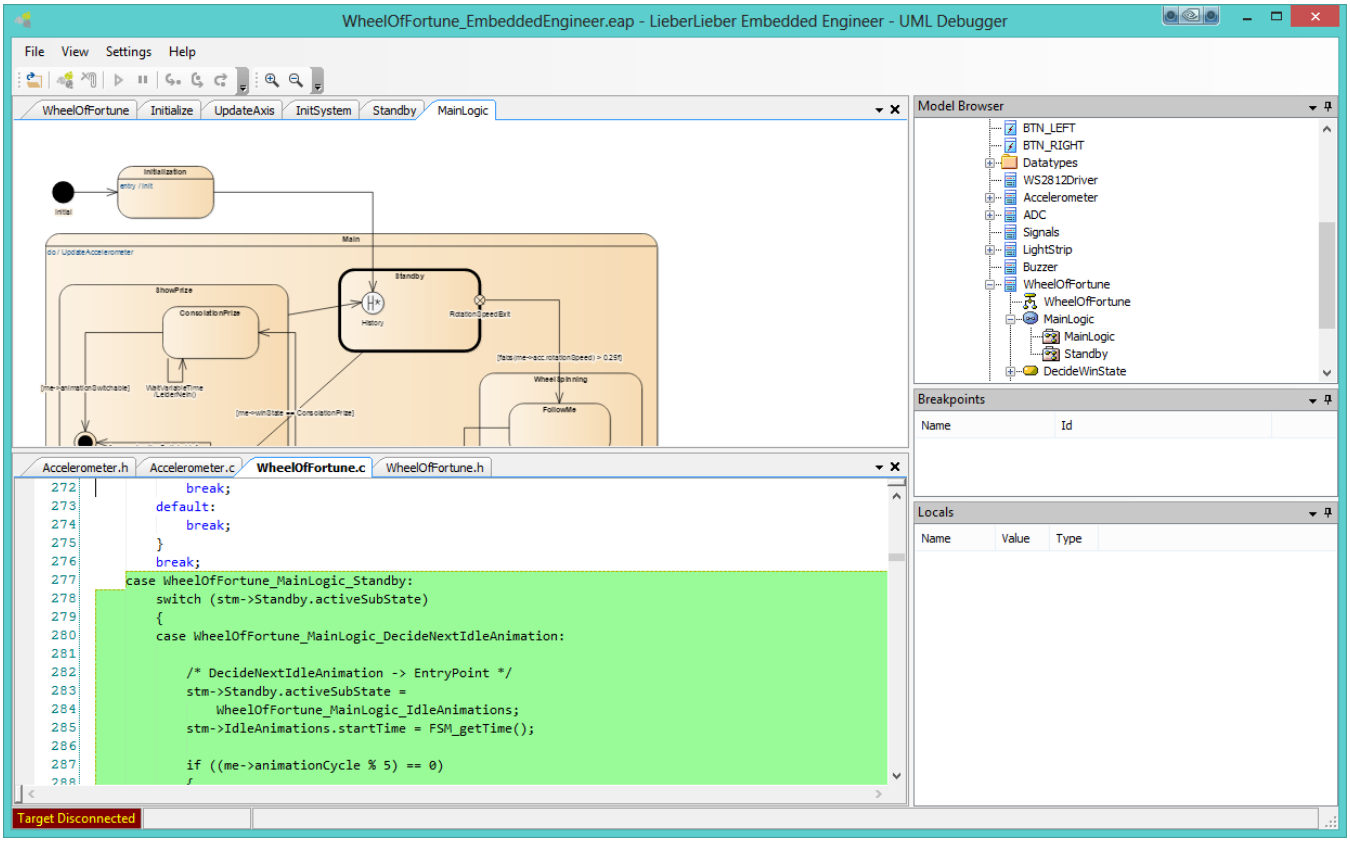
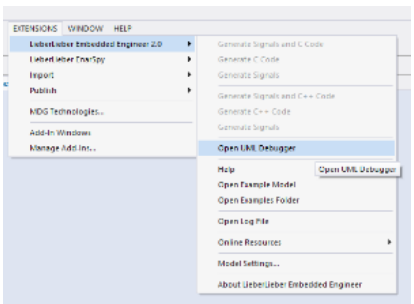
UML Debugger

The UML Debugger establishes the link between the generated code and the model at runtime.



How to start the UML Debugger

In Enterprise Architect, with an opened Model, go to *Extensions → LieberLieber Embedded Engineer 2.0 → Open UML Debugger*, this will open the UML debugger with the currently opened model.



You can use the Model Browser to the right to browse the model, just like in Enterprise Architect.

> To simplify navigation, only Packages, Classes, Diagrams, Activities and State Machines are shown in the Model Browser

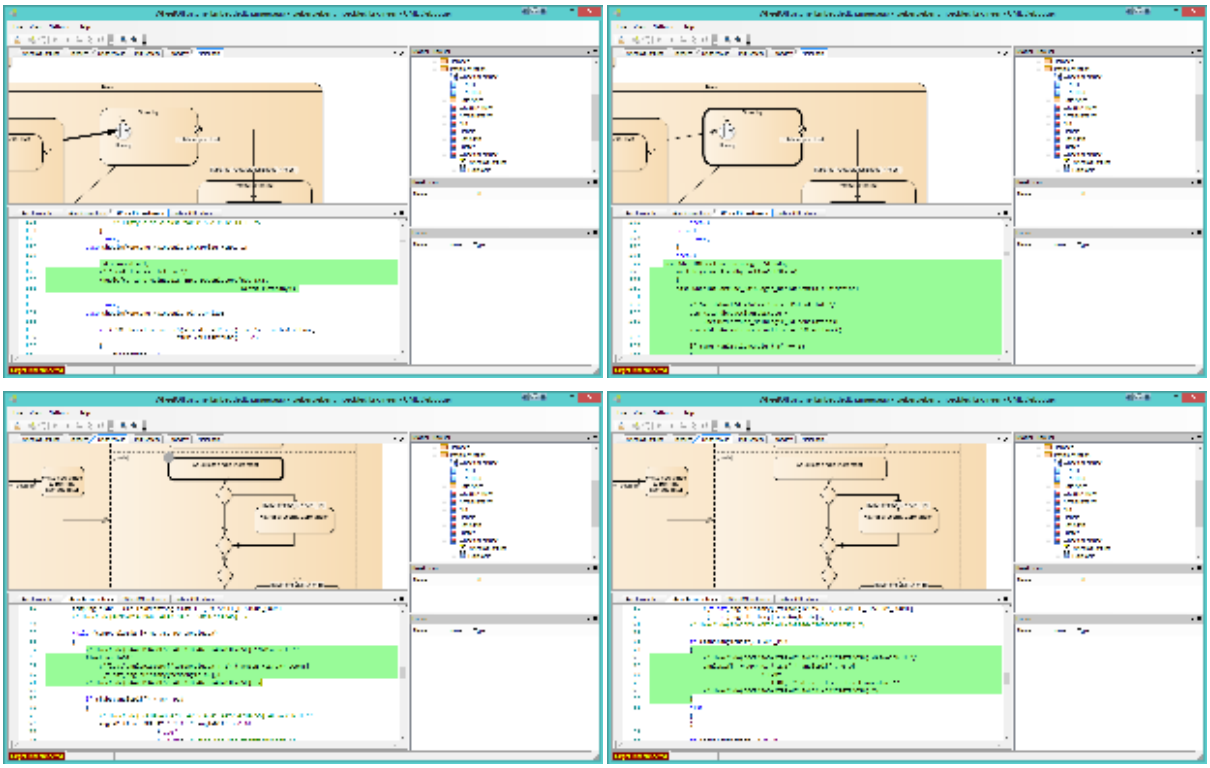
The main area of the UML Debugger consists of a diagram view and a source code view. Code generated for selected elements in the diagram view will be highlighted in the source code view.

Exploring Model and Code

You can use the UML Debugger to gain a deeper understanding of your model, as well as the code generated for it. Select the elements you're interested in in the diagram viewer, and examine which code was generated for it by browsing the source code.

This feature is particularly useful with the following elements:

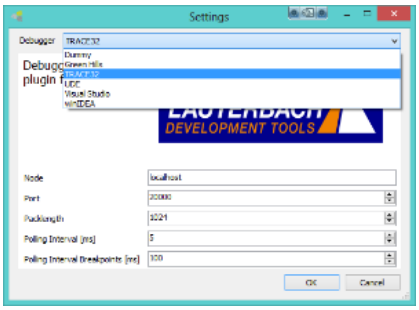
- Transitions
- States
- Actions
- Pins
- Control Flows



Debugging

Setting up the debugger

To debug on your target, first configure the connection by selecting *Settings → Connection Settings*. Select your debugger in the drop down menu.



Different debuggers require different settings. For the settings we've used the terminology of the target debugger vendors, please consult their documentation to find out how to configure the target debugger and the correct value for the settings.

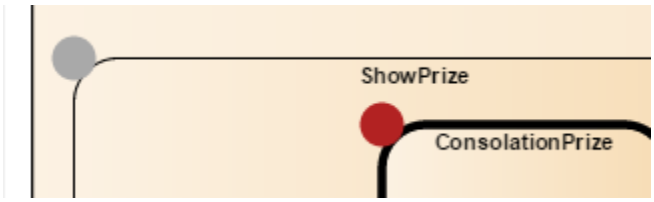
Connecting the debugger

Now that you've configured the debugger, connect to it by pressing on the *Connect Debugger* button ( ). In the lower left corner, you should see the status *Target Connected.*

Using the debugger

The UML Debugger offers very basic debugging capability. You can add breakpoints on states and actions, and you can examine the values of properties of the active class.

Adding breakpoints on diagram elements is done by hovering over the element, and clicking on the breakpoint which appears in the upper left corner of the element.



> You can set breakpoints in the diagrams as well as in the source code.

Note that breakpoints are synchronized between the target debugger and the UML debugger. Whenever you encounter a debugging situation which requires the debugging power of the target debugger, just do your debugging tasks there.

> Conditional breakpoints are one example for this. Set a breakpoint in the UML debugger, switch to your target debugger, and modify the breakpoint to be a conditional breakpoint. You can also set the breakpoint in the target debugger without using the UML debugger at all - the views (source code- and diagram-) are constantly updated and include fully custom breakpoints.

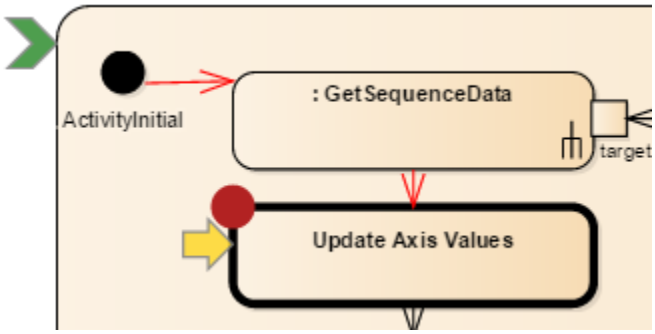
You can also step through your code with the use of the stepping commands. Note that stepping is done on HLL-level, meaning that a step isn't done on the UML level but on C/C++ instead.
This has solely technical reasons and is subject to change.

Troubleshooting / Frequently Asked Questions

What do the green and yellow indicators mean?

You'll notice that there are green and yellow arrows/indicators when debugging your model. This distinction is borrowed from *Microsoft Visual Studio*. Put simply, a green arrow symbolizes that the element pointed to is currently active, that is, the instruction pointer is somewhere between the start and the end of the code generated for the element. The yellow arrow means that the element pointed to will be executed next.

Note that there can only be one yellow arrow, while there can be an arbitrary number of green arrows at a single instance.



I can open and browse the Model, but no source code is shown

UML Debugger relies on the tagged value *LL Embedded::GenSrcDir* to locate the generated source code of a package. Make sure that this tagged value points to the right location, and that the files are accessible from your user account. When debugging, the file information from the target debugger is used to supplement the information about file paths and names, so even if the location of the source code has changed in the meantime, UML Debugger is still able to find the files using this supplemental information.

The source code is shown when I click on a class, but no location is shown for states, actions, activities …

There are a number of possible reasons for this error:

- The model has changed in major ways since generating the code
  - This can happen if, for example, the GUIDs have been reset in the model. Please try to regenerate the source code.
- The source code has changed
  - UML Debugger calculates a checksum for each file, and compares it with the checksum in the debug information file. If the checksums don't match, UML Debugger will refuse to load the debug information as it's impossible to ensure the validity of the debug information. Please try to regenerate the source code.
- When clicking on certain elements, no code is highlighted in the source code view
  - While most elements in a state- or activity-diagram lead to generated code, some have no real resemblance in the source code. Such elements include certain pseudostates, merge nodes, object flows, ...

I can't set any breakpoints

1. Make sure that you're connected to a debugger. You can't set breakpoints while no debugger is attached.
2. Make sure that you can set breakpoints directly in your IDE. UML Debugger relies on the IDE for this functionality, so it can only set breakpoints when the IDE allows it to do so.
3. You can't set a breakpoint on every element. For example, certain pseudostates or merge nodes might have no code attached, and therefore no breakpoint can be set on these elements. Breakpoint functionality heavily depends on the debugger of the IDE. Some debuggers restrict the number of breakpoints you can set, which can lead to this problem.
   There are also some debuggers which don't always remove breakpoints, but sometimes only disable them, which can also lead to problems.