Wind River Workbench

Wind River Workbench is the device software industry’s premier open, standards-based development suite. Through its powerful combination of capabilities, integration, and availability, Workbench enables organizations to standardize on a common environment for device software development, helping developers, project teams, and enterprises improve their effectiveness.

Workbench offers:
- Best-in-class capability at each phase of the development process, including hardware bring-up, firmware development, application software development, advanced visualization, system diagnostics, and test
- Broad availability to support increased standardization across projects
  - Multiple-target operating system support, including support for VxWorks 5.5, VxWorks 6.x, and Linux
  - Target processor support for ARM, IA/Pentium, MIPS, PowerPC, SH, and XScale processors
  - Plug-in architecture enables additional target operating system, target processor, and target connection support to be added
- An extensible framework, based on Eclipse, to seamlessly integrate third-party and in-house plug-ins for total customization and scalability

Workbench addresses the challenges individual developers and project teams face by increasing productivity, enabling collaboration between hardware and software developers, and meeting diverse development needs across an enterprise. The development suite is backed by Wind River’s 20+ years of device software industry experience, a world-class support organization, and a specialized professional services team.

NEW IN WIND RIVER WORKBENCH 2.5
Workbench 2.5 includes significant new capabilities in support of increased development team productivity:
- General Workbench enhancements
  - Updated to Eclipse 3.1.2
  - Flexible managed build support
  - Team-sharing of static analysis data
  - Support for C/C++ code templates in the editor
  - vi-compatible mode in the editor
- VxWorks platform integration
  - Expanded VxWorks 6.x processor support
  - Edit, compile, and debug support for VxWorks 5.5 for PowerPC targets on Windows hosts
  - Improved kernel configuration editor
- Linux platform integration
  - Expanded processor support
  - Improved project setup
  - Enhanced kernel configuration tools
  - User space configuration tool for RPM configuration
  - Core file debugging
  - User mode debug over USB
  - Core file analysis
  - Tighter integration of ScopeTools with Linux kernel and file system configuration
- On-chip debugging
  - Wind River Trace
  - Scripting
  - Auto-attach to core; workflow enhancement
  - Linux kernel debug on ARM 9 and 11 via on-chip debugging
  - Expanded processor support, including the following new architectures, new processor families, and new processors:
    - Atmel; AMCC PPC44x additions; ARM 11; Broadcom BCM 11xx, 74xx, and 73xx additions; Freescale i.MX36, MPC83xx, MPC85xx, and MCF5xxx additions; Intel IOP 3xx additions; Marvell MV88F5181; Philips PNX additions
- New Workbench products
  - Workbench Unit Tester support for VxWorks 5.5
  - Workbench Diagnostics for Linux and VxWorks 5.5

INCLUDED IN WIND RIVER WORKBENCH

Application Components
Wind River Workbench 2.5 offers the following capabilities and components:
- Eclipse framework
- Project System
- Build System
- Editor
- Source Code Analyzer
- Wind River Compilers
  - Wind River Compiler (for VxWorks 6.x and other target operating systems)
  - Wind River GNU Compiler (for VxWorks and Wind River Linux platforms)
- Wind River Debugger
Wind River Workbench components

- Wind River shell environments
- ScopeTools
  - StethoScope for data analysis
  - MemScope for memory analysis
  - ProfileScope for performance analysis
  - CoverageScope for code coverage analysis
  - TraceScope for code execution flow analysis
- Wind River System Viewer for system event visualization
- Workbench Unit Tester for unit and integration test
- Workbench Diagnostics for dynamic instrumentation and root-cause analysis
- Wind River on-chip debuggers
  - Wind River ICE
  - Wind River Probe
  - Wind River Trace

Eclipse Framework

Because of its openness, capability, and strong community support, Eclipse was chosen as the framework for the Wind River Workbench development suite. The Eclipse 3.1.2 framework supplies the necessary infrastructure to graphically and functionally integrate the components of Workbench. Open, extensible, and backed by a strong community of commercial and open-source developers, the Eclipse framework provides developers using Workbench with a wide range of additional integrated functionalities.

Eclipse integrated capabilities are provided by commercial development tool providers (such as IBM, Hewlett-Packard, and Borland) and an active developer community. As a result, developers have access to a wide range of value-added plug-ins from third-party and in-house sources that can be used to extend the capabilities of Wind River Workbench. Examples include Eclipse-integrated configuration management (CM) systems and editors, which offer simple plug-in integration with Wind River Workbench through standard Eclipse interfaces.

More information on Eclipse and available third-party plug-ins is available from the Community Projects and Plug-Ins section of the Eclipse website, among other resources:

- http://www.eclipse.org
- http://www.eclipse-plugins.info/eclipse/index.jsp
- http://www.eclipseplugincentral.com

In many cases, users will need to validate the utility and compatibility of these plug-ins with Wind River Workbench.

Project System

The Workbench Project System allows developers to organize and manage the primary components in a device software development project, including source files and target systems. By design, Workbench enables users to manage multiple projects simultaneously.

Features of the Project System include:

- Drag-and-drop manipulation of projects to increase speed and enhance simplicity
- Wizards and templates to accelerate setup of common project types, including downloadable kernel modules (DKMs) and real-time processes (RTPs) for VxWorks 6.x
- The ability to import projects from existing Tornado 2.2 and SNiFF+ 4.x application projects to DKM or RTP projects, saving the time that would be needed to manually re-create them; this includes the ability to migrate toolchains (Wind River GNU to Wind River Compiler or vice versa) at the same time
- The ability to share projects among team members, enabling parallel development; each team member can work in one or more workspaces with local copies of source files, builds, and debugging
- The ability to work with multiple projects in a workspace simultaneously, reducing redundant setup and configuration and eliminating wasted time between projects
- The ability to provide project types to create native applications for executables, libraries, DLLs, and shared libraries
- Automatic version control to keep track of changes, reduce errors, and allow restoration of previous project environments
- Use of structured projects that reflect the build hierarchy by linking subprojects to multiple parents, so you can easily see the relationship between the project and the way it is built
- The ability to provide and associate different build rules with different components of a project
- Easy identification, visualization, and management of changes to the hierarchy of files, folders, and projects
- Ability to update workspace and project data via the command line
Build System
The Workbench Build System specifies the tools, options, and parameters to use when building device software projects, enabling you to set build parameters easily from the project level down to the individual file level. The Build System allows for use of simple global build-setting, fine-grained control at the level of an individual file, and everything in between.

Features of the Build System include:
- Managed application build process, based on make, is simplified through a series of wizards and self-guided dialogs—through the build system, you can specify how you want to generate Makefiles specific to a project
- Automatic generation of dependencies (compiler-dependent)
- The capability to generate include search paths through source code analysis
- GUI configuration of build tools and build specifications that can be designated at the project, folder, or file level, with property sheets for advanced customization
- Compiler configuration GUI that allows custom compiler target and optimization settings to be specified at the project, folder, or file level; GUI supports both Wind River Compiler and Wind River GNU Compiler
- Easy switching between build specifications
- Multiple targets supported per project, including specific build settings, selective objects list, and custom link ordering for each target
- Command line build capability for scripted and nightly builds
- Default settings, which shorten the time to an initial build; the most frequently used settings are set as defaults
- Shared Library build support
- Custom link order, file-level build, settings, and multiple build targets give you expert-level control and customizability
- Wizards to create new projects, such as shared libraries, VxWorks file system, downloadable modules, bootable ROMs, and RTPs
- Option for user-defined builds using existing Makefiles
- Support for remote builds

Editor
The Workbench Editor provides state-of-the-art editing capabilities, including a number of performance-enhancing features—such as code completion, parameter hinting, and syntax highlighting for source files—that serve to speed the development process and make the edit-compile-debug cycle less frustrating and prone to error.

Since the Workbench Editor is fully integrated with the Project System, Build System, Source Code Analyzer, and Debugger, you can use it to perform all source-code-oriented tasks.

Features of the Workbench Editor include:
- Code completion enabled by source code analysis:
  - Provides reasonable proposals for code symbols (macros, functions, methods, variables) that could be used at the given location
  - Takes the current scope into account
  - Reduces typos and repetitive typing
  - Includes parameter hinting that provides visual help, identifying the parameters expected in the argument list, including VxWorks 6.x and external API calls
- Code folding
- Ability to perform active vs. inactive code analysis in the editor by highlighting code disabled by macros
- Syntax highlighting, which identifies language keywords and constructs, including comments, strings, and constants
- Extremely configurable, so you can change key bindings, font type, font size, color scheme, and syntax coloring
- Other editor views include:
  - Line numbers
  - Non-printable characters
  - Debugger breakpoints and program counter
- Symbol highlighting with integrated navigation capability:
  - Symbols can be filtered by file, type, and name
  - Symbols can be searched
  - Symbols are an entry point to the editor and provide a way to navigate within the editor
  - Navigation is allowed between different types of symbols (e.g., definition and declaration, base and derived objects, overloaded and overridden symbols, defined and instantiated objects)
Editor emulations
- Complete vi emulation mode, including keyboard shortcuts and command line mode
- Emacs keyboard shortcuts
- Support for third-party editors through use of Eclipse plug-ins
- Support for large source files
- Support for multiple-byte characters

Source Code Analyzer
To improve productivity and reduce costs, organizations strive to reuse as much legacy code as possible, incorporate significant third-party code into their products, integrate software components from multiple developers into a single application or system, and increase developer understanding of code components to reduce integration and debug time.

The ability to quickly and completely understand code written by someone else—or to assess the impact of a change under consideration—is vital to development productivity. Wind River Workbench source-code analysis capabilities enable this function. Integration of these capabilities into the editing and debugging functions of Workbench speed both code creation and debugging.

Features of the Source Code Analyzer include:
- Proven and mature source-code analysis technology from SNiFF+ (no compilation required)
- Fault-tolerant source-code parsing that accurately analyzes incomplete source code and source code that cannot be compiled
- Precise evaluation of macros and complex C/C++ constructs using the context of the current build settings
- Source-code completion and parameter hinting for C/C++, even for source code not loaded into the development environment, such as OS header files
- Symbol-based source navigation based on static analysis of code
- Symbol highlighting in the editor based on static analysis of code
- High-quality cross-referencing, including:
  - Support for any type of code symbol, including functions, variables, method, and macros
  - Refers-to and referred-by information
  - Complete and accurate cross-reference data, including external declarations
  - Navigation between references in the source code
  - Ability to look up symbol definitions from within any part of the development environment
  - Displaying the file to which a reference belongs
  - Identity of symbol reference type for any variable (read, write, read and write)
  - Tool tips for symbols to show symbol declarations
  - Virtual methods (object-oriented languages) included in the list of references for a variable
  - C/C++ parser support for pre-processing source code and handling various dialects
  - Ability to perform active vs. inactive source-code analysis in editor
  - Background parsing without blocking the development environment
  - Visualization of source and header file include diagrams
  - Dependency trees that aid in analyzing the impact of code changes
  - Class hierarchy browsing:
    - Inheritance relationships (derived from)
    - Class contents (data members, member functions)
    - Ability to filter the hierarchy tree
  - Symbol definition lookup and quick source-code navigation to reduce comprehension time of less familiar code
  - Filters and collapsible or expandable views to avoid information overload
  - Source browsing features with local C/C++ symbols (e.g., local variables)
  - Ability to parse the entire source of the Linux and VxWorks kernels, providing increased visibility into system operation
  - Support for C, C++, Assembler, and Ada
  - Source-code analysis features that are directly accessible within the Workbench Debugger, providing additional perspective during debug
  - High-speed text search-and-replace tool (Retriever)
  - Improved accuracy of parsing during source-code analysis using build settings
  - Refactoring support for symbol rename
Wind River Compilers
Wind River provides two compilers for use in Wind River Workbench when developing with VxWorks 6.x: Wind River Compiler and Wind River GNU Compiler. Both compilers are included and supported as part of Wind River Workbench for VxWorks 6.x.

Wind River Compiler
Wind River Compiler is the default C/C++ compiler configured for building the VxWorks 6.x kernel, libraries, board support packages, and applications in Wind River Workbench. It also supports stand-alone (no RTOS) development. This compiler’s optimization capabilities are based on and extend the industry-hardened Diab compiler technology, and it produces robust, tight, and fast-executing code.

Features of Wind River Compiler include:
- Superior optimization technology to generate fast, compact, high-quality code
- 100 percent compatibility with the latest ANSI C++ specs (ISO/IEC 14882:1998(E) C++ standard) and the ANSI C spec (X3.159-1989)
- Standards conformance (ANSI and EABI) for maximum tool interoperability
- Complete control of code and data memory allocation
- Position independent code (PIC) and position independent data (PID) support
- Proven performance with VxWorks

Wind River Compiler also supports run-time error-checking that detects and corrects hard-to-find problems, such as memory leaks and out-of-bounds pointers, to aid in producing higher-quality code.

For more detailed information, please see the Wind River Compiler Product Note.

Wind River GNU Compiler
Wind River GNU Compiler is based on the Free Software Foundation (FSF) distribution of gcc and g++. It provides support for VxWorks 6.x and Linux. Wind River has modified an off-the-net version of gcc specifically for use with VxWorks 6.x. The primary areas of modification deal with support for RTPs and shared libraries.

Features of Wind River GNU Compiler include:
- cpp, the C preprocessor
- gcc, the C and C++ compiler
- ld, the programmable static linker
- as, the portable assembler
- binary utilities

Wind River Debugger
The Wind River Debugger addresses the common and unique needs of developers involved with hardware bring-up, firmware/driver/BSP development, kernel development, and application development. It incorporates the feature set of best-in-class on-chip debugging environments, source-level debuggers, and target OS-aware development environments.

The Wind River Debugger combines the power of a direct hardware on-chip debug connection and a target agent connection, providing the ability to debug complex environments and complex device software applications. Innovative multi-context debugging capabilities allow developers to debug code running in multiple contexts simultaneously. Multiple contexts means any of the following:
- Multiple cores
- Multiple tasks/processes/threads
- Multiple physical processors
- Multiple processor types
- Multiple boards
- Multiple target operating systems

With support for on-chip-debugging-based and target-agent-based debugging in the same environment, Workbench gives you access to a wider range of debugging options than any single debug method can provide. For example, you can use the Wind River Debugger connection to on-chip debugging to bring up new hardware designs and develop board initialization code, code for ISRs, code for device drivers for the kernel, and code for bootloader applications. You can use the Wind River Debugger target agent to task mode debug a user application—and also be able to switch to the JTAG connection and determine the state of the target, if it suddenly crashes and takes the target agent down with it.

Wind River also allows you to connect to a target running VxWorks 6.x using Transparent Mode Driver technology, in which a virtual connection is made to the target agent through the on-chip debugging hardware and connection. Workbench includes the ability to debug VxWorks 6.x targets via full kernel, task, and real-time process (RTP) debugging, including visibility and synchronous control of tasks associated to an RTP, as well as full kernel and user mode debugging of Linux targets.

Tracking down kernel and interrupt handler bugs is difficult and time-consuming. A solid kernel mode debugging solution can save days or weeks in comparison to printf/printk, which requires multiple edit-compile-debug cycles. Workbench offers both on-chip debugging and software debugging capabilities for Linux kernel debug.

User applications (both VxWorks and Linux) involve multiple interacting tasks/processes, so debugging these applications is greatly enhanced if more than one process or task can be debugged concurrently. The Wind River Debugger offers best-in-class kernel and user application debugging for both VxWorks 6.x and Linux.

Other Eclipse-based development environments allow only one view of each type to be present within the perspective at one time, forcing you either to view relevant data for one processor, process, or thread at a time, or to place the data from multiple debug targets into a single view. Wind River has created colored differentiated instances of each debugger view, allowing up to eight instances of each view.
Wind River Debugger administration features enable:
- Multiple-window instantiation, with user-defined labeling/color, user-controlled manual/automatic update
- Color highlighting of changed value in any display
- Saved workspace settings by saving and restoring the state of the debugging environment
- Option to disable window updates on breakpoints or when execution stops
- Data tip support when in an appropriate stopped mode

C++ debugging capabilities include:
- Ability to debug code containing namespaces
- Support resolution of ambiguous namespaces
- Debugging of derived classes
- Breakpoint support for a single in-line function
- Support for stepping in-line functions
- Debugging of templates
- Breakpoint support
- In-lined template functions
- Debug implicit and explicit instantiations

To provide additional debugging capabilities, Wind River has taken object browsing and inspection capabilities from earlier Wind River products and focused them on the key areas necessary to provide world-class support for debugging VxWorks 6.x and Linux-based applications. With the object browsing capability, Wind River Debugger can inspect:
- RTPs
- Objects
- Tasks
- Shared libraries
- Semaphores
- Watchdog timers
- Message queues
- Memory partitions
- Modules
- Symbols
- ISR objects
- Triggers
- File descriptors
- I/O devices

Features of Wind River Debugger include:
- Basic execution control (step into, step over, step out, go, and stop)
- Advanced execution control (go all, stop all)
- Full-featured source-level, assembly-level, and mixed-mode debugging
- Comprehensive debugging views:
  - Watch view
  - Breakpoint view
    - Software breakpoints
    - Hardware breakpoints
    - Number of software breakpoints limited only by host and target resources
    - Counted breakpoints
  - Global variable view
  - Local variables view
  - Register view
  - Stack view
- Task view
- Memory view
- Function view
- Modules view
- Target selector facility/target connection dialog/wizards
- Specific support for debugging multiple execution paths or contexts
- Wind River Debugger target agent (see details below)
- Debug multiple processors, multiple threads/tasks, and multiple processes simultaneously, without having to detach and reattach to different execution paths
- Attach to and debug multiple VxWorks 6.x RTPs and tasks
- Debug code in shared libraries
- Separate debug views for each execution path within the same debugger pane; each view can be organized for ease of comprehension
- True multiple-context debugger with simultaneous viewing and control of real-time data relevant to each entity being debugged
- Debugging via multiple simultaneous target connections to processors and target boards, each of which can be using a different operating system

Wind River Target Debug Agents for Linux

Wind River Workbench and Wind River Linux platforms together provide extensive debugging capabilities for the kernel, kernel modules, and user mode applications. User mode applications are supported using an advanced agent based on the ptrace API. Advances in Linux 2.6 kernel multiple-threading enable developers to debug individual threads reliably inside processes without stopping the entire process. Because ptrace is applicable only with the debugging of applications in user mode, another method is required for debugging the kernel.

Kernel debugging for Linux 2.6 kernels, including device drivers, kernel modules, and interrupt handlers, is achieved by using the open-source “KGDB-2” agent, or the kernel-debugging version of GDB. With standard debugging tools, this would be an issue, as connecting to KGDB is not typically performed using the same debugger as the one used for user mode applications. Workbench’s multiple-context technology allows multiple simultaneous target connections, allowing a simultaneous
connection to KGDB-2 and the target agent through the same interface. Workbench provides the synchronization necessary to debug a multiple agent connection in a synchronized manner, despite the fact that multiple target debugger agents are in use.

Wind River Target Debug Agent for VxWorks
VxWorks 6.x requires a robust debugging agent capable of debugging one or more tasks within one or more real-time processes. The debug agent must be able to provide this functionality across multiple memory boundaries protected by hardware MMUs. Wind River has extended the capabilities of the wdb agent used in previous versions of VxWorks to include debugging and controlling one or more tasks contained within an RTP or multiple RTPs. At a high level, RTPs can be selected and controlled with specific actions. For example, issuing a “run” or a “stop” command results in the overall control of each task state within the RTP. Tasks can also be individually selected within an RTP and debugged without stopping any other tasks. Breakpoints can be created that are only detected when code is executed within the context of a specific task or RTP, greatly simplifying the debugging of a complex device design.

Additional capabilities of Wind River Debugger include:
- Kernel and user debugging
- Multi-process debugging through a single debugger
- Ability to debug a task or thread independent of a process (Linux) or real-time process (VxWorks 6.x)
- Process-qualified breakpoints that can stop a single process or a specified group of processes
- Forked-process debugging
- Ability of the wdb agent to stop the entire system when it panics, allowing attachment of Workbench to track the cause of the panic (typically, memory access error in a kernel thread or interrupt handler)
- Process fatal exception debug: The agent can attach to a process before it generates a core file, and the debugger can display all process and stack back trace information
- Kernel fatal exception debug: The agent can remain in system mode after the exception occurs, allowing the debugger to display kernel stack trace and other useful information
- Debugging of “stripped” target applications in which symbolic information only exists in the host copy (in order to save target memory resources)
- Support for hardware and software breakpoints
- No impact on the behavior of the scheduler
- Proven stability and robustness through 20+ years of use as the debug agent for VxWorks in all major processor architectures

Wind River Shell Environments
To enhance developers’ work environment and improve their effectiveness in developing VxWorks 6.x–based device software applications, Wind River Workbench provides three command line environments, or shells:
- VxWorks Host Shell
- Linux Host Shell
- VxWorks Kernel Shell

VxWorks Host Shell
The VxWorks Host Shell, previously known as WindSh, provides a command line interface that lets you download application modules and invoke both VxWorks 6.x and application module subroutines. This facility has many uses:
- Interactive exploration of the VxWorks 6.x operating system by calling any VxWorks routine and API
- Interactive exploration of VxWorks 6.x RTPs
- Debug and monitor processes
- Prototyping
- New to VxWorks 6.x:
  - Interactive development by calling any application (RTP) routines
  - VxWorks 6.x application (RTP) and kernel testing
  - Error management support through the output of error dumping; the ability to turn on/off error management on a per-task or per-RTP basis
  - Wind River Architecture for Messaging Protocol (WRAMP) support through text dump of the message traffic

The Host Shell executes on the development host, not the target—but it enables you to spawn tasks, look at RTPs, read from or write to target devices, and exert full control over the target. The Host Shell receives your commands, executes them locally on the host, and dispatches requests to the target server for any action involving the symbol table or target-resident programs or data.

Because the shell executes on the host system, you can use it with minimal intrusion on target resources. As with other VxWorks 6.x tools, only the target agent is required on the target system. Thus, the Host Shell can always remain available; you can use it to maintain a production system, as well as to experiment and test during development. Since you do not need to rebuild the VxWorks 6.x image, the Host Shell is useful on targets with restricted memory and permits system-mode debugging, which is helpful for debugging drivers.

Capabilities of the VxWorks Host Shell include:
- RTP debugging
- Kernel task debugging
- Task-specific breakpoints
- Task-specific single-stepping
- Symbolic disassembler
- Task and system information utilities
- Ability to call user routines
- Ability to create and examine variables symbolically
- Ability to examine and modify memory
- Exception trapping
- Uses wdb agent to access the target through three connection types: network, serial, or a pipe connection
- Supports four interpreters (C, CMD, Tcl, GDBmi)
- Supports Tcl scripting and is backward compatible with Tcl 8.0
- GDB command line mode
- Command mode (CMD)
Host Shell capabilities include:
- debugging drivers.
- permits system mode debugging, which is helpful for development. Since you do not need to rebuild the kernel image, you can use it to maintain only the target agent is required on the target system. Thus, the Host Shell can always remain available; you can use it to maintain with minimal intrusion on target resources. As with other tools, because the shell executes on the host system, you can use it with minimal intrusion on target resources. As with other tools, only the target agent is required on the target system. Thus, the Host Shell can always remain available; you can use it to maintain a production system, as well as experiment and test during development. Since you do not need to rebuild the kernel image, the Linux Host Shell is useful on targets with restricted memory and permits system mode debugging, which is helpful for debugging drivers.

Host Shell capabilities include:
- Kernel debugging
- Breakpoints
- Symbolic disassembler
- Process and system information utilities
- The ability to examine and modify memory
- Exception trapping for the debugged processes

Combined with the currently supported modes (C interpreter mode and Tcl interpreter mode), the Host Shell now supports four key command modes to increase the effectiveness of device software developers. The C mode enables the interpretation of simple C language expressions. Within C mode, most C operators can be executed, symbolic data references resolved, and subroutines invoked. This mode gives developers a prototyping and debugging tool to use within the kernel space. Users have access to the Tcl interpreter. The WTX Tcl API and all Tcl commands can be accessed from within Host Shell’s Tcl mode. Command mode follows a UNIX-style model that allows arguments to be specified by flags such as –f <arg>, with a flexible order and number of arguments. Command mode also allows the passing of string parameters without having to delimit with quotation marks. This mode lets users attach to RTPs and essentially provides a monitoring and debugging mode for a VxWorks 6.x-based device software application.

GDB mode enables users to debug the system using GDB commands by directly interfacing with the Wind River Workbench debugging server. The GDB mode provides an extended subset of the GNU GDB/CLI set of commands, such as file, run, attach, continue, interrupt, break, print, list, frame, up, down, next(i), step(i), disas, x, source, info functions, args, regs, locals, path, and cd. In addition, the three command modes (C, GDB, and CMD) can be accessed from within Tcl mode, allowing users to exploit Tcl’s complex scripting capabilities.

**Linux Host Shell**
The Linux Host Shell, similar to the VxWorks Host Shell, provides a command line interface that lets you download, invoke, and debug user processes. The Linux Host Shell executes on the development host, not the target, but it enables you to spawn processes, read from or write to target devices, and exert full control over the target. The Linux Host Shell receives your commands, executes them locally on the host, and dispatches requests to the debugger framework for any action involving the symbol table or target-resident programs or data.

Because the shell executes on the host system, you can use it with minimal intrusion on target resources. As with other tools, only the target agent is required on the target system. Thus, the Host Shell can always remain available; you can use it to maintain a production system, as well as experiment and test during development. Since you do not need to rebuild the kernel image, the Linux Host Shell is useful on targets with restricted memory and permits system mode debugging, which is helpful for debugging drivers.

**VxWorks Kernel Shell**
The VxWorks 6.x Kernel Shell, formerly known as the Target Shell, runs within the VxWorks 6.x kernel and provides direct access to VxWorks 6.x through a console or a network connection, such as Telnet. While the VxWorks Host Shell and VxWorks Kernel Shell have similar capabilities, there are several key differences:

- VxWorks Kernel Shell implements fewer commands than are provided in VxWorks Host Shell
- Both shells include a C and command mode CMD interpreter; the Host Shell also provides a Tcl and a GDB interpreter
- Host Shell is always ready to execute, provided the wdb target agent is included in the system; the Kernel Shell, as well as its associated target-resident symbol tables and module loader, must be configured into the VxWorks 6.x image by including the appropriate components
- Kernel Shell’s input and output are directed at the same view by default, usually a console connected to the board’s serial port; for the Host Shell, these standard I/O streams are not necessarily directed to the same view as the Host Shell

Features of the VxWorks 6.x Kernel Shell include:

- Multiple session support to enable a user to spawn multiple target shells:
  - Turn this feature on or off statically or dynamically
  - Used for Telnet, rlogin access, and virtual consoles
  - Enables independent and simultaneous interaction with the target from several remote connections and from the console
  - Possible to create a new shell session redirected to the host computer through the wdb virtual I/O
  - Design allows an unlimited (except by the target memory size) number of Kernel Shell sessions running at the same time; each session is connected to a different set of I/Os and has its own context information (such as internal parameter of the shell task, default task ID for debugging command, last memory address accessed, and command line history)
- Possible to create new Kernel Shell sessions from the host tool wtxConsole using the “-s” option
- Still possible to set up a VxWorks 5.5-compatible mode; in this case, only one shell session can be created, and it is shared between the connections
- Global standard I/O of VxWorks is set to one of the shells

- Multiple interpreter support: You can integrate your own interpreters (Tcl, GDB-like interpreter), dynamically switch between them, and have one interpreter evaluate statements by another one
- General shell enhancements
- Inclusion of the shell in the kernel without spawning an initial shell task
- The length of the input line of the target shell is configurable upon kernel creation and dynamically from a shell task
- Symbol name, task name, and path completion are provided
- Configurable shell prompt with format strings like the current path, and target name is provided
- RTP support: The real-time process, a new entity introduced by VxWorks 6.x, is an application in a protected memory context that executes at the user level. The VxWorks kernel lives in its protected memory area and executes at supervisor level. The RTP application is a fully linked module that can be relocated in memory and has no unresolved symbols. The RTP accesses the VxWorks 6.x kernel function through system call, and does not directly access the kernel memory area. The Kernel Shell allows users to monitor and debug the RTP in the following ways:
  - Launch and delete a RTP
  - List the RTPs and RTP tasks
  - Stop/continue an RTP (stop and continue all RTP tasks)
  - Add/remove a breakpoint on an RTP application
  - Disassemble/display/modify the RTP memory
  - List the existing symbols name or modify the existing symbols value of an RTP
  - Shared library support
  - Symbol management support
  - Execution path settings
  - Environment variable settings
- Command interpreter mode: A new UNIX-like interpreter is available to access the RTP; also available in the kernel
- WRAMP support: Several APIs are available in the shell to give information about WRAMP messages
- Fault management support: Target shell provides commands to display information reported by the operating system error management capabilities, as well as help debug applications using the fault management capabilities
- Secure target shell: You can protect access to the Kernel Shell with a login and password. This feature is available for remote connections and for a connection over the console/serial line. You can also configure an inactivity timeout delay, after which the shell returns to the login/ password prompt.
- New C interpreter: The existing C interpreter has been replaced to ease future maintenance and extension of C language support
- Multiple line edit mode support: You can integrate your LED modes and dynamically switch between them; there is an “EMACS-like” LED mode, in addition to the traditional “vi-like” LED mode
- Target Shell access via a Workbench virtual console

**VxWorks Kernel Configurator**

The VxWorks 6.x Kernel Configurator is a graphical utility that simplifies and accelerates the task of selecting the operating system components that need to be included in a bootable VxWorks image. The Kernel Configurator is backward compatible with Tornado 2.2 and VxWorks 5.5. A command line utility, vxprj, is available; it enables the ability to perform a kernel build within scripts used as part of nightly builds.

When creating a new bootable kernel image, Workbench analyzes available kernel components and BSP and compiler selections. The Kernel Configurator displays a summary of key configuration data, such as number of selected components or data and text size. A bundle selector allows users to quickly and easily include or exclude dedicated configuration bundles composed of multiple components from a kernel image. Sample configuration bundles include components needed for POSIX compliance, real-time process development, and error management.

Selecting kernel components individually gives you greater flexibility and control over your VxWorks image. The Kernel Configurator analyzes component dependencies and highlights conflicts when components are required but not selected, or if components are not compatible with one another. An AutoScale feature analyzes the entire VxWorks image and removes unused kernel components that may unnecessarily increase the size of a bootable image.

It is also possible to include custom component definitions for specialized purposes or from third parties; the Kernel Configurator verifies if component selections are valid and conflict-free.

Workbench offers the ability to assemble projects in hierarchies as a means of designing the structure of a target system. A bootable kernel project may include a file system project, as well as several RTP projects. Each RTP project may consist of one or several libraries and other source code modules. The advantage of creating a project structure over defining loosely coupled
projects is that the build system will create bootable kernel images in a single run, compared to multiple build stages that must each be started individually. It is also a more logical and natural representation of system components.

Features of the VxWorks Kernel Configurator include:

- Component bundle selector
- Improved system builder
- Automatic scaling of the kernel size (AutoScale)
- Dependency control and conflict resolution
- Ability to create project hierarchies to reflect the structure of the final system, including bootable, file system, and RTP projects
- Command line availability of all GUI features
- VxWorks Image Builder

Wind River ICE

Wind River ICE, our newest BDM/EJTAG/JTAG hardware run-control device, takes serial Ethernet communications coming from the host development platform and transfers them to the appropriate communications protocol required to communicate with the CPU’s on-chip debugging capability.

Through its JTAG server capability, Wind River ICE supports multiple JTAG/EJTAG devices on a single scan chain, as well as multiple debugger connections to these devices. Wind River ICE supports connections for up to eight devices simultaneously in a scan chain of up to 128 individual EJTAG/JTAG devices. Wind River ICE also has the ability to support the internal Trace buffer provided on Freescale’s MPC85xx processor family.

In addition, Wind River ICE has the ability to support externally Trace-buffered SoCs, such as AMCC’s PPC405 and 440 families and Freescale’s MCF5xxx family. This is supported through addition of our Wind River Trace module, an adapter module that plugs into the end of the Wind River ICE cable.

Wind River Trace

Wind River Trace allows developers better visibility into the hardware/software interaction within their device. It provides a GUI within the development environment to show Trace configuration parameters and display Trace data. Wind River Trace also offers a hardware adapter for Wind River ICE, enabling it to capture and buffer more than 900,000 lines of Trace data from the target.

Wind River Probe

Wind River Probe uses on-chip debugging services embedded in a microprocessor and the Wind River JTAG Accelerator technology to provide the industry’s most feature-rich tool for board bring-up, flash programming, and production/test.

Wind River Probe provides a high-speed USB connection between a host debugger and target microprocessor for PC-based development, as well as a high-performance emulation system with quick download speeds, overall debug throughput, and unprecedented debug efficiency.

Wind River ICE also has the ability to support the internal Trace buffer provided on Freescale’s MPC85xx processor family.

For more information on Wind River ICE, Wind River Trace, and Wind River Probe, please see their corresponding Product Notes.

Wind River System Viewer

System Viewer, formerly known as WIND®VIEW, is the run-time analysis tool for device software developers who need to inspect the dynamic behavior of device software systems to detect runtime problems and improve system performance. System Viewer captures the dynamic interactions of the operating system, device software applications, and target hardware.

System Viewer provides detailed analysis and graphical visualization of system events, revealing the complex interactions of tasks, threads, interrupts, and system objects of applications executing on a target. Context changes are clearly shown, as well as such system events as semaphores, message queues, signals, tasks, timers, and user events. This tool is best suited for use when developers need to diagnose and solve one or more of the following problems:

- Task/process/thread scheduling problems, such as deadlocks, starvation, and race conditions
- Performance problems, such as priority setting, resource contention, and mutual exclusion
- Timing problems arising from the interaction of interrupts and tasks

System Viewer allows device software developers to detect anomalous behavior quickly, then understand the cause and effect by reviewing the complete history of events leading up to the problem, including error management events from VxWorks 6.x. This tool operates independently of the target architecture and can be used with Wind River VxWorks Simulator (for VxWorks 6.x) before hardware is available, and with Wind River Linux.
platforms using Linux Trace Toolkit (LTTng) instrumentation. System Viewer comprises Event Log Collection and Event Log Analysis, and it supports the definition of custom events that can be used to identify the occurrence of events manually inserted in the executable. Custom events are supported for VxWorks only.

**Event Log Collection**
System Viewer is instrumentation-based to provide exact timing of information about system events. System Viewer allows specification of the event-logging level (such as logging events from context switches), from tasks-changing state, or from events that operate on objects. This allows developers to control the amount of data collection and balance the intrusion caused by the logging.

For example, users may configure which time-stamp level should be used for event logs, such as sequential, high-resolution, or custom time stamps. This method allows developers to choose the desired level of instrumentation without impacting the edit-compile-debug cycle.

Developers can also use the triggering facility on VxWorks targets to start and stop data collection, so data is collected from a precise interval under precise conditions; and they can manage data upload to the host to optimize system resources. For those unique conditions, developers can create their own application-specific events.

**Event Log Analysis**
System Viewer provides graphical tools to make data analysis efficient and effective. The Event Graph is the main System Viewer display, presenting events as a scrollable graph. For each event, status information—such as parameters, time stamps, and argument values—is shown to aid measurement of the elapsed time between events. When analyzing large amounts of data, you can select from various filtering options to focus on specific tasks and events you want to study. Bookmarks indicate interesting locations in the event log, and icons help identify event types. Stippled lines indicate the state of each task at any given time. VxWorks 6.x real-time processes (RTPs) are supported in the display, allowing the expansion and collapse of RTPs to analyze tasks within their RTP contexts.

Linux applications can also benefit from System Viewer’s graphical analysis presentation. With Linux, data collection and presentation are based on the support of Linux Trace Toolkit (LTTng) data collected from an instrumented Linux kernel.

The System Viewer Radar presents a time-based view of an entire log and provides mechanisms to modify the selected display in the main tool windows. It offers several display modes that are unique ways of visualizing areas of activity within an event log. Device software developers can use the radar to show all events, peak activity, and event intensity, and to quickly zoom into areas of particular interest in the log.

The Event Table presents the events in the log as rows of information. In conjunction with the event container tree, the Event Table displays only events from selected tasks and interrupts.

**ScopeTools**
ScopeTools are powerful and dynamic visualization tools for device software applications. They provide developers with visibility into the entire platform: application code, third-party libraries, and the operating system. You can monitor variables, optimize performance, and find memory problems—all while the system is still running.

An integral part of Workbench, ScopeTools support both VxWorks 6.x and Wind River Linux platforms. Three ScopeTools are included with Workbench: ProfileScope, MemScope, and...
StethoScope. TraceScope and CoverageScope are optionally available for VxWorks 6.x at an added cost.

Features of ScopeTools include:

- Learn where the processor spends its time, so you can focus on optimizing areas that contribute the most to overall performance
- Analyze all the code running in the system, including the kernel and third-party applications
- Understand more about the behavior of your platform through clearly diagrammed details of application and system behavior
- Quickly uncover and resolve system problems faster through superior data analysis and display
- Change variables on the fly, allowing more flexible experimentation and immediate observation of the results
- No hardware setup or code instrumentation required (except for CoverageScope), allowing easy installation of the tools with minimal system impact

Included with Wind River Workbench:

- **ProfileScope**: Profiling is critical for real-time systems. Once you understand performance bottlenecks, it becomes easier to optimize application code. ProfileScope is a dynamic execution profiler that provides detailed function-by-function performance analysis, specifying individual routines within the program that are consuming the CPU cycles. ProfileScope pinpoints inefficiencies and shows how performance changes over time.

- **MemScope**: Ensuring optimal use of memory is a critical activity in device software design. In many applications, memory usage is not fully understood, and a large portion of available memory is wasted. Systems can run for days before failing due to non-characterized memory leaks. MemScope is an instant memory analyzer that provides greater visibility into memory usage. Without any special compilation or instrumentation, you can monitor available memory, detect leaks that occur due to system calls or third-party libraries, and even watch leaks as they happen.

- **StethoScope**: This real-time graphical monitoring tool is used to examine variables, data structures, or memory locations in your system. You can watch any set of variables, see peak values and out-of-range settings you would otherwise miss, trigger collection on specific events, change variables while your program runs, and save collected data to disk. StethoScope presents this live analysis of your program without stopping or slowing your code.
OPTIONAL ADD-ONS FOR WIND RIVER WORKBENCH

ScopeTools for Test and Validation
In addition to the standard ScopeTools included with Wind River VxWorks and Linux platforms, two other tools are available as options:

- **TraceScope**: Trace your code execution in real time by providing function call sequences as your code executes. Included with the function call displays are the provided parameters, as well as the returned values, to allow you to identify when function behavior and execution timing change.

- **CoverageScope**: Analyze your code to determine which code segments are executed during testing. Visibility into the execution of individual statements, decisions, and conditions enables you to create more thorough test scenarios, ensuring delivery of higher-quality devices. It also becomes easy to identify and remove code that is never executed, thus preventing future problems and reducing your overall memory footprint.

Workbench On-Chip Debugging (OCD)
The Workbench development environment can be enabled for on-chip debugging (OCD). Wind River's OCD capability, along with Wind River ICE, Wind River Trace, or Wind River Probe hardware, provides access to significant additional capability within Workbench.

In the early stages of hardware and software development, a robust connection to the microprocessor through its run-control port is essential. Workbench OCD provides connectivity between the host development environment and the target device via the JTAG or OCD interface of the microprocessor residing on the device. The OCD interface of most microprocessors enables full control of the microprocessor itself, access to core and peripheral registers, and access to on-chip switch fabrics and memory controllers, along with access to external buses and many devices attached directly to the bus. In addition, some microprocessors support either internal or external Trace buffers, allowing developers to capture information about the exact code that ran on the target and when.

On-chip debugging provides developers with complete system-level control of their environment at all times, enabling more efficient and effective hardware bring-up, firmware development, and device driver and BSP generation. OCD can also be a useful alternative to agent-based debugging in applications where serial, Ethernet, or USB interfaces are not available, or in environments where agent instrumentation of the OS is not desired.

Extended Workbench capabilities offered through the OCD connection include:

- OCD target connection manager
- OCD command shell
- OCD console
- Flash programming
- Hardware and memory diagnostics
- CF options
- JTAG editor
- Extensions to register view, including:
  - Bit-level register details
  - Additional peripheral register support for most processors
  - Combined register view with agent views and perspectives
  - OCD user’s perspective within Workbench
  - Wind River Trace (may require additional hardware, to be purchased separately)
  - Firmware update
  - Cache memory view
  - Statistical performance analyzer (PFA)
- OCD reset and download/launch
- Linux OS awareness via JTAG
- VxWorks 6.x OS awareness via JTAG
- VxWorks 5.5 OS Awareness via JTAG
- All targets
- All targets with BSP available
- ThreadX OS awareness via JTAG

Figure 14: ScopeTools—TraceScope

Figure 15: ScopeTools—CoverageScope
Workbench Unit Tester

Workbench Unit Tester, available at an added cost for use with VxWorks 5.5- and VxWorks 6-based platforms, is an integrated set of tools that allows developers greater efficiency in completing unit testing, integration testing, and code coverage analysis on the tests. The integration between Workbench Unit Tester and the rest of the development suite places these capabilities within easy reach of every developer. Workbench Unit Tester increases software quality, decreases time-to-market, and reduces support costs through better, faster, more automated testing in the development life cycle. For more detailed information, please see the Wind River Workbench Unit Tester Product Note.

Workbench Diagnostics

Workbench Diagnostics, available at an added cost for use with VxWorks 5.5- and VxWorks 6-based platforms, allows developers to instrument a running system and collect valuable system status and diagnostic information easily. Enabling users to insert code sections into a system without recompiling or reloading the applications on the device, Workbench Diagnostics also offers a valuable new software debugging capability, increasing developer productivity by compressing debug cycles and eliminating unnecessary instrumentation-compile cycles. In the event of a system failure, Workbench Diagnostics lets developers conduct “forensic” analysis on a core file effectively, uncovering system behavior step-by-step prior to the system failure. For more detailed information, please see the Wind River Workbench Diagnostics Product Note.

TECHNICAL SPECIFICATIONS

Workbench 2.5 Host OS Support
- Red Hat Enterprise Linux 3, update 5
- Red Hat Enterprise Linux 4, update 1
- Solaris 8
- Solaris 9
- SuSE Desktop Linux 9.3
- SuSE Desktop Linux 10
- Windows 2000 Professional with Service Pack 1
- Windows XP Professional with Service Pack 2

Workbench 2.5 Target OS Support
- VxWorks 6.0, 6.1, 6.2, and 6.3
- VxWorks 5.5
- VxWorks 653
- Linux: Wind River Linux platforms
- Native Linux development on Red Hat Enterprise Linux (3 and 4)
- ThreadX 4.0 (Workbench, On-Chip Debugging Edition, only)
- Customizable target OS awareness capability for Workbench, On-Chip Debugging Edition, enables support for other target operating systems to be added

Workbench 2.5 Target Processor Support
New processor support is added on a regular basis. Please contact your Wind River Account Manager for the latest information.

Workbench Processor Family Support for VxWorks 5.5
(only available for Windows hosts)

PowerPC Architecture
- PPC40x
- PPC44x
- PPC60x
- PPC74xx
- PPC75x
- MPC8xx
- MPC82xx
- MPC85xx

Workbench Processor Family Support for VxWorks 6.x

ARM Architecture
- ARM9
- ARM 9E
- ARM 11

Intel Architecture
- Pentium
- Pentium II
- Pentium III
- Pentium 4

Intel XScale Architecture
- IXP425
- IXP465
- IXP23xx

MIPS Architecture
- BCM1250
- MTI 24K
- MTI 4K
- MTI 5K
- NEC vr54xx
- NEC vr55xx
- PMC-Sierra RM9K
- Toshiba Tx49xx

PowerPC Architecture
- PPC40x
- PPC44x
- MPC5200
- PPC74xx
- PPC75x
- MPC8xx
- MPC82xx
- MPC83xx
- MPC85xx
- MPC86xx
- PPC970FX

SuperH Architecture
- SH-4
- SH-4A
Workbench Processor Family Support for Linux

ARM Architecture
- ARM 9
- ARM 11

Intel Architecture
- Pentium II
- Pentium III
- Pentium 4
- Pentium M
- Xeon
- EM64T

MIPS Architecture
- MTI 4K
- MTI 24K
- BCM 1480

PowerPC
- MPC82xx
- MPC83XX
- MPC85xx
- PPC44x
- MPC74xx
- PPC75X
- PPC970FX

XScale Architecture
- IXP465
- IXP425
- IXP2850
- PNX270

Workbench Processor Family Support for VxWorks 653

PowerPC
- MPC74xx
- PPC75X

Processor Family Support for Workbench, On-Chip Debugging Edition

ARM
- ARM 9

ColdFire
- MCF52xx

MIPS
- BCM11xx
- BCM33xx
- BCM47xx
- BCM 7038
- BCM71xx
- BCM73xx
- RM7900
- RM9xxx
- 4K
- 5K
- 20K
- 24K
- 25K

- VR41xx
- VR54xx
- VR55xx
- VR77xx
- PR19xx
- PR39xx
- PR44xx
- PNX30xx
- PNX83xx
- PNX85xx
- TX49xx

PowerPC
- PPC40x
- Xilinx X2VP
- Xilinx XC4VFX
- PPC44x
- MPC52xx
- MPC5xx
- PPC60x
- PPC7xx
- MPC 8xx
- MPC82xx
- MPC83xx
- MPC85xx
- MPC74xx
- MPC970

XScale
- IOP80219
- IOP80315
- IOP310
- IOP321
- IOP331
- PXA2xx
- IXP42xx
- IXP45x
- IXP465
- IXP2400
- IXP2800
- IXP2325
- IXP2350

PROFESSIONAL SERVICES

Wind River Professional Services enable companies to reduce risk and improve competitiveness. Our team delivers device software expertise within structured engagements that directly address key development challenges and contribute to the success of our clients. Our track record of timely delivery and in-depth understanding of market and technology dynamics makes Wind River a valuable implementation partner for clients worldwide. Based on our commercial-grade project methodology, service offerings include device design, BSP and driver optimization, software system and middleware integration, and legacy application and infrastructure migration.

Workbench Services

Whether you select Wind River Workbench as a stand-alone product or as part of our platform solutions, Wind River Professional Services knows how to jump-start your development
efforts. Even if you opt for a non-Wind River platform, Linux distribution, host operating system, or target architecture, we can help. No matter which development environment you use, Wind River Professional Services can extend Workbench to adapt to your needs with the following offerings:

- Extend Workbench processor support
- Extend Workbench target OS support
- Validate Workbench on Linux host environment
- Validate Eclipse plug-ins
- Integrate agents

**Installation and Orientation**

Proper installation and orientation of Wind River VxWorks and Linux platforms means you won’t waste time solving easily avoidable problems before you can begin your next development project. Wind River offers an Installation and Orientation Service to ensure that your project starts on time and without hassle by delivering:

- **On-site installation:** Guided install on your hardware and host platform, along with a sample build process, demonstrations, and examples of customizations
- **Hands-on orientation:** Architecture, development file system, adding open-source packages, porting drivers, addressing design issues
- **Advice:** Introduction to Wind River support channels and processes, additional services, project review, and consultation

The Wind River Installation and Orientation Service will expedite your path to productivity, allow you to rest assured that we have eliminated a common source of user error, and help you realize all of the platform’s potential.

**EDUCATION SERVICES**

Education is fundamentally connected not only to individual performance, but also to the success of a project or entire company. Lack of product knowledge can translate into longer development schedules, poor quality, and higher costs. The ability to learn—and to convert that learning into improved performance—creates extraordinary value for individuals, teams, and organizations. To help your team achieve that result, Wind River offers flexible approaches to delivering product education that best fits your time, budget, and skills development requirements.

**Personalized Learning Program**

Wind River offers a unique solution to minimize the short-term productivity drop associated with the process of adopting new device software technology, and optimize the long-term return on investment in a new device software platform. The Wind River Personalized Learning Program delivers the right education required by individual learners to accomplish their jobs. The program identifies work-related skill gaps, generates development plans, materials, and learning events to address these skill gaps, and quantifies the impact of the development activities for each individual user.

This programmatic, focused, and project-friendly approach to skills development results in a significant increase in the personal productivity of your team, improved efficiency in the processes they employ, and faster adoption of the technology you have purchased. Personalized Learning Programs deliver improved business performance—customers have reported a return on investment ranging from 18% to 30% over a traditional training approach.

Please consult your local Wind River sales representative for more information on Personalized Learning Programs.

**Public Courses**

Wind River’s public courses are scheduled for your geographical convenience. They are conducted over one to five days, using a mixed lecture and interactive lab classroom format that leverages the experience of Wind River instructors and other course participants. Courses provide a fast, cost-effective way for students to become more productive with Wind River technology.

Benefits of public courses include:

- A conceptual introduction that orients students to the subject matter
- A selective examination of the details, focusing on the most commonly used areas, or on areas with which users tend to be least familiar
- Personal guidance and hands-on application of individual tools and course concepts
- The chance to grasp device software concepts, as well as the fundamental issues involved in real-time design
- The knowledge needed to develop device drivers, perform hardware porting, or develop applications
- Answers to specific questions about topics addressed in the course

Please consult your local Wind River sales representative for more information on course schedules and fees.

**On-Site Education**

If you have a large project team or a number of new users, you may benefit from custom on-site education. Instructors will consult with you and, based on the workshop series curriculum, determine which topics should be included and emphasized. This type of education offers an opportunity for one-on-one discussions with our instructors about your specific project needs, technical requirements, and challenges—all in the comfort of your own office.

Advantages of on-site education:

- The entire team gains a common knowledge base
- On-site education helps ensure that knowledge and skills will transfer from the classroom to the workplace
- The location saves employees both travel expenses and time away from the office

Please consult your local Wind River sales representative for further information.
SUPPORT SERVICES

Wind River provides full technical support for VxWorks 6.x, Wind River Workbench, and Wind River Linux platforms. Our global support organization is staffed with engineers who have extensive experience with Wind River products and device software development. At major support centers worldwide, our local experts can help diagnose problems, provide guidance, or answer “How do I…” questions. Support is also available 24 hours a day at our Online Support website (www.windriver.com/support) or by email at support@windriver.com.

Visit Wind River Online Support (OLS) for fast access to product manuals, downloadable software, and other problem-solving resources for Wind River Workbench. Additional features, including patches and technical tips for common problems, are available for all customers on subscription. OLS visitors can also access a community of developers to discuss their issues and experiences.

If you cannot find the information you need through Online Support, please contact our global support team for access to the industry’s most knowledgeable and experienced support staff:

North America, South America, and Asia/Pacific
support@windriver.com
Toll-free tel.: 800-872-4977 (800-USA-4WRS)
Tel.: 510-748-4100
Fax: 510-749-2164
Hours: 6:00 a.m. to 5:00 p.m. (Pacific time)

Japan
support-jp@windriver.com
Tel.: +(00)81-3-5778-6001
Fax: +(00)81-3-5778-6003
Hours: 10:00 a.m. to 5:00 p.m. (local time)

Europe, the Middle East, and Africa
support-EC@windriver.com
Toll-free tel.: +(00)(800) 4977-4977
France tel.: +33(0) 1 64 86 66 66
France fax: +33(0) 1 64 86 66 10
Germany tel.: +49(0) 899 624 45 444
Germany fax: +49(0) 899 624 45 999
Israel tel.: +972(0) 9741 9561
Israel fax: +972(0) 9746 0867
UK tel.: +44(0) 1793 831 393
UK fax: +44(0) 1793 831 808
Hours: 9:00 a.m. to 6:00 p.m. (local time)

The Wind River logo is a trademark of Wind River Systems, Inc., and Wind River and VxWorks are registered trademarks of Wind River Systems, Inc. Other marks used herein are the property of their respective owners. For further information regarding Wind River trademarks, please visit our website at www.windriver.com/company/terms/trademark.html.