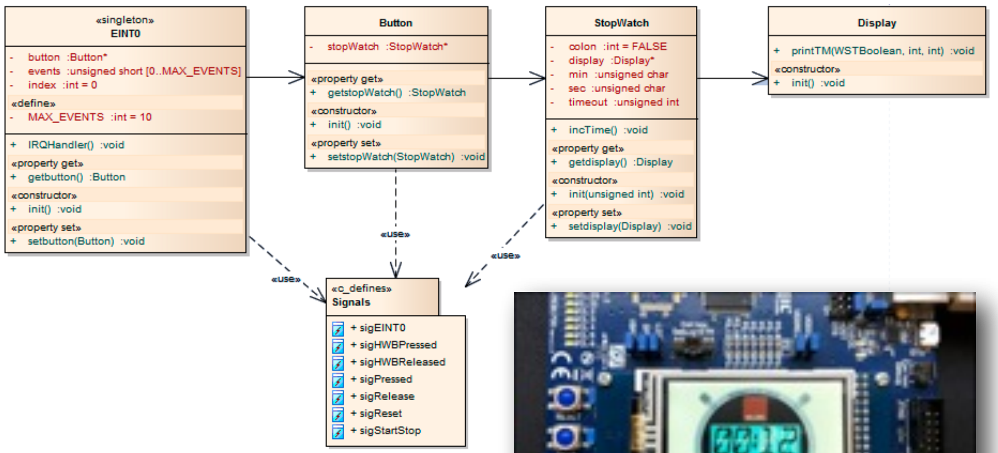
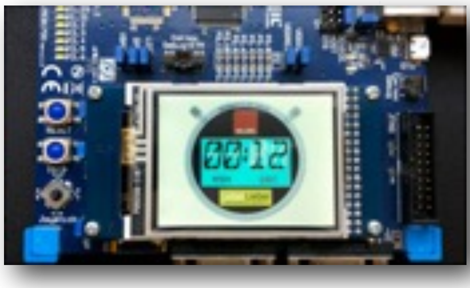
Modeling Embedded Systems

Using embedded UML tools to create high quality embedded software on hardware with limited resources «

WILLERT.





For 15 years the UML plays an increasingly important role in the development of software. However, in the field of embedded software, especially embedded software on target environments with limited resources, ‚C' is still the development environment of choice.

Actually strange, because the often safety related embedded systems would benefit immensely from the increase in overview, changeability, testing possibilities, links with requirement management systems and so on.

For all UML with code generation could give a real boost to the quality of embedded systems and support in certification processes. Willert Software Tools offers several solutions for UML environments tailored for even the smallest embedded systems. All Solutions with their own specific features, benefits and price.

What must be considered, however, is that the UML has a much larger „instruction set" than ‚C'. To apply the UML on a target environment this must first be implemented in an efficient way.

This is where Willert Software Tools shows it's real strength. We have put all the experience in real-time embedded software development that we have collected since 1992, in our UML Framework & Debug Technology. This enables you to use the UML in it's full strength for your embedded software development!

Index:

Evolution of Programming

What is the essential problem, we are facing today in software engineering?

What keeps us from becoming more productive?

Why do we need more and more capacity for testing?

The answer is short and simple and reveals itself, if we take a look at the history of programming.

1.GL - First Generation Language

First generation actually means pre-historic. Computers were programmed using wires and/or switches. Binary codes needed to be checked, programming was very tiresome. Inserting statements meant recalculating all offsets and jumps. Errors were, off-course, made very easily so people started searching for better ways to program computers. After some years they found:

2.GL - Second Generation Language

Assembly language took away lots of the disadvantages of the binary methods used before. The introduction of labels took care of the error-prone recalculation of jumps and offsets. The mnemonic notation allowed a quicker and better understanding of what really happened. Programming was still a job for experts but could be learned more easily now. Errors were still made but they were no longer in trivial parts. Programming was done more and more often and programs got bigger. This caused that the nice assembler programs from the early assembler days were no longer easy to maintain. The search for better methods started.

3.GL - Third Generation Language

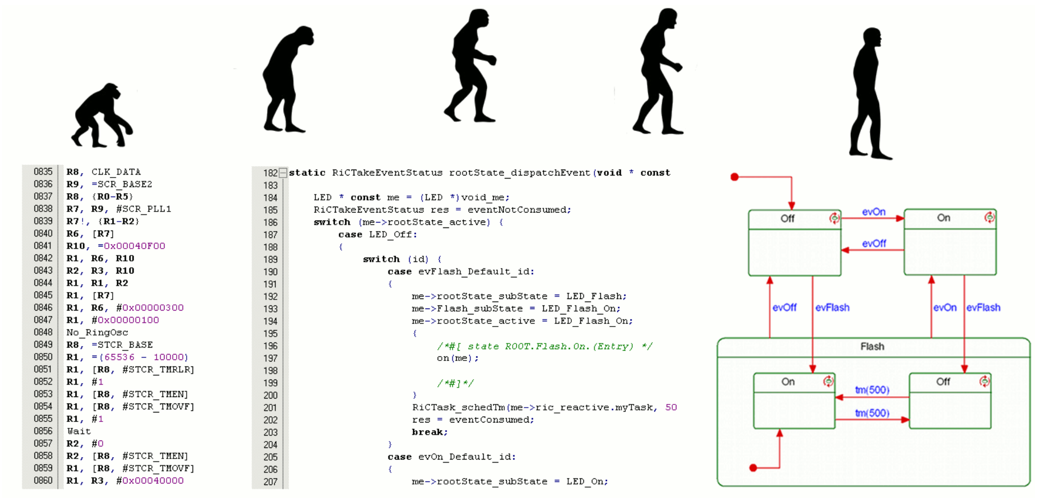
3GL languages come in a lot of flavors. For administrative environments there was COBOL, more technical environments used FORTRAN. For embedded programming the switch from Assembler to 3GL took longer, lots of us can still remember it. The wealth of choice for implementation languages was actually a big disadvantage in the embedded world where stability and re-use were already common before the PC world ever heard of it.

The arrival of a standard (ANSI-C) caused that even the conservative embedded world finally gave in to 3GL. Nowadays there are highly optimized compilers that make us forget that assembler ever existed.

But our software continues growing and is therefore more difficult to handle. Still the need for better programming methods continued. Other languages were introduced (C++) Operating Systems are more widely used but they are only tools.

4.GL - Fourth Generation Language

Since 1996 there is the UML. Actually nothing new but a collection of best practices. The advantages of Object Orientation, State-charts, sequence diagrams, it was all invented before the UML, but it was all combined in one language (Hence unified). The possibility to extend the language in a standardized way helps to create a powerful language for any domain.



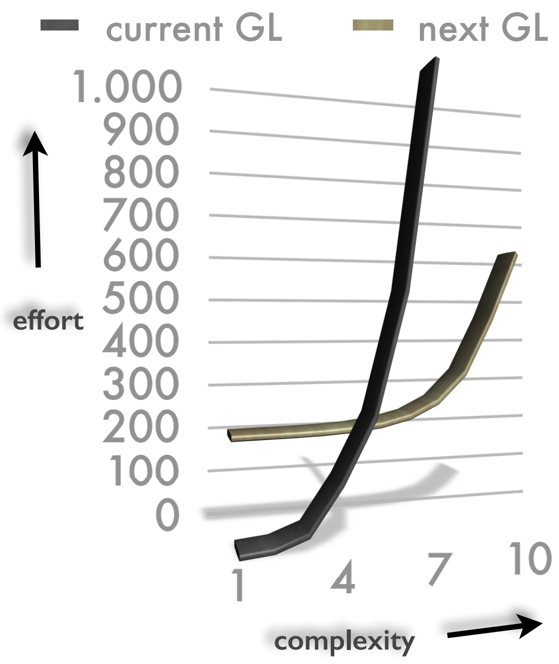
```
R8, CLK_DATA
R9, =SCR_BASE2
R8, {R0-R5}
R7, R9, #SCR_PLL1
R7!, {R1-R2}
R6, [R7]
R10, =0x00040F00
R1, R6, R10
R2, R3, R10
R1, R1, R2
R1, [R7]
R1, R6, #0x00000300
R1, #0x00000100
No_RingOsc
R8, =STCR_BASE
R1, =(65536 - 10000)
R1, [R8, #STCR_TMRLR]
R1, #1
R1, [R8, #STCR_TMEN]
R1, [R8, #STCR_TMOVF]
R1, #1
Wait
R2, #0
R2, [R8, #STCR_TMEN]
R1, [R8, #STCR_TMOVF]
R1, R3, #0x00040000
```

```
static RiCTakeEventStatus rootState_dispatchEvent(void * const

    LED * const me = (LED *)void_me;
    RiCTakeEventStatus res = eventNotConsumed;
    switch (me->rootState_active) {
        case LED_Off:
        {
            switch (id) {
                case evFlash_Default_id:
                {
                    me->rootState_subState = LED_Flash;
                    me->Flash_subState = LED_Flash_On;
                    me->rootState_active = LED_Flash_On;
                    {
                        /*#[ state ROOT.Flash.On.(Entry) */
                        on(me);

                        /*#]*/
                    }
                    RiCTask_schedTm(me->ric_reactive.myTask, 50
                    res = eventConsumed;
                    break;
                }
                case evOn_Default_id:
                {
                    me->rootState_subState = LED_On;
```

900 lines Assembler　　300 lines of C-Code　　15 grafik elements in UML

Why UML Modeling ?

So what is the answer to our questions and what has brought us from one language generation to the next?
It was the rising complexity from our software which was caused by increasing requirements. Before each step, the complexity of our software has reached a critical level. It became more and more expensive to program, handle, and test the current software generation. Only the use of a new technology can minimize the complexity. Of course, a step to another language generation was associated with a lot of effort. But has someone repented the step from assembler to
ANSI C? Or would you still write your programs in assembler today? I hope not. Thus, this effort has been worthwhile, right?



Back to our software requirements. Do you think, they will stagnate or even decline? Unfortunately, there is only one direction, upwards, even in the embedded world. And it gets even worse. Complexity is rising much faster than our requirements.

Therefore, we are back to the step of using a new technology.
A fourth generation language like the UML, which brings us from programming to modeling. The UML approach is to simplify a complex program by dividing it into several graphical charts. For each point of view (e.g. structural, behavioral or runtime), there is a suitable diagram. Moreover, graphical notations have a higher information density than conventional high-level languages like ANSI C. This reduces our complexity again.

What do I need to take full advantage of the UML?

Modeling Tool to create a UML model

First, we need a modeling tool, to create our UML model. Programs like "MS Visio" are often used for documentation. But is it that what we want? UML for documentation only and a high level language for coding? Have we documented our assembly code with high level languages in the past? That would be exactly the same.
We need to keep model and code congruent. That is no job for a human being. Finally, you would also not perform the tasks of a compiler. What we need is called code generation. This must be the first important requirement of our UML tool.

Many UML tools do not have a built-in repository. But UML elements are often used several times inside the same model. For example, an attribute could be used simultaneously in statecharts, flowcharts, methods, and so on. If we want to change the name of this attribute, we have to search the entire model, unless our tool has a repository. With a repository, we would always access the same attribute. Therefore we have to change the name only once.

Of course, our tool should be established and widespread used. There are two tools that meet these requirements.

- IBM® Rational® Rhapsody®
- Sparx® Enterprise Architect

Rhapsody is definitely the Mercedes of all UML tools. Unfortunately, the price is also like a Mercedes. However, we can offer you along with our Embedded UML Studio III a more affordable version. It's an Architect version with code generation for embedded systems. This version costs only a fraction of a full developer version.

New in offer is our Embedded UML Studio III with Sparx Enterprise Architect. UML and Enterprise Architect simply belongs together. Sparx Systems Enterprise Architect is due to its comparatively low price and its early implementation of UML 2 widespread used as UML editor. Moreover, Enterprise Architect is expandable by many plug-ins. All this makes Enterprise Architect to a good alternative solution to IBMs Rational Rhapsody.

Framework for the step from 4.GL to 3.GL

To be able to execute generated code from a UML model, the generated code needs to rely on a strong base of functionality. The UML notation is much more powerful and on a higher abstraction level than any high level programming language. A third generation language can not directly reflect the UML-instruction set, which includes asynchronous communication, active classes, complex state behavior and many other powerful features.

If no standard framework would be chosen, all of these functionalities need to be implemented by the user, including lots of beta testing in a complex environment and staying with limited functionality and flexibility. Also, without a framework, target and RTOS specific elements would be spread all over the application model. This corrupts one of the major advantages of model driven development.

Selecting a stable framework gets the project productive much faster while benefitting from a lot of advantages including best practice solutions for static memory management, how to avoid RTOS pitfalls (like deadlocks), get an optimized IRQ to model latency etc. A solid base framework with a good user base is very important for successful production code generation on an embedded system.

The RXF also fills the gap between UML modeling tool and IDE and it optimizes the code-generation especially for a resource friendly use in the embedded realtime world.

IDE for compiling, linking, HLL debugging, flashing

Just like before, we need a normal IDE to compile and link the generated source code. Nothing has changed here.

RTOS (or run-time system)

There are a lot of embedded real-time projects around where, when asked, the developers clearly state that they have not used an RTOS or a run-time system. This may be true they mean that they did not explicitly bought a commercial RTOS. But every application, no matter how small, contains a run-time system. Even a simple loop program (while(1) { do(); }) is a run-time system and must be considered when designing an application like that. That global variables are used for communication is also a run-time decision. So the statement: "I don't need an RTOS" should in fact be "Gee, I didn't know I already use an RTOS".

As already stated, the UML does not contain just programming language elements that can be projected on a common high-level language, some elements represent run-time environment settings. Things like Active Classes require the use of a preemptive RTOS to create the threads needed to make a class active. Sending and receiving events in a statechart requires that there are message queues that can do that for us. Then there are timers, mutexes, semaphores, etc. It is possible to build this all in the UML Framework (In fact, this is what Willert Software Tools has done with the OO-RTX). The RTX RTOS functionality is not comparable to a full third party RTOS! However, it is optimized to fulfill the minimal RTOS requirements, to use the UML for embedded realtime systems.
Often there is already an RTOS in use that could be used to implement all this functionality. Therefore, the RXF Framework has an RTOS Interface that allows the easy integration of most off-the-shelf RTOS's.

Target Debugger for UML level debugging

With the UML Target Debugger™, the UML model can now also be debugged on the actual hardware. Take a look on your animated sequence- and timing diagrams in realtime. One advantage of MDD (Model Driven Development) is that models can be tested and debugged at an early stage, usually based on simulation. According to experience, this method helps to identify numerous errors in early process phases.

The RXF Framework

The Embedded UML RXF™ (Real time eXecution Framework) is the solid foundation for your software architecture.

It forms the interface between a UML model and the target platform consisting of CPU, compiler and runtime system or RTOS. It also efficiently allows to combine legacy code based parts of your software with generated code.

It contains an abstraction layer to support most real-time operating systems in the market. Thus, mechanisms like timers or events can be employed independent of the operating system, resulting in a high degree of portability, reusability and platform independence. Your software design will be target-independent and guarantees maximum return on invest.

With Embedded UML RXF™, you implement your UML Models designed with Rhapsody or Enterprise Architect in target code. Each RXF variant is specially designed for a tool chain combination and integrates it tightly. This also includes deploying the generated code in your IDE project, allowing the use of the IDE's target configuration wizard. Short processing cycles are provided between IDE / debugger and your UML model.

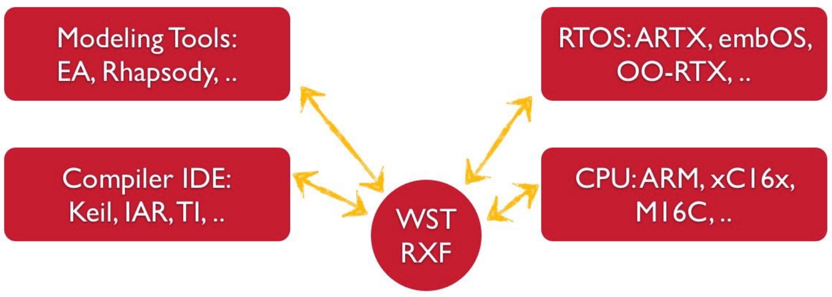
Our highly optimized framework is suitable for code generation for embedded applications with limited resources.

Configuration settings for the real-time operating system and RXF can be stored where all your software is designed and implemented: inside the UML model. Also adaptations to different target hardware requires a minimum effort.

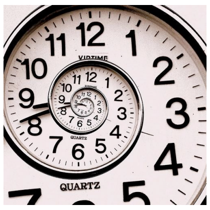
The delivered installation routine provides a fast start-up with the RXF and integrates the complete tool chain.

The RXF's debug interface allows you to debug and test your model on UML level, tool independent, without destroying your timing behavior, as it uses a highly optimized monitor to communicate with a PC in real-time.

- Adaptations for a large number of target platforms
  ARM – C167 – Blackfin – M16C – MSP430 – V850 – TMS320 – ATMega – RL78 – PIC32 – RX – RaspberryPi…
- Support of many real-time operating systems
  CMSIS-RTOS – Linux – embOS – OSEK – MQX – CMX – FreeRTOS – Keil-RTX – µC/OS – RTEMS…
- Support many IDE's like Keil MicroVision, Eclipse, IAR Embedded Workbench, Tasking EDE, and many more
- Support 2 of the most widely used UML Modeling Tools: Sparx Enterprise Architect and IBM Rational Rhapsody
- Optimized for use in embedded systems with limited resources
- Deterministic real-time behavior (in connection with OO RTX, no interrupt latency for 32 bit CPUs and max. 4 instructions interrupt disabling for 16 bit CPUs)
- Reliable and with a long-year and wide user base to ensure a stable base for your software development
- Best practice solutions for modeling event based logic with time based software controllers
- Supports integrating generated components into an AUTOSAR RTE, including communication between tasks
- A certification package is available for safety critical software development projects
- Delivery with source code
- UML-level debugging with the Embedded UML Target Debugger
- 12 months warranty
- No royalties
- Highly optimized versions for 'C' and 'C++'


Modeling Tools: EA, Rhapsody, ..
RTOS: ARTX, embOS, OO-RTX, ..
Compiler IDE: Keil, IAR, TI, ..
CPU: ARM, xC16x, M16C, ..
WST RXF

Features of the RXF



Memory Management with Deterministic Behavior

RXF features memory management with user-defined block sizes. A configurable number of static block sizes can be configured. Also the number of blocks can be defined while modeling in your UML Tool.
Dynamic memory can thus be used without encountering the common disadvantages such as the need for defragmentation and non-deterministic time behavior during memory allocation.

Deterministic Real-Time Behavior

Compared to most other UML framework solutions, RXF operates with only minimum manipulation of interrupt latency. There is no interrupt locking on 32 bit CPUs and one interrupt locking of max. 4 command cycles on 16 bit CPUs.

HIGH WATER MARK
40.29 Feet (15.29 Feet above Flood Stage)

APRIL 1, 1886

ON THIS DAY, WATER FROM THE OOSTANAULA RIVER ROSE TO THIS LEVEL AT THIS LOCATION. FOR MORE INFORMATION, CONTACT:

NATIONAL WEATHER SERVICE PEACHTREE CITY, GA
www.srh.noaa.gov/ffc

SOUTHEAST RIVER FORECAST CENTER
www.srh.noaa.gov/serfc

USGS GEORGIA WATER SCIENCE CENTER
ga.water.usgs.gov

High Water Marks

The Framework can be used in a fully static way. The static memory blocks can be fully tailored for optimal use. The utilization of these memory areas and also event queues and timer queues across application runtime can be monitored and optimized.

Fast Event Implementation

Fast events are available especially for the integration of interrupt service routines in the UML model. Thus, an object-oriented interface based on static events from interrupt routines can be implemented with minimized overhead.
The speed of events is measured in PEPS (Processed Events Per Second). A small target like a C167 running with 20 Mhz can already handle more then 10.000 PEPS.

Low-Power Support

RXF automatically detects periods in which the system is idle and jumps to a routine in which the user can switch the CPU to low-power mode. This is fully configurable for your own CPU. For the MSP430 an example application is included. Enjoy the advantage of an event drive system.

Best Practice Solutions

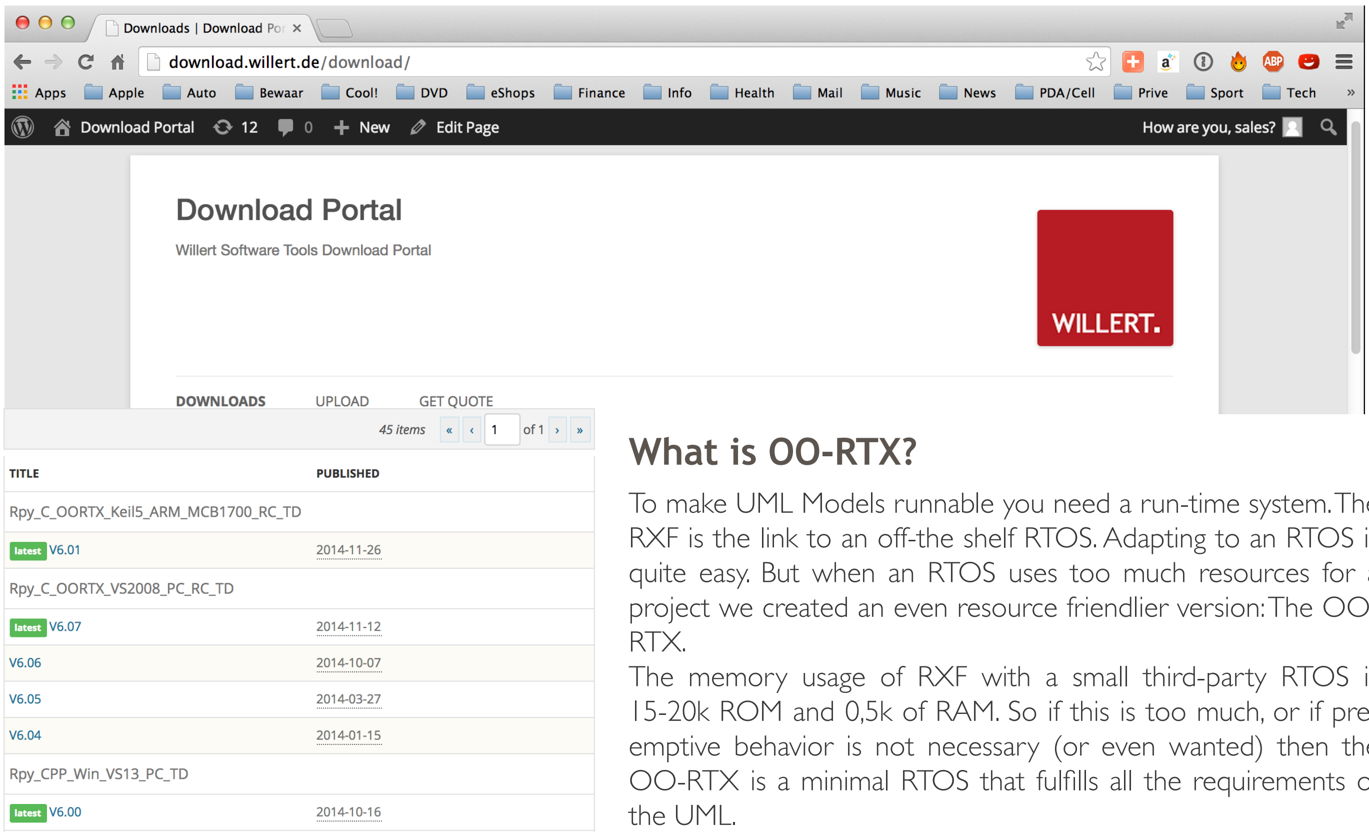
- AUTOSAR integration as well as import ARXML and code generation
- single threaded RTOS for very small applications
- Traceability with REQXChanger
- Certifiable Version available.

NOTE

This product is designed to be used together with a UML Tool like Rhapsody or Enterprise Architect.

Add-ons and Plug-ins

The RXF is built up so that it is fully configurable for almost any embedded environment. You only need a base license to use all available bridges. The Willert Software Tools Download Portal allows you to download the adapters that are included in your license.
The list below gives an overview of actually available bridges. If your desired environment is not on the list, then don't worry.
Adaptations can be made with a minimal effort, you can do it yourself or you can order us to do the necessary adaptation for you. Contact us for the details.



Download Portal

Willert Software Tools Download Portal

45 items « ‹ 1 of 1 › »

TITLE	PUBLISHED
Rpy_C_OORTX_Keil5_ARM_MCB1700_RC_TD	
latest V6.01	2014-11-26
Rpy_C_OORTX_VS2008_PC_RC_TD	
latest V6.07	2014-11-12
V6.06	2014-10-07
V6.05	2014-03-27
V6.04	2014-01-15
Rpy_CPP_Win_VS13_PC_TD	
latest V6.00	2014-10-16

What is OO-RTX?

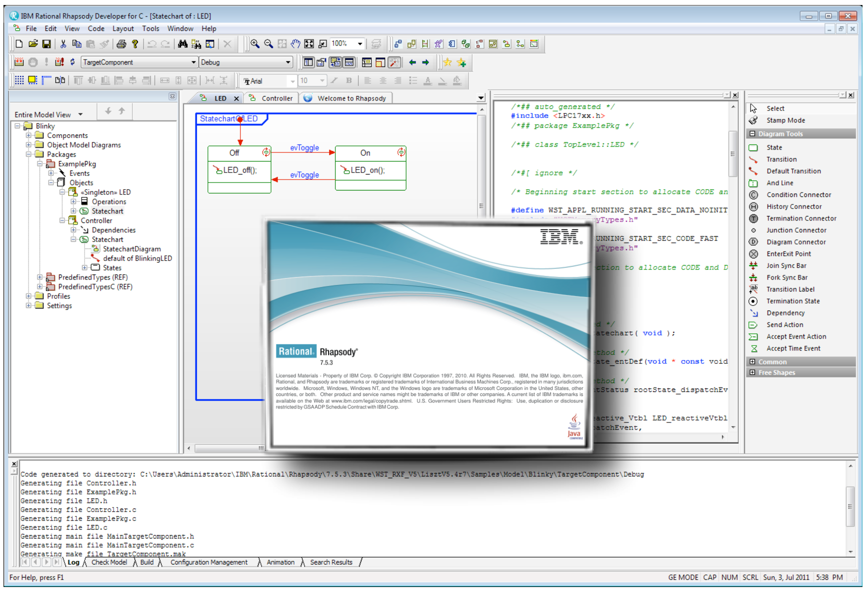
To make UML Models runnable you need a run-time system. The RXF is the link to an off-the shelf RTOS. Adapting to an RTOS is quite easy. But when an RTOS uses too much resources for a project we created an even resource friendlier version: The OO-RTX.
The memory usage of RXF with a small third-party RTOS is 15-20k ROM and 0,5k of RAM. So if this is too much, or if pre-emptive behavior is not necessary (or even wanted) then the OO-RTX is a minimal RTOS that fulfills all the requirements of the UML.
Due to it's compact integration in the RXF it is much smaller and much faster then a third party RTOS. Depending on your compiler the OO-RTX consumes 3-6k ROM and 200b of RAM. Also the speed is higher than that of an of the shelf RTOS with RXF.

- Adaptations for a large number for target platforms
  C167 – TriCore - ARM – M16C – BlackFin – MC16 - Coldfire – ATMega – 8051 – PIC32…
- Super fast timer implementation.
- ROM Usage between 3k and 6k
- RAM Usage 200 bytes
- Highly optimized for use in systems with limited resources
- Delivery with source code
- Convenient configuration through properties
- Optional interface with UML Target Debugger
- 12 months warranty
- Supported target platforms see www.willert.de
- No royalties

NOTE

The OO-RTX is not a stand-alone RTOS. It can be used ONLY in cooperation with the RXF.

IBM® Rational® Rhapsody®



```
/*## auto_generated */
#include <LPC17xx.h>
/*## package ExamplePkg */

/*## class TopLevel::LED */


/*#[ ignore */

/* Beginning start section to allocate CODE an

#define WST_APPL_RUNNING_START_SEC_DATA_NOINIT
```

```
Code generated to directory: C:\Users\Administrator\IBM\Rational\Rhapsody\7.5.3\Share\WST_RXF_V5\LisztV5.4r7\Samples\Model\Blinky\TargetComponent\Debug
Generating file Controller.h
Generating file ExamplePkg.h
Generating file LED.h
Generating file Controller.c
Generating file ExamplePkg.c
Generating file LED.c
Generating main file MainTargetComponent.h
Generating main file MainTargetComponent.c
Generating make file TargetComponent.mak
```

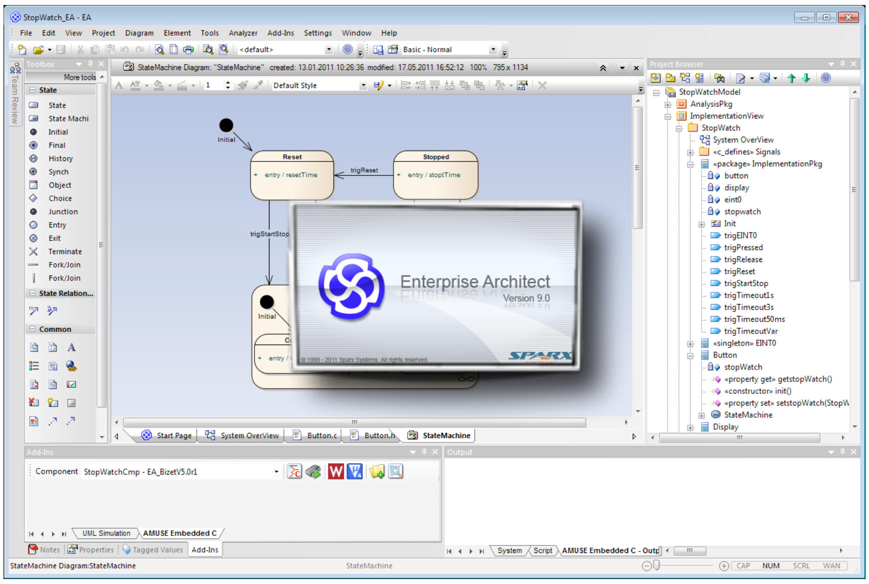
Rational Rhapsody® 7.5.3

Licensed Materials - Property of IBM Corp. © Copyright IBM Corporation 1997, 2010. All Rights Reserved. IBM, the IBM logo, ibm.com, Rational, and Rhapsody are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml. U.S. Government Users Restricted Rights: Use, duplication or disclosure restricted by GSAADP Schedule Contract with IBM Corp.

The IBM® Rational® Rhapsody® Developer is a embedded and real-time software development environment based on industry standard UML/SysML.
For over 10 years Rhapsody® sets the standard for UML development in the embedded domain. Willert Software Tools RXF is the perfect match to allow development on even the smallest of targets.
There are Rhapsody® Versions for System Engineering, Software Engineering (Architect) and for Software Development.
Add-ons are available to connect to Doors® or other Requirements Management Tools, test environment. For testing there is Test Conductor that is based on the UML testing profile.

- Full Simulation of Diagrams, even on Target. (Requires TCP/IP and high-performance target)
- C, C++, Java, and Ada code generation of state-charts.
- Model multicore affinity,
- Generate code and build files for leading embedded and real time development environments.
- Visualize C# code from Microsoft Visual Studio and generate C# code from Rational Rhapsody
- Integrates within Eclipse development environment
- Import existing C, C++, Java or C# code for visualization and documentation
- Flexible development environment synchronizes code and model.
- Requirements traceability.
- Develop software on host before target hardware is available.
- Maintain consistency of architecture, design, code and documentation automatically
- Architect Data Distribution Service for Real-Time Systems (DDS) applications to manage the complexity of inter-connected components
- Automate documentation across product lifecycle with Rational Publishing Engine integration
- Develop automotive applications using AUTOSAR from concept to code
- Leverage MARTE profile for architecting multi core applications
- Collaborate using model-based differencing and merging features, including an integration with the Jazz- based IBM® Rational® Team Concert solution
- Automate model based testing with Rational Rhapsody TestConductor Add On
- Extendable and customizable modeling and code generation
- Rational Rhapsody, DoDAF, MODAF, and UPDM Add On assists in delivering compliant and consistent architectures for these architectural frameworks support with ability to develop your own profiles
- Rational Rhapsody Developer 7.5 adds support for AUTOSAR 4.0 system authoring and behavioral design, token based activity diagram simulation, improved performance for code generation, and improved code customization.

Sparx Enterprise Architect with LieberLieber embedded coder RXF



Enterprise Architect is a high performance modeling, visalization and design platform based on the UML 2.x standard.

With complete traceability from mind mapping, through requirements to business and software design and deployment , Enterprise Architect provides the kind of robust and efficient visualization and collaboration required in today's large and demanding modeling environments.

A truly agile modeling solution, Enterprise Architect provides a low installation overhead, sparkling performance and an intuitive interface.

Keep your entire team on the same page with Enterprise Architect, a tool priced for team deployment and designed for real-world situations.

Enterprise Architect is the worlds leading tool for UML and SysML. It's versatility allows usage on many different area's. MDG Technologies allow numerous Add-ons exists for Requirements Management, Testing, Code Generation and Simulation.

LieberLieber

Embedded Engineer
for Enterprise Architect

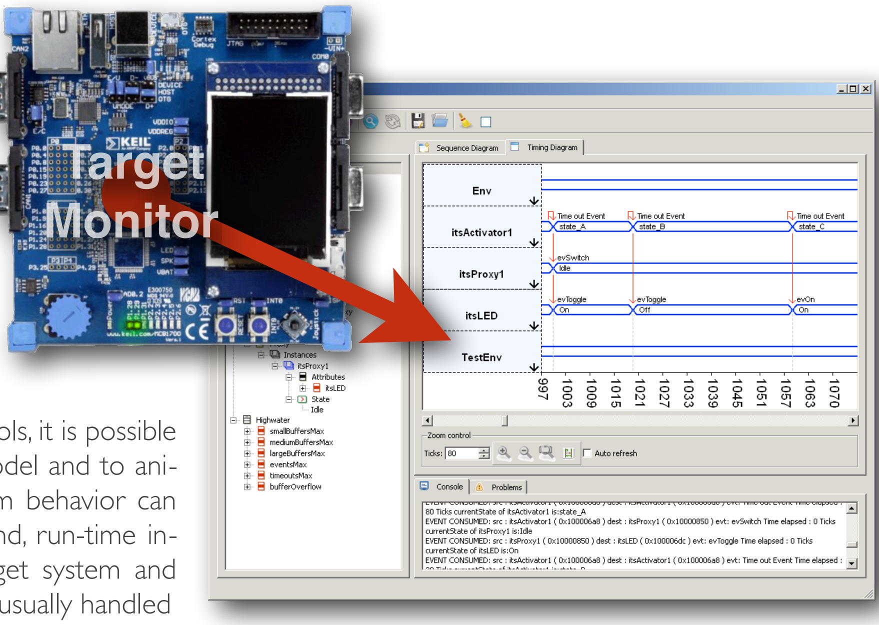
In contrast to IBM's Rational Rhapsody, Enterprise Architect doesn't have a built- in embedded code generator. Fortunately, we can add this feature through a plugin by the Enterprise Architect specialists "LieberLieber". The LieberLieber Embedded Engineer RXF plugin generates ANSI 'C'- code from the EA UML model. Moreover, with the help of another plugin: LieberLieber AM|USE, we can simulate and execute our UML diagrams directly in Enterprise Architect.

LieberLieber Embedded Engineer RXF is your tool for interactive UML & SysML modeling. Whether you want to maximize the output of your modeling endeavor or you are just starting to use MDD - embedded coder RXF will be your best ally.

Embedded Engineer RXF integrates with Enterprise Architect from Sparx Systems and guarantees a consistent way of modeling. Enterprise Architect is one of the leading modeling-tools on the market. If your focus in software- and systems-development is on quality and efficiency, Enterprise Architect and LieberLieber Embedded Engineer RXF is the combination to use.

The Code generator and the RXF together produce optimized production code that runs directly on your target environment.

Embedded UML Target Debugger™



```
80 Ticks currentState of itsActivator1 is:state_A
EVENT CONSUMED: src : itsActivator1 ( 0x100006a8 ) dest : itsProxy1 ( 0x10000850 ) evt: evSwitch Time elapsed : 0 Ticks
currentState of itsProxy1 is:Idle
EVENT CONSUMED: src : itsProxy1 ( 0x10000850 ) dest : itsLED ( 0x100006dc ) evt: evToggle Time elapsed : 0 Ticks
currentState of itsLED is:On
EVENT CONSUMED: src : itsActivator1 ( 0x100006a8 ) dest : itsActivator1 ( 0x100006a8 ) evt: Time out Event Time elapsed :
```

Even software modeled in UML isn't always free from errors. Therefore, debugging remains one of the key tasks in software development. In the context of so-called embedded systems, the hardware used is often proprietary, and some of the errors probably occur only when the software is executed on this hardware. In addition, many embedded systems have to manage limited hardware resources (e.g. memory).

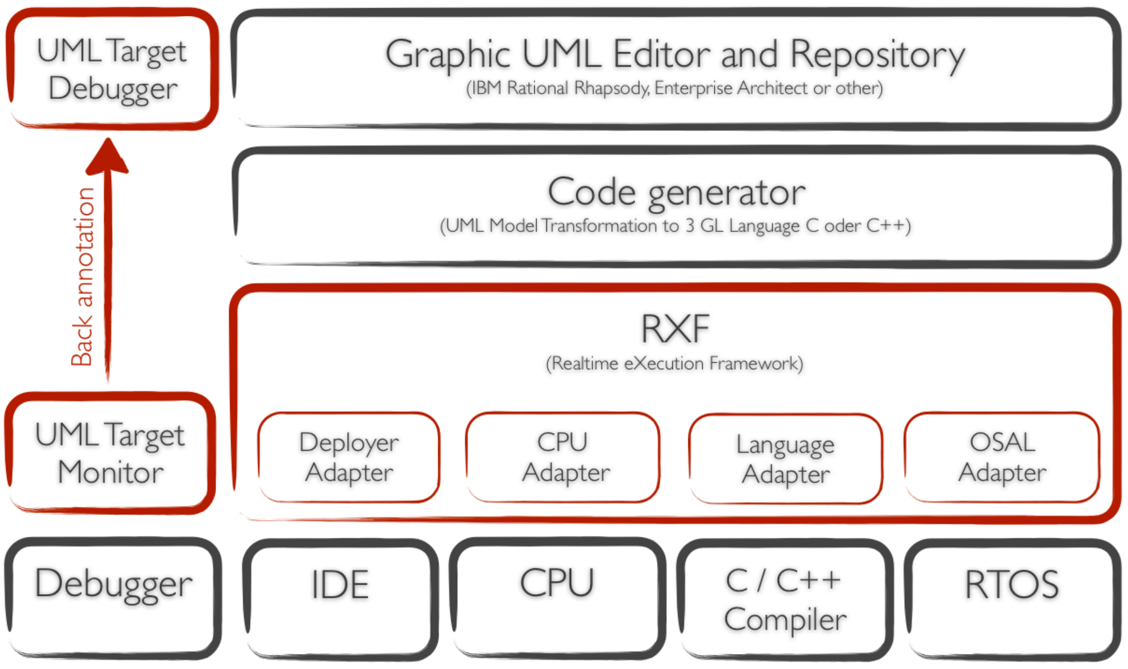
When debugging with conventional CASE tools, it is possible to execute the code generated from the model and to animate the model simultaneously. The program behavior can thus be monitored at model level. To this end, run-time information has to be generated on the target system and transmitted to the development PC which is usually handled through so-called code instrumentation. However, there is a major disadvantage to this approach. The code to be executed is unnecessarily loaded with overhead which considerably slows down the actual run-time. A debugging solution is now available that is compatible with Embedded UML Studio III™ and based on a monitor like conventional embedded high-level language debuggers. It offers the benefit of real-time debugging with minimum overhead at UML model level.

- deterministic real-time behavior (other than with instrumented code, the new technology does not affect the run-time behavior)
- live animation of sequence diagrams to trace the event order on target
- see which instances of your reactive objects (with a statechart) have been created and destroyed
- navigate through a model element tree to show attributes or the current state of a statechart for any instance
- display the current values of the highwatermarks when used with the OO-RTX
- inject events from the host-side and track the reaction of the target (currently only events without arguments are supported)
- monitor the elapsed time-ticks between two events

One advantage of MDD (Model Driven Development) is that models can be tested and debugged at an early stage, usually based on simulation. According to experience, this method helps to identify numerous errors in early process phases. But in many cases, errors occur later, when the code is generated from the UML model and executed on the actual hardware.
Another disadvantage of simulation is the need to provide all external interfaces required for the simulation procedure which can be quite laborious. With the UML Target Debugger™, the model can now also be executed on the actual hardware. All required run-time information is provided through a monitor at model level. Thus, animation takes place at model level in parallel to model execution.
In addition, information can be transferred from the development PC to the target system, so that UML based tests can be automated on real hardware, e.g. with Test Conductor.

Embedded UML Studio™ Third Generation

Based on 10 years of experience in development of UML-solutions and specialized for use in Embedded Software Engineering, we have responded to the market needs. The answer is Embedded UML Studio III™, which is now available in two different editions. First of all, based on the market-leading technology from IBM® Rational® Rhapsody®. An established, high quality tool. And now, also available with the widespread, low budget solution - Sparx Enterprise Architect.
This technologies are enhanced by adding the proven RT Framework from Willert Software Tools. The outcome is an UML-based engineering-environment at a low price, which provides everything for a successful development of embedded software.


UML Target Debugger
Graphic UML Editor and Repository
(IBM Rational Rhapsody, Enterprise Architect or other)
Back annotation
Code generator
(UML Model Transformation to 3 GL Language C oder C++)
RXF
(Realtime eXecution Framework)
UML Target Monitor
Deployer Adapter
CPU Adapter
Language Adapter
OSAL Adapter
Debugger
IDE
CPU
C / C++ Compiler
RTOS

- Supports all required diagrams for embedded software engineering.
- Target-specific highly optimized ANSI-C generation. Enables reaction-times down to the interrupt-response-time of the implemented CPU. (Adaptations are available for a large number of target-platforms, including ARM, C167, AT-Mega, PIC, Blackfin...)
- Upgradable by implementing the Embedded UML Target Debugger™ for UML Animation and Test-automation.
- Interfaces to other tools (Configuration Management, Requirements Management, Testautomation ...)
- Based on the proven and market-leading technology Embedded UML Studio™
- Is delivered royalty free, complete with source code
- Free of charge access to all available bridges.

Download an evaluation DVD for free

We have provided a Demo DVD for both tool chains. Apart from a Windows PC, you do not need additional hardware. The target hardware (LPC1768 Cortex M3) will be simulated by the Keil's µVision IDE.
Please visit our website for detailed informations: http://www.willert.de/uml-getting-started/

UMLforum.de

Author:
WALTER VAN DER HEIDEN

Editor:
WILLERT SOFTWARE TOOLS GMBH
Hannoversche Straße 21
31675 Bückeburg
www.willert.de
info@willert.de
Tel.: +49 5722 9678 - 60