

OO RTX Restrictions

OO RTX based products are to be used with Rhapsody in C, so we are dealing with several interfaces:

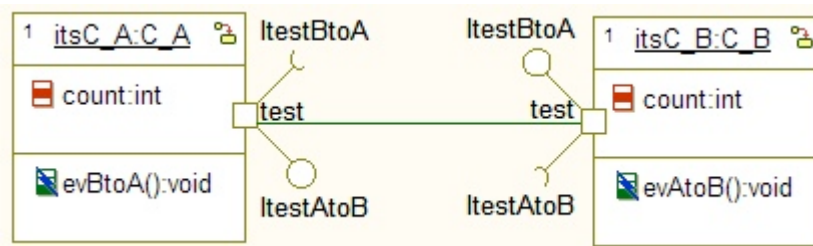
- modelling. Rhapsody in C supports the so-called **Operating System Abstraction Layer**, but OSAL also includes services for which no code is generated like semaphores. In a product which uses the OO RTX without extensions for an RTOS, real task switching and preemptive scheduling is not possible, but scheduling is done on statechart level by using messages and by using a heartbeat like for example a timer interrupt. As a consequence, no active classes are supported, but we will come back on this later.
- generated C code. The C code generated by Rhapsody is not using the Object eXecution Framework by Telelogic, but the OO RTX with optional extensions for an RTOS 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 OO RTX. This includes classes, objects, singleton 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.
- Active Classes:
The OO RTX does not support preemptive multitasking. Thus it is not allowed to use active objects in the UML model. Scheduling between the UML model objects is done event driven (via the event queue). This mechanism is based on the run-to-completion approach (e.g. a state's entry action commands need to be finished before another event can be consumed by any object).

- 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. Depending on your product, the property `CPPCompileSwitches` sets the constant `WST_PORTS_DISABLED` by default, to disable support for UML ports in the OO RTX to prevent extra code overhead.
 - The usable ports are restricted to simple behavioral ports. Through these ports it is possible to send events from one object to another without a tight coupling meeting defined interfaces.



- the macro `RiCGEN_PORT_ISR(Port, Event)` should not be used. The event is allocated dynamically which results to crashes when done within a ISR.

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

OO RTX

The following features are not supported by the OO RTX:

- Timer objects
Timeouts still can be modelled using timeout transitions like `tm(...)` and are implemented using events. There is no `RiCOSTimer` class.
- Semaphores
There is no support as Rhapsody will not generate code needing them and you can not use

them in your project code as well. In an event driven, mostly asynchronous software architecture there is no need to use semaphores. This also has the advantage that you will not run in typical semaphore problems like deadlocks and priority inversion. There is no RiCOSSemaphore class.

- Event Flags
There is no RiCOSEventFlag class.
- Mutexes
There is no support as Rhapsody will not generate code needing them and you can not use them in your project code as well. The OO RTX does not support preemptive multitasking and therefore does not need mutexes. This also has the advantage you will not run in typical mutex problems like deadlocks and priority inversion. There is no RiCOSMutex class.
- Tasks
The OO RTX does not support preemptive multitasking. Thus it is not allowed to use active objects in the UML model. There is a RiCOSTask class, but it is only used internally by the OO RTX itself and may not be referenced by user code.
- Message Queues
Events with arguments can be passed within your Rhapsody model. These are passed internally via pointers (call by reference). However, there is no RiCOSMessageQueue class.
- Sockets
The OO RTX 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.
 - Active Classes
The OO RTX does not support preemptive multitasking. Thus it is not allowed to use active objects in the UML model. Scheduling between the UML model objects is done event driven (via the event queue). This mechanism is based on the run-to-completion approach (e.g. a state's entry action commands need to be finished before another event can be consumed by any object).
 - 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.
The property CPPCompileSwitches may set the constant WST_PORTS_DISABLED to disable support for UML ports in the OO RTX in a profile which comes with your release.
 - Dynamic memory management.
Use of malloc() and free() is not supported when the OO RTX is used in the form of a Shared Library PMF even when the constant NO_MALLOC is not set.

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 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.