MULTI® 8

Integrated Development Environment

Green Hills® SOFTWARE

The suite of tools in MULTI were created by Green Hills Software developers to form a complete environment specifically made to fit the needs of embedded software developers:

- Safety-certified C/C++ compilers and run-time libraries
- Multicore debugger
- Advanced system viewer
- Back-in-time debugging
- OS awareness
- Static code analyzer
- Run-time error detection
- Memory leak detection
- Performance profiler
- Editor
- Instruction set simulator
- Code coverage
- Project manager
- Version control integration
- Flash programmer
- Python interface
- MATLAB/Simulink integration

Tools that make embedded software developers more productive

Used by thousands of developers for over three decades, the MULTI IDE is the industry's unrivaled integrated development environment used to create, debug, and optimize code for embedded processors. With MULTI, developers can easily see and fix difficult bugs, pinpoint performance bottlenecks, and prevent future problems. Key benefits are: lower cost to deliver on-time software, avoidance of costly recalls due to poor software quality, and lower processor and memory costs because more efficient code uses less memory and smaller CPUs.

- Find and fix bugs faster
- Replay programs to pinpoint problems
- Uncover bugs before run-time
- Spot and fix performance bottlenecks
- Analyze and optimize complex systems
- Increase code security and reliability
- Maximize performance
- Reduce processor cost
- Run Simulink® and other models
- Meet functional safety requirements

MULTI supports all popular processor architectures, runs on Linux or Windows desktops, debugs code from multiple compilers, connects to targets over various links, and supports multiple operating systems running on silicon, simulators, or emulators.

Find and fix bugs faster

In complex real-time systems, unexpected time spent on hunting for sporadic and deeply buried bugs or finding performance bottlenecks can take weeks or even months, potentially risking an on-time product launch.

With MULTI, problems that once took days, weeks, or months are found and fixed in hours or even minutes. In complex systems, the most time-consuming bugs are sporadic, hidden and not evident with code inspection. MULTI enables developers to quickly pinpoint problems caused by memory corruption, unintended dependencies, unexpected interrupts, multicore complexity, inter-task corruptions, opaqueness from missing source code or virtualization, missed real-time requirements, and asynchronous hardware events … just to name a few.

Here are a few examples of reports from customers that highlight the effectiveness of the MULTI Debugger and its ability to solve challenging problems that less capable tools cannot achieve:

- Using MULTI, a customer found and fixed a bug in two hours after spending two person-months searching for the bug
- A customer reported that after using MULTI for only a couple minutes, they found a three-month-old bug that was causing infrequent hangs on their target due to unexpected interrupts in start-up code

- A customer had been unable to find the source of a sporadic audio pop in their automotive audio system for six months. The deadline for production was looming and executives had experienced this bug. Even with missing source code, the Green Hills system visualization tool, History, easily displayed an unexpected execution flow that—after a few hours of analysis—pointed to the root cause within a graphics application. Once the engineers knew the cause, this issue was fixed in minutes

The MULTI Debugger includes three powerful capabilities:

- **History® viewer** shows how the program got to its current state & what the system was doing
- **TimeMachine®** uses trace data so you can run, step, & analyze back-in-time in the Debugger
- **Debug Snapshot** saves a debugging session that you can easily share with coworkers around the world

Make sense of complex systems

- *How did the program get here?*
- *Where is time being spent?*
- *Is the program doing anything unexpected?*

These three questions are at the foundation of debugging code and making your system run faster. The History viewer answers these questions.

History viewer displays the last few seconds, minutes or days of program execution across complex heterogenous multicore systems with a natural and intuitive GUI. For the first time you now have a clear, complete view into a murky hardware and software system. You can zoom deeply into processor behavior at the micro-second level or zoom out to see system behavior spanning minutes and days. This new kind of visibility empowers you to find difficult bugs in seconds, see hidden bottlenecks and dependencies, and analyze execution times.

History viewer capabilities

- Halt the MULTI Debugger and view the system's history leading up to any moment in the History viewer
- Gain an instant understanding of the flow of your program with task call stacks. See how the program got to this point and where the code diverted from an expected path
- Clearly see the causes of bugs and bottlenecks across multiple cores, such as unexpected execution flow, incorrect task priorities, excessive interrupt level processing, too many calls to the operating system, and unusual interrupt latencies
- Zoom in to inspect and measure micro-second OS events and dependencies and zoom out to see system-level execution patterns
- Click anywhere in the various History viewer panes and go to the corresponding line of source code
- Gain a new understanding of your complex program including kernel execution, virtual address spaces, multicore CPU utilization, inter-component communications, exact timing, user-defined events, and more
- Ingest terabytes of trace data and/or instrumented code running on simulators and boards
- Display third-party logging data as well, including Linux ftrace and COVESA DLT (Diagnostic Log & Trace)
- Inject an almost unlimited number of user-defined, low-overhead variables and view in History viewer
- Find and analyze complex conditions with advanced search options
- Easily save a History viewer snapshot of the system, bookmarked and annotated at important hotspots, and send to fellow developers, testers or technical support so they can instantly see the entire system and continue debugging
- Measure times of events with very high resolution
- Handles cases when source code is missing or software is running in a virtual machine
- See deeply into Linux and Android
- Minimal processor overhead and no additional hardware needed

History provides unprecedented visibility into your system by displaying the last few seconds, minutes or days of program execution across complex heterogenous multicore systems.

Back-in-time debugging

By automatically capturing actual program execution data **TimeMachine** enables the **Debugger** to run, step and debug code backward to any problem area shown in **History**. It also powers other tools such as the **Profiler**.

TimeMachine enables the MULTI Debugger to run, step and analyze your program forward and backward in time to find the root cause of problems.

Prevent new problems

DoubleCheck™, Green Hills Software's integrated static source code analyzer, saves you time and money by identifying programming errors *before* running your program. Because DoubleCheck is part of the Green Hills optimizing compilers, there's no setup hassle, and minimal overhead to use it. Simply set an option and DoubleCheck automatically reviews source code each time it's compiled. Since it's part of the compilers, DoubleCheck runs much faster than traditional static analysis tools and that means that every developer can have it turned on all the time.

Moreover, DoubleCheck is more reliable than traditional code reviews and can find bugs that may never show up during regular system testing.

Run-time Error Checking complements DoubleCheck by finding bugs that cannot be identified by static analysis alone. Bugs due to memory corruption are especially heinous and difficult to find, and can remain silent for a long time. With run-time error checking, you're alerted to the cause of the problem when it happens. Run-time error checking identifies problems such as invalid memory accesses through pointers, out-of-range value assignments, unhandled cases in switch statements, and division by zero.

Clean code is less likely to contain errors and is easier to test, understand, and modify. All of these factors contribute to fewer bugs and greater reliability. Green Hills optimizing compilers enable enforcement of clean coding conventions defined by industry standards including the **MISRA 2012** and **2004** coding standards, which includes more than one hundred rules for safe programming. You can also choose to enforce a customized subset of these rules to meet your specific requirements.

DoubleCheck lowers development costs by identifying problems early in the "Develop" stage which saves time and resources.

GHS Standard Mode is a collection of compiler warnings and errors that enforces a stricter coding standard than regular C and C++. It's based on the experience Green Hills has gained through the years to help increase the reliability of code. We use GHS Standard Mode internally and find it provides good protection against common mistakes and bad coding stye, without causing too many false positives.

Advanced debugging

The MULTI **Debugger** is a powerful tool for examining, monitoring and changing source code, while running on complex heterogenous multicore target processors and simulators. When TimeMachine is used, it can even run backward in time. The Debugger is seamlessly integrated with other tools within MULTI and can be invoked by clicking inside various MULTI tools such as the History viewer.

- Debug applications, kernels and device driver code on various operating systems or bare board code
- Debug virtualized operating system kernels and their applications
- One debugger for run-mode and freeze-mode debugging and run-control of synchronous (SMP) and asynchronous multicore (AMP)
- Traverse the call stack, view cache, browse objects, and view memory and registers
- Use the **OSA Explorer** to see deeply into the status of kernel objects, tasks, and resources
- Debug task interactions at the source level
- Find specific data points by using powerful execution and data breakpoints
- Debug code running in virtual address spaces

With the MULTI Debugger, debugging multitask, multicore systems is as straightforward as with single core systems.

When you debug a multitasking application on an OS like INTEGRITY® or Linux, MULTI can interact with the multiple tasks in run-mode, in freeze-mode, or in both modes simultaneously. In run-mode, the operating system kernel continues to run as you halt and examine individual tasks. In freeze-mode, the entire target system stops when you examine tasks.

Key among these features is synchronous run control, which halts all cores as a unit when any core encounters a debugging condition. For instance, when a core hits a breakpoint the target list clearly shows:

- which core(s) have hit the breakpoint
- which thread was executing on the core at the time
- what the other cores in the system were doing

Ordinary tools often make downloading code to a multicore target a complex and tedious process. MULTI can automatically handle downloading the INTEGRITY RTOS to these targets. For other types of operating systems and applications a simple configuration file can be used to automate the download and setup process.

MULTI synchronous run control halts all cores in a multicore system simultaneously. This enables you to debug one core without worrying that operations running on another core will affect shared memory.

Debug embedded Linux

MULTI for Linux brings advanced debugging to engineers developing embedded Linux software. It dramatically improves their productivity and helps them bring a more reliable, higher-performing product to market faster.

Traditional debugging tools for Linux offer only limited visibility and usefulness. GDB—and front-ends such as Eclipse—are cumbersome to set up, difficult to use, slow in responsiveness, and not always trustworthy when debugging embedded, optimized code. As a result, developers often resort to the hit-and-miss effectiveness of printf and printk commands. Unfortunately—due partly to the significant performance impact—these techniques introduce their own problems and provide only a limited window into complex system execution.

With MULTI for Linux and Green Hills consulting services, you can now visualize and debug all aspects of an embedded Linux system—including the Linux kernel, kernel threads, interrupt services routines, user mode threads and processes—faster and more easily than ever before.

Linux debugging capabilities

- Comprehensive visibility into the Linux kernel, including shared memory regions, semaphores, message queues, and kernel modules
- Support for SMP multicore debugging
- A single debugger for kernel code, drivers, user-space applications and virtualized code
- Debug embedded Linux or virtualized Linux (on INTEGRITY Multivisor® virtualization) on the target and native Linux on the desktop
- See virtualized Linux and INTEGRITY events alongside each other in History viewer
- Debug loadable kernel modules and device drivers
- Easily debug Linux's process startup code
- Set any thread-specific breakpoints in kernel code
- Debug shared objects in virtual address spaces
- Handle breakpoints with minimum performance cost
- Debug multi-threaded interactions across processes

An IDE that helps you manage your project

The MULTI IDE's simplified project management frees you to spend more of your valuable time developing your product. With its **Builder**, there's no complicated build infrastructure to maintain, and build configurations can be easily changed. The Builder automatically analyzes the dependencies of your entire project and compiles and links as many files in parallel as possible, taking full advantage of modern multicore systems.

Along with the Builder, the seamlessly-integrated **Project Manager**, **Editor**, **Flash Programmer**, and **Instruction Set Simulator** help you spend less time on build management and more on your code.

The MULTI Project Manager simplifies how you build and update your application. Tabbed block diagram and memory layout views provide an at-a-glance understanding of application structure. Here, green signifies read/write memory and red signifies read-only memory.

Maximize performance

For applications that require the highest possible performance, the MULTI IDE offers both the industry's best embedded C/C++ compilers and the best performance analysis tools. To reach the ultimate performance, you can also enlist a Green Hills Software performance-tuning expert for a solution customized to your device.

Generate faster, smaller code

Compilers are the essential ingredient to leverage processor performance and the Green Hills C/C++ optimizing compilers are the best in the industry. On the widely-accepted EEMBC benchmarks for embedded processors the Green Hills Compilers consistently outperform competing compilers to generate the fastest and smallest code for 32- and 64-bit processors.

Processor manufacturers publish certified EEMBC benchmarks to showcase their processor's performance on real-world applications. These manufacturers have chosen to publish the highest scores generated with Green Hills Compilers more than twice as many times as any other compiler.

The Green Hills Compilers use hundreds of the most advanced compiler optimizations to significantly increase program execution speed and decrease size.

While the Green Hills Compilers provide an excellent baseline of default optimizations and settings, we understand that your appli- cation is unique. You can fine-tune Compiler output by enabling different combinations of optimizations and by configuring optimiza-tions on a per-project, per-file, per-function, or even per-source-line basis. Our consulting services team will help you use the highly configurable Green Hills Compilers to give you the best performance results, on any project. On many programs, our consulting services team has helped customers use the Green Hills Compilers to optimize programs to run 20% faster than with competing compilers.

Pinpoint performance bottlenecks

The second key ingredient to maximizing your program's performance is using the best performance analysis tools. With the help of **History** and the **Profiler** tools, you can easily identify and eliminate performance bottlenecks in your code. The result is a faster, more efficient application.

When it comes to maximizing performance, time is the most important dimension to consider. It's also the dimension that most profiling tools throw away. The innovative History visualization tool provides a high-level, time-based view of every function executed in your program, making it easy to identify patterns in program flow and easily spot where time is being spent.

The Profiler's interface streamlines analyses by showing how often and for how long different tasks, functions and instructions were run so you can pinpoint where to focus optimization and test efforts.

Profiler capabilities

The powerful Profiler provides additional insight into program behavior, using data gathered from trace or instrumented code. It shows how much processor time is consumed by each task, function, source line, and assembly language instruction, making it easy to:

- find performance problems
- find coverage gaps and track test coverage of specific code
- share profile data with other users
- collect and merge profile data from several tests or profile recordings

History's Graph pane, **Activity Graph** pane and **Thumbnail** pane provide unprecedented views into where the program is spending its time. When pinpointing a performance problem during your optimization work, History can save you days or weeks.

- Displays the length and frequency of RTOS events making it obvious what operations take the most time and where optimization efforts should be focused
- See incorrect task priorities, excessive interrupt level processing, too many calls to the operating system, and unexpected task context switches
- Intuitive display of core execution and relationships between tasks

With three decades of experience using and creating tools to solve customer problems, our skilled experts can help you to attain the best possible performance and reliability.

Benefit from our expert knowledge

To reach the highest possible performance, bring in the performance-tuning experts from Green Hills Software. Our custom services will help you get the most out of our development tools and compilers, assess your design, and even create a custom compiler optimization tailored to your specific goals.

Lower your production cost

In high-production environments, saving even pennies per unit can result in big overall savings. The MULTI IDE can significantly reduce memory and processor costs to boost your bottom line.

Reduce your processor costs

You don't always need a faster processor to meet increasing performance requirements. Our superior C/C++ compiler optimizations are tuned to get the most out of the processor you use. Additionally, our performance analysis tools such as the **History** viewer and **Profiler** give you a bird's-eye view of where your program is spending its time, helping you to focus your optimization efforts. How much money would you save if you could use a lower-cost processor to meet the same real-time requirements?

Improving your code's performance pays off because it gives your design more CPU headroom to add more features. It can even enable you to choose a lower cost processor that uses less power and cooling.

Reduce your memory costs

Green Hills compiler optimizations reduce your memory costs by reducing the size of your executable. Most programs see at least 10% improvement relative to the competing compilers. Additionally, **DoubleCheck** and the MULTI Debugger's **Memory Allocations** tool help you to make more efficient use of your memory, futher reducing the amount of memory needed in you embedded system.

The MULTI Debugger's Memory Allocations tool lets you easily identify memory leaks at run-time.

Functional safety confidence

The MULTI C/C++ IDE and toolchain are certified to meet the highest safety levels:

- ISO 26262:2018 (Automotive); IEC 61508:2010 (Industrial); EN 50128:2011 and EN 50657 (Railway) functional safety standards
- Certificates received from both TÜV NORD and exida
- Satisfies the highest achievable levels SIL 4 (Safety Integrity Level) and ASIL D (Automotive Safety Integrity Level) and meets C/C++ run-time libraries certification requirements
- Supported on a broad set of target processors

The benefits are manifold:

- Lower cost and shorter time to certification and production
- Reduced certification maintenance after product release
- Higher quality and confidence targeting code to highest ASIL/SIL levels

Green Hills training and consulting

Green Hills expert training and consulting services provide the most direct route to understanding the full potential of the MULTI IDE, so you can reach maximum productivity in the shortest time.

Managed implementation program

This program is designed to make development teams more efficient—and productive—with the Green Hills Software product environment. Services offered are completely customizable in content and duration and include:

- expert training
- environment setup
- application design mentorship

Starting with the design phase, our experts will help you focus on the product features that best match your needs. Along with customized training, the managed implementation program offers proactive, ongoing mentorship to ensure you know how to best use our environment as well as have regular access to a Green Hills Software expert.

Customized training

Our training curriculum spans a wide range of material, from INTEGRITY programming to advanced History-guided debugging. Courses are taught by experts with hands-on experience developing with Green Hills tools. Classes can be taught at your facility and on your schedule, providing maximum value with minimal interruption.

Open enrollment training

Teams with smaller training budgets can attend our popular open-enrollment courses at scheduled locations around the world. These classes are also perfect for new hires who have just joined a team that has already completed a Green Hills Software training class.

MULTI ecosystem

Contact Green Hills Software for the latest support for new processors and operating systems.

Target processors

- ARM (AArch32 & AArch64)
- Power Architecture
- TriCore
- RH850/ V850
- Intel (x86 & x64)
- RISC-V
- MIPS

Development hosts

- Windows
- Linux

Operating systems

- INTEGRITY
- Linux
- *µ-velOSity™*
- AUTOSAR
- VxWorks
- FreeRTOS

JTAG & Trace Probes

- Green Hills Probe
- Selected third-party probes
- Green Hills SuperTrace Probe

Languages

- C
- C++
- EC++
- Ada (with AdaMULTI)
- GNU C/C++ extensions

Third-party integrations

Compilers

- GCC & those that generate DWARF or Stabs debugging info

Design & test tools

- Rhapsody
- Simulink/MATLAB®
- SCADE
- LDRA
- VectorCAST
- Python

Corporate Headquarters

30 West Sola Street ▲ Santa Barbara, CA 93101
ph: 805 965 6044 ▲ fax: 805 965 6343 ▲ email: info@ghs.com ▲ www.ghs.com

European Headquarters

Fleming Business Centre ▲ Leigh Road ▲ Eastleigh ▲ Hampshire S050 9PD ▲ United Kingdom
ph: +44 (0)2380 649660 ▲ fax: +44 (0)2380 649661 ▲ email: info-emea@ghs.com

Safety & Security Critical Products

34125 US Hwy 19 North ▲ Suite 100 ▲ Palm Harbor, FL 34684
ph: 727 781 4909 ▲ fax: 727 781 3915 ▲ email: info-sscp@ghs.com

Green Hills, the Green Hills logo, INTEGRITY, MULTI, TimeMachine, DoubleCheck, Multivisor, µ-velOSity, SuperTrace Probe, and History are trademarks or registered trademarks of Green Hills Software in the US and/or internationally. All other trademarks (registered or otherwise) are the property of their respective companies. © 2022 Green Hills Software. v1022