

How Eclipse fits with embedded development

By Madison Turner and Robert Day

Providing tools for the embedded software developer is a very complex task. The wide range of processor architectures, development host systems, Real-Time Operating Systems (RTOSs), and application-specific requirements has traditionally meant these tools have been proprietary in nature.

Unfortunately, this means the embedded software developer has had to relearn and rebuy solutions all essentially doing the same thing. No standard environment or tool suite could meet all these diverse requirements ... until now.

Enter Eclipse. Born from the enterprise space, this open environment is now providing a common platform that embedded developers can *buy once, use many*. But, how exactly can this nonembedded platform meet the needs of embedded developers?

Eclipse is not a product; it is a framework. It allows embedded vendors to plug in proprietary tools into a common environment. Part of its appeal is that it actually goes much further and defines the look and feel of plug-ins, too.

For embedded developers, this means:

- They have a common Integrated Development Environment (IDE) meeting many of their common software development needs, such as project management, version control, and source code browsing
- They also have access to tools meeting specific embedded needs that plug into the environment and offer a common look and feel

Let's examine a couple of examples of embedded software tools used for different purposes, but share this common IDE.

Debugging a Linux application on a standard COTS board

In this case, the embedded software team is very software and application centric.

The team does not need deeply embedded probes and hardware tools, as the application is running on debugged COTS hardware. Instead, much more attention is spent on the software issues, not dissimilar to the enterprise space. The standard Eclipse platform offers a rich project navigation and project management plug-in focused on developing application software. The CDT project from Eclipse adds some specific C and C++ build tools that give specific C/C++ editors and a sophisticated build environment built around the GNU compilers. For a Linux developer, this is a good starting point to help manage source files and Linux builds.

Each of the embedded Linux providers also offers a debug plug-in to Eclipse that allows the developer to debug their embedded Linux applications, with full awareness of the target hardware and what the Linux operating system is doing as the developer steps through the application. This debugger is often connected to the COTS board using standard Ethernet connections, and hence doesn't even need hardware connection technology to facilitate it. All of this is achieved without leaving the Eclipse environment, and better still, is integrated with available software management products not specific to embedded development.

Linux developers can take advantage of open source for both the OS and the tools, but also have an environment that is embedded aware, of high quality, and with a common Application Programming Interface (API) to the tools used in hard real-time systems if needed. This last point becomes relevant when embedded systems have a large application part that can be well served by Linux and a hard real-time part that needs to be serviced by a hard RTOS. Eclipse can be used for both.



Figure 1 displays an example of how Eclipse can be used to debug Linux to the thread level, showing LynuxWorks' Luminosity IDE in action.

Debugging a hard-real time system on proprietary hardware

For more deeply embedded devices, the Eclipse framework can utilize a debug perspective that allows connection to the target via the JTAG port available on most embedded devices. Embedded developers face the issue of the large number of target boards that exists, each configured slightly or significantly different. Connection wizards ease the process of establishing a connection to an embedded target by providing standard configurations for popular targets and connection devices. These wizards offer a centralized and easy-to-use interface for specifying debug session parameters. For example, the user can choose an output file to load automatically and specify a symbol to run to upon loading.

While a register view displays and modifies register values, a variable view operates similarly for the contents of variables and data structures, with structures and their members laid out hierarchically for a logical and usable view of the target data. A memory view displays and modifies an address or address range. In these views, values are displayed in the radix specified by the user: binary, decimal, hexadecimal, or octal. A memory map view graphically displays the layout of the application in target memory in terms of program sections, files, or functions.

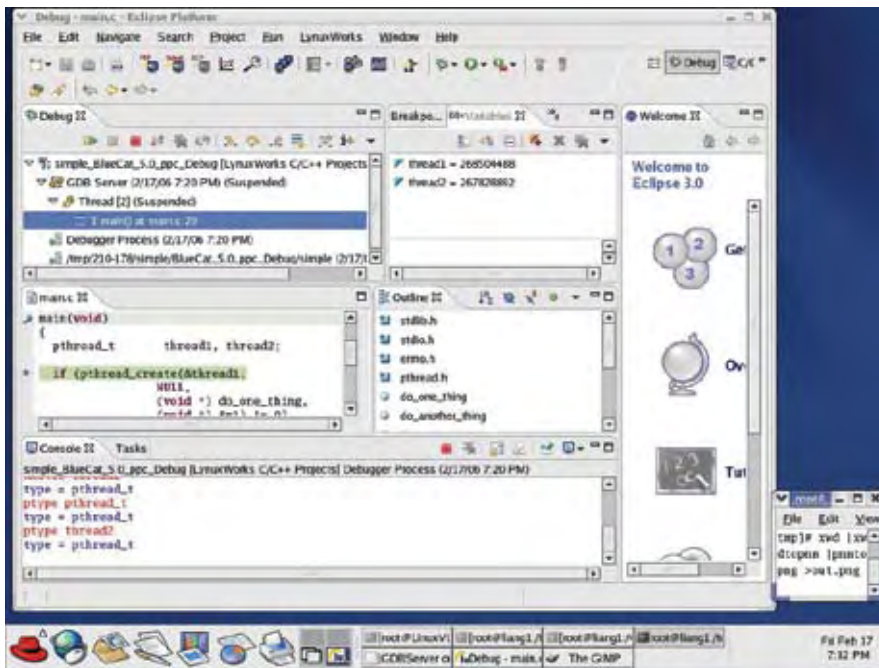


Figure 1

over the JTAG connection and made easily navigable within the debug view. Kernel-aware debugging makes it easy to analyze task interaction, real-time logic, and memory usage. It also makes task-specific break-points available.

These examples show how the different hardware, software, and application-specific requirements typical in today's diverse embedded applications can be united in a single development environment – powered by an open source platform called Eclipse, the today and tomorrow of embedded tool environments.

Madison Turner is a technology analyst at Mentor Graphics Embedded Systems Division, where he focuses on next-generation development tools and methodologies.

Robert Day is the vice president of marketing for LynuxWorks. His responsibilities include leading program management teams and driving worldwide marketing initiatives, including corporate communications and brand strategy.

For more information, contact Madison or Robert at:

Mentor Graphics	
739 N. University Blvd. Mobile, AL 36608 Tel: 251-208-3400 E-mail: madison_turner@mentor.com Website: www.mentor.com/embedded	
LynuxWorks	
855 Embedded Way San Jose, CA 95138-1018 Tel: 408-979-3900 E-mail: rday@lynuxworks.com Website: www.lynuxworks.com	

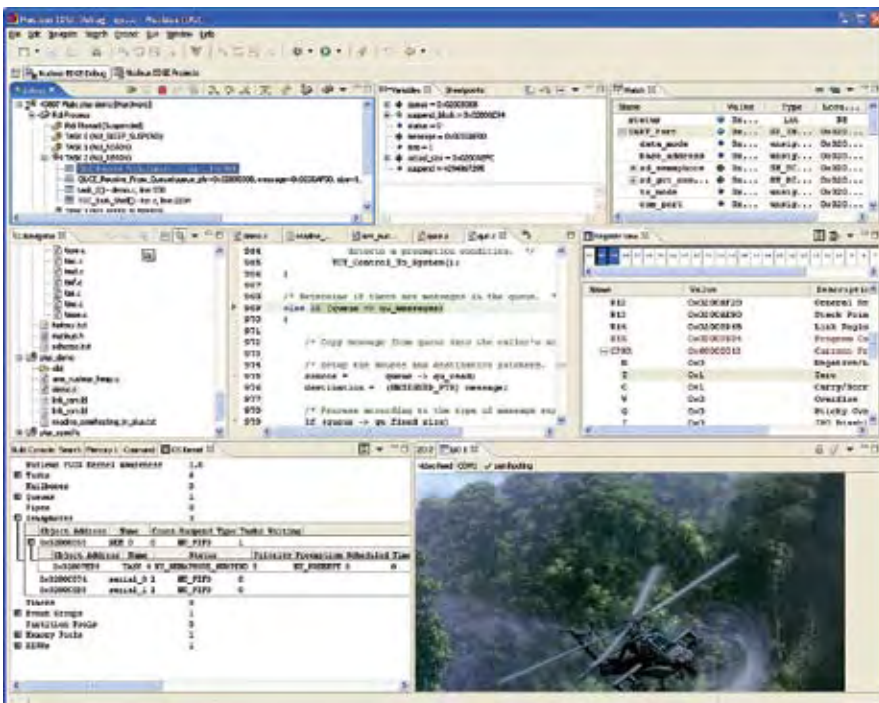


Figure 2

With the addition of a scripting language for application-specific debugging support, the combination of an Eclipse-based embedded debugger and JTAG probe can provide unprecedented levels of insight into the target application. In this scenario, scripts carry out debugging operations on the target and format, manipulate, and display that information on the host. In this way, extremely sophisticated debugging techniques can be developed specific to the application under test. For example, a block of target memory values

representing a video file can be uploaded via JTAG and viewed in a media player on the host to ensure the integrity of the content. A tool such as the Mentor Graphics Nucleus EDGE IDE shown in Figure 2 supports these functions.

Another powerful feature that relies on the Eclipse framework in concert with a JTAG connection is kernel awareness for a hard RTOS. Kernel structures, including tasks, communications mechanisms, timers, and memory pools, are monitored