

# Integrated Environment is Key to Real-Time Software Development

Software engineers searching for the perfect development environment for an important real-time software project should consider integration a major selection criterion. Successful development of real-time software applications requires the tight integration of a software development toolset, the real-time operating system (RTOS), and compilers that are targeted to the microprocessor architecture. Because mission critical, real-time applications require a reliable operating system environment, the incorporation of a UNIX-based RTOS is another important consideration.

The integration of the software development tool set and the RTOS is especially critical in real-time applications because there is a need to correlate events that are occurring within applications to the RTOS activity. Standalone tools may have the ability to correlate application events, but lack the ability to provide the critical timing relationship between the RTOS activity and application activity.

The ability for the various tools to communicate with one another provides other advantages. The developer may use an analyzer tool to determine when and where a failure has occurred. The next step will be to invoke a debugger to implement and test a change. The ability to quickly switch from one tool to another saves time and reduces the possibility of errors. It is also a significant advantage if the debugging tool can communicate with the appropriate language compiler and have the compiler generate executable code that can be inserted into the program by the debugger without having to re-compile the application. This saves time. Equally important, the inserted code will be less intrusive and, consequently, will preserve the real-time characteristics of the application.

I can cite a specific example where this integration proved extremely important. An engineering group was developing a series of applications to gather and process data from a targeting radar system. One of the

applications was experiencing a failure after 15,000-plus iterations. It required multiple integrated tools to detect, locate, and correct the problem. Standalone tools would have been of little use because they would not have been able to provide the time correlation required finding this single point failure.

## AN INTEGRATED TOOLSET PROMOTES EFFICIENT DEVELOPMENT

There is a high probability that all of the components of an integrated development environment (i.e.: the toolset, RTOS, drivers, and compilers), designed and developed within the same organization, will work well together. There needs to be specific linkages and communications between the various components to ensure integration, and this is best accomplished in a single organization. As an example, let's use the integrated NightTrace analysis tool -- part of Concurrent Computer Corporation's NightStar package. NightTrace allows the developer to analyze activity within the RTOS kernel. Because it is integrated with the OS, events such as context switches, interrupts, and system faults can be accurately tracked to within microseconds -- and then those events can be correlated with application activity. The ability for the tools to work together makes the developer's job much eas-

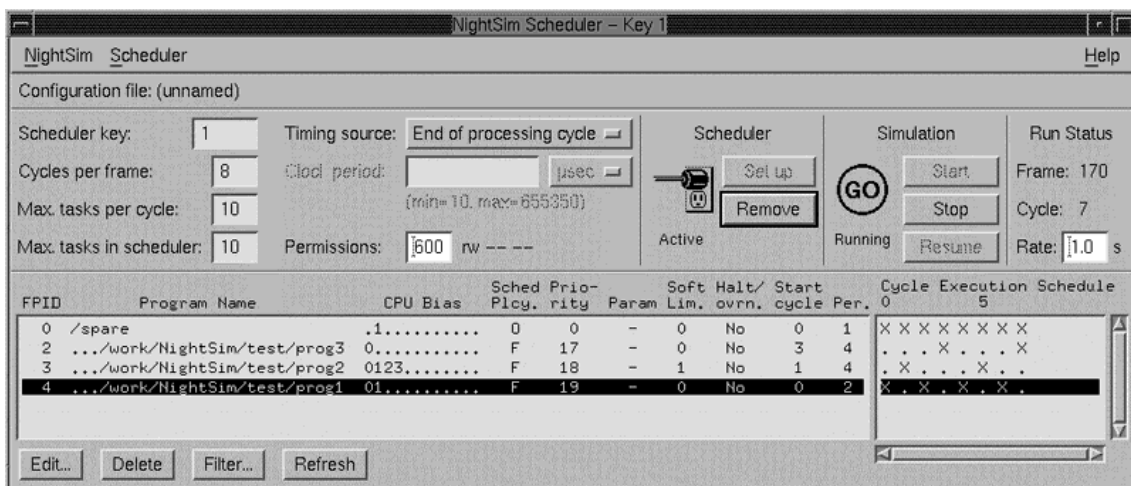


Figure 1. NightSim Scheduler

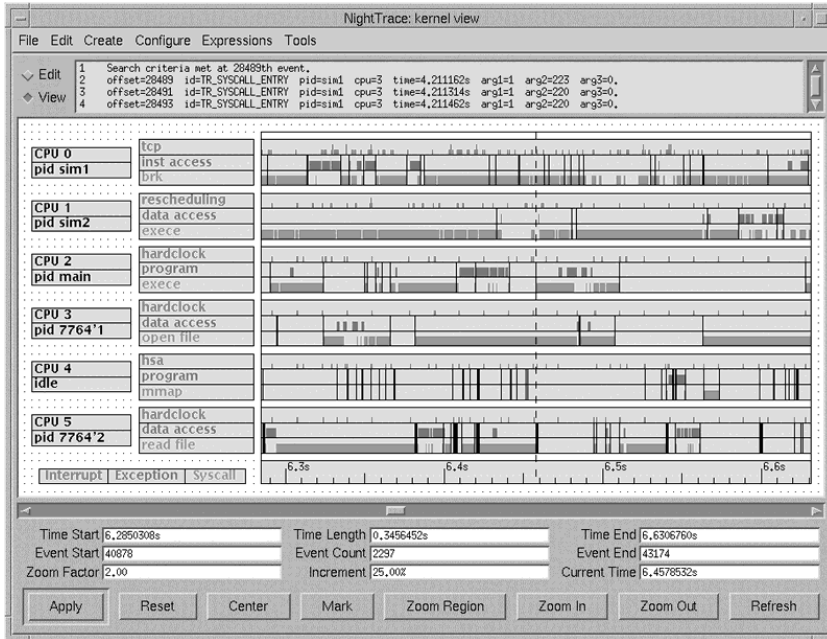


Figure 2. NightTrace

ier and, consequently, more efficient in the development of a software project.

## COMPLETE SET OF FULL-FEATURED TOOLS IS ESSENTIAL IN THE DEVELOPMENT OF COMPLEX REAL-TIME PROJECTS

The focus on the importance of integration should not overshadow the importance of having a complete set of full-featured tools. The primary tool required for any software project is a debugger. A full-featured debugger, such as Concurrent's NightView, will be enormously valuable in the development process. However, if this were the only tool available, the software engineer would have a very difficult and costly effort ahead.

The debugger is essential in helping the software engineer get his application "working." This usually means the application appears to do what was intended and there are no syntax errors. Unfortunately, in complex real-time systems involving a number of applications, this definition of "working" will not hold up. Complex real-time systems involve a number of applications that must all work together as a closely synchronized unit. The point at which the various applications are integrated into a complete system is when the

most costly and difficult task of debugging starts. It is at this time that a full complement of integrated tools is required and can yield the most benefits.

Real-time systems generally require that applications run in a cyclical fashion with some applications required executing more frequently than others did. A scheduling tool like Concurrent's NightSim will help the software engineer organize and schedule the various applications. The Graphical User Interface is very helpful here because it allows scheduling to be visualized and easily modified. The NightSim tool has many other features that help the software engineer

in the scheduling process but none is more important than the monitoring of scheduled activity while the system is running. If a scheduled activity does not complete before it is scheduled to start again this is a hard overrun, and it must be corrected. NightSim will detect and report this type of error as well as other types of errors.

The NightSim scheduling tool reported the hard overrun but another tool will be required to help correct the problem. The solution involves determining when in time the hard overrun occurred and what other activity was happening that could have caused the problem. Remember that the software engineer had determined earlier that the application was performing as expected, so the problem must be external activity. The tool that is needed here is an analyzer such as Concurrent's NightTrace. The analyzer tool will allow

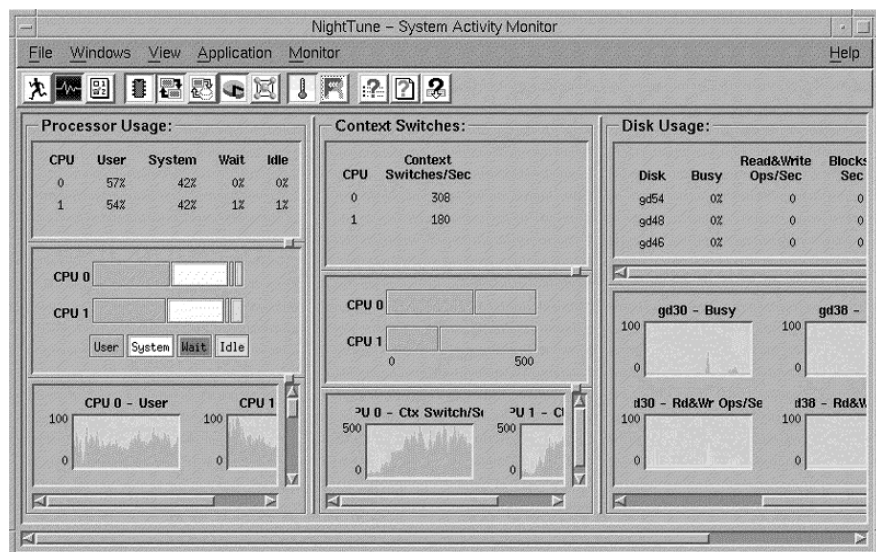


Figure 3. NightTune

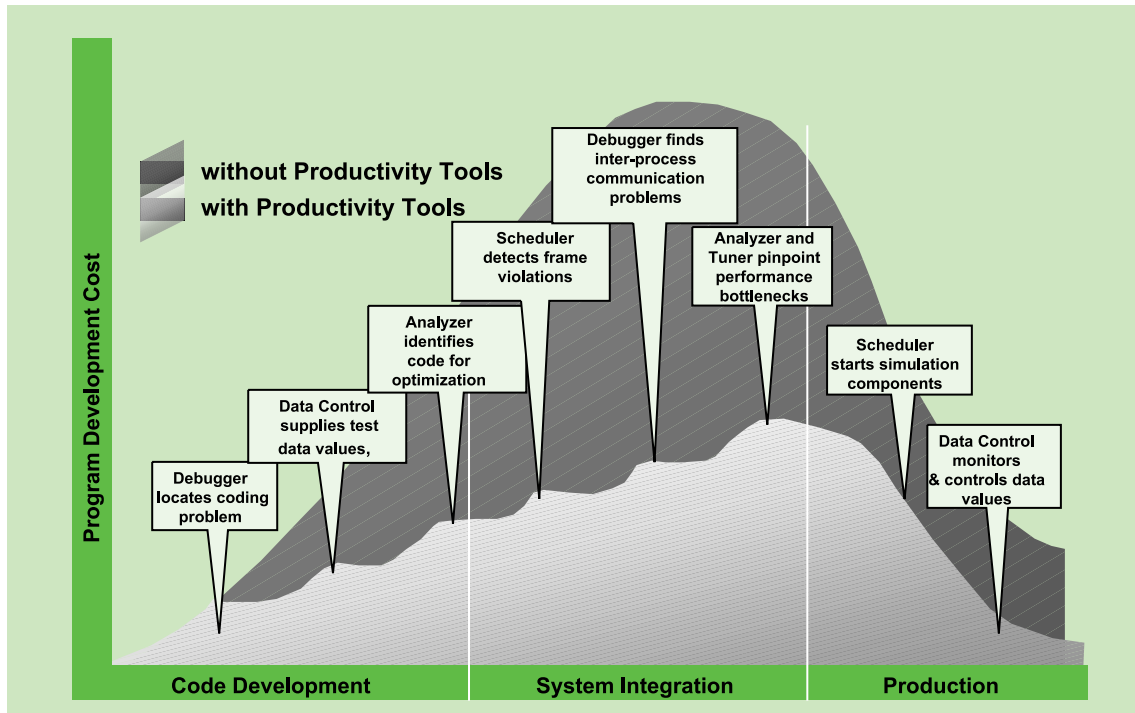


Figure 4. As a project progresses, the cost to find problems increases

the software engineer to view all activity -- within the RTOS and any other applications -- that was occurring at the time the overrun occurred. This information will lead the software engineer to the source of the problem.

The software engineer will want to implement changes

to try and correct the problem causing the hard overrun. Concurrent's NightView debugger has a unique feature that expedites this situation. The software engineer can use the debugger to modify code and compile and insert new code without having to recompile the application. The new modification can be implemented and tested immediately.

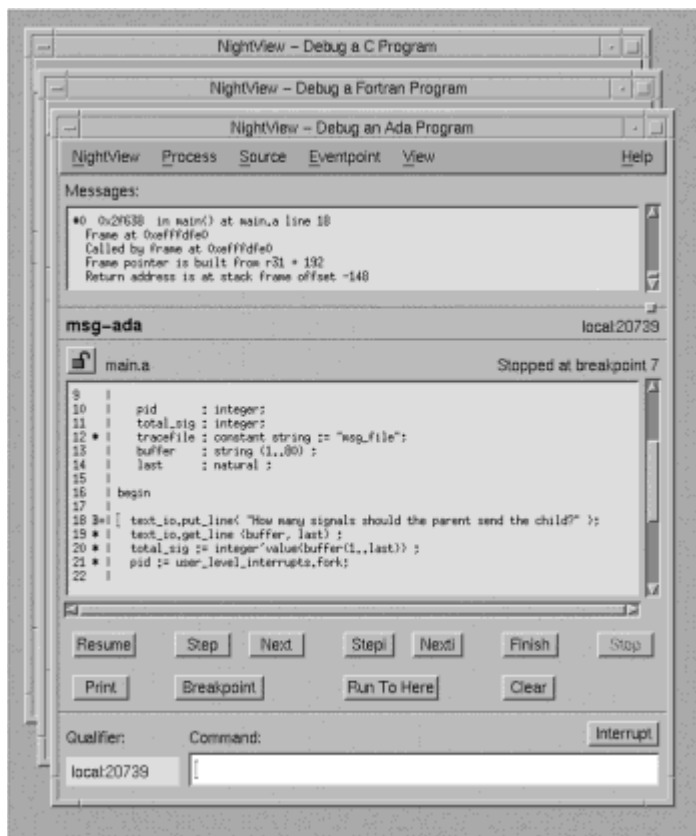


Figure 5. NightView

Efficient testing of real-time systems requires another tool, such as Concurrent's NightProbe monitor, that can simulate the varying data inputs that would be present in the system. The problem is very easy to solve with the NightProbe tool because the software engineer can supply varying values to any of the global or local static variables while the applications are running. Standalone tools cannot provide this type of capability because the tool must work with the compiler to determine physical memory locations. NightProbe not only plays a role in the debugging and testing of real-time applications, but a number of customers have chosen to use it as a run-time control mechanism because of its easy-to-use GUI interface and integrated capabilities.

One final tool that completes the complement of the NightStar toolset is NightTune. NightTune is a tool for "tuning" the system to achieve maximum efficiency. This tool graphically displays the utilization of the various system resources and lets the user easily move applications to balance the load.

Figure 4 shows how as a project progresses, the cost to find problems

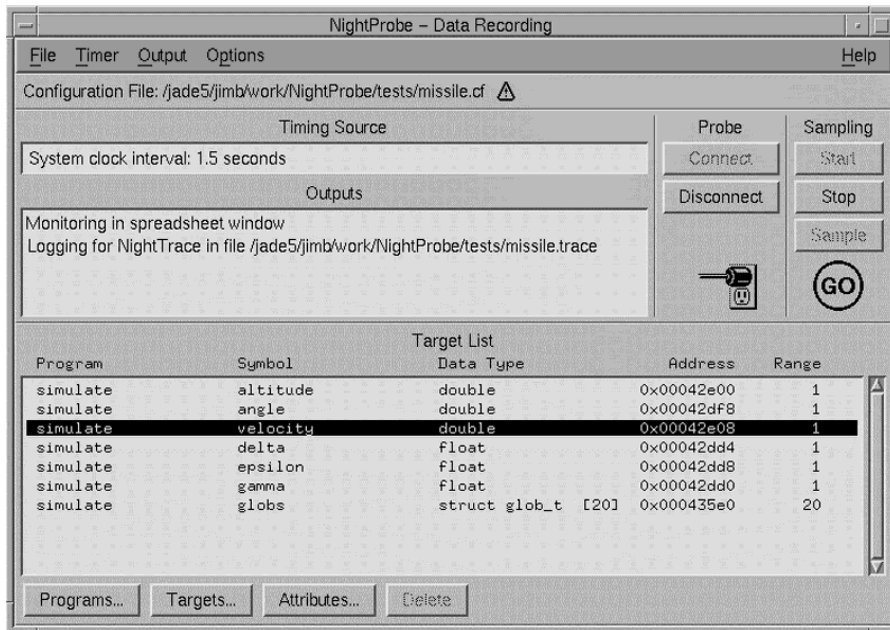


Figure 6. NightProbe

increases. Concurrent's PowerWorks Integrated Development Environment exemplifies a completely unified environment. PowerWorks includes Concurrent's PowerMAX OS UNIX-based RTOS, the NightStar productivity toolset, a compiler family, and utilities and device drivers. PowerWorks has been successfully used in dozens of mission-critical and life-critical applications in flight simulation, aircraft engine testing and medical applications.

### A UNIX BASED RTOS IS THE CLEAR CHOICE FOR MISSION-CRITICAL APPLICATIONS

Among software engineers responsible for complex real-time projects, UNIX-based real-time operating systems are becoming the overwhelming choice for mission-critical applications. This preference for a UNIX-based RTOS among real-time engineers is easy to understand when the true meaning of real-time is considered. When an activity must occur on time every time, there is a hard real-time requirement.

The key to a reliable RTOS is its maturity. Common sense dictates that the more exposure any piece of code gets, the greater the likelihood that more problems will be found and corrected. It is also important that the developers can accurately identify errors so they can be reproduced and corrected. Real-time development engineers are traditionally very cognizant of the operating system environment and are capable of defining problems and requiring their resolution. A mature UNIX-based RTOS, such as PowerMAX OS, is a necessity for mission critical, real-time applications.

#### **Integrated Approach Facilitates the Most Demanding RT Applications**

Perhaps one of the best examples of the benefits of an integrated development environment can be observed in the aircraft engine testing area. Pratt & Whitney is using the PowerWorks environment to collect real-time data from aircraft engines under test. This task is per-

formed in a networked environment with a tremendous amount of data coming from multiple engines. You can imagine the critical nature of the data received, and the importance of reliability.

Specifically, in this application a single-board computer is mounted on the engine during the test, and the information is transferred back to the target system via Ethernet. The integrated PowerWorks system shines in this envi-

ronment because the development of applications and the analysis of activities must occur simultaneously. Because a tremendous amount of data must be input in a cyclical manner, it is critical to gather the information quickly and correlate the data on a time basis to determine what activity occurred at any given time.

This ability to integrate disparate tools is also critical in the aircraft simulation applications deployed by Flight Safety International's Simulation Systems Division. In these applications, PowerWorks again benefits the real-time engineer by helping to accurately re-create the aircraft environment. Pilots have a keen sense of time relationship to the visual events, which means all of the activity in the simulator has to be accomplished within a frame time that is imperceptible to the pilot. For example, when the pilot moves the navigation stick, gauges and images change. All of those changes must be conveyed to the pilot every time. If the computers on-board the simulator don't receive the data in time, the pilot won't get an accurate representation of the data.

I hope these examples help illustrate the importance of incorporating an integrated environment for true real-time software development. Without the benefit of integration, software engineers can waste a significant amount of time evaluating standalone tools. ■

*Don Noel is product manager, real-time software products for Concurrent Computer Corporation in Fort Lauderdale, Florida. Mr. Noel has more than 20 years experience in the computer industry. He is responsible for worldwide marketing of Concurrent's PowerWorks Integrated Development Environment, and for developing and supporting third-party alliances with other major software vendors. He earned his Bachelor of Science degree in Computer Science from Florida Atlantic University. He can be reached at don.noel@mail.ccur.com or at (954) 973-5278.*