SWG - Rational

Test RealTime 7.5.0 Integration side by side with Rhapsody 7.4

Document goal

The goal of this document is:

- to explain how to setup Test RealTime 8.0 in Rhapsody 7.4
- to demonstrate with a Rhapsody example how this integration works

Test RealTime for Rhapsody Installation

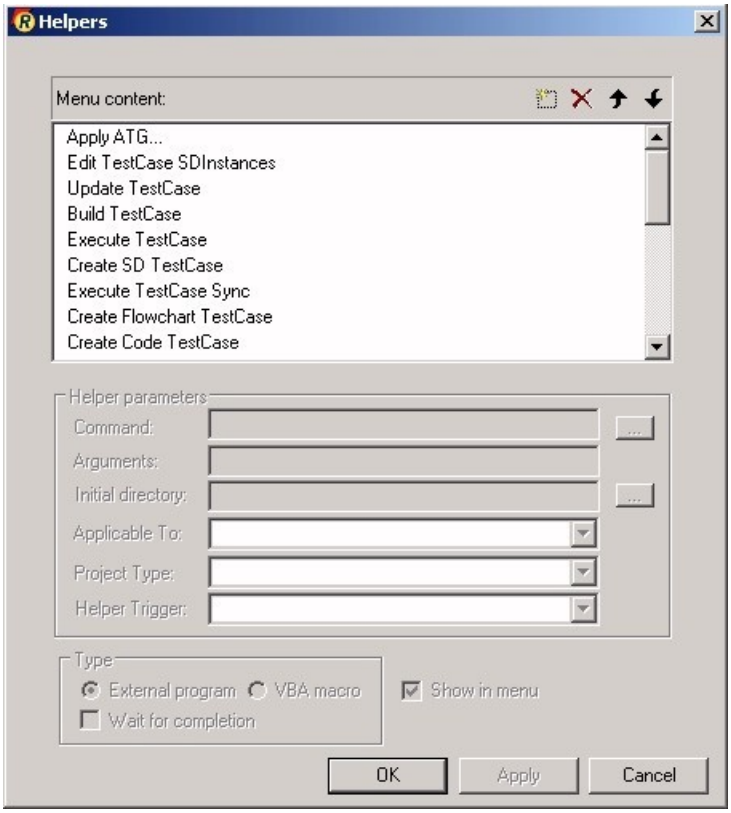
1.1 Automatic Installation of Test RealTime integration in Rhapsody

To install the TestRT Rhapsody plugin automatically, run the following command:
TestRTperl.exe "–I%TESTRTDIR%\lib\perl" "%TESTRTDIR%\lib\scripts\RhapInstall.pl"
This process should copy required files in the Rhapsody install and create the customizable menus in the Rhapsody GUI.

1.2 Manual Creation of Test RealTime menus in the customizable Rhapsody Tools menu

This creation have to be done only if the automatic installation did not work.
Nevertheless the customized menu can be checked.
Launch Rhapsody and open your project:
e.g. C:Telelogic\Rhapsody74\Samples\CppSamples\Pacemaker\pacemaker.rpy

Then in the Rhapsody Tools menu select Customize…
The following dialog box opens:



Let's now create the Test RealTime Results menu.

Click on the square icon  to create a new customized menu.

In the Menu content list, type Test RealTime Results and then fulfil the Command field below with:
`RhapTestRTreport.exe`
The Arguments field is left blank.

If not done already, Tick the check box "Show in the Tools menu".
Once you are done, click the OK button.

Repeat the same sequence of operations for a last TestRT menu entitled Test RealTime Clean.
You must use the following Command: `RhapTestRTreport.exe`
In the Arguments field, please type: **`-cleanall`**

Repeat the same sequence of operations for a last TestRT menu entitled Test RealTime Clean Results.
You must use the following Command: `RhapTestRTreport.exe`
In the Arguments field, please type: **`-clean`**

Repeat the same sequence of operations for a last TestRT menu entitled Test RealTime Enable.
You must use the following Command: `RhapConf.exe`
In the Arguments field, please type: **`-add "$OMROOT/Profiles/TestRealTime.sbs"`**

Repeat the same sequence of operations for a last TestRT menu entitled Test RealTime Disable.
You must use the following Command: `RhapConf.exe`
In the Arguments field, please type: **`-remove "$OMROOT/Profiles/TestRealTime.sbs"`**

These customized menu will be then available for all project.

Test RealTime configuration for Rhapsody project

Launch Rhapsody and open your project:
e.g. C:Telelogic\Rhapsody74\Samples\CppSamples\Pacemaker\pacemaker.rpy

1.3 Enable TeatRealTime RuntimeAnalysis

menu Tools > Test RealTime… Enable

this make multiple commands in One action:

1. add a Reference on the TestRealTime package to the loaded model:
   Menu File > 'add to Model' the package <RhapsodyInstall>/Share/Profiles/TestRealTime.sbs.
2. activate the TestRealTime stereotype for the current configuration:
   select the configuration > feature > stereotype > RuntimeAnalysis
3. add the TestRealTime package to the active component scope:
   select the component > feature > scope > TestRealTime

Note: This action may be done multiple time without any problemes

1.4 Disable TeatRealTime RuntimeAnalysis

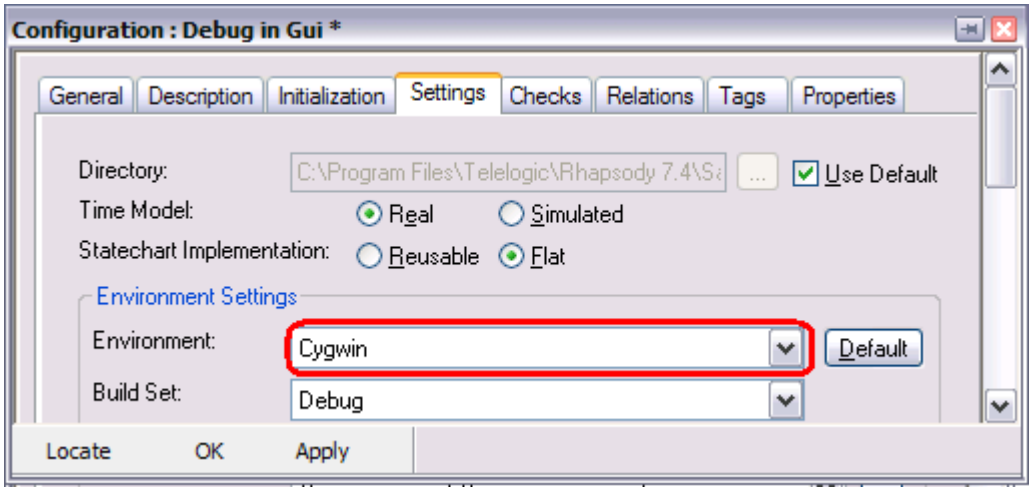
menu Tools > Test RealTime… Disable

this make multiple commands in One action:

1. disable the TestRealTime stereotype for the current configuration:
   select the configuration > feature > stereotype > RuntimeAnalysis
2. remove the TestRealTime package to the active component scope:
   select the component > feature > scope > TestRealTime

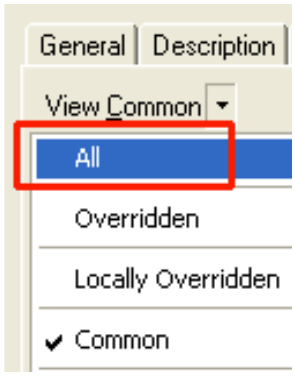
while the RuntimeAnalysis stereotype is checked in this configuration the build will be done through the Test RealTime instrumentation using the right TDP.

The TDP is then automatically choosen in accordance with the used Settings : Environment

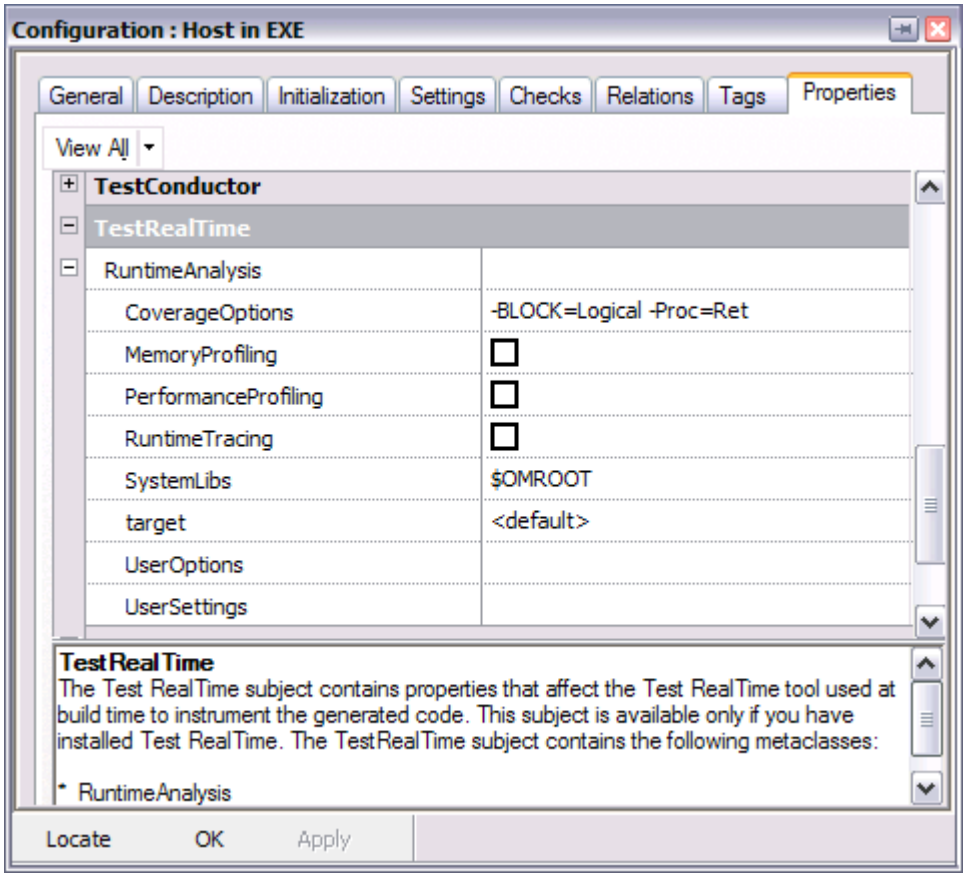


1.5 Change the Test RealTime instrumentation options

From the same dialog box as above, **select** the Properties tab and **change** the View to All:



Scroll down and Select the **TestRealTime** property and **expand** it.
The online Help is available for each item.



select **UserSettings** to add **--ATC_ON_THE_FLY=2** ; This setting is used to generate the differential coverage result on the fly without any 'pragma attol' insertion into the code sources.
=2 here is the number of method enter between each results flush. Range is [1.. 4294967295]

select **UserOptions** to add **-NO_TEMPLATE_NOTE** ; This option allow to avoid to insert notes for class template.

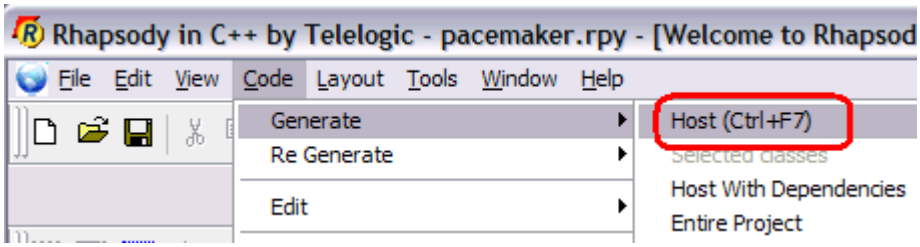
SystemLibs property allows to set the option –noinstrdir=<SystemLibs>. Values must be separated by a comma.
Target property allows to force Test RealTime to use an other TDP than the predefined one.
Modify the properties as wanted. E.g add **–verbose** in the **UserOptions** property
Finally **click OK**.

Note: The **verbose** option is used to generate more logging messages at instrumentation and build time, and to avoid analyzing the files belonging to the Rhapsody framework.

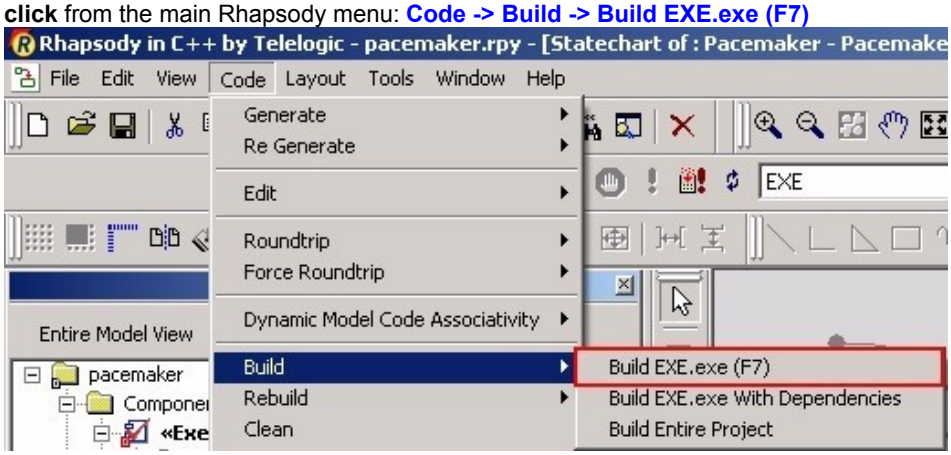
instrumentation with Test RealTime in Rhapsody

1.6 Generation of the code with instrumentation enabled

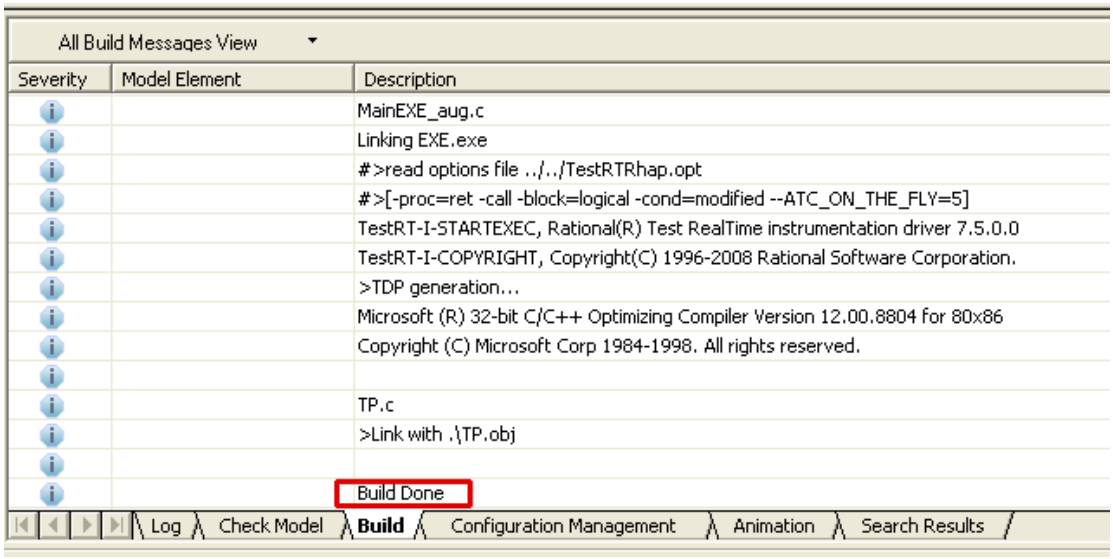
click from the main Rhapsody menu: **Code -> Generate -> host (Ctrl+F7)**
To make sure the TestRealTimeObject will ba part of the application.



click from the main Rhapsody menu: **Code -> Build -> Build EXE.exe (F7)**



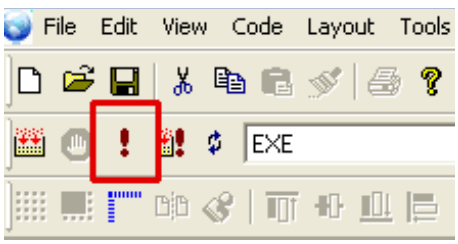
Note: Only the component with the newly modified configuration will be instrumented by TestRT. At build time, pay attention to the appearing instrumentation messages.



When you arrive at this point of the build process, the Pacemaker application has been reconstructed with the TestRT code coverage instrumentation.

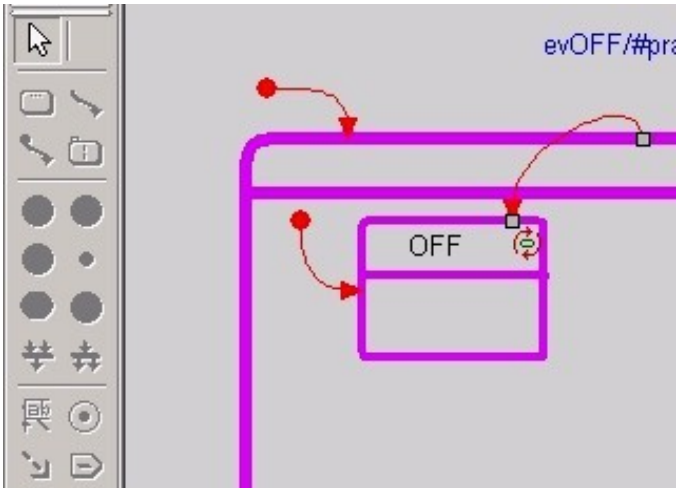
1.7 Run the instrumented application

To proceed, **click** on the following icon:

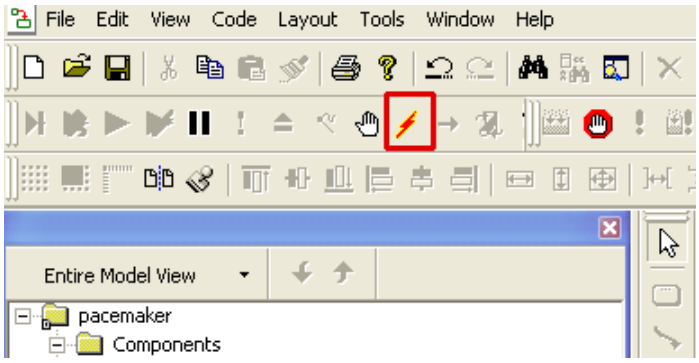


Then **click** this icon: 

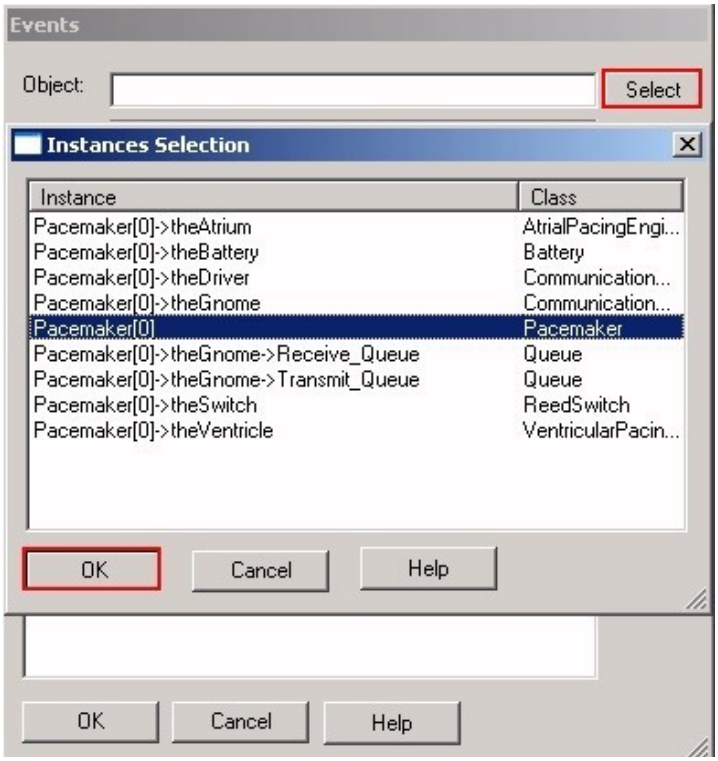
The **Animated Statechart** should be at the **OFF** state as shown below (make sure that the statechart animation is turned on):



Now we can start sending events to stimulate the system under test. To proceed, **click** on the **Event Generator** icon:

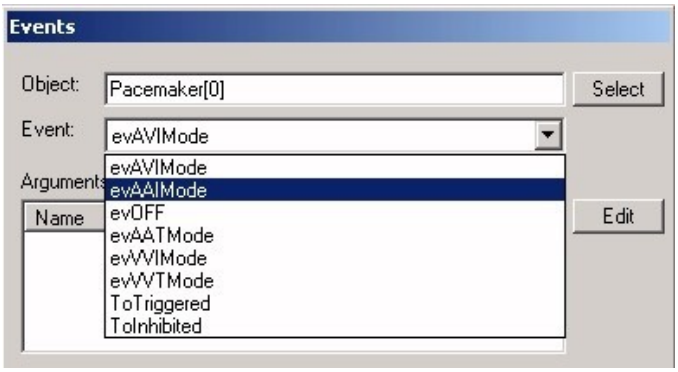


A new window named Events pops up and select the **Pacemaker[0]** instance in the drop down menu:

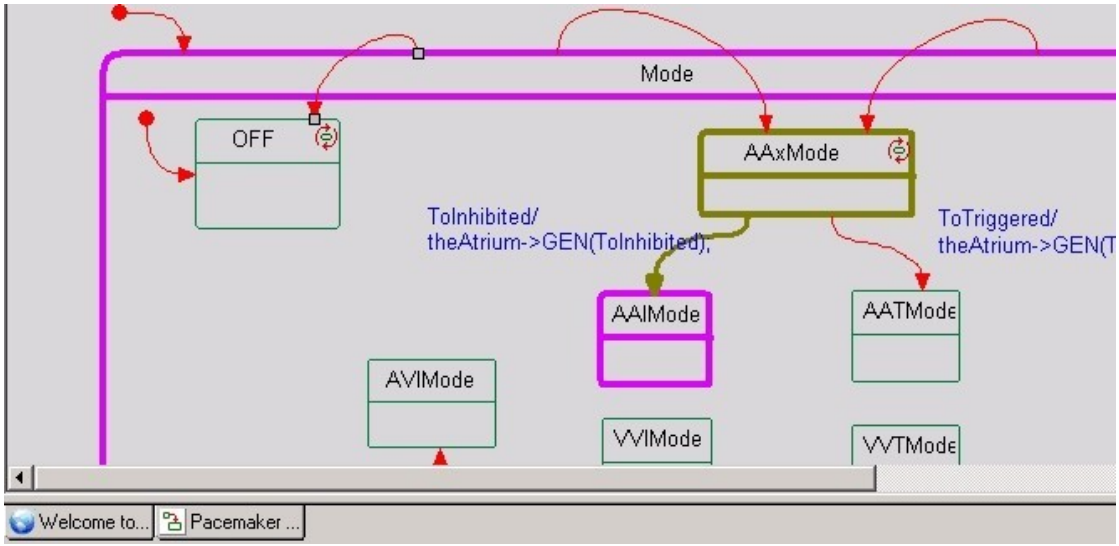


Click OK.

Then **pick up** the **evAAIMode** even in the drop down selection of events and **click OK** to send it to the Pacemarker.

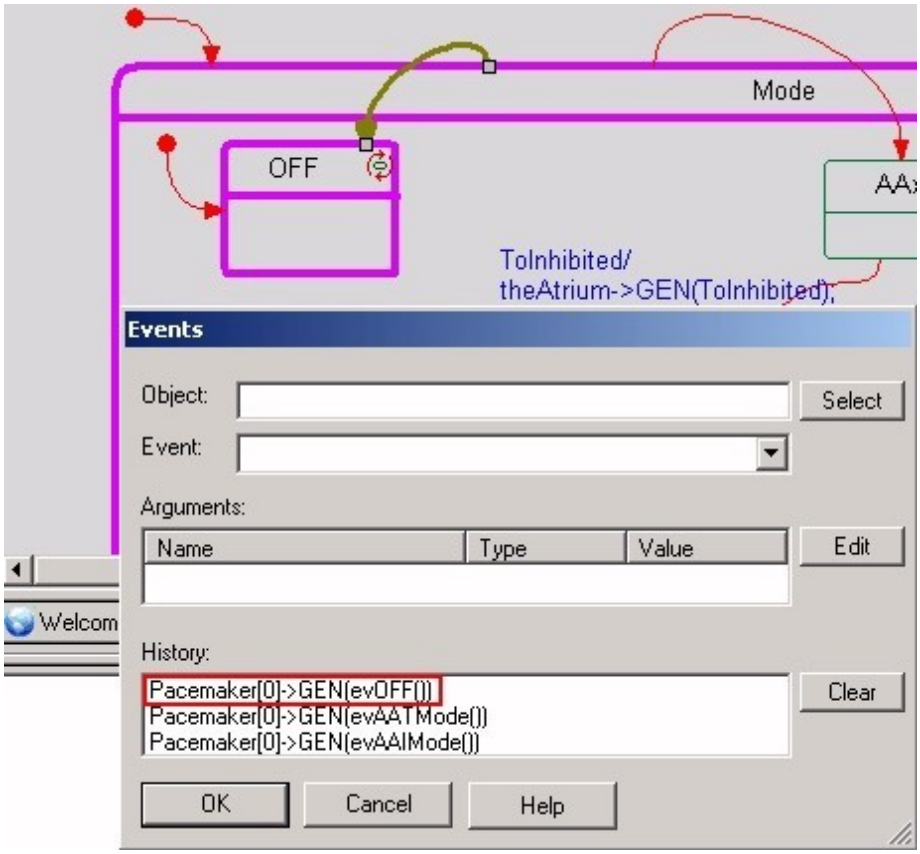


You should see that the corresponding state has been reached in the animated chart:



Proceed in the same way to inject a second event (**evAATMode** for instance).

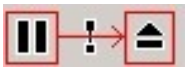
Feel free to play a little bit with animation as long as you inject as a last event **evOFF** which permits to fire the transition which contains the test results dump command.



1.8 Dump Test RealTime Results

When you done your tests it is time to request the TestRealTime RuntimeAnalysis results.

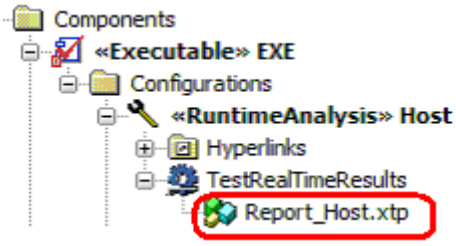
click on the **Event Generator** as shown above, then select the
TestRealTimeObject-> TestRealTimeDumpResults event.

Then break and quit the animation: 

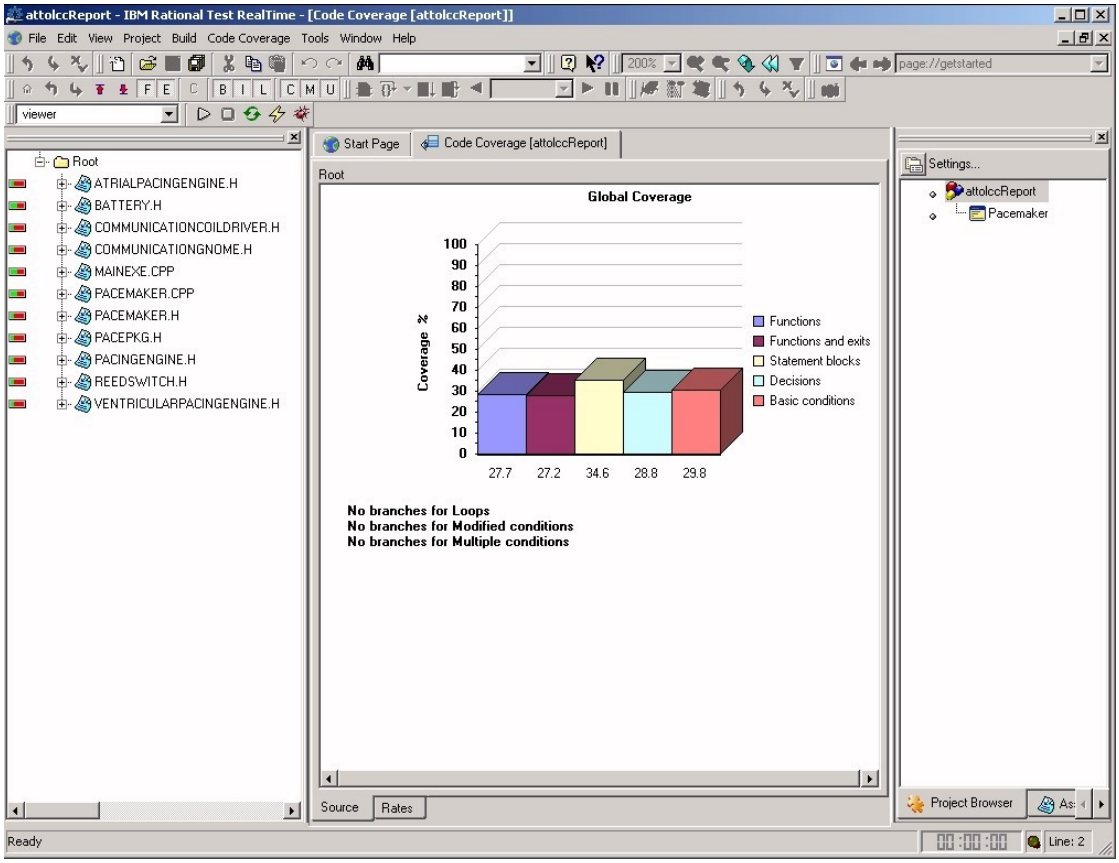
And let's have a look at the generated Coverage results…

1.9 Evaluate Test RealTime Code Coverage Report

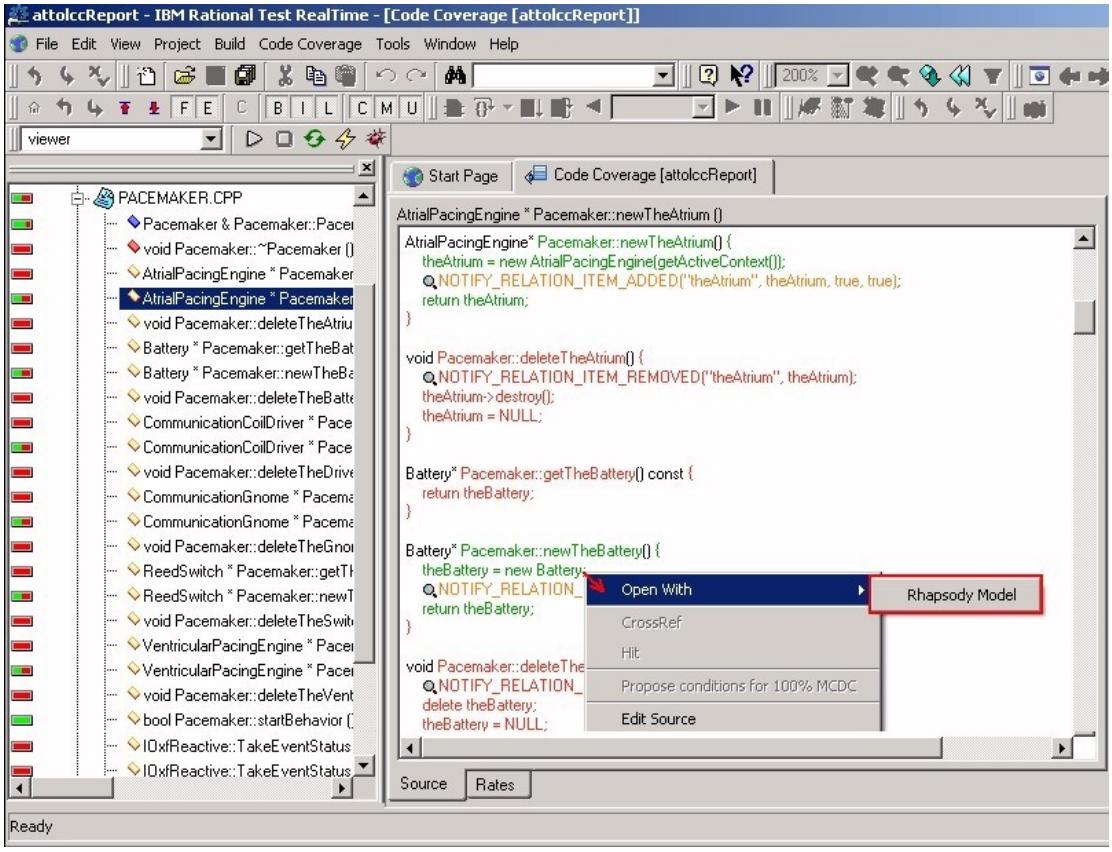
From the Rhapsody Model browser, **double-click TestRealTimeResults/Report_Host.xtp**:



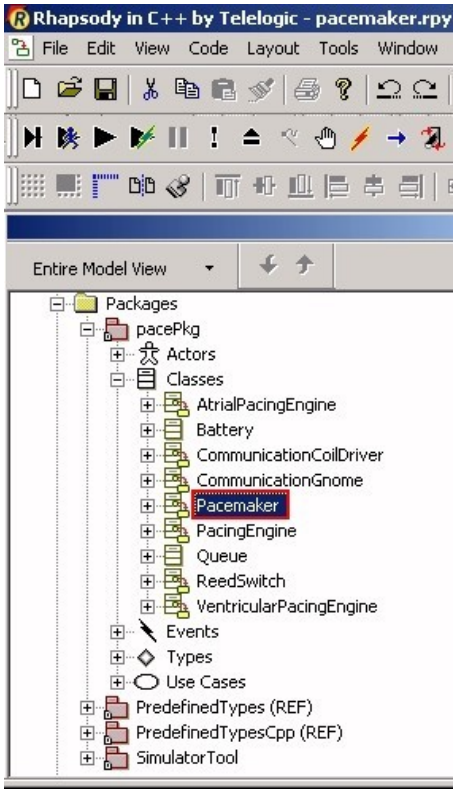
That should open the Code Coverage results report in TestRT studio viewer:



The source code coverage results are linked with the Rhapsody model elements. To evaluate this feature, please bring a piece of code coverage report in the Source tab and **right-click-hold** a covered statement block and **select**: **Open With -> Rhapsody Model**

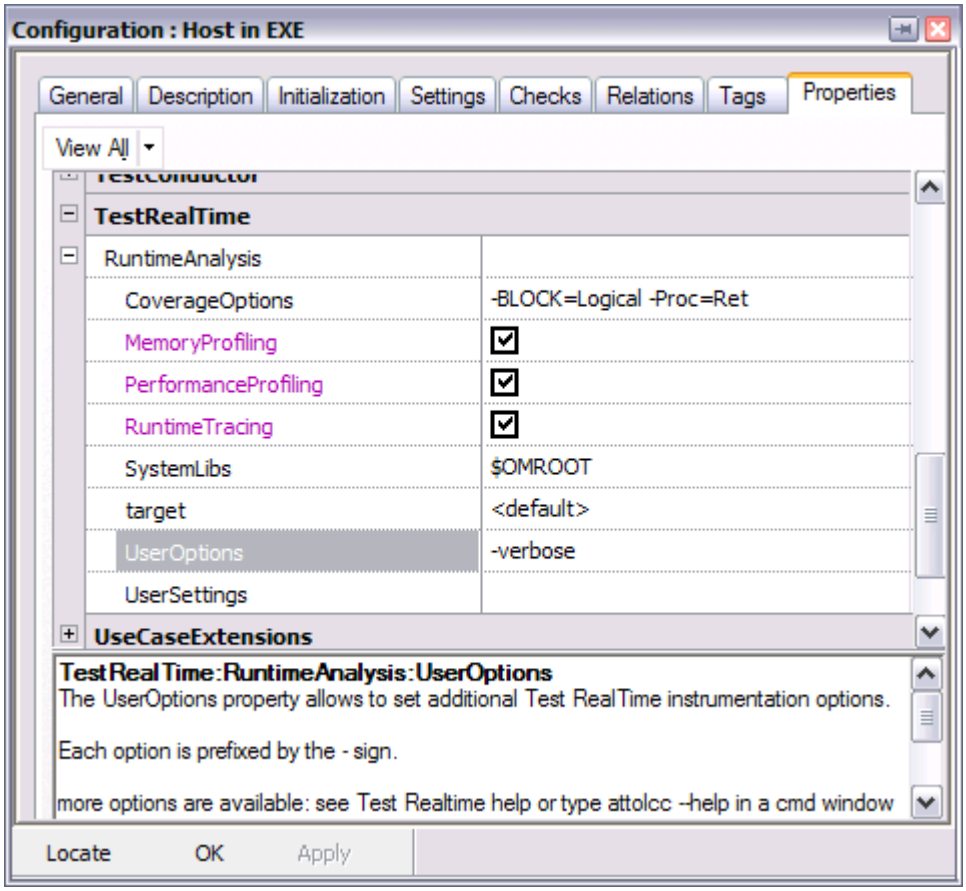


If you bring the focus on the Rhapsody GUI, you can see the **corresponding class** is **highlighted** in the model:



1.10 Generate more Runtime Analysis Results

From the **Test RealTime Properties**, tick the **memoryProfiling, performanceProfiling** and **runtimeTracing** boxes:

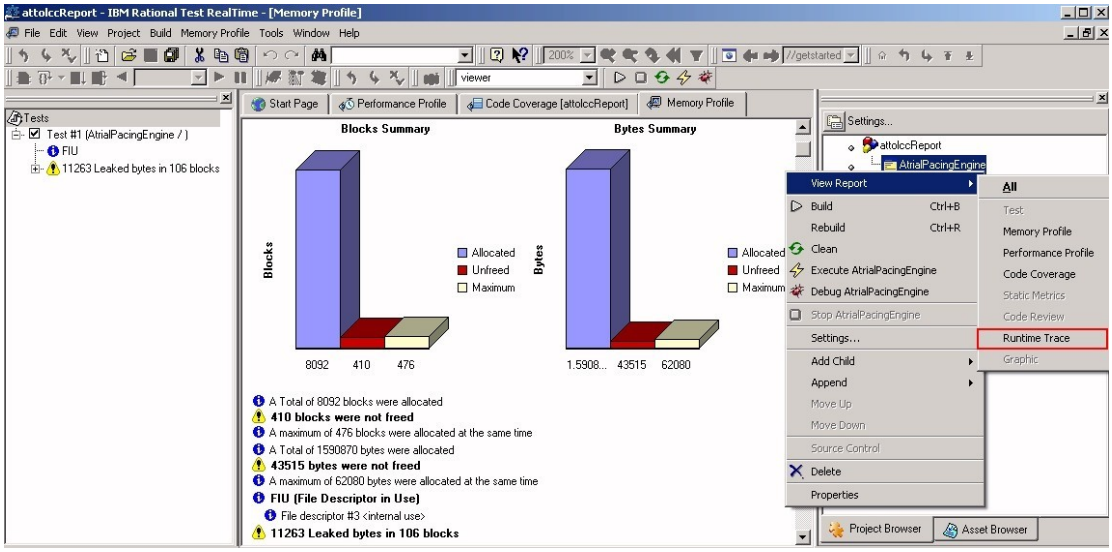


Click **OK** when done

then **rebuild** your component to be instrumented:

Run the instrumented application one more time and play with the **Animated Statechart** as described in **Run the instrumented application** (do not forget to shoot the event **evOFF** which permits to fire the transition which contains the dump command).

From the Rhapsody Tools menu, **select Test RealTime Results** to bring Test RealTime Studio with the corresponding reports:

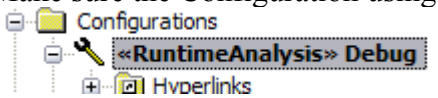


IMPORTANT NOTES:

- Only the code coverage, memory and performance profiling reports automatically displayed. For the runtime trace report, you need **to right-click** the **AtrialPacingEngine** test node on the right-hand side of the UI and then **select View Report → Runtime Trace** (this small bug should be fixed when the official 7.5 version of TestRT is released).
- You may have noticed that the memory profiling report shows many memory leaks. It comes from the fact that the Rhapsody Framework performs memory allocations which are not seen at the application level. To work around that, we need to instrument the Framework as well.

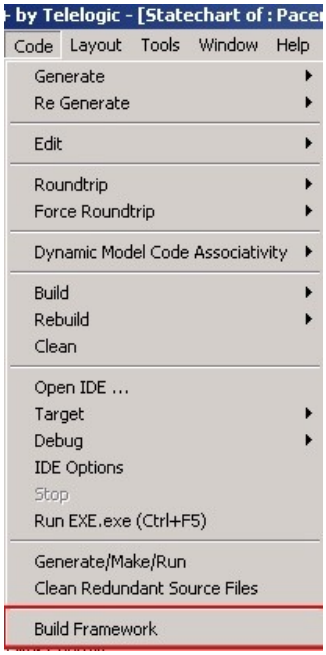
1.11 Instrument the Rhapsody Framework to get accurate memory profiling results

Make sure the Configuration using the TestRealTime stereotype is selected.



make sure the property TestRealTime::RuntimeAnalysis::memoryProfiling is ticked to involve the memory profiling feature.

Once it is done, rebuild the libraries of the Rhapsody Framework with the appropriate memory profiling tracking by **clicking** from Rhapsody main menu: **Code → Build Framework**

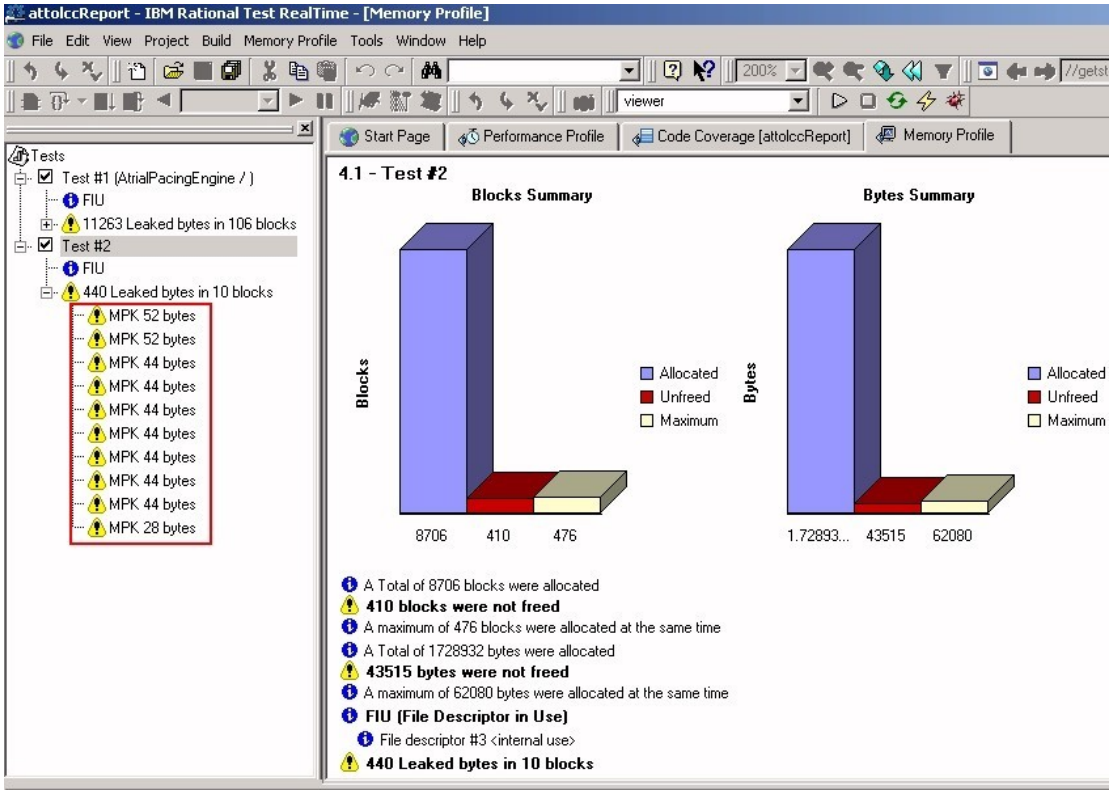


Once the Framework instrumentation and build process is over (it takes approximately 5 minutes in total), **rebuild** the Pacemaker **EXE** file to be able in order to integrate the new instrumented libraries of the Framework in the final executable code.

Note: since Microsoft environment uses framework as DLL the instrumentation of this FrameWork is more complex to instantiate than for other environment like Cygwin which uses Framework as archive. So FrameWork instrumentation process for Microsoft environment is not detailed here.

Basically, **complete** all the steps of the chapter §3.6 (except the first one about the selection of TestRT instrumentation options) in order to re-run the Pacemaker code with the animation and generate a new Memory Profiling report at the end.

This time, the new instance of the memory profiling report contains only potential memory leaks (MPK):



Well, this is why you should use a verification testing tool… These MPK are actually present in the animation code of the Framework. This information has been passed to the Rhapsody development team which should fix the problem in the upcoming releases. In the meanwhile, you can work around that by:

- Saying it's just a memory warning and not a severe error (it affects the animation code only)
- Turning the animation off to avoid generating the MPK messages…

Test RealTime Manual configuration

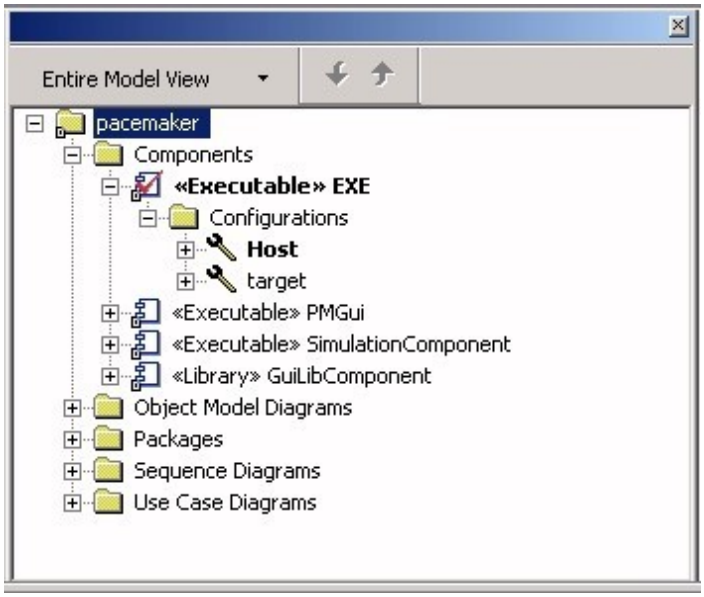
This chapter explains How the TestRealTime package can be used to in your project to instrument and get results. The TestRealTime package is made of the TestRealTimeObject and the Stereotype RuntimeAnalysis.

The goal of the TestRealTimeObject is to insert a simple class/object to the model in order to have access to TestReaTime results Dump whatever the state of the application.
So At any time the event TestRealTimeObject->TestRealTimeDumpResults() can be sent to the application to dump TestRealtime results.
This package can be removed from the component scope as soon as stereotype RuntimeAnalysis is not used (instrumentation is off) OR when the user uses its own mechanism to dump the TestReaTime results like the pragma attol insert _ATCPQ_DUMP(1); in any transition.

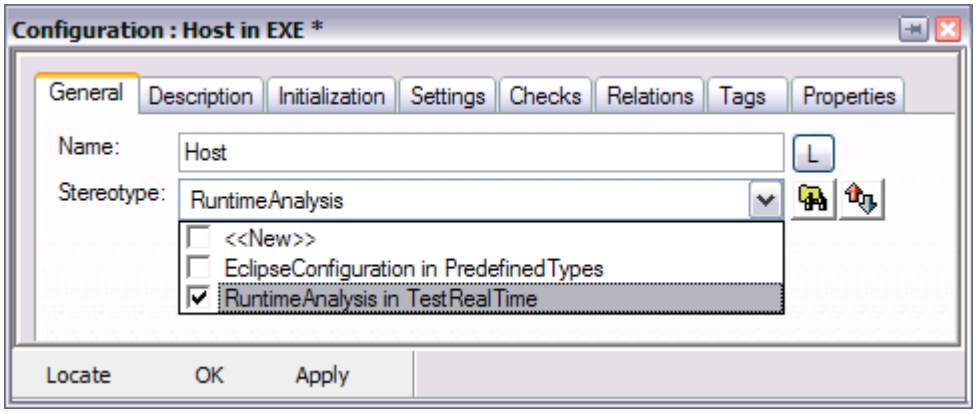
The goal of the RuntimeAnalysis stereotype is to change the InvokeMake property to use TestRT.bat instead of the .bat dedicated to the build environment. It is the responsibility to TestRT.bat to run the requested .bat once TestRT variables are set.
The RuntimeAnalysis stereotype offers TestRealTime properties used by the instrumentation.

1.12 Use the Test RealTime Stereotype for your configuration

Once the TestRealTime package added to the model, the following actions can be done to enable/disable the instrumentation:
In the project tree, expand the Components folder, then select the component named **<<Executable>> EXE. Right click -> Set as Active Component.**
It appears now with a bold font as show below:



Let's now use the **Test RealTime stereotype** for the active Configuration node.
Select the existing **Host** configuration and (double-click) to open the configuration dialog box.

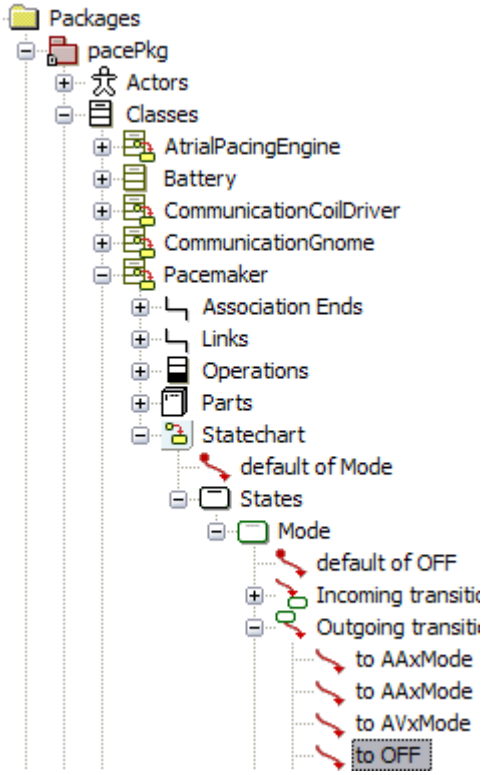


click in the Stereotype listbox to select RuntimeAnalysis in Test RealTime

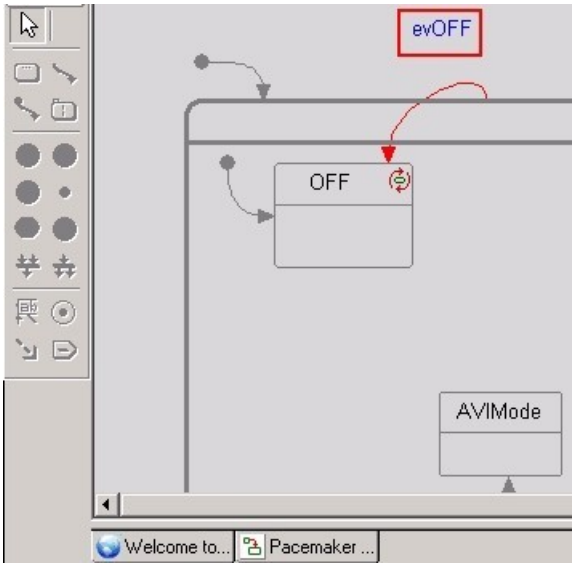
1.13 Add a Test RealTime analysis dump point in the model

This is a means to get results without using the additional TestRealTimeObject package.
Test RealTime has been designed for embedded and real-time systems which frequently never end. Therefore runtime analysis results are evaluated in the target memory endlessly until you decide to dump them back to the host machine.

Let's add such a test results dump command in one of Pacemaker transitions.
The Pacemaker statechart should be present in the main view. If it is not the case, please open the model as follow:

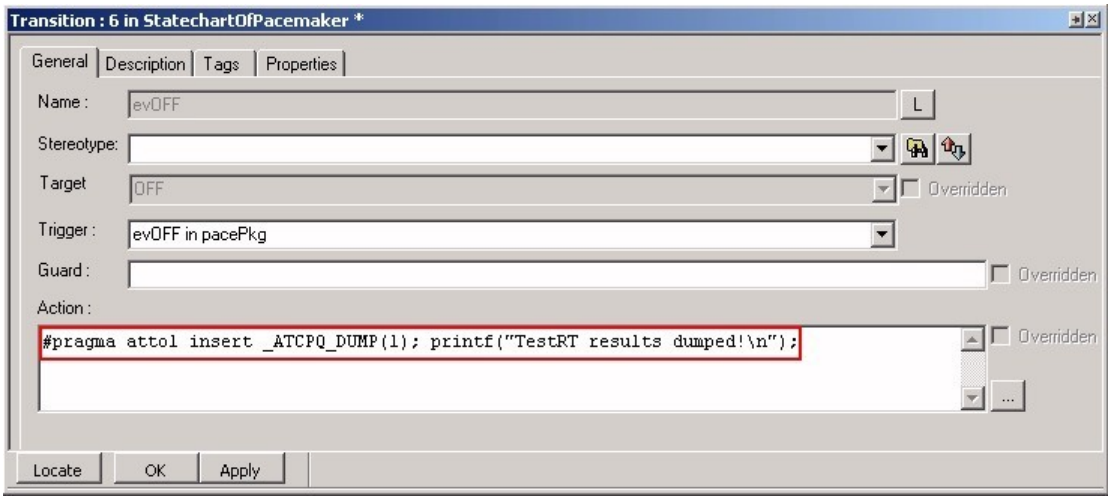


Then **select** the transition **evOFF** as shown above
Right-click and select **Locate On Diagram** to open the following diagram



then **double-click** it:
In the **General** tab **action** box, please type the following statements:

#pragma attol insert _ATCPQ_DUMP(1); printf("TestRT results dumped!\n");



Click the **OK** button.

That's all folks!