

ZIPC++

Advantage to the embedded system development!
 "State Transition Design" is adopted to assure "Reliability" & "Quality!"
 "UML" is adopted in object-oriented analysis & design that improves
 "Software Product Line!"

Recently the embedded system is expanded and complicated and high reliability is required. However, software development period is expected to be shorter! Thus it is difficult to keep the existed development style. The important topic in the embedded system is releasing a product with low cost and short period with high quality.

Embedded System development by UML

Recently, object-oriented method became popular method in embedded system development industries.

UML(Unified Modeling Language) is a standard notation of object-oriented is used widely in many domains.

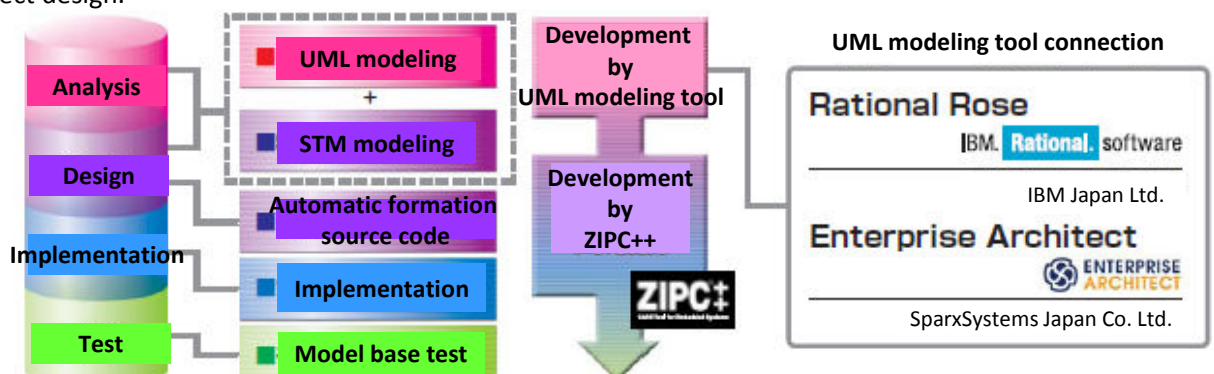
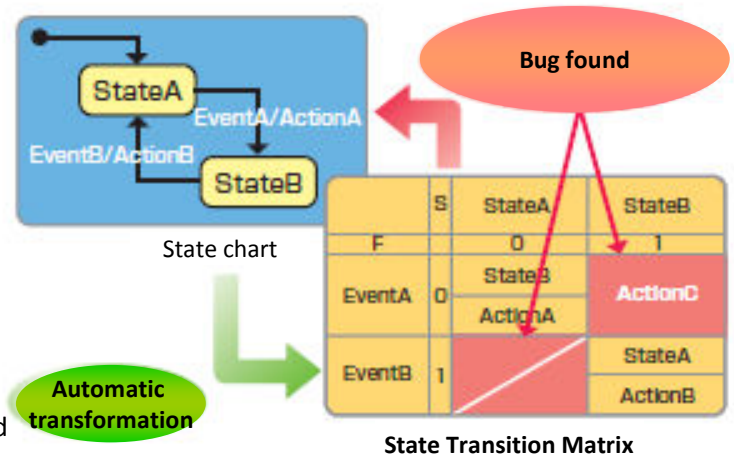
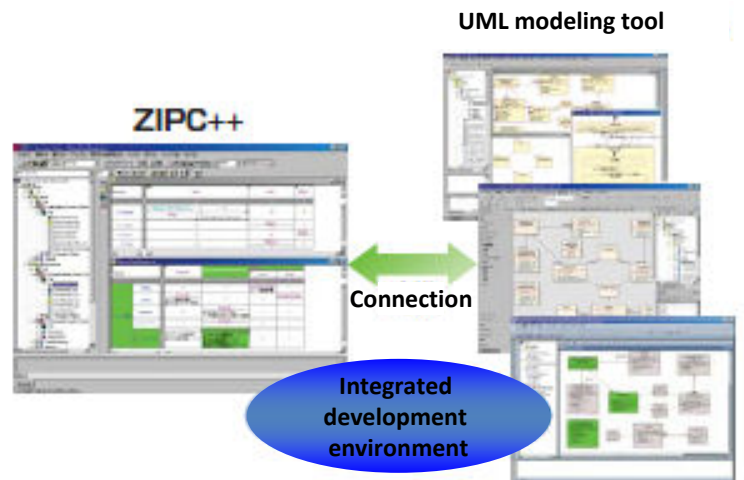
ZIPC++ provides the integrated development environment for the UML embedded system software development by , using the UML modeling tool.

ZIPC++ is equipped with various functions to largely improve efficiency in development and reliability systems. We strongly support the users in the process of "analysis, design, implementation, tests" when they are willing to develop a system by using object-oriented.

"State Transition Matrix" complements UML development

In order to create high quality modeling in embedded system, UML is necessary. In the embedded system, you have to make clearness of the "WHEN" "WHERE" "WHAT". If you neglect these parts, it can become the origin of a fatal bug. To design "WHEN" "WHERE" "WHAT", State Transition Matrix is optimal for it. The state chart, one of notation of UML, is for a document designing the normal case of an event to occur and the process. (Refer to the right picture.)

The state-chart the concept of processing is easy to understand but isn't comprehensive especially, in the embedded system. When unexpected event occur in the system, the state-chart does not have enough reliability. So that is why STM is necessary for a perfect design.



Behavior Convert

The state chart information which is a UML class behavior can be converted into the STM (State Transition Matrix), it has a two way converter which can convert from STM on a state chart. As a result, after designing the process flow in the state chart, a perfect design can be done by converting to the STM.

Model Checker

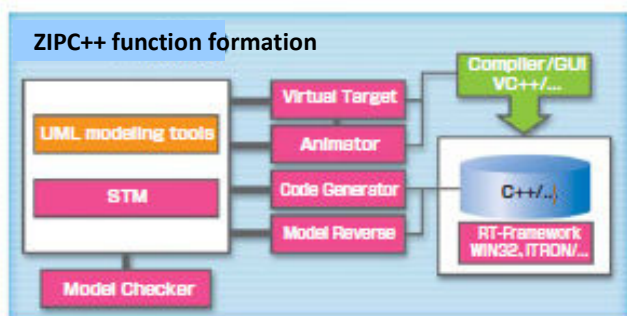
When code-generating, it can be model checked. The UML class information and the STM are the target of the checker. The class information mainly check's out the unfilled parts. The STM check, confirms on the unfilled part and form and on the un-reached state. By this a static check is possible in the modeling phase. When selecting an error or warning in STM, it can jump to the problem part promptly. A simple mistake can be avoided early in the design phase by the model checker.

Code Generator

The code generator automatically generates C++ code from class information on the UML model and the STM of the class behavior. The source code can be extracted from the STM and the code is not like the skeleton level, but it is completely execution level. The state transition logic with the code generator is generated by an excellent algorithm in a suitable code size and speed for the embedded system. The source code which is product dependable can be auto-generated by using various options. And also it can code generated in the Japanese model.

Model Reverse

When customizing a source code generated by a code generator, changing contents to the source code can be reflected (round trip) in a model. Normally, when there is a change to the source code, first correct the model and then try the code generation, when the model cannot be changed on the target environment, model and code can be combined even when model reverse is performed to the source code directly.



ZIPC++ operating environment

OS: Japanese Version Windows 2000/XP/Vista
Disk capacity: 120MB or more (when installing)

Other required conditions:

- *UML Modeling tool should be installed.
- *To use Animator function Microsoft Visual C++ is required.
- *To operate the virtual target Microsoft Visual Basic or Microsoft Visual C++ is required.

Communication
Art
Technology
Systems

CATS Co., Ltd.
Software division

Kawaasa Bldg. 2-11-5 Shin-Yokohama Kohoku-ku, Yokohama 222-0033 Japan
TEL : +81-45-473-2816 FAX : +81-45-473-2673
<http://www.zipc.com/> E-mail : info@zipc.com

* Material will change without previous notice.

* Company Names that are written in the material are a trademark or a registered trademark.

Animator

The animator is a function that executes, and debugs the source code generated with the code generator. When the code is executed in the Animator, the execution process is displayed visual in the STM. As a result, in what condition and what is executed can be understood in visual.

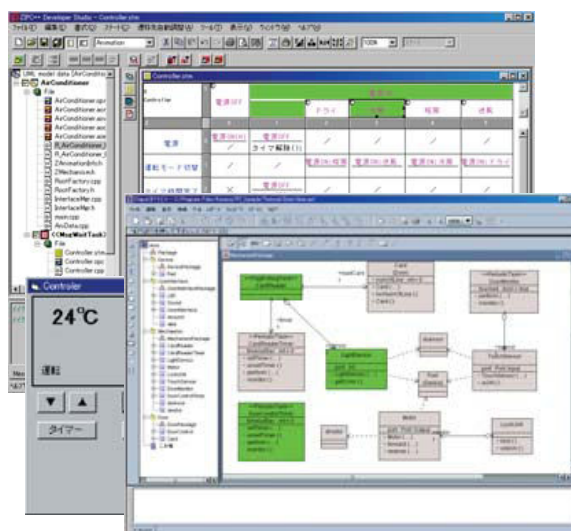
The Animator uses Microsoft Visual C++ as compiler to make the execution environment.

The Animator, runs the execution environment that is compiled linked, and not just showing the execution process in visual, event issuance and break configuration can be done on the STM. But also calculating STM coverage from a log, and the actions can be possibly included in RTOS(Win32,ITRON).

By using these functions, debug is possible on the model base and also easy to understand in visual. Debug efficiency rises when, just only using the source code for debugging, debug functions are able to be used on the STM.

Virtual Target

GUI which is made from Visual Basic or Visual C++, is connected as a function. So, the model verification by controlling Virtual Target can be done. Thus, not only model based animation can be done, but also, intuitive verification can be done.



RTOS Frame Work

In the Animator execution environment, it corresponds to Win32 and ITRON as operation RTOS. The verification including ITRON can be done by fully equipping ITRON with ITRON OS that operates on Windows also on Windows.

When original RTOS is used, the verification including original RTOS can be done by setting to operation on the Windows.