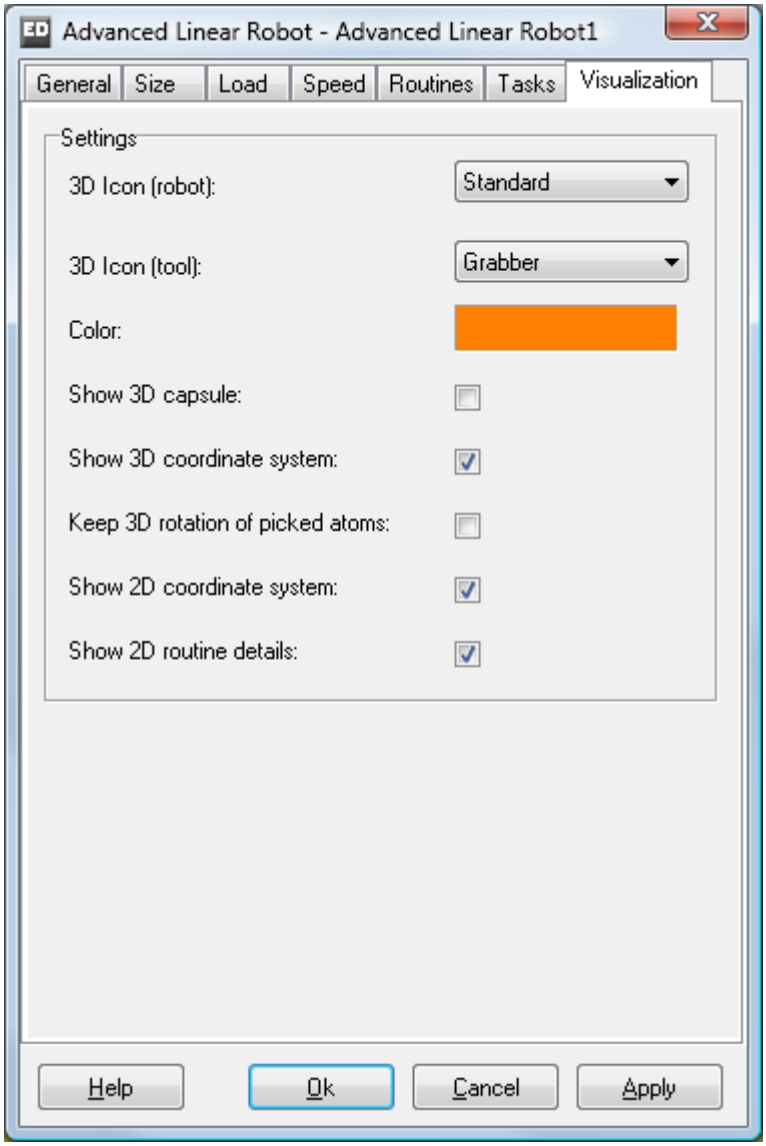
- *Relative destination (z)*
  The tool (Grabber) relative x destination.

- *Relative tool rotation [°]*
  Relative rotation of the tool (Grabber) arm at the destination.

There are 5 predefined relative value definitions available:

1: No relative value – the parameters from the Routines table are used on any account
2: Conditional row statement – cases are defined based on the current row index of the active Routine

3: *Conditional content statement - cases are defined based on the current content of the tool (Grabber)*
4: *Conditional pick label value statement - cases are defined based on Label values of Products waiting at input channel of the tool (Grabber)*
5: *User defined statement – define your own statement using 4DScript*



Picture 18-12: Visualization panel

- *3D Icon (robot)*
  A 3D icon that can be used to visualize the Advanced Linear Robot in 3D. Change this icon to switch between standard visualization and other customized 3D icons.

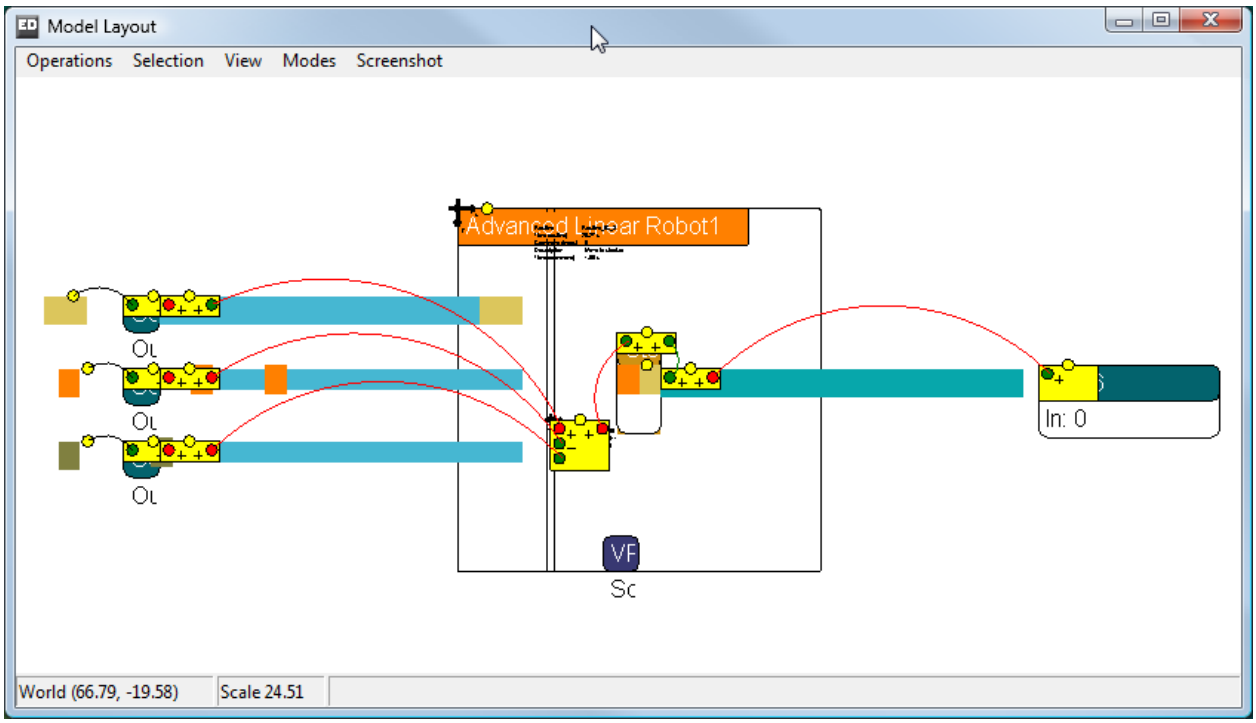
- *3D Icon (tool)*
  A 3D icon that can be used to visualize the Grabber in 3D. Change this icon to switch between grabber, welding head and other customized 3D icons.

- *Color*
  The color of Advanced Linear Robot.

- *Show 3D coordinate system*
  If this box is checked the origin of the 3D coordinate system and the centre of the Grabber will be displayed to ease routine definition.

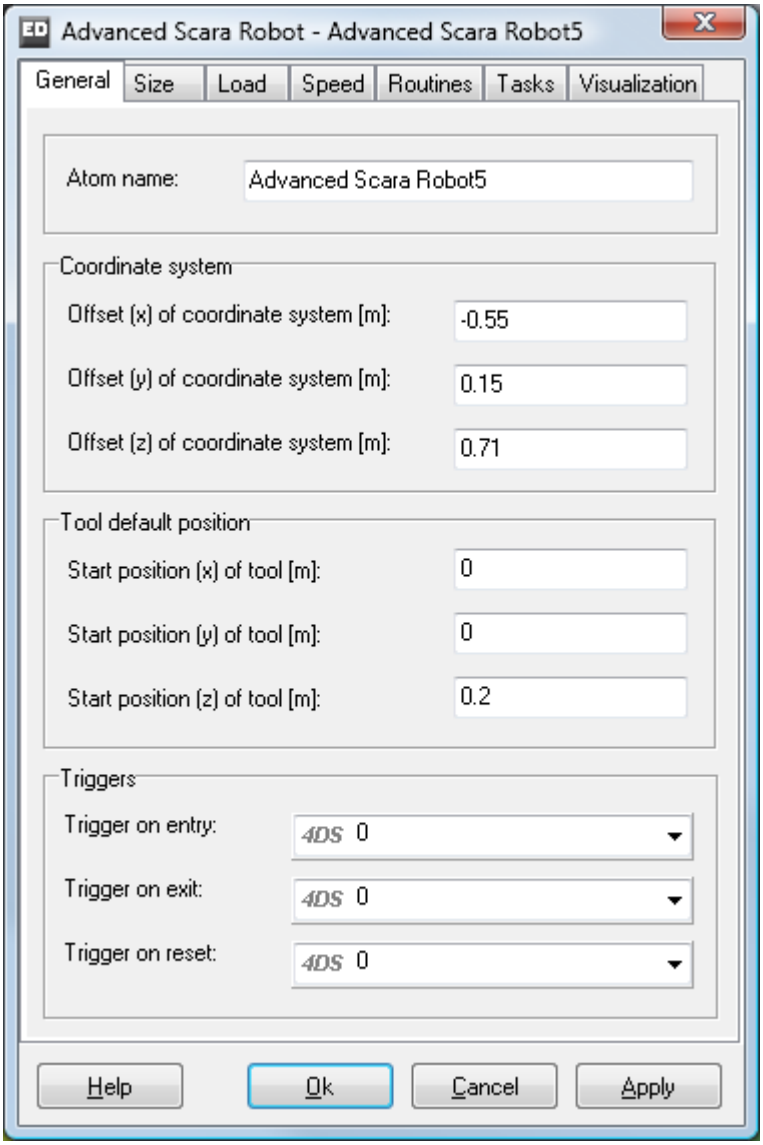
- *Show 2D coordinate system*
  If this box is checked the origin of the 2D coordinate system and the centre of the Grabber will be displayed to ease routine definition.

- *Keep rotation of picked atom*
  If this box is checked picked atoms will not be rotated according to the rotation of the robot joints, but will keep their global rotation.



Picture 18-13: 2D Layout of a running pick example

19 THE ADVANCED SCARA ROBOT ATOM



Picture 19-1: The Advanced Scara Robot atom

A Scara Robot is a selective compliance assembly robot arm for any kind of pick-, place- and motion- tasks. The sphere of action is limited by the z-size and the radius of action. A layout of at least 4 axis are necessary to reach any destination within the sphere of action. Motions are defined based on a world coordinate system. The destinations of the Tool are defined with xyz-coordinates. All kind of action like loading, unloading, moving, delays are defined within Routines. Those Routines are stored with global references, so they can be used by other Scara Robots as well. Every kind of command has its specific parameters like load time or speed to define the behaviour of the robot. There are two ways to define routine motion commands. First one is to move the tool of the robot by

hand and then get the coordinates in order to teach the command into the routine. Second one is the other way around when the robot is assigned to drive to user edited coordinates. If the destination has been reached successfully, the coordinates can be integrated into an motion command. Routines can be assigned to tasks. Routine execution is triggered if there is a task available. Strategies can be defined to decide which routine has to be executed, for what kind of task. If a Routine has been started it executes all commands in sequence before it becomes idle again. Due to high level of danger Advanced Scara Robot systems are driven without human interference.

Important: There a two ways to use the Advanced Scara Robot. For any kind of pick and place tasks, the robot channels have to be connected to previous and following atoms to make it part of the material flow. Motion without material handling do not need channel connections, but the Routines need to be started by external call.

Note: This atom is part of the ED Logistics version of Enterprise Dynamics. However the ED Logistics version of Enterprise Dynamics edition also has a more simplified and therefore faster Robot atom.

- *Atom Name*
  The name of the atom.

- *Offset (x/y/z) of coordinate system [m]*
  All motions of Advanced Scara Robot are determined based on the origin of its coordinate system. Per default it is located in the rotation center of the robot socket at z-level zero. Edit offsets to shift that point to a more comfortable location, before starting routine definition.

Note: It can be helpful to lock the position of the robot to protect it from unintended dislocation.

- *Start position (x/y/z) of tool [m]*
  Default position of the tool center relating to the coordinate system. The position is set on reset.

Note: Choose a start position that can be reached according to robot geometry and sizes.

- *Trigger on entry*
  Determines what kind of action needs to be executed when a Product atom enters the Advanced Scara Robot atom. There are a number of rules predefined (see also Trigger on exit), but you can also create your own rule via a 4DScript expression.

Note: The trigger on entry is executed from the Tool sub-atom, not from the Advanced Scara Robot atom itself.

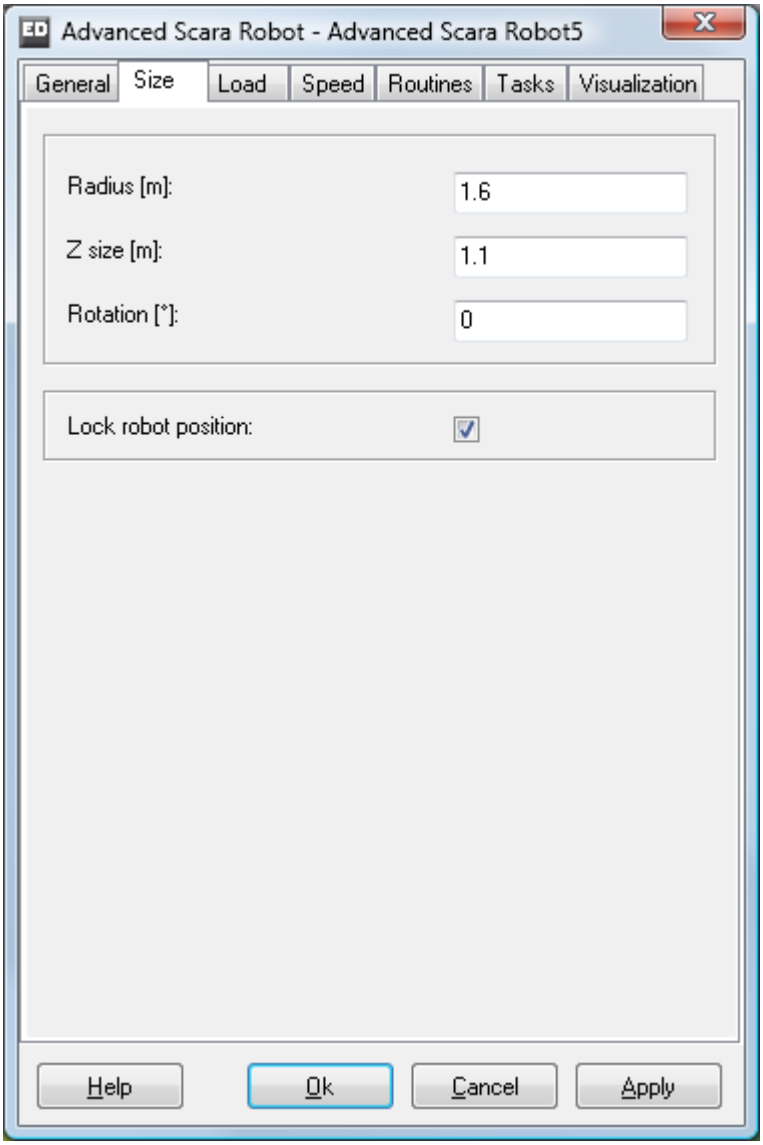
- *Trigger on exit*
  Determines what kind of action needs to be executed when a Product atom exits the Advanced Scara Robot atom. There are a number of rules predefined (see

also Trigger on entry), but you can also create your own rule via a 4DScript expression.

Note: The trigger on exit is executed from the Tool sub-atom, not from the Advanced Scara Robot atom itself.

- *Trigger on reset*
  Determines what kind of action needs to be executed when the Advanced Scara Robot atom is reset. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

Note: The trigger on reset is executed from the Tool sub-atom, not from the Advanced Scara Robot atom itself.



Picture 19-2: The Advanced Scara Robot size parameters

Design of robots is defined by number/kind of articulated joints and the length of arms. The Advanced Scara Robot atom has 4 axis. Starting with a vertically rotating socket

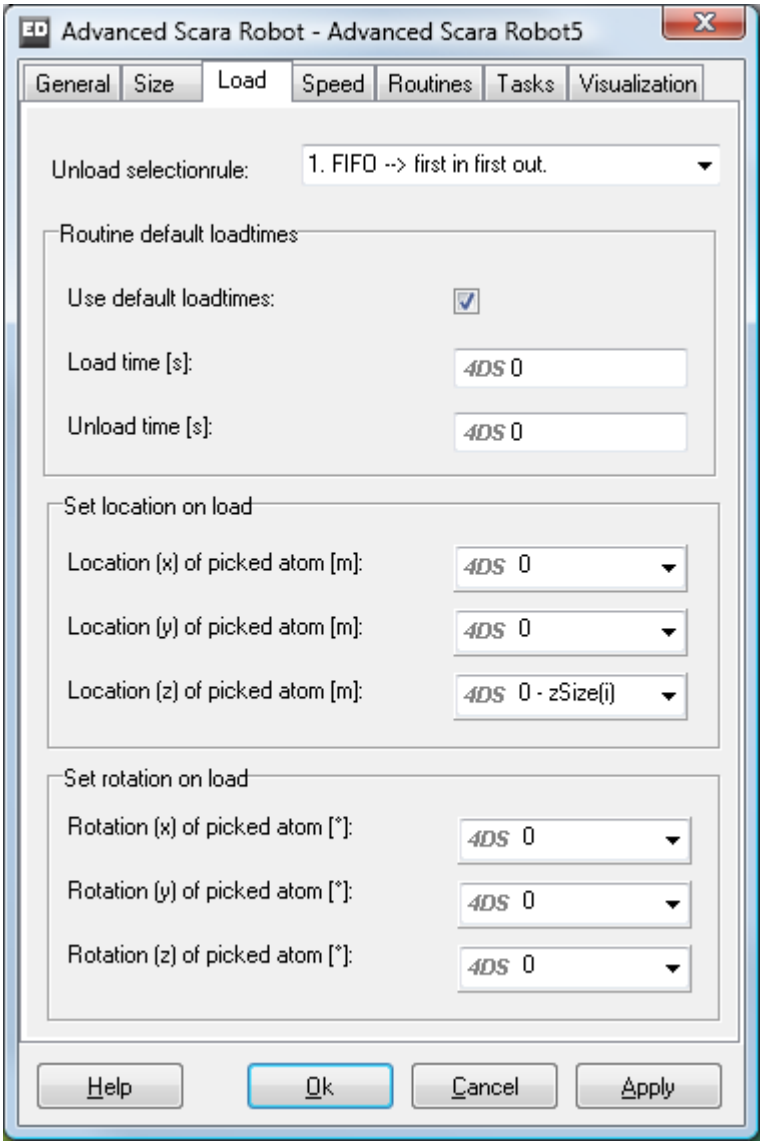
there are two vertical main arms of equal length, followed by another vertical Tool arm that finally can be rotated around its vertical center.

- *Radius [m]*
  Determines the length of the robot arms along x- and y-axis. The radius of the workspace in x/y is two times the length of each robot arm.

- *Z size [m]*
  Define the height of the workspace along the z-axis.

- *Rotation [°]*
  Determines the rotation of the robot around the z-axis.

- *Lock robot position*
  If this box is checked the position of the robot is locked to protect it from unintended dislocation.



Picture 19-3: The Advanced Scara Robot load parameters

- *Unload selectionrule*
  This option lets unload the picked atoms in a specified sequence. The internal queue of the Advanced Scara Robot is sorted in the selected sequence before unloading. Default is first in first out (FIFO). There are 6 strategies available.

  1: FIFO --> first in first out
  2: LIFO --> last in first out
  3: Label minimum (picked atoms)
  4: Label maximum (picked atoms)
  5: Icon index minimum (picked atoms)
  6: Icon index maximum (picked atoms)

Note: Picked atoms are stored in the sub-atom Tool of Advanced Scara Robot.

- *Use default load times*
  If this box is checked the default load and unload times are directly assigned to Routine editfields in order to reduce the number of edit parameters.

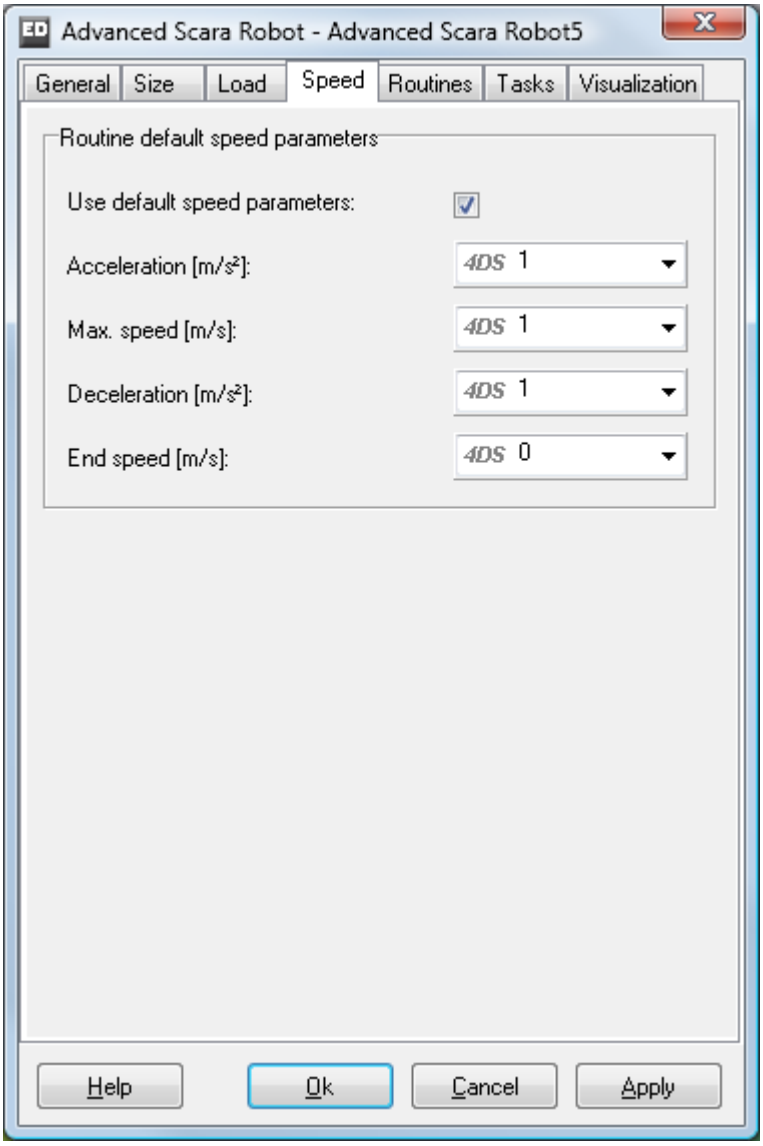
Note: Use this parameter for simplified and fast Routine definition. Uncheck for detailed un-/loadtimes.

- *Load time [s]*
  The default time (s) that is needed to load something to Advanced Scara Robot. Assigned to the Routine load command when checkbox use default load times is checked.

- *Unload time [s]*
  The default time (s) that is needed to unload something from Advanced Scara Robot. Assigned to the Routine unload command when checkbox use default load times is checked.

- *Location (x/y/z) of picked atom [m]*
  For picked atoms you can define location offsets for each individual x, y, and z.

- *Rotation (x/y/z) of picked atom [m]*
  For picked atoms you can define rotation offsets around each axis individual x, y, and z.



Picture 19-4: The Advanced Scara Robot speed parameters

- *Use default speed parameters*
  If this box is checked the default speed parameters are directly assigned to routine editfields in order to reduce the number of edit parameters.

Note: Use this parameter for simplified and fast Routine definition. Uncheck for detailed speed definitions.

- *Acceleration [m/s²]*
  Acceleration of Tool center in relation to origin of coordinate system.

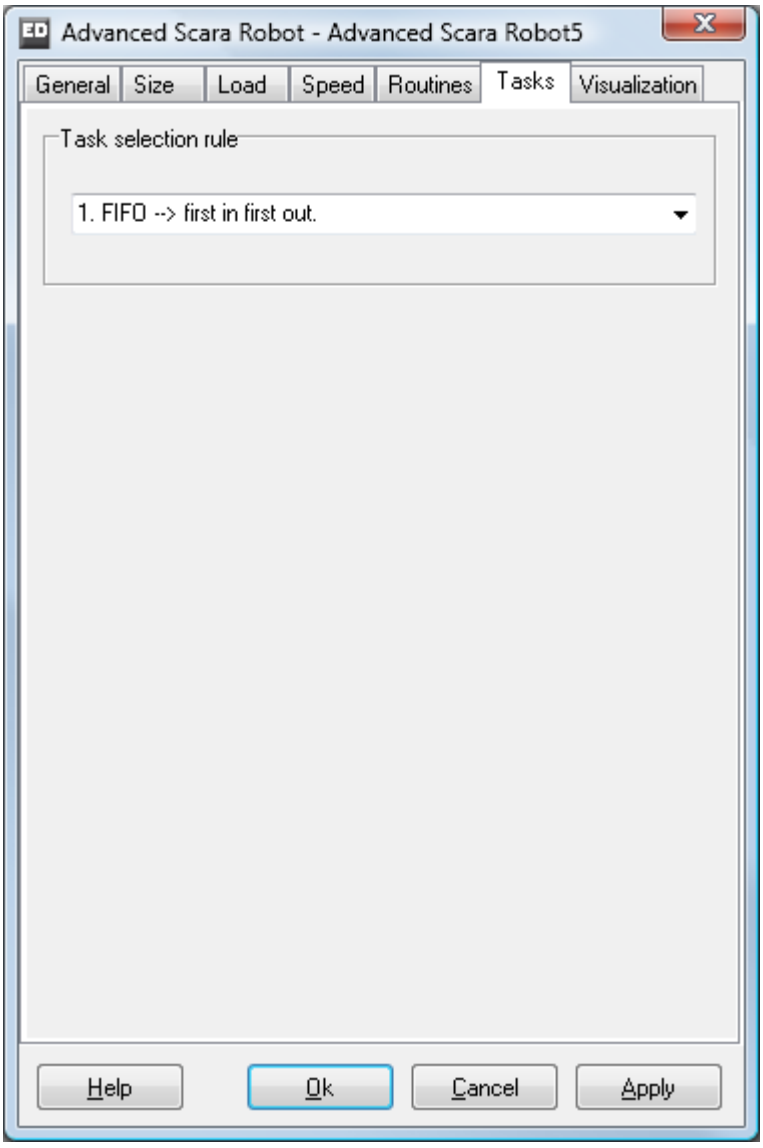
- *Max. speed [m/s]*
  Maximum speed of Tool center in relation to origin of coordinate system.

- *Deceleration [m/s²]*
  Deceleration of Tool center in relation to origin of coordinate system.

- *End speed [m/s]*
  End speed of Tool center in relation to origin of coordinate system.

Note: The Advanced Scara Robot uses the MovingTo command to define the motions between the origin of the coordinate system and the center of the Tool. In MovingTo the acceleration and deceleration are leading. The end speed will be tried to reach, but when this is impossible the end speed is the last speed taking acceleration and deceleration into account.

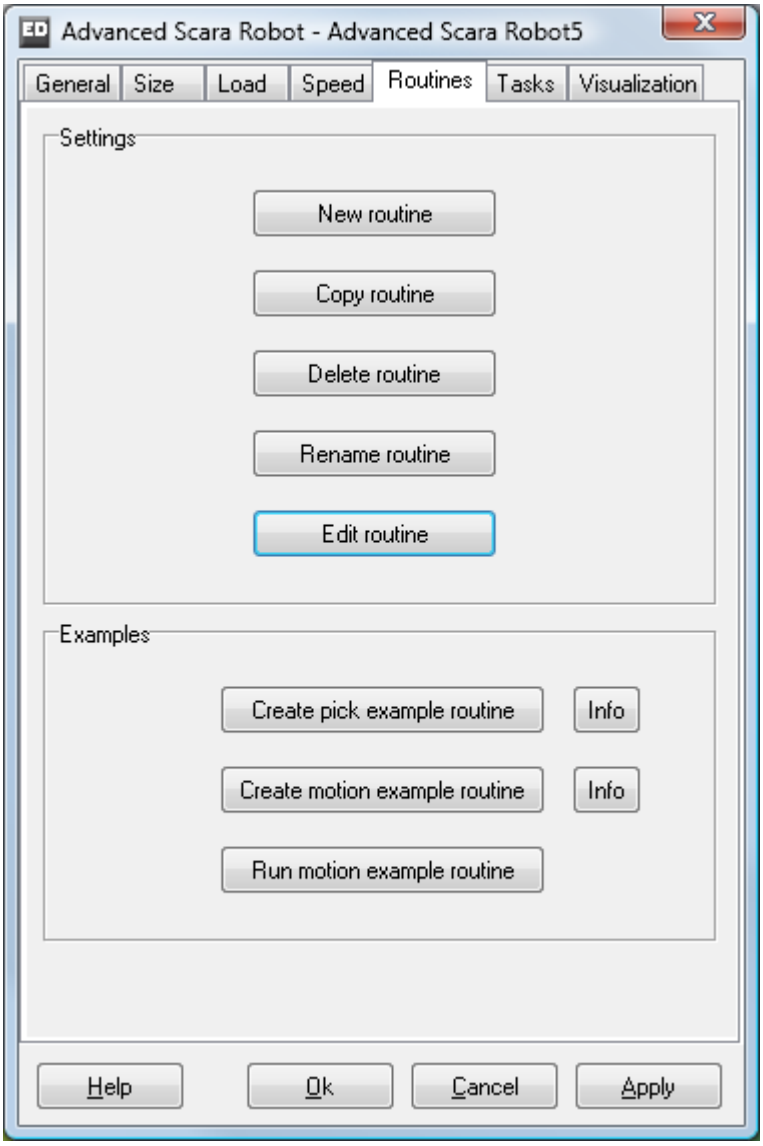
Note: Tasks are calls to the Advanced Scara Robot to execute a specified Routine. There are two ways to create a Task. Pick and place Task are created every time an atom connected to an input channel of the robot causes an IcReady. Motion Tasks are created if there is an external call created with the "Advanced_Scara_Robot_create_motion_task" command. If the robot is busy, the Tasks are stored until the robot is idle again. The Tasks selection rule determines what Tasks has to be done next.



Picture 19-5: The Advanced Scara Robot task parameters

- *Task selection rule*
  This option lets execute the Tasks in a specified sequence. The internal queue of the Advanced Scara Robot is sorted in the selected sequence before choosing the next Tasks to execute. Default is first in first out (FIFO). There are 10 strategies available.

  1: FIFO --> first in first out
  2: LIFO --> last in first out
  3: Content minimum --> atoms in the container with the smallest content first
  4: Content maximum --> atoms in the container with the largest content first
  5: IC minimum --> lowest input channel of robot first
  6: IC Maximum --> highest input channel of robot first
  7: Label minimum (atom) --> the atom with the lowest value on label first
  8: Label maximum (atom) --> the atom with the highest value on label first
  9: Label minimum (container) --> atoms in the container with the lowest value on label first
  10: Label maximum (container) --> atoms in the container with the highest value on label first



Picture 19-6: The Advanced Scara Robot routine parameters

Note: Routines are sub atoms of the Advanced Scara Robot. Storing all information about repeatable sequences into tables, the behaviour becomes reproducible in order to execute frequently necessary actions like motions, loading, unloading or waiting.

- *New Routine*
  Click this button to create a new Routine. You will be asked to enter a Routine name. If the Routine has been created successfully, you are asked to edit commands to the Routine table by using the edit commands GUI.

- *Copy Routine*
  Click this button to copy a existing Routine with all its commands. Select the sample Routine via atom selector. If selection was successful, you will be asked to enter a diffrent Routine name. If the Routine has been created successfully, you are asked to edit Commands to the Routine table by using the edit routines GUI.

- *Delete Routine*
  Click this button to delete a existing Routine with all its table-stored Commands.

Note: The global reference to the Routine will be unregistered too. Ensure, that the Routine is not needed by any other Advanced Scara Robot.

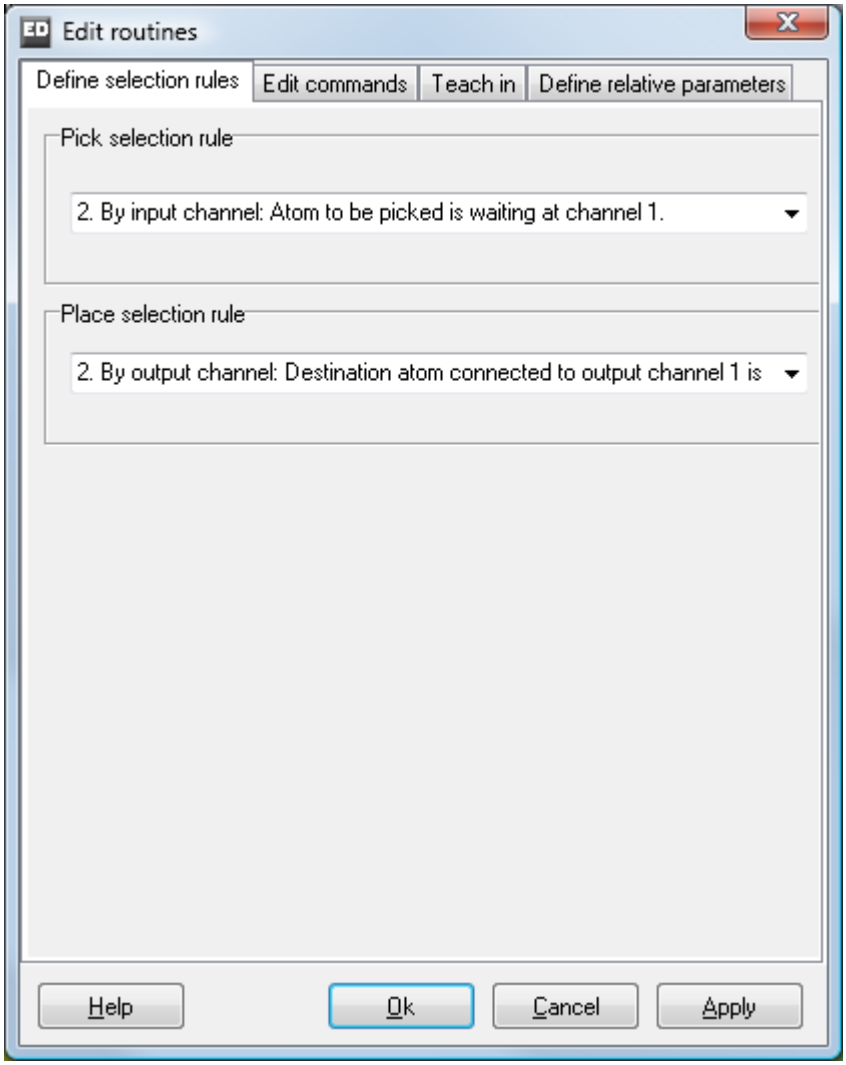
- *Create pick example Routine*
  Click this button to create a Pick Routine example with the name 'Routine_pick_example'. If the Routine has been created successfully, you are asked to edit Commands to the Routine table by using the edit commands GUI.

Warning! Pick example Routine only works if input channel one of tool is connected to pick atom and output channel one of Tool is connected to place atom.

- *Create motion example Routine*
  Click this button to create a Motion Routine example with the name 'Routine_motion_example'. If the Routine has been created successfully, you are asked to edit Commands to the Routine table by using the edit commands GUI.

- *Run motion example Routine*
  Click this button to run the 'Routine_motion_example' routine.

- *Edit Routine*
  Click this button to select the Routine you want to edit via atom selector. If selection was successfully, you are able to edit Commands to the Routine table by using the Advanced Scara Robot Edit Routines GUI.



Picture 19-7: The selection rule definition at the routine editor

All kind of action like loading, unloading, moving, delays are defined within Routines. The Advanced Scara Robot Edit Routines GUI is the tool to apply, insert, copy, delete and edit Routine Commands. It is also possible to define selection rules for the Routines, so the Robot is able to select the Routine that fits best to the current Task. The Teach In functionality allows to create motions in an easy way.

Note: Some of the parameters used for the Edit Routines GUI are defined within the Advanced Scara Robot GUI.

- *Pick selection rule*
  Determines the Pick condition to select the Routine. The Routine is selected, if the selected statement is true. There are 16 strategies available.

  1: No pick restriction or motion routine
  2: By input channel
  3: By pick atom name
  4: By pick icon name
  5: By pick icon number

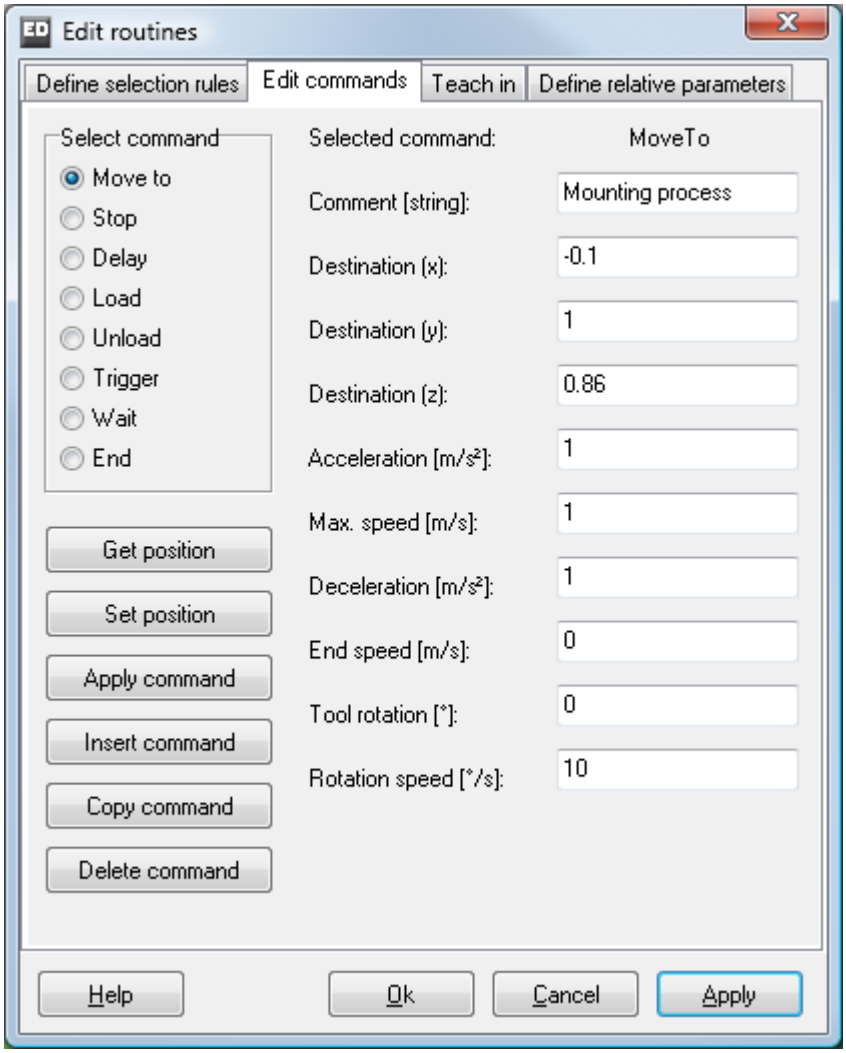
6: By pick label value (direct)
7: By pick label value (conditional)
8: By pick label text
9: By pick location (direct)
10: By pick location (conditional)
11: By pick size (direct)
12: By pick size (conditional)
13: By pick content (direct)
14: By pick content (conditional)
15: Conditional statement
16: By user

Note: Choose a Pick selection rule if there is more than one Routine available and the selection criteria can be found via input channel connection. The Pick selection rule can be combined with a Place selection rule to define advanced strategies.

- *Place selection rule*
  Determines the Place condition to select the Routine. The Routine is selected, if the selected statement is true. There are 11 strategies available.

  1: No place restriction or motion routine
  2: By output channel
  3: By output channel fail safe
  4: By place location (direct)
  5: By place location (conditional)
  6: By place size (direct)
  7: By place size (conditional)
  8: By place content (direct)
  9: By place content (conditional)
  10: Conditional statement
  11: By user

Note: Choose a Place selection rule if there is more than one Routine available and the selection criteria can be found via output channel connection. The Place selection rule can be combined with a Pick selection rule to define advanced strategies.



Picture 19-8: The Advanced Scara Robot command editor

- *Select command*
  Select between the predefined kinds of Commands to define robot behaviour. There are 8 predefined Commands available:

  1: Move to - defines any kind of robot motion
  2: Stop - stops any kind of robot motion
  3: Delay - makes the robot waiting for a specific time
  4: Load - executes picking of incoming products
  5: Unload - executes placing of outgoing products
  6: Trigger - allows the execution of 4DScript code
  7: Wait - let the robot wait for an external trigger to continue
  8: End - finishes a routine

Choose a Command to edit the necessary parameters. The number and kind of parameters depends on the type of selected Command.

Move to command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Destination (x/y/z)*
  Tool center destination according to the origin of the coordinate system.

- *Acceleration [m/s²]*
  The tool center acceleration to reach the destination.

- *Max. speed [m/s]*
  The tool center maximum speed to reach the destination.

- *Deceleration [m/s²]*
  The tool center deceleration to reach the destination.

- *End speed [m/s]*
  The Tool center end speed to reach the destination.

- *Tool arm deflection [°]*
  Absolute deflection of the tool arm at the destination.

- *Deflection speed [°/s]*
  The deflection speed to reach the absolute Tool arm deflection.

- *Tool arm rotation [°]*
  Absolute rotation of the Tool arm at the destination.

- *Rotation speed [°/s]*
  The rotation speed to reach the absolute Tool arm rotation.

Note: You are able to edit detailed speed parameters, if the default speed parameters checkboxes is selected at the Advanced Scara Robot GUI.

Note: The Advanced Scara Robot uses the MovingTo Command to define the motions between the origin of the coordinate system and the center of the Tool. In MovingTo the acceleration and deceleration are leading. The end speed will be tried to reach, but when this is impossible the end speed is the last speed taking acceleration and deceleration into account.

Note: You can get the current position of the Tool into the editfields by click on the Get position button.

Note: You can set the position of the Tool according to the editfields by click on the Set position button.

Stop command:

- *Comment*
  Comment the Command to make the Routines more readable. This command has no other parameters it just stops all Robot motion.

Delay command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Time [s]*
  The delay before the Routines is continued with the next command.

Load command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Time [s]*
  The load time of the atom to be picked.

- *Channel [index]*
  The input channel of the robot, where the atom is waiting.

Note: You are able to edit detailed load parameters, if the default load parameters checkboxes is selected at the Advanced Scara Robot GUI.

Unload command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Time [s]*
  The unload time of the atom to be placed.

- *Channel [index]*
  The output channel of the robot, where the atom has to be sent to.

Note: You are able to edit detailed unload parameters, if the default load parameters checkboxes is selected at the Advanced Scara Robot GUI.

Trigger command:

- *Comment*
  Comment the Commands to make the Routines more readable.

- *Trigger [4DScript]*
  The code that has to be executed before the Routine is continued with the next

Command. Select between the predefined Commands defined in the kernel function “getTriggerEnterExit” or enter your own 4DScript code.

Wait command:

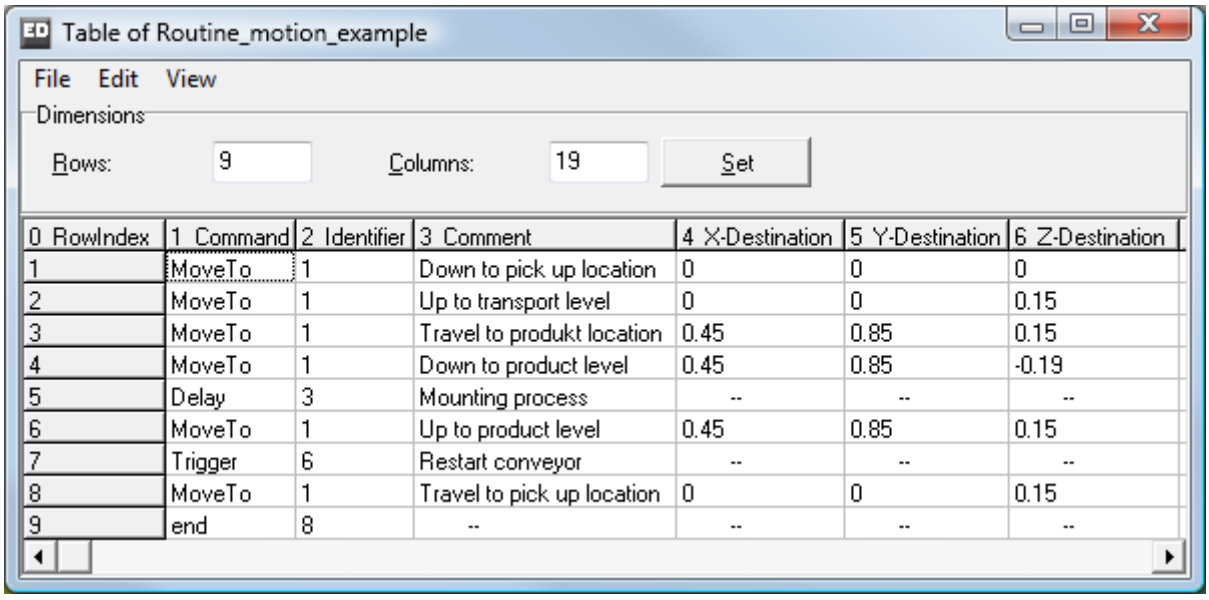
- *Comment*
  Comment the Commands to make the Routines more readable.

- *Awaited message [string]*
  The text message that has to arrive in the OnMessage eventhandler of the Advanced Scara Robot before the Routine is continued with the next Command.

End command:

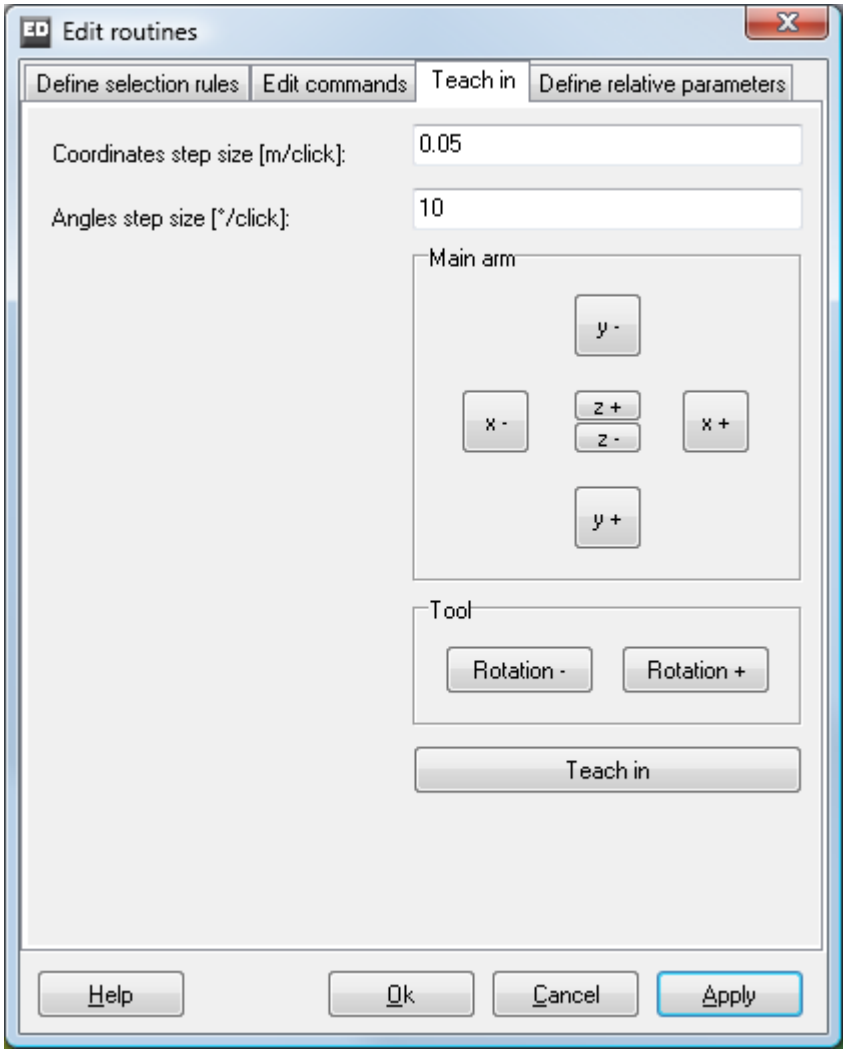
If you select the End Command no parameters can be changed, but another row is added to the Routine in order to set a flag, that that there will be no further Commands. The Robot is set to status Idle and it searches for other Tasks available.

There are four buttons to apply, insert, copy and delete Commands. On click you are asked to edit the row index of the Command that you want to apply, insert, copy or delete.



0 RowIndex	1 Command	2 Identifier	3 Comment	4 X-Destination	5 Y-Destination	6 Z-Destination
1	MoveTo	1	Down to pick up location	0	0	0
2	MoveTo	1	Up to transport level	0	0	0.15
3	MoveTo	1	Travel to produkt location	0.45	0.85	0.15
4	MoveTo	1	Down to product level	0.45	0.85	-0.19
5	Delay	3	Mounting process	--	--	--
6	MoveTo	1	Up to product level	0.45	0.85	0.15
7	Trigger	6	Restart conveyor	--	--	--
8	MoveTo	1	Travel to pick up location	0	0	0.15
9	end	8	--	--	--	--

Picture 19-9: Routine example (motion)

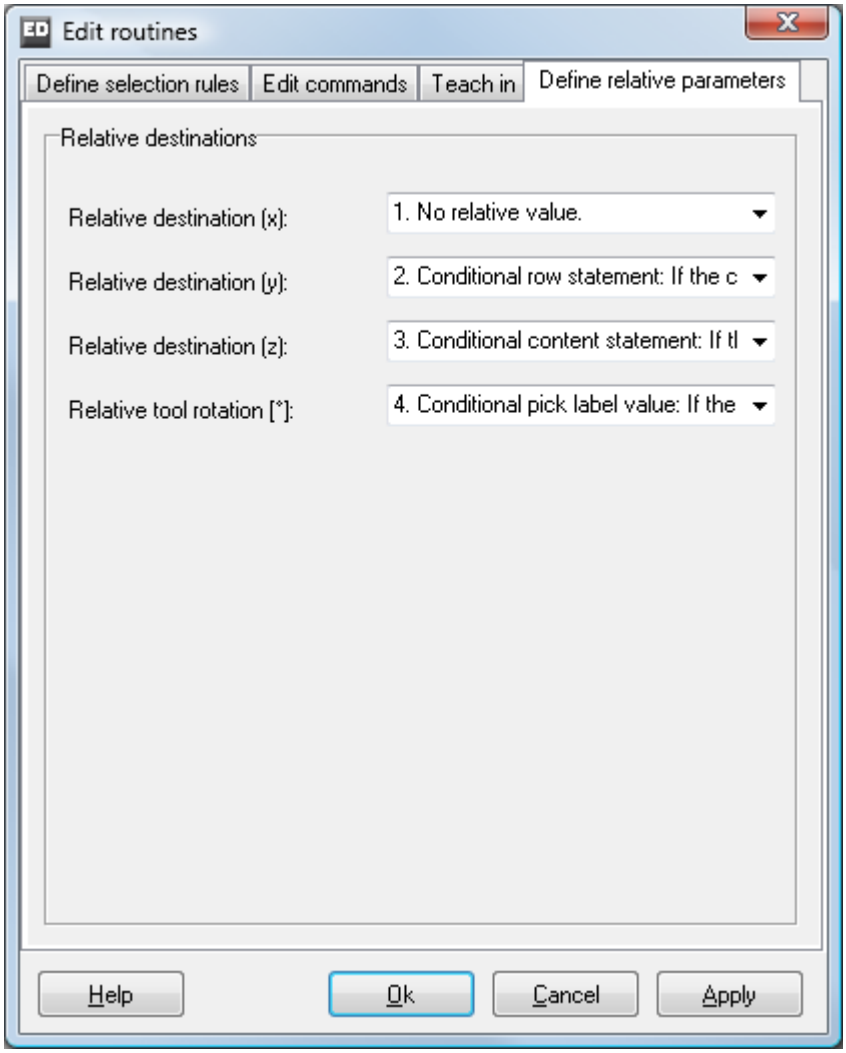


Picture 19-10: Routine teach panel (move to command)

- *Coordinates step size [m/click]*
  On click of main arm buttons the tool position will move according to the coordinates step size.

- *Angles step size [°/click]*
  On click of tool arm buttons the tool deflection will rotate according to the angles step size. On click of tool buttons the tool will rotate according to the angles step size.

- *Teach in button*
  On click all motion parameters are updated into the editfields of the Edit Command page and the destination is stored to the active routine as Move to Command.



Picture 19-11: Relative destinations panel

- *Relative destination (x)*
  The Tool relative x destination.

- *Relative destination (y)*
  The Tool relative x destination.

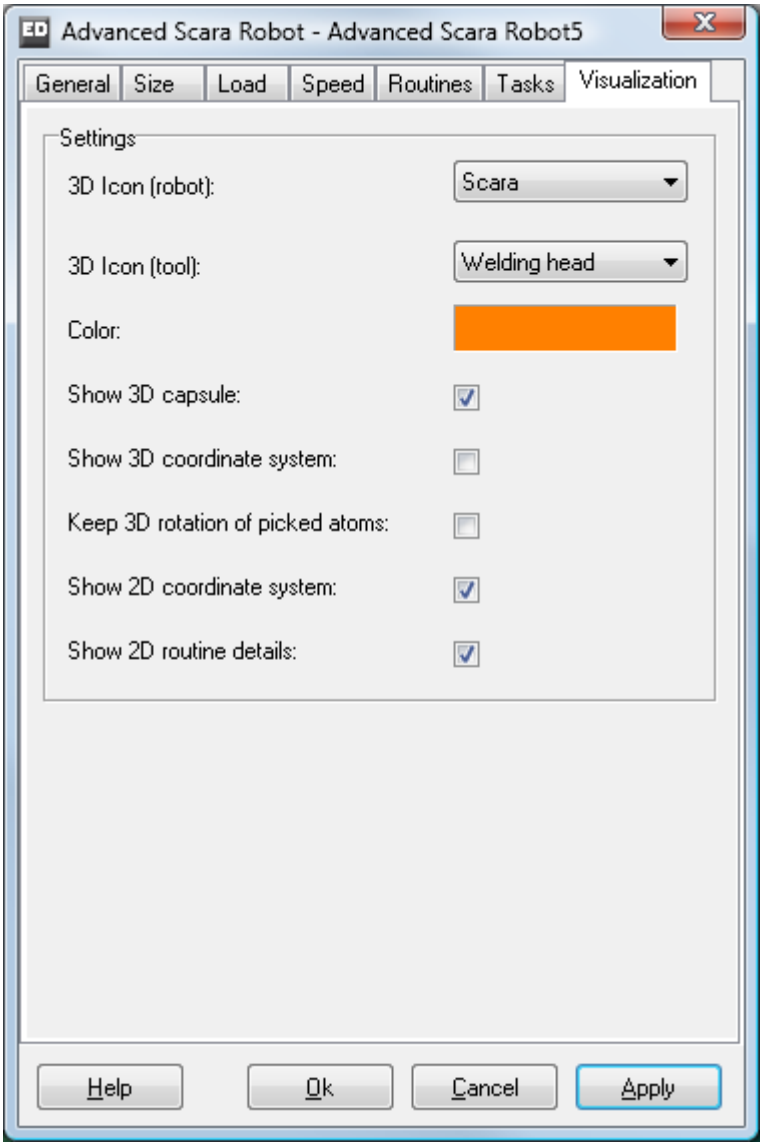
- *Relative destination (z)*
  The Tool relative x destination.

- *Relative tool rotation [°]*
  Relative rotation of the Tool arm at the destination.

There are 5 predefined relative value definitions available:

1: No relative value – the parameters from the Routines table are used on any account
2: Conditional row statement – cases are defined based on the current row index of the active Routine

3: *Conditional content statement - cases are defined based on the current content of the Tool*
4: *Conditional pick label value statement - cases are defined based on Label values of Products waiting at input channel of the Tool*
5: *User defined statement – define your own statement using 4DScript*



Picture 19-12: Visualization panel

- *3D Icon (robot)*
  A 3D icon that can be used to visualize the Advanced Scara Robot in 3D. Change this icon to switch between standard visualization and other customized 3D icons.

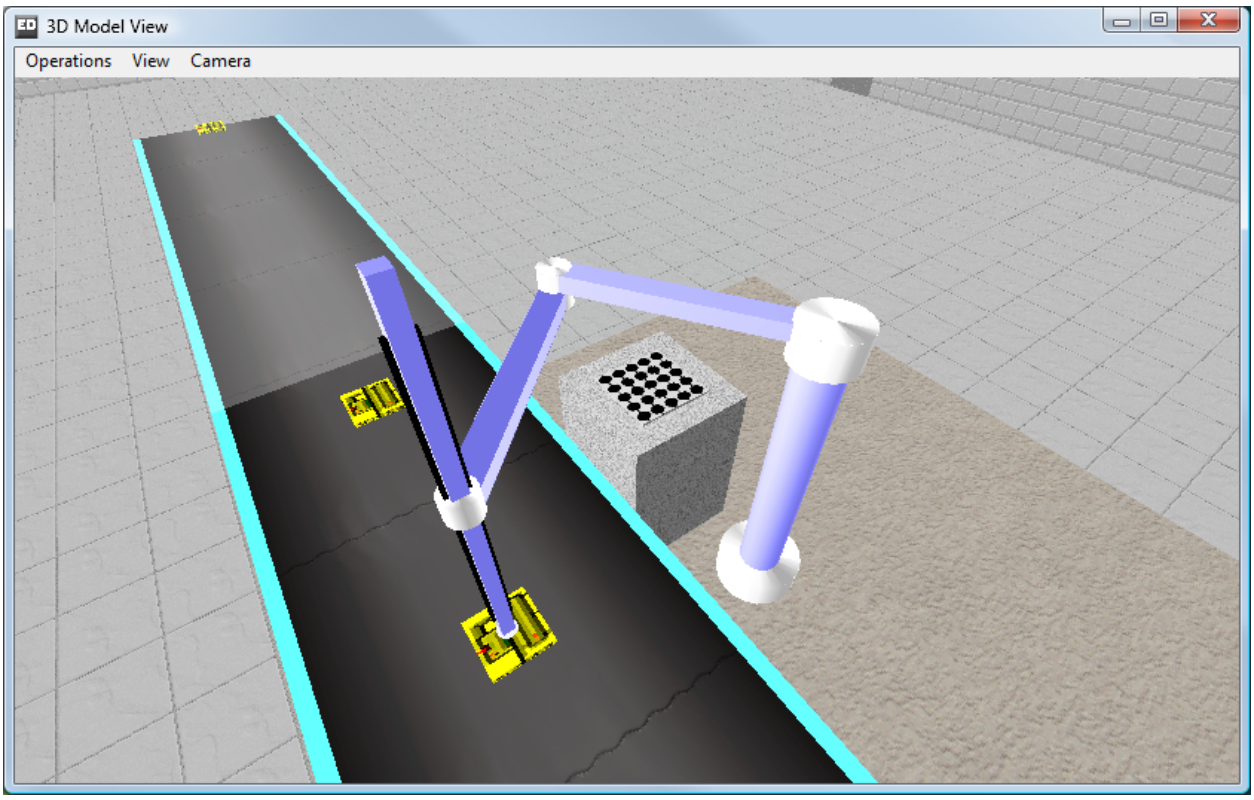
- *3D Icon (tool)*
  A 3D icon that can be used to visualize the Tool in 3D. Change this icon to switch between grabber, welding head and other customized 3D icons.

- *Color*
  The color of Advanced Scara Robot.

- *Show 3D coordinate system*
  If this box is checked the origin of the 3D coordinate system and the centre of the Tool will be displayed to ease routine definition.

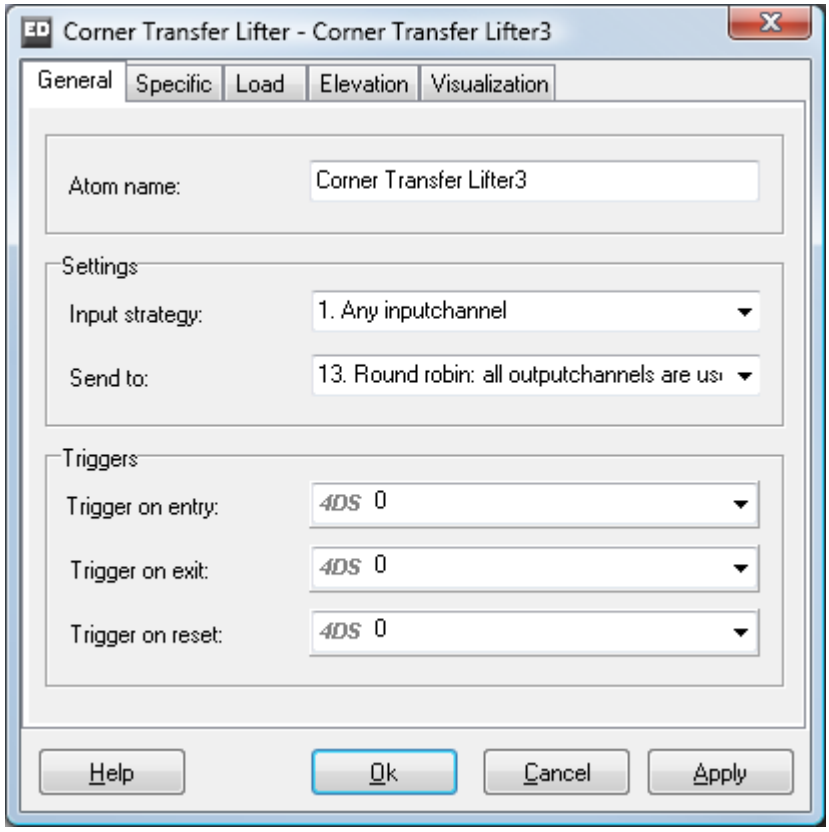
- *Show 2D coordinate system*
  If this box is checked the origin of the 2D coordinate system and the centre of the Tool will be displayed to ease routine definition.

- *Keep rotation of picked atom*
  If this box is checked picked atoms will not be rotated according to the rotation of the robot joints, but will keep their global rotation.



Picture 19-13: 3D view of a running motion example

20 THE CORNER TRANSFER LIFTER ATOM



Picture 20-1: The Corner Transfer Lifter atom

A Corner Transfer Lifter is an intersection between conveyors of any kind, if elevation to another floor level is required. Unlike the Turntable Unit Atom it does not rotate incoming goods into transport direction before lifting and forwarding them to an outgoing conveyor. Products rather keep their rotation when turning left, right or moving straight ahead. The number of possible changes of transport directions is limited for each z level. Products can turn left around a corner of -90 degrees, turn right around a corner of 90 degrees or go straight ahead. It is even possible to travel into counter direction on another z level. Often a turn around a corner is not possible without a transfer time to slow down speed, change the drive mechanism and speed up into the new direction. There is a minimum of one input and a maximum of four inputs for each z level. The number of outputs is at least one but has a maximum of four for each z level. To control more than one input a strategy is required. More than 1 outputs requires a strategy too.

Note: This atom is part of ED Logistics. ED Logistics edition also has a Turntable Unit atom that rotates incoming goods into transport direction before forwarding them to an outgoing conveyor.

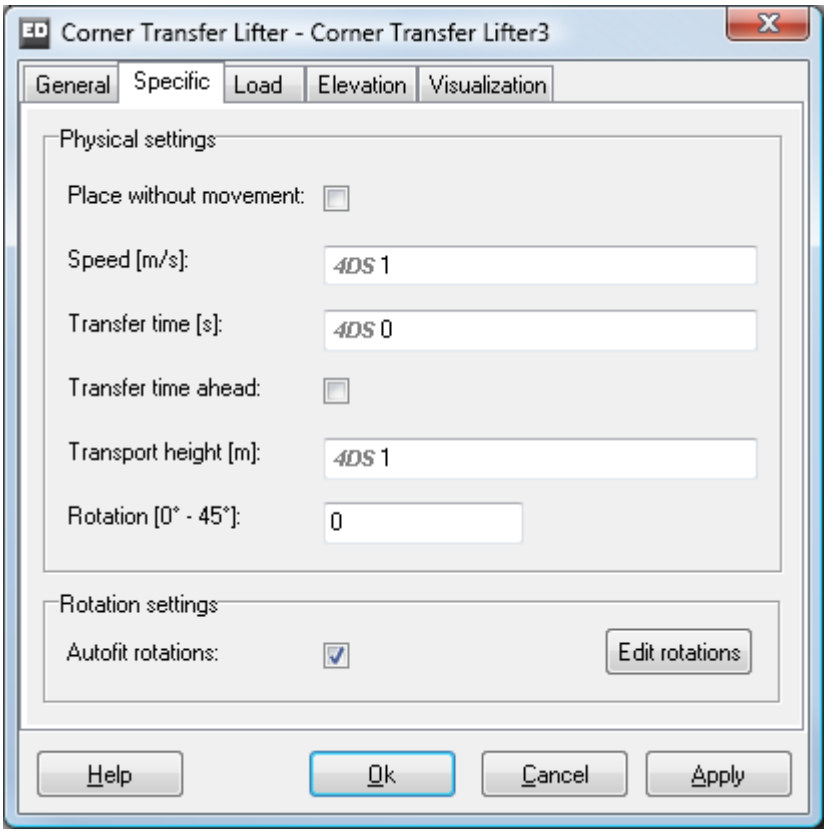
- *Atom name*
  The name given to the atom.

- *Input strategy*
  A rule that determines how Product atoms may be let into the inbound atom of the Corner Transfer Lifter. There are a number of rules predefined (see also Input strategy), but you can also create your own rule via a 4DScript expression.

- *Send to*
  A rule that determines to which output channel a Product atom needs to be sent. There are a number of rules predefined (see also Send to), but you can also create your own rule via a 4DScript expression.

- *Trigger on entry*
  A rule that determines what kind of action needs to be executed when a Product atom enters the inbound atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.

- *Trigger on exit*
  A rule that determines what kind of action needs to be executed when a Product atom exits the inbound atom. There are a number of rules predefined (see also Trigger on entry and exit), but you can also create your own rule via a 4DScript expression.



Picture 20-2: The specific parameters

- *Place without movement*
  If this box is checked the Corner Transfer Lifter will create no input movement. Instead incoming products will be placed immediately in the center of the atom. This functionality may be helpful if the Corner Transfer Lifter receives goods from an atom of the transportation section like the Portal Crane atom or a Robot atom.

- *Speed [m/s]*
  The speed for the input movement of the products to the center of the Corner Transfer Lifter. The speed for the output movement is controlled by the atom connected to the output channels.

- *Transfer time [s]*
  Time to slow down speed, change the drive mechanism and speed up into the new direction when turning around a corner.

- *Transfer time ahead*
  If this box is checked the transfer time is also necessary for straight on going products.

- *Transport height [m]*
  The length of the leg supports of the conveyor (in meters).

- *Rotation [+/- 45°]*
  At default the conveyor is positioned from left to right in your screen. However you can also rotate the conveyor to display the flow of atoms over the conveyor matches reality.

Note: Due to the layout of the Corner Transfer Lifter atom rotation is limited +/- 45° because any other rotation can be modelled by turning left or right.

- *Autofit rotations*
  If this box is checked the Corner Transfer Lifter tries to detect the rotation of atoms connected to input and output channels OnReset. The Corner Transfer Lifter needs to know the rotation of connected atoms to calculate the rotation of passing products.

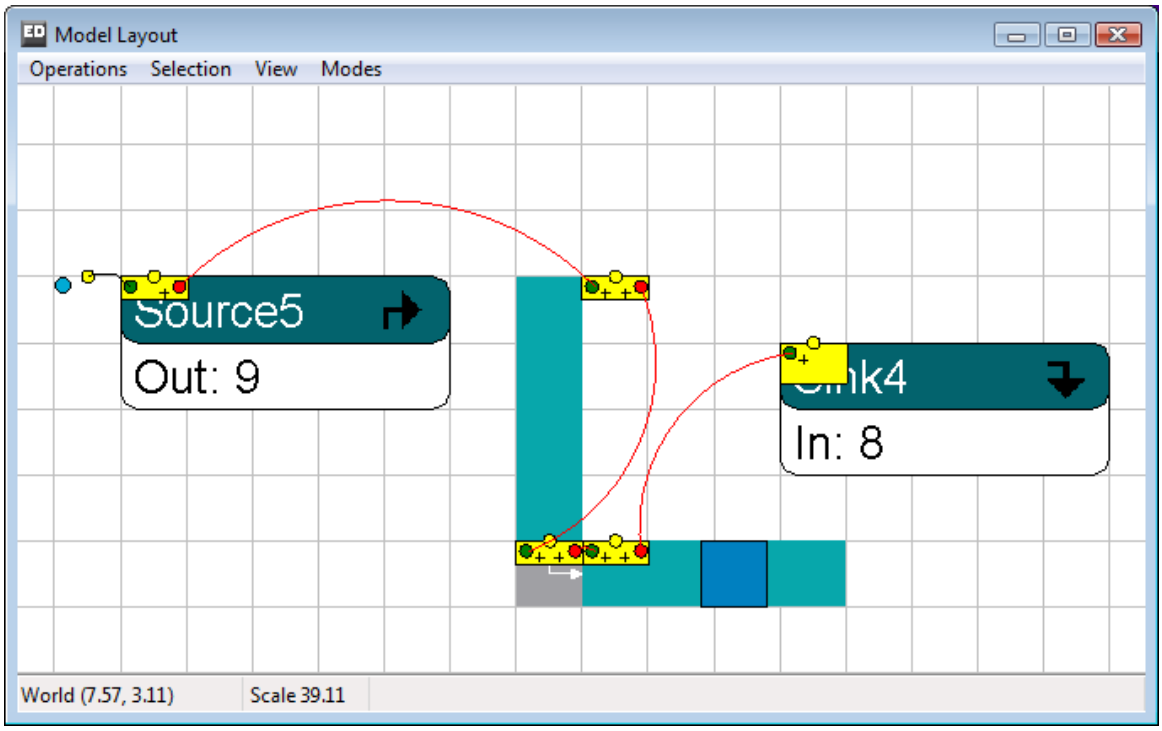
Note: However not all kinds of atoms may be detected by the auto fit functionality. Therefore it can be switched of in order to edit the table of rotations by hand.

- *Edit rotations*
  Click this button to edit the rotation around self of atoms connected to the Corner Transfer Lifter. Edit the motion flags for the input and output motion directions. Edit 1 for positive x/y directions and -1 for negative x/y directions.

Note: The number of rows is updated OnReset according to the number of input and output channels. Channels not connected to an atom cause an error message to prevent user from unexpected behaviour later on.

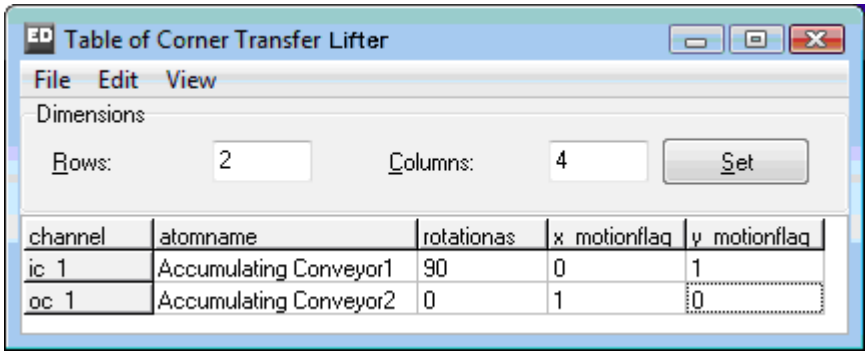
Note: Set only one motion flag for each row, either an motion along x-axis or along y-axis.

Example: Connect a Source with an 90° rotated Accumulating Conveyor. The conveyor leads into a Corner Transfer Lifter atom. The Products are sent on to another Accumulating Conveyor with a rotation of 0°, before they are destroyed in a Sink.



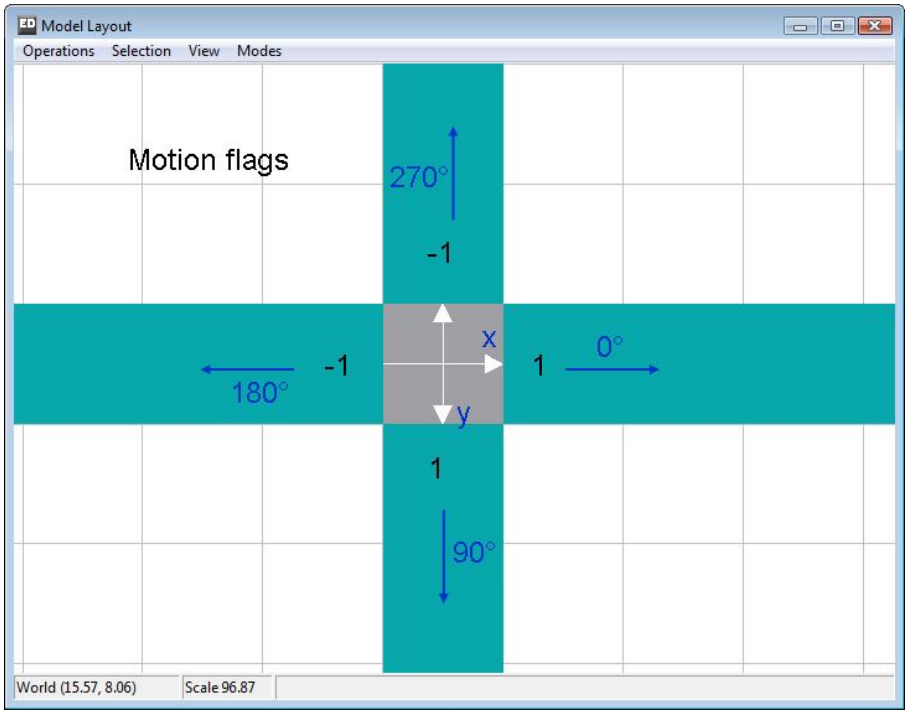
Picture 20-3: 2D Model Layout example

Set x-motion flag to 0 and y-motion flag to 1 for Products coming from an input (with rotation 90° around self) to move in. Set x-motion flag to 1 and y-motion flag to 0° for Products going to an output (with rotation 0° around self) to move out.



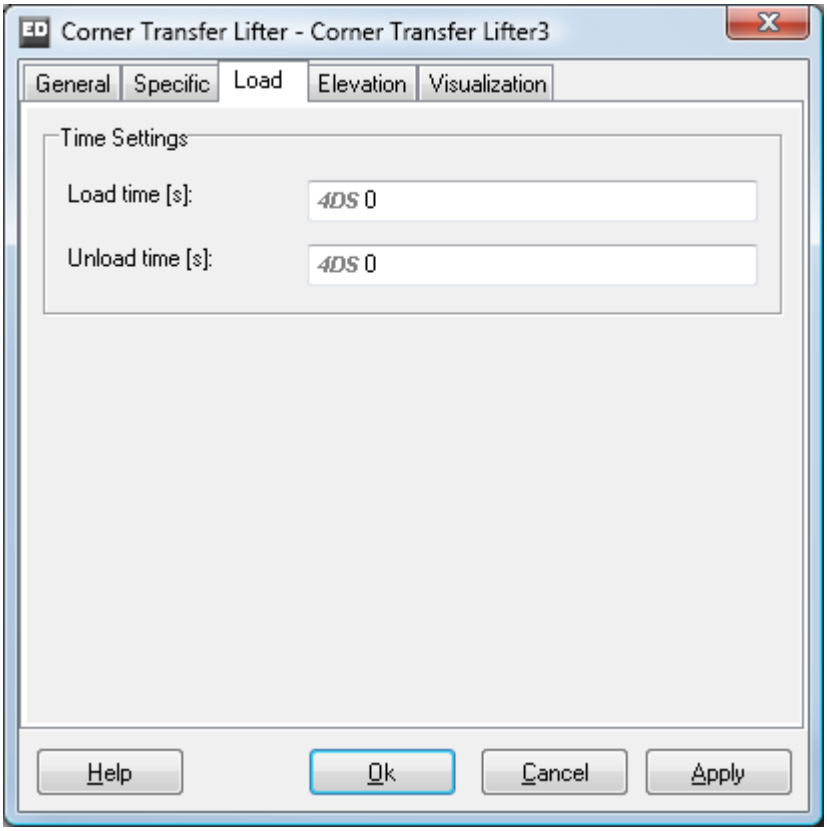
Picture 20-4: The table of rotations settings

If you select the 'Autofit rotations' Checkbox in the GUI of the Corner Transfer Lifter atom, the table is filled automatically.



Picture 20-5: Motion flags

The result of the edits is visible in the 2D Model Layout. White arrows show the calculated transport directions.



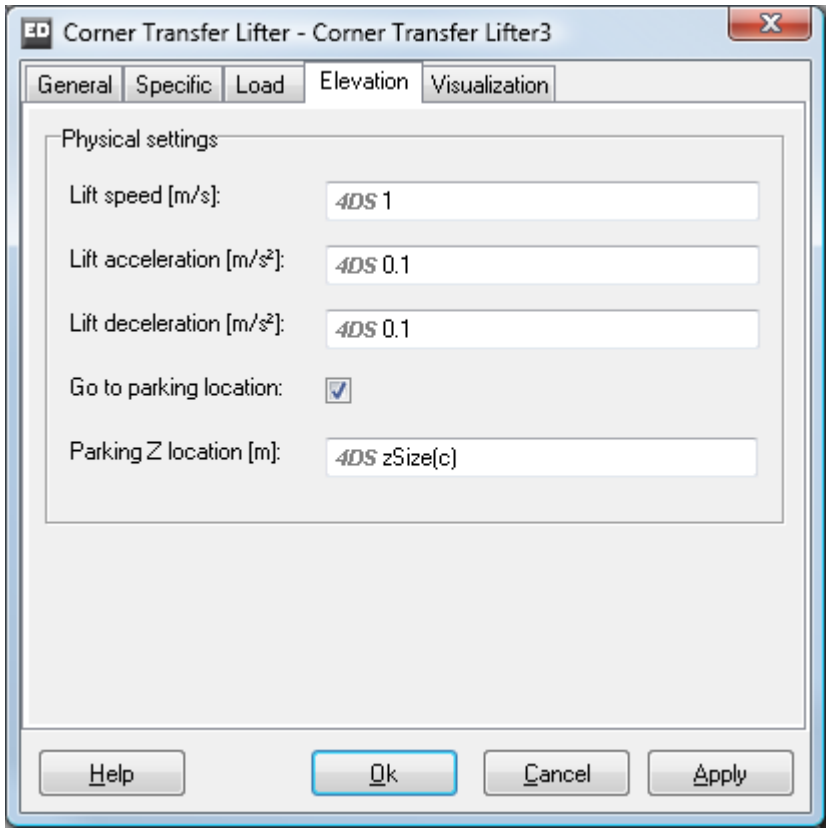
Picture 20-6: The load parameters

- *Load time [s]*
  The time (s) that is needed to load something to Corner Transfer Lifter.

Note: Do not confuse load time with transfer time, as the load time is executed on entry of Product.

- *Unload time [s]*
  The time (s) that is needed to unload something from Corner Transfer Lifter.

Note: Do not confuse unload time with transfer time, as the unload time is executed on exit of Product.



Picture 20-7: The elevation parameters

- *Lift speed (m/s)*
  The maximum speed for the elevation movement of the products to the destination z level of the Corner Transfer Lifter.

- *Lift acceleration (m/s2)*
  The acceleration for the elevation movement of the products to the destination z level of the Corner Transfer Lifter.

- *Lift deceleration (m/s2)*
  The deceleration for the elevation movement of the products to the destination z level of the Corner Transfer Lifter.

- *Go to parking location*
  If this box is checked the Corner Transfer Lifter is travelling to the default parking location after every finished transportation task, before it becomes idle again. The Corner Transfer Lifter needs to know the destination z location.

- *Parking Z location (m)*
  The destination z location of the Corner Transfer Lifter for the default parking location.

Annex 3 A first start in 4DScript

4DScript is the programming language of Enterprise Dynamics. Everything that is executed in Enterprise Dynamics is or can be done via 4DScript.

This document provides for beginners the structure of 4DScript and a list of often used commands, illustrated with examples.

We start with the syntax rules together with the mathematical and logical operators. Second, the concept of referencing is illustrated with examples. The third part consists of the most common commands in ED.

1. The basics of 4DScript

The basic syntax of 4DScript is simple and is valid at any position where you can write 4DScript:

1 the language contains a number of words (commands);
2 these 4DScript commands can have (up to 255) parameters;
3 the parameters are placed between parenthesis ( );
4 parameters can be *values*, *strings* or *expressions* (other 4DScript words);
5 parameters will always be separated by commas;
6 if a parameter should be interpreted as a string, the string parameters are always placed between square brackets [ ]. If the parameter should be executed as 4DScript code, the parameter is written in a normal fashion;
7 comments can be placed between squiggly brackets { }.

These rules apply everywhere and are always valid. However, for some statements these rules do not result in very readable code. In order to make parts of the code easier to understand for others, the 4DScript syntax makes an exception in some cases. This applies in particular to mathematical and logical symbols.

Mathematical symbols

Consider the following valid statements:

A more natural way:
+(12,7) results in 19
12+7
+(/(100,10),6) results in 16 100/10+6 or 6+100/10

All the special mathematical operators are evaluated in the following order:

*	=	multiplication
/	=	division
+	=	addition
-	=	subtraction

The normal priority rules for these operators apply. In case you have doubts, use parenthesis ()!

Logical symbols

For logical operators (like >, < , =) it is also possible to disregard the rule that the 4DScript command needs to be written first and then the parameters

Consider the following valid statement:

>(10,6) can also be written in the `normal' fashion as: 10>6
<=(23,12) can also be written as: 23<=12

This example can be extended to more 4DScript logical symbols:

=	=	equal
>	=	larger than
<	=	smaller than
>=	=	larger than or equal to
<=	=	smaller than or equal to
<>	=	not equal

and
syntax: and(e1,e2)

Returns 1 if e1 and e2 are true (1), returns 0 if not. If e1 is not true, then e2 is not evaluated. Instead of writing and(e1,e2) the user can also write e1 **and** e2See also **or**

So, both commands are used as usual…

max
Syntax: max(e1,e2)

Returns the maximum of e1 and e2. See also **min**

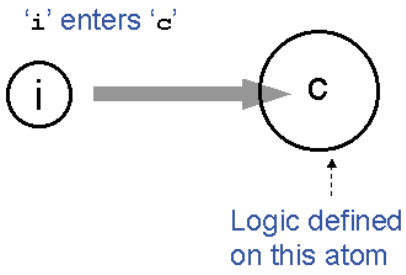
We end this chapter by a few commands on time and time conversion. Remember that ED `thinks' in seconds, but it is easy to convert time:

time	returns the current model time in seconds
mins(e1)	returns e1 (minutes) in seconds: multiplies e1 with 60.
hr(e1)	returns e1 (hours) in seconds: multiplies e1 with 3600.

2. Referencing

A very important subject in ED is referencing: if we are talking about the `next' atom, what is our viewpoint? It gets more complicated if there are atoms contained in other atoms like products in a queue.

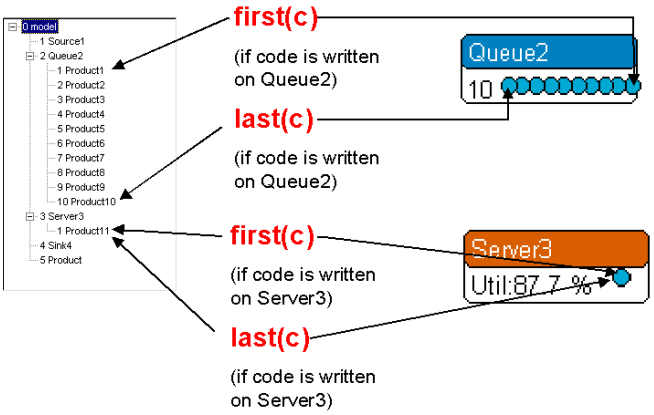
We have seen the letters c and i in different statements, now we will explain them:
`c' refers to the **c**urrent atom where the statement is written on
`i' refers to the **i**nvolved atom, the one entering or leaving the current atom



Picture 1: References i and c

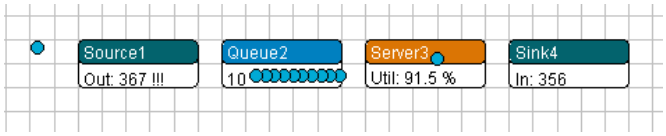
To keep it simple: the `i' is used only in statements written on Trigger on Entry or Trigger on Exit!

The key to understanding the concept of referencing is the next example:



Picture 2: Model Tree of a simple Queueing System. Explaining First and Last atom references

On the left of picture 2 you see the Model Tree of a simple queuing system, frozen at one moment in time:



Picture 3: Model of a simple queueing system

On the same level we find the Source, the Queue, the Server and the Sink and the Product atom placed before the Source (number 5 in the model tree, because it is mostly added after you reset your model the first after making the lay-out).

Queue2 contains 10 products, named Product1 to Product 10. They form a second level, compared to the Product, Source, Queue, Server and Sink. Of course Product1 and, lets say, Product5 are on the same level. Server3 contains 1 product, named Product11.

The highest level is the model itself. In picture 1 you can see this in the model tree! Can you predict how the model tree changes if you reset this model?

If you understand the hierarchy in this model, you will understand referencing…

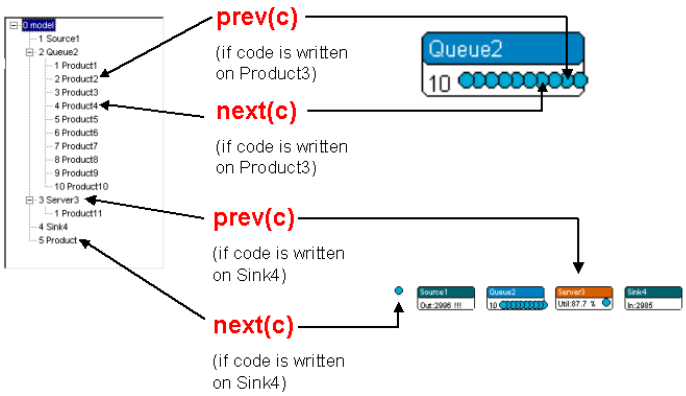
First a few commands which are often used for referencing:

first(e1) refers to the first atom inside atom e1
last(e1) refers to the last atom inside atom e1
next(e1) refers to the next atom on the same level as atom e1
prev(e1) refers to the atom in front on the same level atom e1

So first(c) refers to the first atom inside the current. If the current is a Queue it could be the first product in the queue. Now look at picture 1 again and the examples on *first* and *last.*

So, first and last are always looked at one level deeper from the atom where the statement is written on!

The commands next and previous are explained below:



Picture 4: Model Tree of a Queueing System. Explaining Prev and Next atom references

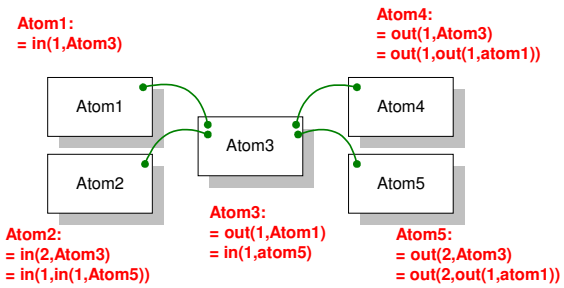
Now look at picture 3 and the examples on *prev* and *next.*

So, prev and next are always looked at the same level from the atom where the statement is written on!

There are a few commands often used for flow control and related to channels:
out(e1,e2) refers to the atom connected to output channel e1 of atom e2
in(e1,e2) refers to the atom connected to input channel e1 of atom e2
OpenOutput(e1) opens the general output of atom e1
CloseOutput(e1) closes the general output of atom e1. If closed, the channels cannot be ready, regardless of individual channel settings.

In a similar way you can define **OpenInput** and **CloseInput.**

Take a look at picture 5 regarding the use of *in* and *out:*



Picture 5: Explaining in and out atom references

Example 1:
So refers *in(1,c)* (written on Atom3, the point of view) to Atom1, but the same statement written on Atom4 refers to Atom3!

Example 2:
Content(in(3,c)) returns the content (number of atoms) contained in the atom which is connected to input channel 3 of the current atom.

Example 3:
CloseOutput(in(2,c)) closes the output of the atom connected to the second input channel of the current

Notice the way commands are nested…

3. Important commands

In every programming language there are important commands, which perform as some kind of building blocks.

3.1 For conditional statements: **if**

Syntax: if(e1,e2 {,e3})
Executes e2 if e1 is true (1); otherwise executes e3 (if specified). Returns result of e2 or e3.

Example 1:
if(time>3600, msg[more than an hour], msg[less than an hour])

Translated: if time>3600 seconds then give message `more than an hour' else give message `less than an hour'.

Example 2:
if(content(c)>10, closeinput(in(1,c))

Translated: if the content of the current atom is more than 10 then stop the input of the atom connected with the first incoming channel else do nothing

Don't worry about the used commands like msg, content or closeinput. We'll explain them later!

3.2 For executing more than one statement: **do**

Syntax: do(e1,e2,...,e25)
Executes e1, e2, etc. in order of sequence. Returns result of last expression. You can use up to 25 parameters. If more parameters are needed, several do loops can be nested.

Example:

do(set(color(i), colorred),set(icon(i) ,2),
setlabel([temperature], uniform[20,40], i)
)

Three things are done at the **i**nvolved atom (mostly a product): the color is set to red, the icon is changed to iconnumber two and a label with the name *temperature* is stamped on the product, its value is chosen randomly between 20 and 40.

3.3 Labels

The use of labels is the way to store local (temporary) variables by users. They are mostly attached to products and can represent a weight, a customernumber, a productiontime etcetera. They are defined with the command *setlabel* and reproduced with *label.*

a. setlabel(e1,e2,e3)

Defines a label on atom e3 where e1 is the name and e2 is the new content of the label. Labels do not have to be created, at the moment they are referenced they exist. The *sddb* command does the same.

Example 1:
setlabel([Weight],10.24,i)

First select an atom and create (and give a value to) a label with the command setlabel. Suppose the name of the label is v1 and the value is 100:

Example 2:
setlabel([v1],100,animatom)

label([v1],animatom) now returns the specified value (100) of the selected atom.

b. label(e1,e2,{e3})

Returns the contents of label named e1 defined on atom e2. Labels do not have to be created, at the moment they are referenced they exist. The label name e1 is always a string and is case sensitive.

The result is a string or a value, depending on the contents and on e3: if e3 is not specified or e3=0 then the result is a value if the contents can be converted to a value, otherwise the result is a string. If e3=1, the result is always a value. If the contents cannot be converted to a value the value is 0. If e3=2, the result is always a string. If you use long names and many labels you lose speed.

3.4 We conclude with a few other important commands:

age

Syntax: age(e1)

Returns the age of atom e1. The age is the difference between current time and creation time. When a run is reset, the creation time of all atoms becomes 0.
See also **maxage, minage, avgage**

content

Syntax: content(e1)

Returns the contents (the number of atoms contained at the highest level) of atom e1.
See also **maxcontent, avgcontent**

avgstay

Syntax: avgstay(e1)

Returns the average staying time of atoms in atom e1 over the total runtime.

input

Syntax: input(e1)

Returns the input of atom e1. The input is defined as the number of atoms that have entered atom e1 until now. The opposite is the output of the atom.
See also **output**