

aicas news

News for software developers of critical applications

Spring 2008

↓ Editorial

Dear Reader,

Spring time is a time of renewal and aicas is no exception. Spring brings a fresh release of JamaicaVM with many new features for Java on embedded systems as well as new tools such as thread monitoring and a JamaicaVM for CoDeSys. Tools to ease the development of your newest embedded project.

This newsletter describes the new JamaicaVM features and aicas products. Inside, the main article describes how thread monitoring can aid in your system development. The back page also describes our JamaicaVM for CoDeSys and our newly supported ARINC 653 platform.

The technical side is only part of the story. This issue also presents our new distribution and sales partners around the world. So wherever you are, you can enjoy better world wide support for aicas' products.

Whether you are building systems for avionics, industrial automation, or some other market, I hope you will find something useful for your own projects.

Sincerely,

Dr. James J. Hunt, CEO

JamaicaVM Version 3.2

Generics and Enumerations with Power for Embedded Systems

Version 3.2 of JamaicaVM is now available. This release supports the language changes in Java 5. Now embedded programmers can use the full power of generic classes and enumerations in an embedded, realtime environment with the full features of the RTSJ and latencies of a few μ seconds.

Generic and enumeration support improve program type safety. Generics enable the construction of a subclass that shares code with its superclass, but has a distinct type thereby disallowing inappropriate class substitutions. Likewise, enumerations aid the compiler's type checker to find inappropriate uses of constant values. Both features are important for demonstrating program correctness for certification.

Aside from language features, this release also provides options for configuring a JamaicaVM application. The garbage collector can be tuned to provide a better tradeoff between GC response time and throughput resulting in optimized allocation performance. This change gives developers more control over GC and allocation behavior for better overall performance.

The builder now compiles applications by default, yielding good performance out of the box without limiting options for op-

timiza-
tion.

The VM itself has become more capable. It can detect dead-

locks in running code. When one occurs, the VM throws an exception. Also fatal POSIX signals can now be handled. Both features are useful for debugging and error recovery.

Even with these new features, this release is the fastest JamaicaVM ever. The scheduler has been revamped to improve both realtime scheduling on non realtime operating systems and faster thread switching. The interpreter loop has been rewritten for better performance. Important system libraries such as ZIP and SSL have been optimized as well. Even JNI calls are more than twice as fast as the previous release.

Along with the new thread monitor tool described within, this release will provide even better support for embedded development. The resultant applications will also run faster. Evaluation copies are available for download from the aicas' web site: <http://www.aicas.com/download>.



Thread Monitor

↓ News

SuReal R&D Project

aicas is a partner in the German national research project SuReal to improve software development tools for realtime and safety critical systems. Industrial partners include AbsInt for WCETA, Symtavisision for Scheduling Analysis, and ScopeSET for UML tools, as well as DFKI and the Technical Universities of Dresden, Braunschweig, and Munich. The aicas team is working on combining tools from its partners with data flow analysis for accurate timing and schedul-

ing analysis as well as Java code generation from UML. These new tools will be made available at the end of the project.

Microsoft Embedded Partner

aicas has joined the Microsoft Windows Embedded Partner Program. JamaicaVM brings the power and safety of Java to Windows Embedded CE, giving CE developers true write once, run anywhere programming. JamaicaVM is the only JavaVM for Windows CE combining Swing and the RTSJ APIs.

JEOPARD Project started

Europe invests over 3 million Euro (almost \$5 million) in research and development of multicore capable realtime Java technology. Lead by aicas and the Open Group, a consortium of leading European companies and universities will work on tools and an execution environment for deterministic, multiprocessor systems. The consortium will also contribute to relevant standards such as the RTSJ. More information can be found at <http://www.jeopard.org>.

Jamaica Thread Monitor

Understanding Multithreaded Programs Down to the Nanosecond

In the course of improving the JamaicaVM scheduler, aicas has developed a new tool to assist with understanding the behavior and debugging of multithreaded applications. The Jamaica Thread Monitor can peek inside a Java application running on JamaicaVM, enabling the collection and visualization of various types of data about events that occur inside a VM.

The events that can be tracked range from context switches over monitor enter and exits to garbage collector activity. An application developer can also introduce additional events to make code points in the application visible in the Jamaica Thread Monitor. All events can be viewed down into the nanosecond range.



Figure 1: Thread Monitor Control Window

Event collection for the Jamaica Thread Monitor is controlled on the VM side by the two system properties *jamaica.scheduler_events_port* and *jamaica.scheduler_events_port_blocking*. To enable the event collection in the JamaicaVM, a user sets the value of one of these properties to the port number to which the Jamaica Thread Monitor will connect later. If the user chooses the 'blocking' property, the VM will stop after the bootstrapping and before the main method is invoked. This enables a developer to investigate the startup behavior of an application. When event collection is enabled, the requested events are written into a buffer and sent to the Jamaica Thread Monitor by a high priority periodic thread. The amount of buffering and the time periods can be controlled from the monitor tool.

When starting the Jamaica Thread Monitor, a control window opens. This window is used to configure the

connection to the target VM. The user can enter the network address and port of the scheduling event server inside the VM, as well as which of the following types of data to be collected.

- Thread state changes record how the state of a thread changes over time, including which threads cause state changes in other threads.
- Thread priority changes show how the priority changed due to explicit calls to *Thread.setPriority*, as well as adjustments due to priority inheritance on Java monitors.
- Thread names show the Java name of a thread.
- Monitor enter/exit events show whenever a thread enters or exits a monitor successfully, as well as when it blocks due to contention.
- GC activity records when the incremental garbage collector does garbage collection work.
- Start execution shows when a thread actually starts executing code after it was set to be running.
- Reschedule shows the point when a thread changes from running to ready due to a reschedule request.

- All threads that have the ready state within JamaicaVM are also ready to run at the OS level. The OS can choose a thread to run that does not correspond with the running thread within the VM. In such cases, the thread chosen by the OS performs a yield to allow a different thread to run.

- User events contain user defined messages and can be triggered from Java code.

Between clicks on 'Start recording' and 'End recording' the tool receives events from the target VM. After completion of data collection, a new window will open for the visualization. Within this visualization, the user can zoom in or out and contract or expand the time scale. The tool also provides for saving the collected data to a file.

To better understand the Thread Monitor output, it is helpful to have some understanding of the JamaicaVM scheduler. The new JamaicaVM sched-

uler has been overhauled for release 3.2. The new scheduler provides real-time priority enforcement within Java programs on operating systems that do not offer strict priority based scheduling (e.g. Linux for user programs). This scheduler reduces the overhead for JNI calls and helps the operating system to better schedule CPU resources for threads associated with the VM. These improvements let JamaicaVM integrate better with the target OS and increase the throughput of threaded Java applications.

The VM scheduler, as in previous version of JamaicaVM, controls which thread runs within the VM at any given time. This means it effectively protects the VM internal data structures like the heap from concurrent modifications. The VM scheduler does not replace, but rather supports, the operating system scheduler. For example, this allows for a light implementation of Java monitors instead of using heavy system semaphores.

As depicted in Figure 2, threads in JamaicaVM can either be attached or detached. All threads created in the VM are, by default, attached to the VM (i.e. they are controlled by the VM scheduler). Threads that execute system calls must detach themselves from the VM. This allows the VM scheduler to select a different thread to be the running thread within the VM while the first thread blocks, for example on an I/O request. Since it is critical that no thread ever blocks in a system call while it is attached, all JNI code in the Jamaica VM is executed in detached mode.

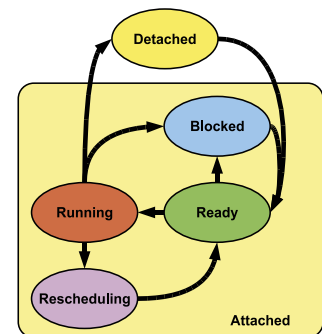


Figure 2: Thread States

For the interpretation of the Thread Monitor data, the distinction between attached and detached mode is impor-

tant. A thread that is detached could still be using the CPU, meaning that the thread that is shown as running within the VM might not actually be executing any code.

Threads attached to the VM may be in one of the states running, rescheduling, ready or blocked. Running means the thread that currently executes within the context of the VM. Rescheduling is a substate of the running thread. The running thread's state is changed to rescheduling when another thread becomes more eligible to execute. This happens when a thread of higher priority becomes ready either by unblocking or attaching to the VM. The running thread will then run to the next synchronization point and yield the CPU to the more eligible thread. Ready threads are attached threads which can execute as soon as no other thread is more eligible to run. Attached threads may block for a number of reasons, the most common of which are calls to *Thread.sleep*, *Object.wait*, and entering of a contended monitor.

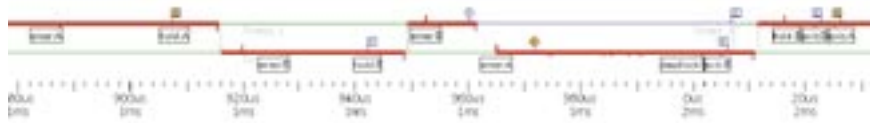


Figure 3: Simple Dead Lock Situation

The following example illustrates another new feature in JamaicaVM 3.2: the Jamaica Thread Monitor can be used to look at how an application may recover from a deadlock using the new deadlock detection feature.

Figure 3 shows a simple two threads with two locks situation. It begins with *Thread_1* as the running thread as shown by the thick red line. This thread enters monitor A. The white boxes *enter:A* and *hold:A* represent user events immediately before and after the 'synchronized' statement. The small flag between the two user events shows that the thread successfully entered a monitor. The color and the number identify the monitor. After it has acquired the lock, *Thread_1* yields to *Thread_2*. *Thread_2* then enters monitor B and yields back to *Thread_1*. When *Thread_1* now tries to enter monitor B, it blocks while entering a contested monitor (shown by the little octagon). Since *Thread_1* is now blocked, *Thread_2* regains the running state and tries to enter monitor A resulting in a deadlock. JamaicaVM

detects this situation and generates a *DeadLockError* which can be caught and handled by the application.

Beneath the line of *Thread_2*, some garbage collector activity is shown. This is caused by the allocation of the error object and the stack trace. The example catches the *DeadLockError* and safely releases monitor B, which now can be acquired by *Thread_1*.

When collecting monitor data is turned on, there will be a lot of data on monitors within the system libraries. The user can hide certain monitors by right clicking on one of the monitor's events and selecting the "hide" option.

In a recent project, the Jamaica Thread Monitor helped to identify a performance bottleneck in a customer application. The application has one communication thread that does a cyclic poll of data items over the network. Each data item is checked to see whether or not its value has changed and if its change listeners were fired. This thread's priority was set to a higher priority than any other thread in the

VM. This should allow the AWT event dispatcher to coalesce all paint events into a single screen update. However, the user interface did not show the desired update behavior. Instead of updating all UI elements in one refresh, there were several refreshes with varying numbers of elements updated each time.

The output in the Jamaica Thread Monitor showed that the AWT event dispatcher thread ran before all changes were communicated over the network due to blocking I/O in the communication thread. On each network communication, the high priority thread blocked for a short amount of time, which gave the event dispatcher thread the possibility to process paint requests already in the queue. Simply queuing up the communication results and processing that queue after all network communication was done solved the problem. Without the Jamaica Thread Monitor, this problem would have been exceedingly difficult to find.

Multithreading brings with it a new class of potential errors to application

programming. These errors may result not only in application failures such as deadlocks, but also simply cause the application to perform more poorly than expected. The detection of such problems is difficult because simply adding probes to test behavior can change thread timings enough to hide the causes of poor performance. A tool, such as the Jamaica Thread Monitor, enables examining an application at full speed, is invaluable for understanding the behavior of multithreaded programs.

↓ New Distributors

In 2007, aicas continued to strengthen its position in the market and develop new markets. Partnerships, which enable aicas to offer customers around the world high quality on-site service, are part of this strategy. To this end, aicas has signed new partnership agreements with SYSGO, Realtimewave, and Masterpiece.

SYSGO AG is a global supplier of highly reliable software for critical systems. They have announced the availability of aicas JamaicaVM for PikeOS. The partnership focuses on aerospace and defense, industrial automation, and medical systems, and will strengthen both companies positions on these markets.

Realtimewave Co., Ltd., is a major services provider and systems integrator in Korea. Realtimewave's business is focused on high-tech embedded applications such as avionics, robotics, simulation, and control systems. As a successful player in the flourishing Korean market, Realtimewave is well positioned to support JamaicaVM customers there while broadening its own product offerings.

Masterpiece Technology Co., Ltd. is a pioneering systems integrator and embedded solutions provider in China. Masterpiece adds a well regarded system to their line card and aicas' customers will gain a qualified local contact, giving aicas an effective distribution channel in the fast-growing Chinese market.

These partnerships improve aicas worldwide reach for sales and service of its products.

ARINC 653

An Aviation OS Standard for Running Separately Certified Application on a Single System

The ARINC Specification 653 defines an operating system standard for safety and mission critical systems, particularly in the avionics industry.

ARINC provides an Application EXecutive (APEX) with space and time partitioning, ensuring the safe use of multiple, concurrent applications. Each application runs in its own partition, with multitasking within its partition. ARINC 653 partitions have their own memory space, and an application cannot access the memory space of another application. Furthermore, the APEX assigns

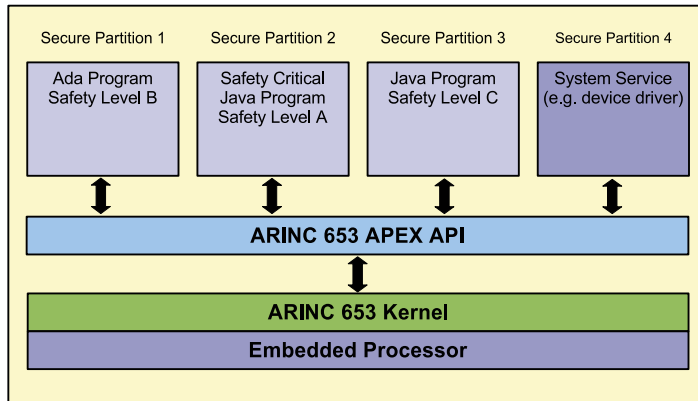
fixed time slices for every partition. The cycle-based scheduling guarantees a deterministic execution flow, where each application receives a guaranteed amount of CPU time.

ARINC also defines unified ports for communication. Partitions can communicate with each other via these

ports. In addition, ports provide data exchange with the external environment.

Using an ARINC 653 compliant OS eases DO-178B certification for running multiple applications on the same hardware. DO-178B is a standard for software certification in airborne systems. Depending on the severity of the problem a failure in an application can cause, the application is categorized in a safety level from E (no effect) to A (catastrophic). The isolation of applications via partitioning in ARINC 653 allows DO-178B certified applications of different levels to run on one machine. The ARINC 653 OS itself must be certifiable up to level A.

aicas currently supports the ARINC 653 conformant operating system PikeOs (SYSGO) and VxWorks ARINC 653 (WindRiver), along with LynxOS 178 (LynuxWorks) and Integrity 178B (Green Hills) upon request.



ARINC 653 Example with Four Partitions

JamaicaVM for CoDeSys WebVisu

IEC 61131-3 Visualizations on Embedded Systems

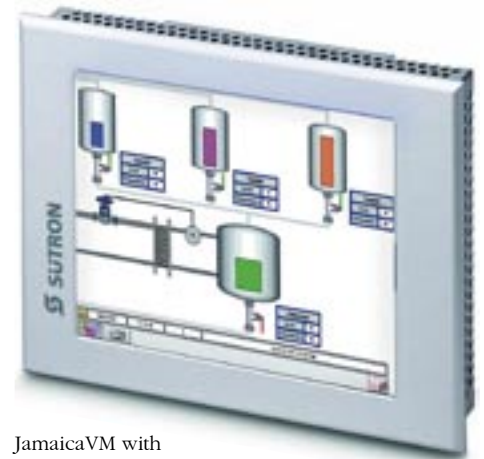
In cooperation with 3S—Smart Software Solutions, aicas now offers JamaicaVM for CoDeSys WebVisu. CoDeSys development software provides quick and effortless creation of visualizations conforming to the IEC 61131-3 standard. This cooperation enables these visualizations to run on a wide range of embedded systems. aicas offers a JamaicaVM that can be installed on many different systems quickly and integrated seamlessly into an existing CoDeSys development environment. Possible targets are embedded systems with dis-

plays and controllers with integrated displays.

The following operating systems are supported: Linux, OS-9, VxWorks, and WindowsCE. Further operating systems can be supported on request.

Displaying the desired visualization on an embedded system is simple and straightforward. First, the customer creates a visualization with CoDeSys. Then, this visualization, together with the projected controller application, is loaded onto a (soft-) PLC. When started on a web panel, JamaicaVM for CodeSys WebVisu retrieves the projected data from the controller and displays the visualization. Changes in the project simply need to be loaded onto the PLC. Using JamaicaVM technologies provides excellent performance, even on small systems.

A new licensing scheme offers customers the advantage that they need no longer to acquire a full project license including the JamaicaVM tool chain, just a precompiled VM for their target hardware. Thus, aicas can offer JamaicaVM for CoDeSys WebVisu for a very low per unit price. This solution, adopted from 3S—Smart Software Solutions, makes Ja-



JamaicaVM with CoDeSys Webvisualization

JamaicaVM for CoDeSys WebVisu attractive for manufacturers and distributors of panel PCs for both large and small quantities.

↓ **Events**

We would like to welcome you at following events.

ERTS 2008,
Toulouse, 29 Jan – 1 Feb 2008

Embedded World 2008,
Nuremberg, 26–28 February 2008

Avionics Event,
Amsterdam, 5–6 March 2008

Embedded Systems Conference
San Jose, 14–18 April 2008

↓ **Contact**

aicas GmbH
Haid-und-Neu-Straße 18
D-76131 Karlsruhe, Germany
+49 721 663 968-0

aicas incorporated
69 West Rock Avenue
New Haven, CT 06515, USA
+1 203-676-9807

email info@aicas.com
web www.aicas.com