

Eclipse: The development system that crosses RTOS boundaries

By Robert Day

In the embedded software world, the choice of real-time operating system has typically dictated the choice of development tools. In the 80's and 90's, close relationships were established between RTOS vendors and tool providers (for example, Microtec Research and Software Components Group with XRAY and pSOS).

At the end of the 90's, industry consolidation resulted in major RTOS vendors and device providers owning their own tools technology. With the increase in software in every new embedded design, the reuse of software is becoming critical, making it very unattractive to keep switching embedded software tools.

This leaves us in an interesting tools conundrum. Which tools do you choose if you are using a given RTOS and device combination? And how can you protect your investment in tools (and code generated using those tools) if you switch your RTOS and/or devices? This is further exacerbated if more than one RTOS or devices are used in the same design – a common occurrence in the System-on-Chip (SoC) era.

Are standards the solution?

Standards are often a good way of helping with code reusability. The widespread use of ANSI C as a development language across embedded systems gives developers a chance for reuse. However, there are no real standards for tools and real-time operating systems. Therefore, each design has to use a specific compiler, which generates code for a specific device, to work with a specific RTOS. The RTOS

and the tools have a proprietary API from compiler flags, through debug GUI, through IDE interface. This means that there is a large amount of relearning and retooling per project.

What is really needed is an environment that is standard across all embedded systems tools. This would allow the tools and RTOS vendors to be able to plug and play with each other, and give users the benefit of a

single tool to manage their projects and code. The environment would also provide standard interfaces and interoperability across the multiple tools and operating systems that are available today. This same environment could also support embedded projects that use a proprietary RTOS, or no RTOS at all.

One of the issues of creating such an environment touches on embedded systems

politics. If either a device company or an RTOS company created this environment, it would become very difficult for their competitors to embrace this as their standard, as it would leave them somewhat at the mercy of their competitor. So having an environment developed and maintained by an agnostic third party is very appealing.

Looking to the desktop or enterprise world is an interesting exercise, as it inherits the benefits of a huge developer network, and the ability to use desktop productivity tools (such as standard editors and version management systems). The problem with this approach is that the desktop world is very host specific (for example, Microsoft Visual Studio). In addition, if embedded features were required, the desktop tools providers would not be readily open to develop those features as they would have a relatively small base of embedded customers.

The Eclipse solution

The good news for the embedded world is that a solution exists. While it may not be well known in the embedded world yet, the solution known as Eclipse is set to revolutionize the embedded software developer's environment. It is as much a culture as it is a product, and it has been embraced by major embedded solutions providers as their standard tools environment.

Eclipse is an open platform for tool integration built by an open community of tool providers. To quote the Eclipse website: "The Eclipse Platform is an open IDE for anything, and for nothing in particular." It was developed by IBM and first released in 2001. In 2004, it was spun out of IBM into a nonprofit corporation called the Eclipse Foundation.

The technology is an open framework that is available in source code. The framework is written in Java, and is highly portable across host environments. To date, it is available under Windows, Linux, Solaris, HP-UX, Mac-OS, and IBM AIX.

Eclipse plug-ins

A key factor that makes Eclipse useful is the notion of plug-ins. Each tools provider can build their tools to a certain set of rules and APIs that allow them to *plug in* to the Eclipse framework. In the embedded world, this enables embedded tools providers to build true embedded products that will simply plug into the IDE, and allow them to focus on their core competencies without worrying about developing IDEs.

If the Eclipse plug-in rules are adhered to, the embedded tools will track the latest versions of the Eclipse framework, as well as plug into other Eclipse-based environments.

Eclipse licensing

Another key factor that makes Eclipse useful is the licensing model under which the Eclipse framework is provided. The Common Public License (CPL) provides royalty-free source code and world distribution rights, and allows tools developers to offer the Eclipse framework and their plug-in products without putting their own Intellectual Property (IP) back into the community.

This makes for a very viable business by allowing an open source framework to be a common vehicle, with a common look and feel. The open source framework can also be maintained by an RTOS- and device-agnostic organization, and contain detailed embedded functionality provided by the embedded companies.

Eclipse user advantages

Many of the embedded RTOS companies are now providing an Eclipse-based solution. This means that the embedded user will have a common platform for the common development functions such as project management, editing, file navigation, and build management. The user will also have a common interface to compilation and debugging tools.

If the RTOS providers have implemented their plug-ins correctly, then this one environment can host multiple operating systems without having to change environments. Each RTOS vendor can provide tools that will have a common Eclipse look and feel, but with specific features that help build and debug applications using that RTOS.

An example of where this is particularly beneficial is in companies with a large product portfolio that serve the same market, but necessitate different user requirements – such as in cell phones.

Cell phone example

High-end cell phones may need a PDA-like operating system such as Palm, Windows CE, or Symbian – whereas middle and low-end phones may use more of a true embedded RTOS like Nucleus PLUS. Much of the software IP will be the same, so having it under a single project manager is very beneficial. The devices are also likely to be the same, so the compilation system can be consistent. Only the RTOS environment and high-end applications will change, and Eclipse can facilitate this.

The cell phone example is also key when considering a multiple-core architecture, as most cell phones use at least one standard processor core, and at least one DSP. Eclipse can host the different compilers, debuggers, and RTOS choices for each processor in one environment.

Perspectives

Eclipse operates using the notion of perspectives, as do engineers. When building, engineers use a build perspective, and when debugging, they use a debug perspective. A nice feature of Eclipse is that plug-ins can cross perspectives. For example, when debugging, the editor and project manager can be made visible and used to make changes to the code if bugs are found, and to navigate around the project to fully understand the context of the debugged code. An example of an Eclipse debug perspective is shown in Figure 1.

Non-embedded plug-ins

Another advantage with Eclipse is the ability to plug in development productivity tools that are not specific to embedded systems. Currently available Eclipse plug-ins provide the following functionality:

- Modeling
- Bug tracking
- Code generation
- Graphics/Drawing
- Source analysis/Testing
- Project/Team management
- Source control (such as CVS, ClearCase, and SourceSafe)

Eclipse provides many advantages for embedded developers; it is now in the hands of the embedded solutions providers to make it a reality.

Nucleus EDGE

At Accelerated Technology, we decided to change our existing development tools environment in 2002. We had based our last generation of tools on Microsoft Visual Studio, as it had offered the only available standard at the time. However, it was proving to be difficult to meet the emerging group of embedded developers who wanted true cross platform support (including Linux host support) and a multi-core embedded development environment.

We took note that Eclipse was emerging as a standard in the enterprise world that offered us a good technical solution and observed that the Eclipse business model allowed us to offer a complete solution to our embedded developers.



Nucleus Embedded Developers Graphical Environment (EDGE) utilizes the latest Eclipse platform (version 3), and realizes all of our embedded technology as Eclipse plug-ins.

We have now developed a number of true embedded tools that interoperate under the Eclipse framework. They provide a complete development suite for embedded developers who are using Nucleus software, their own proprietary RTOS, or no RTOS at all (see Figure 2).

This technology is based on two decades of embedded debuggers, compilers, and RTOS tools. We have added a new project manager interface that complements the Eclipse Navigator view. This allows embedded developers with multiple projects or cores to easily navigate and build their code.

We have also provided our own context sensitive editor that interacts with the build and debug interface, providing a powerful and consistent coding environment throughout the whole software development process. Following the

notion of perspectives discussed earlier, this advanced editor is available in both build and debug perspectives, allowing the engineers to make swift code changes as they find their bugs.

Cross compilers are a very target specific link of the embedded software tool chain. Although code is generally written in ANSI C, the compiler is the last part of the tool chain that engineers are ready to change. Within Nucleus EDGE, we offer a number of compiler options that allow users to select their favorite compiler and still receive all the benefits of Eclipse. We offer GNU cross compilers, processor vendor compilers (for example, the ARM RealView compiler), and our own compiler (Microtec compilers). This also allows users of new device architecture to select which compiler is best for their application.

Embedded debugging is a vital part of the software lifecycle. In Nucleus EDGE, we provide many options while using the debug perspective to best accommodate the embedded engineer's debug environment. We offer a native debugging environment (and native compiler to complement it), which allows users to build their applications to run on the architecture of their host machine (for example, PC). This offers rapid software prototyping and

is very useful early in the project when hardware is not defined or not available.

The same debug interface also plugs into an Instruction Set Simulator (ISS). This allows the user to compile with the cross compiler but to run on simulated hardware. This is very useful to run real system tests if the hardware is not available. The ISS allows detailed examination of code execution and memory, and the debug environment executes this at the source level. The simulation of the clock tick allows the RTOS to also be run at this point, and true system debugging can begin. When target hardware is available, the same debug environment allows connection to the hardware typically through an on-device debug port (JTAG for example).

This debug interface provides a real RTOS-aware view of the system. It provides monitoring of the different RTOS states (such as semaphores, queues, and mailboxes), and information about the state of a particular task or group of tasks. This can be achieved when the system is running (run-mode), or when the system has stopped at a breakpoint (stop-mode). The inclusion of the profiler plug-in can go to the next level, providing full system and timing information down to the task and memory level. This profiler is available as a new perspective, allowing for quick

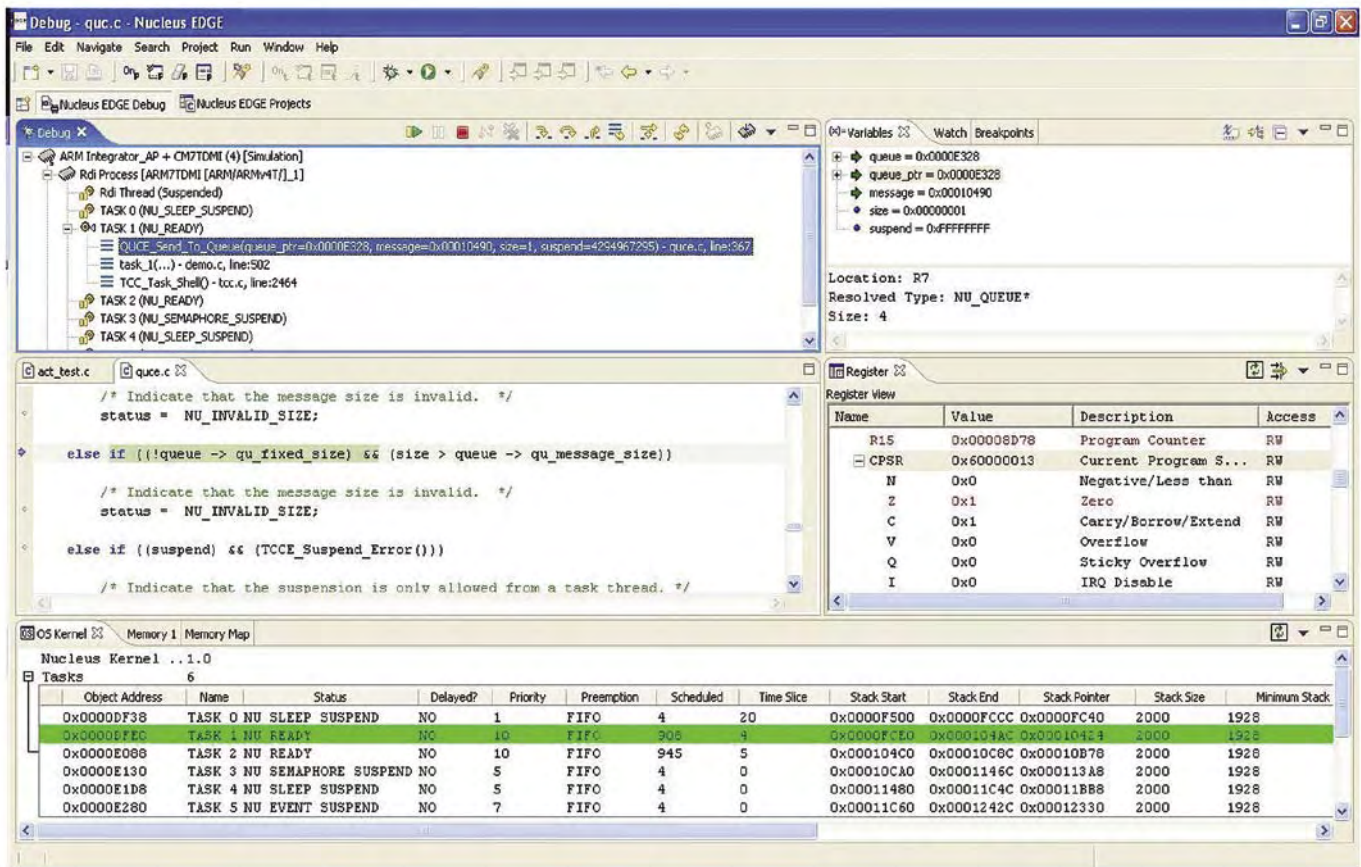


Figure 1

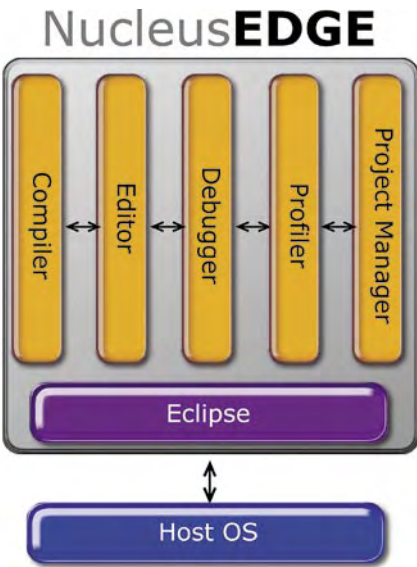


Figure 2

transition from code to system level. The profiler is shown in Figure 3.

An environment that crosses RTOS boundaries

Eclipse and Nucleus EDGE provide a real embedded environment that crosses over RTOS boundaries. Eclipse can offer a solid platform that RTOS vendors can use to build tools to support their RTOS that will have an Eclipse look and feel, and then the embedded developer can take the platform and use whatever RTOS makes most sense for his application without having to change his environment.

Nucleus EDGE provides a proven set of tools that can be used with the Nucleus RTOS, and can also be easily modified to work with another operating system – even a proprietary one. The embedded user then gets the benefit of needing only one set of tools regardless of RTOS. The embedded world is changing for the good of the embedded developer, and Eclipse is helping to fuel the change. **ECD**

Robert Day is the Director of Marketing for Accelerated Technology, a Mentor Graphics division. Since 1987, Day has held a variety of engineering, sales, and management positions at Mentor Graphics and Microtec Research. Day has more than 15 years embedded industry experience. He holds a Bachelor of Science degree in Computing from the University of Brighton, England.



For more information, contact Robert at:

Accelerated Technology
 739 N. University Blvd.
 Mobile, AL 36608
 Tel: 251-208-3400 • Fax: 251-343-7074
 E-mail: info@acceleratedtechnology.com
 Website: www.acceleratedtechnology.com

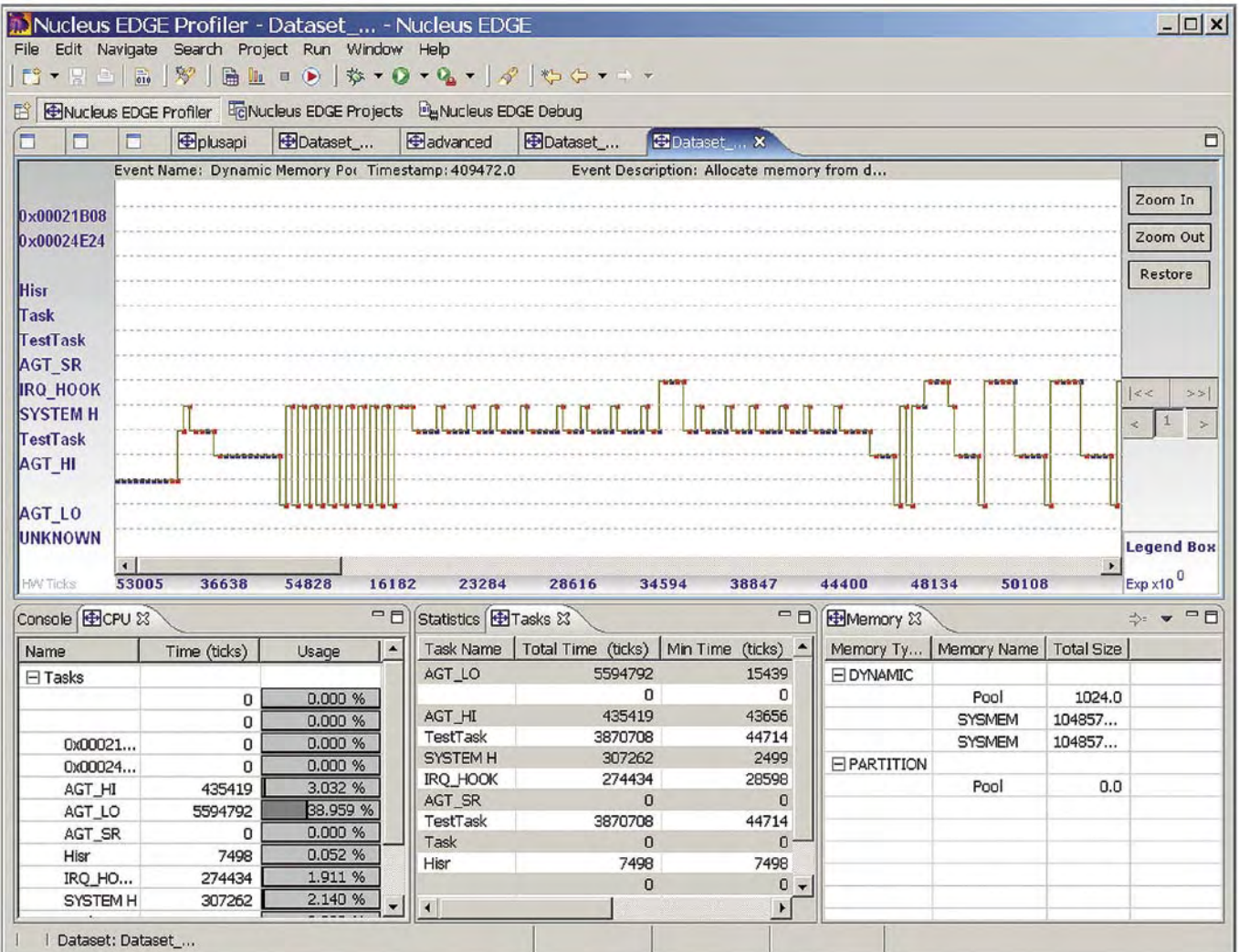


Figure 3