

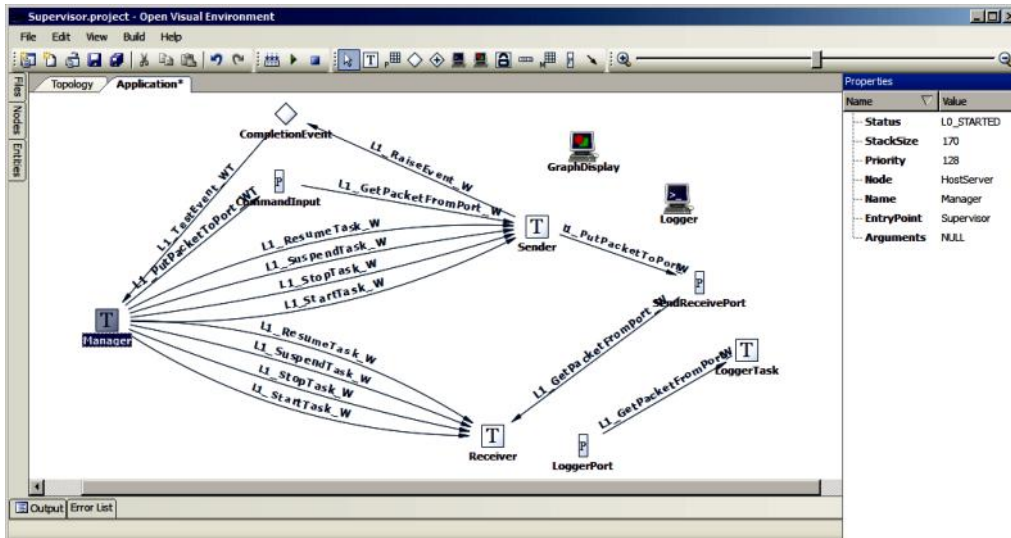
OPENVE

A Unified Visual Development Environment for High Reliability Embedded Applications

Developing distributed real-time embedded applications used to be hard hard. The development is even harder when the requirements dictate the use of heterogeneous target processors. The target processors might use different runtime environments, including legacy Operating Systems. Altreonic's OpenVE provides the solution. OpenVE was developed using the

same principles that govern Altreonic's systems engineering methodology.

Using meta-modeling, embedded software engineers can define their own targets, target topology, application topology, graphically create the program code, build and run the executable images. Before the final



target is used, they can run the application on a host operating system (Windows or Linux) allowing them to verify the behavior. A built in event tracer allows users to verify the scheduling and task interactions. Once done, a small click of the mouse changes the target processor for each node and with little or no source code changes, OpenVE generates code for the final target. The event tracer displays the execution trace on the host. The host nodes can even remain part of the system e.g. for accessing stdio services, graphic displays, file services, TCP/IP sockets, etc.

Five easy steps to develop a distributed multi-tasking application:

1. Define the topology:

- Put the processing nodes on the canvas and connect them

2. Define the application:

- Put the tasks on the canvas, define the properties
- Put the interaction icons on the canvas by selecting from the menu events, semaphores, ports, fifo's, resources, memory pools, packet pool or generic hubs
- Add a hostserver task to interact with the application
- Connect tasks and interaction icons. Select from the available services.
- Add your application specific code.

3. Build:

- OpenVE will generate data structures, add drivers and system tasks,
- Generate routing tables and makefiles.

4. Run and watch.

5. Verify.

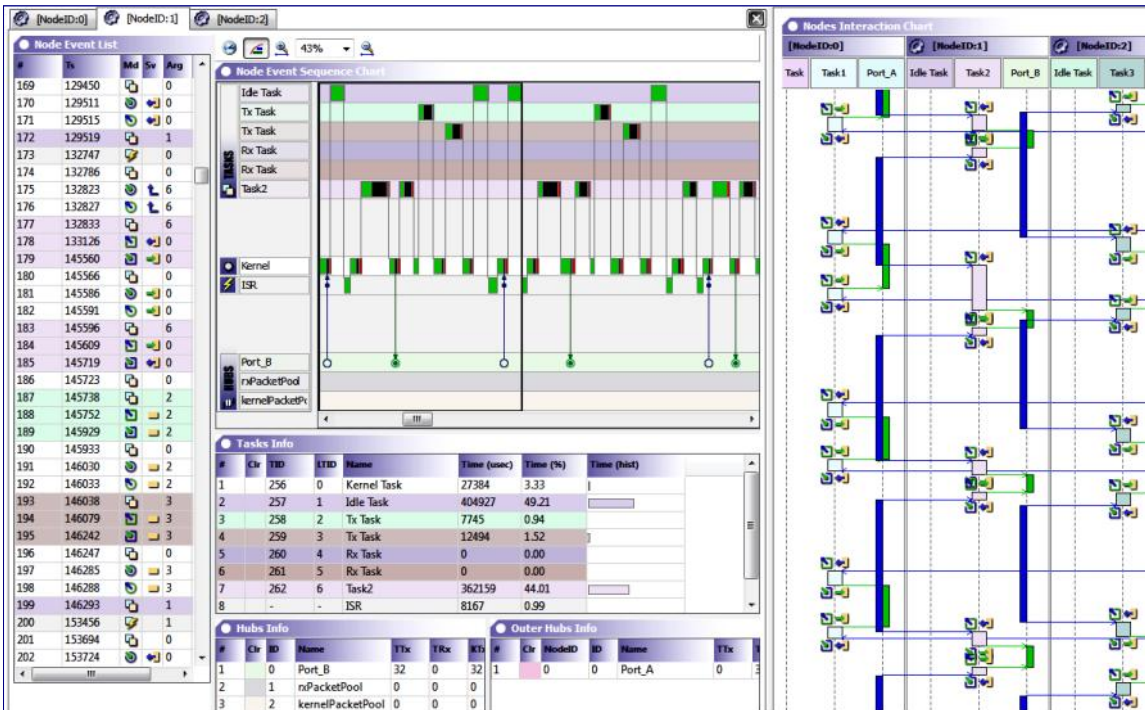
- Just open the tracefiles and analyse the events that happened.

From Deep Space to Deep Sea

WWW. ALTREONIC.COM

Push button high reliability





The event tracer acts like an oscilloscope for the software.

The node interaction diagrams clearly show the interprocessor communication.

Statistical profiling data complement the analysis.

What are the benefits of OpenVE?

1. Productivity.

The real-time embedded engineer only needs to use a single integrated environment for specifying and modeling his embedded software architecture, to simulate it and to generate code for his target.

2. Target independent.

OpenVE is based on a metamodel concept. New OpenComRTOS services, can be added without changing the kernel. Define a suitable icon and the service becomes part of the enhanced RTOS. Adding a new target processor is as simple as adding a folder to the file system. What's more, the target system can be heterogeneous, covering from 8bit and 16bit microcontrollers over high-end 32 or 64bit processors, to legacy operating systems providing access services.

3. Trustworthy development.

Writing software is an error prone process, especially when the application is complex and consists of a large number of interacting and concurrent entities. OpenVE makes this straightforward whereby applications can be drawn on the canvas. OpenVE generates the whole framework so that the developer only has to fill in the core functions.

4. Performance and safety.

OpenVE supports the OpenComRTOS programming model. The latter was developed and verified using formal modeling techniques and is a break-through RTOS design. It has a very clean and safe architecture, 5 to 10 times smaller than equivalent RTOS designs. No buffer overflows are possible and full scalability is guaranteed. The programming model remains the same, whether programming single processors, many-core CPUs or networks of widely distributed processing nodes.

