



eBinder: Integrated Development Environment for Real Time Systems



As the power of embedded processors increases, embedded systems are becoming as complex as yesterday's high-end desktop PCs. In addition to the need to push the processor capacity to its limit, designers of embedded systems must also control many devices within tight time constraints. The requirements for an embedded system are often more complex than those for a desktop application, yet many embedded developers are still using desktop tools to develop and debug complex embedded applications. Today's complex systems demand the ability to debug multiple tasks in real time. Product groups need an efficient development cycle to bring their product to market fast – they can't afford time to integrate platform support, or create their own debug tools. The solution to these problems is an integrated development environment specifically designed for developing complex, multi-threaded real time applications.

Development in Today's Embedded Environment

Today's developers face many issues. The pressure is increasing to develop new products with more features at lower cost, and get them to market faster. At the same time, products are becoming more complex. Developers need solutions to several problem areas to effectively create new products.

Developers need a quick project ramp

To quickly get started on their application, developers need not only an Integrated Development Environment (IDE), but also an integrated development platform. A development platform jumpstarts the project by providing a working launch point. Typically, developers need a complete working platform to include a Real Time Operating System (RTOS), Board Support Package (BSP), drivers, middleware, and a development environment.



Developers must create complex applications

As applications become more complex, more tasks are required to ensure optimum performance. Many IDE solutions in the embedded market are jazzed-up versions of desktop debuggers. These embedded IDEs can display memory and registers, and use hardware breakpoints, but they cannot debug multiple tasks in real time. Today’s complex applications require a development environment that shows what is happening at the task level.

Developers require real time debugging

The majority of embedded IDEs need to stop the target system to gather or examine system information. Halting an entire system designed to run in real time may cause the developer to miss a serious fault that would only be seen when other parts of the system are running. Developers need access to system analysis and debugging tools to work on a live system.

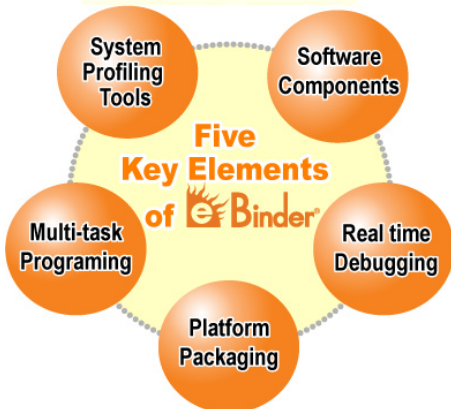
Developers are pressured to do early application development

Time-to-market pressures mean that the application design must start well before real hardware is ready. Developing many parts of the application on a simulator is a viable alternative to prototyping on a board that is different from the target hardware. A good simulator can accelerate development time by providing facilities to develop large portions of the application before the hardware is ready.

eBinder Integrated Development Environment

eBinder is a powerful integrated development environment for real time operating system-based development. eBinder addresses the embedded market’s development needs by providing:

- Integrated development platform
- Multi-tasking debug tools
- System analysis tools
- Real time debugging capability
- Software engineering tools
- Full Instruction Set Simulator for early application development





Debug an individual task – without stopping the rest of the system

eBinder is a complete, graphical Integrated Development Environment that runs on any Windows-based PC. It can connect to a board using serial, Ethernet, or JTAG/ICE interfaces. All of eBinder's features are available through the JTAG/ICE interface: eBinder uses the JTAG as a debug port while preserving the JTAG functionality for execution control.

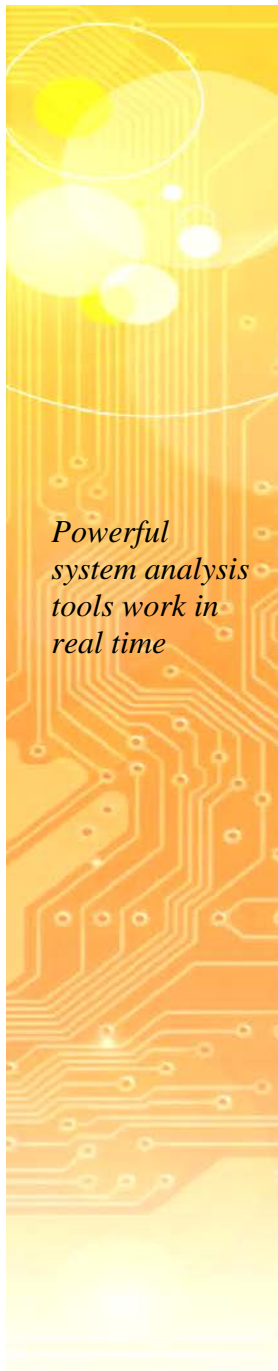
eBinder Integrated Development Platform

The eBinder package provides a full RTOS-based development environment. eBinder includes the PrKERNELv4 real time operating system with full source code, pre-integrated with a Board Support Package (BSP) for reference platforms. PrKERNELv4 is eSOL's open standard multitasking kernel, and is compliant with the μ ITRON 4.0 specification. The BSP provides hardware-dependent code to support selected evaluation boards. eBinder also supplies a target monitor ported to the development board. Multitasking applications require a reentrant C library. Either the bundled compiler or eBinder provides a thread-safe C library that includes support for stream I/O. A Unix-like file system simplifies stream I/O access to media. eSOL's middleware libraries for USB, TCP/IP, and FAT file systems integrate with eBinder to provide a full-base solution. Together, all of these features add up to a complete platform that allows developers to start creating their application without worrying about low-level porting.

Multi-task Debugging

A key to developing a high-quality RTOS-based application is to ensure that each task is individually robust. Many embedded IDEs come with RTOS-aware tools to examine the status of OS objects. But, most IDEs do not allow developers to debug individual tasks. eBinder provides unique capabilities to verify tasks individually and to build a system iteratively.

- **Task Level Debugger:** you can debug a task at the source level while the rest of the system runs. This means that a breakpoint hit in the debugged tasks will not affect interrupts or other tasks that are running. Breakpoints can be set in a specific context so a common function called from the debugged task will only break when that task calls it, and will



Powerful system analysis tools work in real time

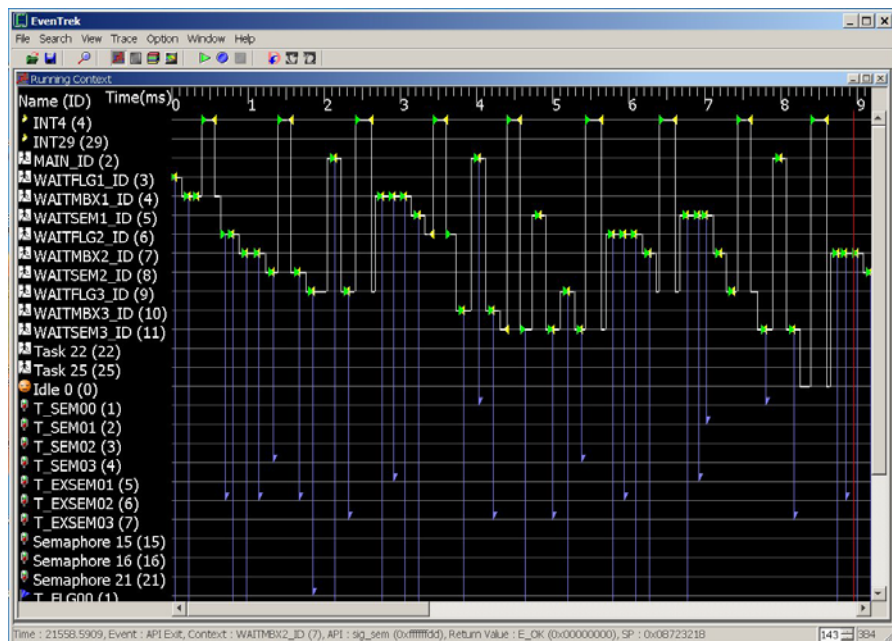
still run when any other task calls it. Up to eight tasks may be debugged individually in eBinder while the rest of the system is running.

- **Iterative Development:** eBinder supports iterative development through dynamic loading: you can load a piece of object code to be dynamically linked to the system while it is still running. After the new object code is loaded, a new task can be created dynamically to execute the new function. In this way, developers can start with a small system and iteratively add capability to build a complex and robust application.
- **System Analysis Tools:** eBinder provides sophisticated tools that enable analysis of complex, real-time systems.

EvenTrek™ collects and analyzes system log information for each running context. EvenTrek analyzes:

- API trace for RTOS, system, and user-defined APIs
 - Includes both graphical and detailed formats for easy understanding and detailed analysis
- CPU utilization per context
- Stack usage per context

User-defined triggers allow EvenTrek to analyze only the selected pieces of the application.

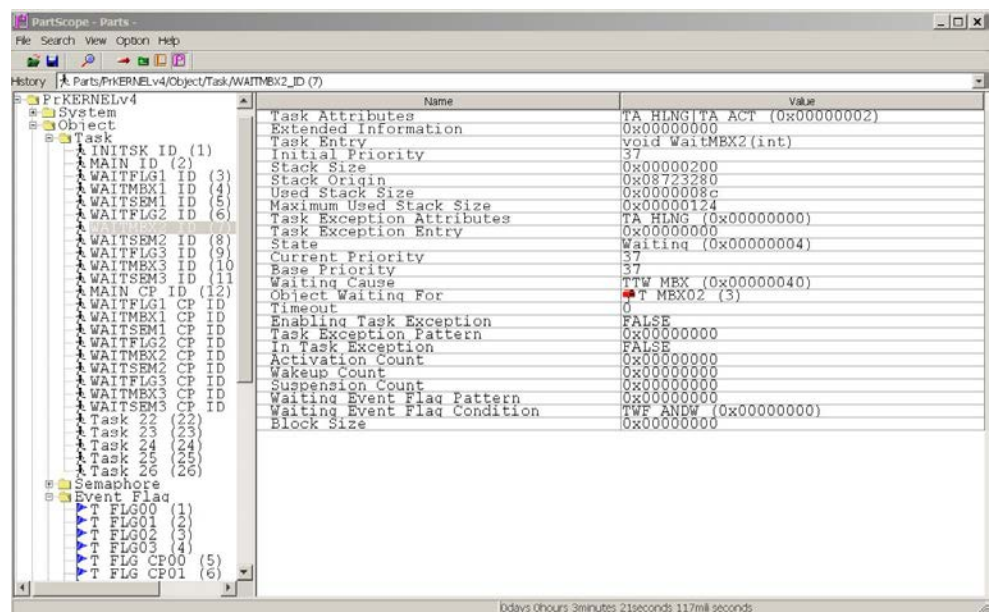


EvenTrek graphical context display



PartScope provides awareness for all RTOS, systems and middleware objects in the application. PartScope displays a real time system snapshot. Information includes:

- RTOS object states such as task status, flag/semaphore state, etc.
- File system status such as files open, media available
- TCP/IP status such as connections' states, interface status
- Other middleware or user-defined information

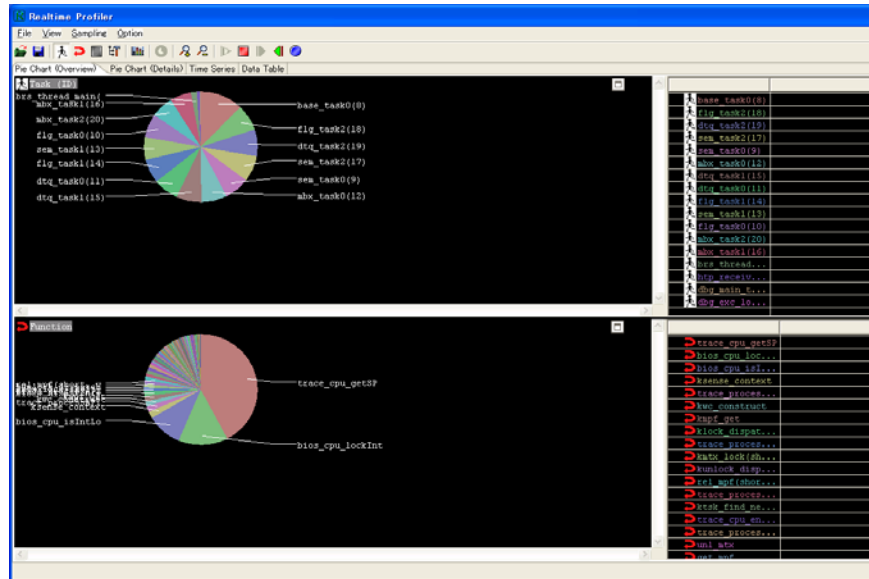


PartScope task status display

Realtime Profiler helps the developer to find out the overall system bottlenecks by showing the system's performance data using easy-to-understand pie charts.

- It collects data by sampling, thus reducing the system overhead compared to other system tracing tools.
- The CPU time taken by each process, task and function is displayed both in relative percentage and microseconds.
- Using the Real time Profiler with EvenTrek gives the exact picture of the system's behavior. Thus the system design can be verified.

Analyze the system in real time – without stopping the target



Realtime Profiler's pie chart display of task and function status

- **Shell:** A powerful Unix-like command interpreter brings the features of eBinder to a shell environment. Users can execute functions on the target, get debug information from the command line, automate execution through scripting, and much more. The eBinder shell implements command-line control of multitasking debug capability.

Real Time Debugging

eBinder's debug tools bring powerful real time debugging to embedded development. eBinder was developed specifically to debug and develop applications without stopping the target.

- **Task-specific breakpoints:** Consider a scenario where a developer wants to debug the next execution of a function, but doesn't know what task will call that function first. In a standard debugger, the developer would set a breakpoint on the function, and the entire application would stop at the breakpoint. eBinder allows a developer to set a breakpoint in an undetermined context, so the context will be chosen when the breakpoint hits. This means that eBinder will stop the next task that executes the selected function, and will bring up



PLP reduces development time by encouraging platform-based system development

a debug window in the context of that task. This feature allows the developer to let the system run and still catch the function that needs to be debugged.

- The **System Analysis Tools**, EvenTrek, PartScope, and Real time profiler run in real time. These tools update all analyzed information without halting the system.
- **Programmable triggers** allow the developer to have fine control over what parts of the system to debug. Developers can analyze and log information for only the part of the system that needs attention.
- The **Shell** provides a Unix-like environment to execute functions in real time. Even if the function was not called in the application, as long as it resides on the target, it can be called from the shell.
- **Dynamic Loading** allows developers to dynamically change their system without restarting the target. Incremental debugging can be performed without having to stop the target and reload the code for each small change.

Software Engineering Tools

- **Platform packaging (PLP) feature**

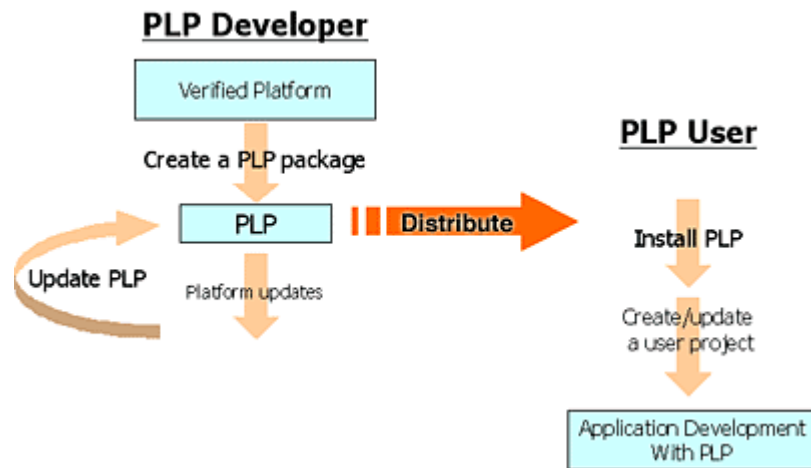
eBinder's Platform packaging feature assists platform-based system development.

A platform is nothing but the verified boot image, including RTOS, BIOS, the basic middleware, and the related configurations.

Platform-based system development is useful in two contexts. One is where a large number of engineers are involved in a large-scale project. In such cases, the system may be divided into different portions, like device drivers, middleware, and applications, and engineers work on each portion separately. In these cases, some developers work first on the platform, which is the base of the whole system, including RTOS and BIOS. Then, using the standardized platform, other developers proceed with each part of the application.



Simulator for early stage application development



In both cases, using eBinder`s PLP feature, platform developers can create a platform package (PLP) from an already-verified and configured platform to distribute to platform users. PLP users are usually middleware developers or application developers.

Key merits of using the PLP feature of eBinder

- In a large team development, PLP developers (driver/middleware developers) and PLP users (middleware/application developers) can be divided and managed to increase the entire system`s productivity.
- PLP can be used to control the usage environment, for instance restricting the modify/build access of the source code and other configuration information to the PLP users.

Simulation for Early Application Development

Application development must often start before the target hardware is ready. A common solution to this problem is to use a reference design or evaluation board. However, development groups may not have enough boards and associated tools for all members of the group, or porting the entire system to a different board may not be worth the time. Doing application development on a simulated system can be a viable alternative to using a reference board. eBinder provides a high-performance Instruction Set Simulator (ISS), eB-SIM, which simulates a target board on a PC. Typical simulators offered by other IDEs are either kernel simulators, or simple Instruction Set Simulators.



Kernel simulators cannot support interrupts, do not support real timing, and cannot run a real cross-compiled RTOS. Simple Instruction Set Simulators generally do not support interrupts, which means they don't support an event-driven RTOS.

eB-SIM combines ISS technology with techniques used in kernel simulation to achieve true interrupt simulation. eB-SIM allows developers to run a real cross-compiled RTOS in the same way it would run on real target hardware. eB-SIM simulates:

- Interrupt controller
- MMU
- Timer (for fully functional RTOS)
- LCD and Touch Panel
- BSD Sockets
- Serial communication
- Peripherals added by the users

eB-SIM simulates ARM cores with the ARMv4T and ARMv5T instruction sets. It also simulates SH cores with the SH-4 architecture, simulating the interrupt controller in the SH7750. eB-SIM comes with an SDK for developers to add simulations for their own hardware and peripherals. eB-SIM was designed with eBinder to provide a seamless transition between the simulation environment and the real target environment. eB-SIM allows product groups to start large-scale development while keeping hardware costs to a minimum.

Summary and Conclusion

eBinder brings value to embedded development by providing a complete solution to debugging complex, multi-threaded real time applications. While most other tools offer some features for multi-threaded debugging, eBinder was designed from the beginning to provide complete support for real time, multi-threaded environments. Using eBinder decreases the time to market by letting product groups focus on enhancing their application instead of concentrating on the platform and tools. eBinder provides an RTOS-based development platform with powerful tools for multi-threaded and real time debugging. eB-SIM provides a way to quickly start development before the hardware is available. By using eBinder, developers can create a high-quality, full-featured product that goes to market quickly.



Founded in 1975 in Tokyo, Japan, eSOL is a leading embedded software developer with core technologies in real time operating systems. We develop, market and sell proven RTOS suites, along with a rich set of vertical oriented middleware libraries. Our rugged software development tools provide optimal reliability in backing up the highly complicated development process for RTOS-based applications. We know that a reliable RTOS and development tools make a significant difference to the quality and timeliness of our customers' products in a continuously growing and competitive world market. Today, our customers - global OEMs and ODMs ranging from consumer electronics to automotive applications - ship millions of products with technologies pioneered by eSOL.

For more information, please visit <http://www.esol.com/>.

eSOL Co., Ltd.
Embedded Products Division

Harmony Tower, 1-32-2 Honcho
Nakano-ku, Tokyo 164-9721, Japan
Tel: +81 3-5302-1360 Fax: +81 3-5302-1361
ep-info@esol.co.jp