WIND RIVER DIAB COMPILER

Boost application performance, reduce memory footprint, and produce high-quality, standards-compliant object code for embedded systems with Wind River® Diab Compiler. Wind River has a long history of providing software and tools for safety-critical applications requiring certification in the automotive, medical, avionics, and industrial markets. And it's backed by an award-winning global support organization that draws on more than 25 years of compiler experience and hundreds of millions of successfully deployed devices.

TOOLCHAIN COMPONENTS

Diab Compiler includes the following programs and utilities:

- **Driver:** Intelligent wrapper program invoking the compiler, assembler, and linker, using a single application
- **Assembler:** Macro assembler invoked automatically by the driver program or as a complete standalone assembler generating object modules
  - Conditional macro assembler with more than 30 directives
  - Unlimited number of symbols
  - Debug information for source-level debugging of assembly programs
- **Compiler:** ANSI/ISO C/C++ compatible cross-compiler
  - EDG front end
  - Support for ANSI C89, C99, and C++ 2003
  - Hundreds of customizable optimizations for performance and size
  - Processor architecture–specific optimizations
  - Whole-program optimization capability
- **Low-level virtual machine (LLVM)–based technology:** Member of the LLVM community, accelerating inclusion of new innovative compiler features and leveraging the LLVM framework to allow easy inclusion of compiler add-ons to benefit customers
- **Linker:** Precise control of allocation, placement, and alignment of code and data
  - Object modules linked into absolute or relocatable modules
  - Fully embedded application binary interface (EABI)–compliant ELF/DWARF output for tool interoperability
  - Stack usage estimates
- **Libraries:** Standard runtime functions to help developers create applications
  - Complete re-entrant C libraries compliant with ANSI/ISO, POSIX®, and SVID standards
  - Complete C++ library and Standard Template Library (STL)
  - Full complement of math libraries, including IEEE-754 appendix functions
  - Fast, efficient floating-point libraries
  - Library source code

- **Link-time optimization (LTO):** Method for achieving better runtime performance through whole-program analysis and cross-module optimization
- **GNU ARM linker support:** Two equivalent options to instruct the driver to call the GNU linker (for greater GNU compatibility), in addition to the default instruction to the driver to call the Diab Compiler linker
- **Simulation with system mode QEMU:** In Diab Compiler 7.0.1, support for the open source QEMU machine emulator; QEMU replaces the Wind River WindISS simulator
- **Xlicense-wait:** Added option to help prevent compilation from aborting if a license is not available
- **Archiver/librarian:** Creation and maintenance of libraries
- **Instruction set simulator:** Simulation of the core instructions of the target processor and ability to run C and C++ programs with the simulated environment
- **Object file converter:** Conversion routines for generating S-Record or IEEE-695 output formats as well as the following:
  - Symbol table management
  - Detailed code size reports
  - C++ symbol name demangler
  - Munch routine
- **Runtime error checker:** Identification of root cause of software errors during program development, including the following:
  - Memory leaks and stack overflows
  - Pointer problems
  - Memory allocation errors
- **Lint program checker:** Compiler-time and link-time static code analysis for ANSI C conformance, finding programming errors such as:
  - Unused variables and functions
  - Used variable before set
  - Missing return statements
  - Out-of-range constants
  - Function call mismatches
- **Eclipse CDT plugin:** Creation of projects and building of Diab Compiler application using the Eclipse integrated development environment
- **Documentation:** Extensive documentation specific to the chosen architecture, with all manuals available in PDF format; detailed "Getting Started" manual enabling users to get up to speed quickly and enhancing the out-of-the-box experience

TECHNICAL HIGHLIGHTS

- **Selectable speed/size optimizations:** Certain compiler optimizations involve trade-offs between execution speed and code density. With Diab Compiler's numerous compiler switches, users can choose whether to optimize for speed or code size.
- **Small data area optimizer:** For certain architectures, "small" data and constant areas use predefined sections that can optionally be created by the compiler to improve reference efficiency for widely used static or public variables.

- **Code factor optimizer:** Diab Compiler finds common code sequences at link time and shares them, reducing code size at the cost of inserting some additional branches.
- **Register coloring:** Diab Compiler locates variables that can share a register to eliminate loads and stores.
- **Global common subexpression elimination:** Subexpressions, once computed, are held in registers and not recomputed the next time they occur. Memory references are also held in registers.
- **Reverse inlining:** This option reduces code size by factoring out repeated code sequences into new functions. This optimization can lead to significant code-size reduction, depending on the structure of the code.
- **Whole-program optimization:** This capability allows the compiler to optimize calls between functions in different source files, improving execution efficiency by allowing function inlining across different modules.
- **Flexible mixing of C/C++ and assembly:** Diab Compiler provides several methods for mixing C/C++ and assembly code. asm macros can be used to inline sections of assembly code that can be invoked as a function. asm strings provide a simple way to embed assembly instructions. Diab Compiler also offers a number of compiler intrinsics that correspond to assembly instructions that improve compiler optimization.
- **Easy interrupt handling:** Diab Compiler makes it easy to handle interrupt processing for embedded systems by providing interrupt keywords and interrupt pragmas.
- **Multiple debugging options:** Diab Compiler provides flexible controls for generating debuggable code. Users can control the trade-offs between the amount of debug information vs. the speed of debugging, and performance optimizations vs. ease of debugging.
- **Position-independent code and data:** Diab Compiler can generate code and data that can be loaded at any address. This is useful in devices that dynamically load/unload modules.
- **Volatile keyword, or all memory is volatile:** Users can mark areas of code as volatile, which prevents the compiler from optimizing away data accesses. This feature is useful for accessing memory-mapped device I/O.
- **Control of structure formats:** Diab Compiler can reduce footprint by packing structures and ensuring that all padding is removed. The compiler can also create byte-swapped structures in which it swaps the byte order for data structures as they are stored in memory, allowing the communication of information in a byte order different than the device's native byte order. This capability can help optimize performance when sharing data between big and little endian processors.
- **Extensive link command language for memory mapping:** Every embedded device has a unique memory layout, with various types of memory available, such as fast RAM, flash, and shared memory. The link command language provides users fine-grained control to lay out code and data in memory in the optimal way.
- **Absolute addressing from C and assembler:** Variables and functions can be assigned to specific memory addresses. This feature is particularly useful for accessing memory-mapped device I/O and for setting up interrupt vectors.

- **Ability to generate PROMable code:** Diab Compiler can generate code that can be burned into ROM and can access memory that is moved into RAM at startup time.
- **Support for multiple object module formats:** The compiler supports ELF, IEEE-695, and S-Records and can generate object modules in multiple formats.

FUNCTIONAL SAFETY AND AUTOMOTIVE GRADE QUALITY

- **Automotive SPICE Level 2 certification:** Diab Compiler is developed using an Automotive SPICE (Software Process Improvement and Capability Determination) Level 2 process. Automotive SPICE is a framework for assessing software development processes created by the consensus of several major car manufacturers such as Audi, BMW, Ford, Fiat, Daimler, Porsche, Volkswagen, and Volvo. Today, Automotive SPICE has become a standard in the international automotive industry to create better processes and better product quality.
- **TÜV SÜD Certification:** Suitable for developing safety-related software for both ISO 26262 (ASIL) and IEC 61508 (SIL) up to the highest defined levels of safety, Diab Compiler is now being offered to customers in a safety certification package with the TÜV certificate, a safety manual, and technical and certification reports. Customers can use this package (following included guidelines, conditions, and restrictions) to deploy Diab Compiler as a TCL1-TCL3 tool for their all their safety projects needing the highest levels of functional safety. Diab Compiler's safety portfolio supports ASPICE and other long life-cycle safety markets such as avionics (DO-178B), nuclear (IEC 60880), railway (EN 50128), and industrial (IEC 61508).

With ongoing Diab Compiler updates (targeting both legacy/microcontroller platforms and high-performance compute platforms), the ISO 26262 (ASIL), IEC 61508 (SIL) TCL3 Cert Qualification and key optimizing enhancements, Wind River reaffirms its commitment to providing the highest-quality safety software tools for automotive and other safety-conscious industries.

TECHNICAL SPECIFICATIONS

Supported Host Operating Systems

- Windows 7 32 bit and 64 bit
- Windows 8.1 32 bit and 64 bit
- Windows 10 32 bit and 64 bit
- Red Hat Linux 6.7 32 bit and 64 bit
- Red Hat Linux 7.2 32 bit and 64 bit
- Ubuntu 14.04 LTS 32 bit and 64 bit
- Ubuntu 16.04 LTS 32 bit and 64 bit
- Fedora 22 32 bit and 64 bit
- Fedora 23 32 bit and 64 bit
- SUSE Linux/Open SUSE 13.2 32 bit and 64 bit
- Open Suse Leap 42.1 32 bit and 64 bit
- SLED 12 32 bit and 64 bit

Supported Targets

Diab Compiler supports a wide range of embedded architectures and provides processor-specific optimizations for each one. This range of choice permits continuity should you decide to migrate from one architecture to another. Compiler support is specific to a processor core and its instruction set. Many processors can be based on a single processor core. The following list provides processor architectures that are supported by Diab Compiler:

- PowerPC
- TriCore
- RH850
- ARM
- MIPS
- SuperH
- ColdFire
- 68K
- M·CORE
- SPARC
- Intel

For more detailed information on processor cores and processors that are supported, contact your Wind River sales representative.

WIND RIVER PROFESSIONAL SERVICES

Whether you select Diab Compiler as a standalone product or as part of our platform solutions, the Wind River Diab Compiler and Wind River Professional Services teams know how to jump-start your development efforts. Types of services provided may include the following:

- Extended compiler processor support
- Application and tuning of compiler optimizations for maximum performance
- Customized support and maintenance
- Updates for end-of-life products
- Safety certification audits
- Code migration

AWARD-WINNING GLOBAL SUPPORT

Diab Compiler is supported by an award-winning and Service Capability and Performance (SCP)–certified organization and the Wind River Support Network website, available 24/7. The website provides patches, manuals, the latest errata, and other announcements, as well as tech tips, application notes, and answers to FAQs. Wind River experts are available for telephone support during standard business hours.

Long Term Support and Frozen Branch Maintenance

In addition to standard support, Wind River offers Long Term Support services for Diab Compiler customers. Long Term Support lengthens the support window beyond the standard product lifecycle for devices that need support for a specific compiler version for many years or even decades.

For customers in the safety-related industry that require complete control of the product lifecycle of the compiler that builds their software, Wind River offers Frozen Branch Maintenance. These maintenance packages allow customers to minimize the impact of compiler changes to their code by having their own branch of the compiler for which they control the lifecycle. They can decide what updates and customized bug fixes to include and when new QA cycles will be run. Frozen Branch Maintenance packages are available for current versions of the compiler and versions that have reached end-of-life.

HOW TO PURCHASE

Visit www.windriver.com/company/contact to find your local Wind River sales contact. To have a sales representative contact you, call 800-545-9463 or write to salesinquiry@windriver.com.