



# Embedded UML RXF Restrictions

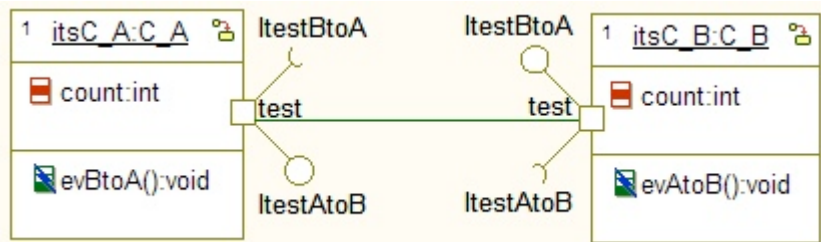
The Embedded UML RXF is to be used with Rhapsody in C, so we are dealing with several interfaces:

- modelling. Rhapsody in C supports the so-called **Operating System Abstraction Layer**, but OSAL also includes services for which no code is generated like semaphores.
- generated C code. The C code generated by Rhapsody is not using the Object eXecution Framework by Telelogic, but the RXF by Willert Software Tools instead. The Framework by Willert Software Tools uses slightly different data structures and a different organized set of include files, which is why we need to use those instead.
- files for properties and profiles.
- IDE. The generated files will be deployed into your IDE project.

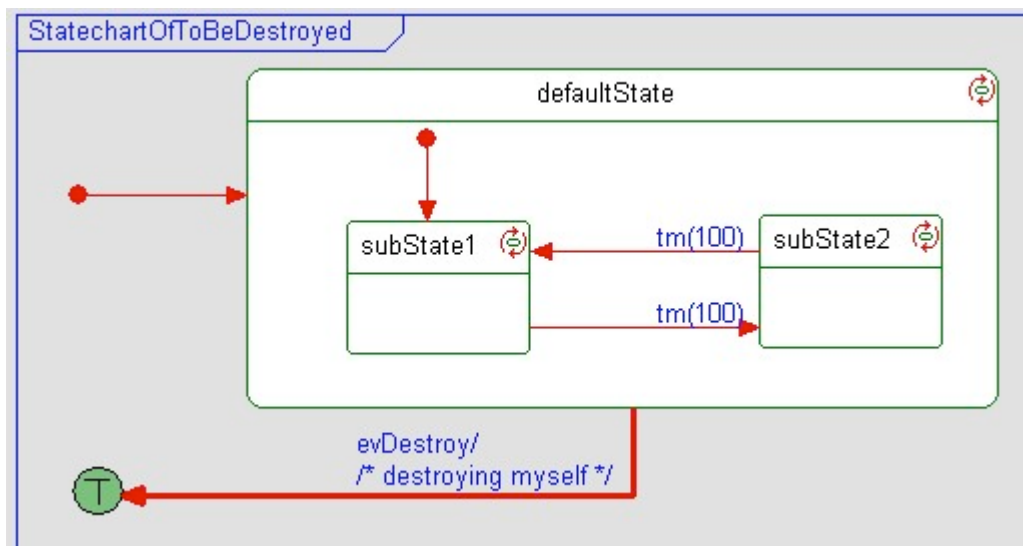
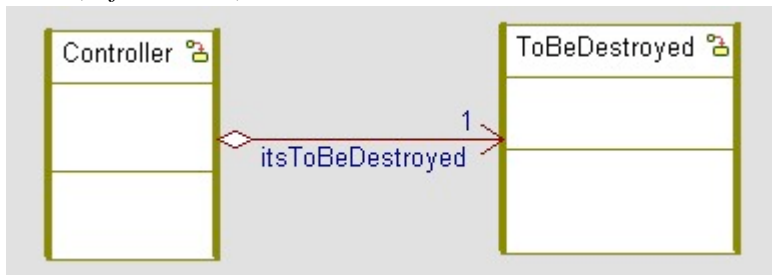
## Modelling

Almost all UML elements can be used in Rhapsody in C with the RXF by Willert Software Tools. This includes classes, objects, singleton and active objects, files, statecharts, several kinds of relations etc. However some UML elements can not be used together with C code generation:

- Inheritance:  
Inheritance is not supported by Rhapsody in C, except inheritance for interfaces which is available as of Rhapsody in C version 7.1.
- Activity Diagrams associated to Operations:  
Code generation is only supported for activity diagrams and statecharts which are associated with a class, an object or a file. Activity diagrams can also be assigned to operations, but no C code can be generated from them by Rhapsody.
- Ports:  
Communication via UML ports is not yet implemented for C code generation in Rhapsody 6.2, but as of Rhapsody 7.0. In products based on the RXF and an RTOS, the constant `WST_PORTS_DISABLED` is set to disable support for UML ports in the RXF because Ports are not implemented.



- Timers. The Framework uses its own `RiCOSTimer` object for the Bridge Task, which handles the timeout events. The Framework does not support further timers.
- Destroying objects remotely from another task: Having two active classes *Controller* and *ToBeDestroyed*, it is not possible to destroy *ToBeDestroyed* by just calling `ToBeDestroyed_Destroy()` from *Controller*. It would not always remove the waiting timeouts and events from the queues, which could cause an undefined behavior. However, there is a simple workaround how *Controller* can get *ToBeDestroyed* to destroy itself. *Controller* can send an `evDestroy` to *ToBeDestroyed*. The statechart of *ToBeDestroyed* must in any state be able to react to that event, which is possible by surrounding the actual statechart with one root state. The transition triggered by `evDestroy` must connect the root state (`defaultState`) to the termination connector.



## Renesas M16C only

- Known Issue #1355: the Telelogic Container Class `RiCString` does not support far string constants for M16C. As a workaround, the macros `STR_ARGUMENT_TYPE` and `STR_RETURN_TYPE` are introduced in `WSTCompiler.h`. The use of the `RiCString` Class is discouraged.
- Known Issue #1440: the Renesas toolchain uses the keyword `_far` for dynamic memory

using malloc(), free() etc. We advice to use the StaticComponent stereotype which prevents usage of these dynamic memory functions

## RXF

The following features are not supported by the RXF based on an RTOS:

- Sockets  
The RXF does not support communication via TCP/IP due to the heavy overhead. As a consequence there is no support for animation. There is no RiCOSSocket class.
- some UML elements can't be used together with C code generation. These are:
  - Inheritance  
To allow inheritance in the C language, all operations would need to be implemented as callbacks or in a virtual function table. This would increase the code size, RAM needs, runtime and calling operations from user code would be more complicated. Thus no inheritance is available in Rhapsody in C.
  - Activity Diagrams associated to operations:  
Code generation is only supported for activity diagrams and statecharts which are associated with a class, an object or a file. Activity diagrams can also be assigned to operations, but no C code can be generated from them.
  - Ports  
Communication via UML ports is not yet implemented for C code generation in Rhapsody 6.2, but as of Rhapsody 7.0.  
The constant WST\_PORTS\_DISABLED is set to disable support for UML ports in the RXF in a profile which comes with your release.
  - Timers. The Framework uses its own RiCOSTimer object for the Bridge Task, which handles the timeout events. The Framework does not support further RiCOSTimer objects.

## IDE

The demo versions of our products use an internal build process within Rhapsody. All non-demo versions of our products use an external build process inside the IDE of the respective toolchain. For that, a deployer is used which copies the generated files to an IDE project and in most products adds the names of the files in the IDE project file. For non-demo products: the Deployer requires the Sun Java VM 1.6 build 6 or higher.

### All IDEs except uVision

The Deployer does not support the Rhapsody feature *Code / Clean Redundant Source Files*. Rhapsody will remove redundant source files, but when these are already deployed to your IDE, those files are not removed from your IDE project.

You can use a workaround for now, by instructing Rhapsody not to use the Default path to generate files which also enables roundtripping

### **CodeComposer and MPLAB IDEs only**

The Deployer does not integrate the generated files in the Code Composer or MPLAB project file, but just copies these. The GettingStarted example includes 'hard coded' the names of the appropriate files.

For any new project, you must add the files by hand in your IDE. Alternatively you can implement the IDE Bridge.

### **Renesas HEW IDE only**

The Deployer will insert files in your HEW project but only supports the HEW build configurations Debug and Release. These build configurations match the Rhapsody buildsets Debug and Release. You should not add another build configuration in HEW because after using the Deployer, HEW refuses to open the .hwp file.

Copyright (c) Willert Software Tools GmbH. All rights reserved.