

Diab™ Compiler Suite

5.0.1

RELEASE NOTES



Copyright © 2002 Wind River Systems, Inc.

ALL RIGHTS RESERVED. No part of this publication may be copied in any form, by photocopy, microfilm, retrieval system, or by any other means now known or hereafter invented without the prior written permission of Wind River Systems, Inc.

AutoCode, Embedded Internet, Epilogue, ESp, FastJ, IxWorks, MATRIX_X, pRISM, pRISM+, pSOS, RouterWare, Tornado, VxWorks, *wind*, WindNavigator, Wind River Systems, WinRouter, and Xmath are registered trademarks or service marks of Wind River Systems, Inc. or its subsidiaries.

Attaché Plus, BetterState, Doctor Design, Embedded Desktop, Emissary, Envoy, How Smart Things Think, HTMLWorks, MotorWorks, OSEKWorks, Personal JWorks, pSOS+, pSOSim, pSOSystem, SingleStep, SNiFF+, VSPWorks, VxDCOM, VxFusion, VxMP, VxSim, VxVMI, Wind Foundation Classes, WindC++, WindManage, WindNet, Wind River, WindSurf, and WindView are trademarks or service marks of Wind River Systems, Inc. or its subsidiaries. This is a partial list. For a complete list of Wind River trademarks and service marks, see the following URL:

<http://www.windriver.com/corporate/html/trademark.html>

Use of the above marks without the express written permission of Wind River Systems, Inc. is prohibited. All other trademarks, registered trademarks, or service marks mentioned herein are the property of their respective owners.

Corporate Headquarters

Wind River Systems, Inc.
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.

toll free (U.S.): 800/545-WIND
telephone: 510/748-4100
facsimile: 510/749-2010

For additional contact information, please visit the Wind River URL:

<http://www.windriver.com>

For information on how to contact Customer Support, please visit the following URL:

<http://www.windriver.com/support>

Contents

1	Introduction	1
2	General Information	2
2.1	New C++ Compiler Front End and Libraries	2
2.2	Compatibility with Earlier Versions	2
2.3	C++ Library Information	4
2.4	Installation and Licensing	5
2.5	Online Documentation	5
2.6	FastJ Availability	5
2.7	Tornado Integration	5
2.8	HP-UX and Linux Installation	5
3	Additions	7
3.1	Diab C Additions	7
3.1.1	Diab C ARM Additions	9
3.1.2	Diab C MIPS Additions	10
3.1.3	Diab C PowerPC Additions	10

3.1.4	Diab C SH Additions	11
3.2	Diab C++ Additions	11
3.3	Diab Assembler Additions	13
3.3.1	Diab Assembler ARM Additions	13
3.3.2	Diab Assembler MIPS Additions	13
3.4	Diab Linker Additions	14
3.5	Library Additions	14
3.6	WindISS (RTAsim) Additions	15
4	Corrections	16
4.1	Diab C Corrections	16
4.1.1	Diab C ColdFire Corrections	18
4.1.2	Diab C M•CORE Corrections	19
4.1.3	Diab C MIPS Corrections	19
4.1.4	Diab C PowerPC Corrections	19
4.1.5	Diab C SH Corrections	21
4.2	Diab C++ Corrections	21
4.2.1	Diab C++ PowerPC Corrections	21
4.3	Diab Assembler Corrections	22
4.3.1	Diab Assembler PowerPC Corrections	22
4.4	D-DUMP Corrections	22
5	Documentation Errata	23
6	Known Problems and Workarounds	24
6.1	Installation	24

6.2	Diab C	24
6.3	Diab Linker	25
7	Customer Services	25
8	Acknowledgments	26
9	Previous Release Notes	27

Diab Compiler Suite

Release Notes

5.0.1

1. Introduction

This document describes recent additions and corrections to the Wind River Diab C/C++ and FastJ compiler suites.

Sections 3 and 4 present additions and corrections, respectively, grouped by version; for pre-5.0 releases, see 9. *Previous Release Notes*, p.27. All products in a release have the same version number, even when a particular product is unchanged. Changes are further organized by language and target:

- Changes marked with a language—Diab C, Diab C++, or FastJ—apply to all targets unless otherwise noted.
- Changes marked “Diab C” also apply to Diab C++ and, if relevant, to FastJ, unless otherwise noted.
- Changes marked with a target name, such as ARM or PowerPC, apply to all languages unless otherwise noted.

2. General Information

2.1 New C++ Compiler Front End and Libraries

Diab 5.0 features a new C++ front end from Edison Design Group (EDG) and new C++ libraries from Dinkumware. With the new front end and libraries, Diab offers improved compliance with the C++ standard, full support for templates and namespaces, more efficient exception handling, and support for precompiled headers. For more information, see 2.2 *Compatibility with Earlier Versions*, p.2, 2.3 *C++ Library Information*, p.4, 3.2 *Diab C++ Additions*, p.11, and chapters 12 and 13 of the *Diab C/C++ Compiler User's Guide*.

2.2 Compatibility with Earlier Versions

Because the new compiler adheres more strictly to the C++ standard, some code written for earlier versions of Diab C++ may not compile. See chapters 12 and 13 of the *Diab C/C++ Compiler User's Guide* for more information. In addition, some options, directives, and nonstandard keywords have changed or are no longer supported. The compiler issues a warning for obsolete or unimplemented options.

For the most part, these changes apply to Diab C++ only. *C programmers should notice few problems compiling existing code.* However, if you are compiling C or C++ code for a MIPS or SH target, see 3.1.2 *Diab C MIPS Additions*, p.10 or 3.1.4 *Diab C SH Additions*, p.11 for information about backward compatibility.

The new C++ compiler is called **etoa** (instead of **dtoa**), but the driver is still called **dplus**. The driver automatically invokes the new compiler and links to the new libraries. C++ programmers who want to continue using the old front end and libraries can do so by specifying the **-Xc++-old** command-line option. For details, see the *Diab C/C++ Compiler User's Guide: 12. Converting Existing Code*.

The following keywords are not supported by the new C++ compiler:

extended
interrupt, **__interrupt__**
pascal

The following special DSP extensions for the SH target, and the intrinsic functions that use them, are not supported by the new C++ compiler:

`__accum`
`__fixed`
`__X`
`__Y`

The following options are not supported by the new C++ compiler:

`-I@` (but `-I` is still supported)
`-Xold-function-decls-...`
`-Xpostfix-inc-dec-...`
`-Xvtbl-...`
`-Xcall-MAIN`
`-Xshow-inst`
`-Ximplicit-templates-...`
`-Xclass-type-name-visible`
`-Xstruct-arg-warning`
`-Xbottom-up-init`
`-Xcpp-no-space`
`-Xswap-cr-nl`
`-Xunsigned-bitfields, -Xbit-fields-unsigned`
`-Xsigned-bitfields, -Xbit-fields-signed`
`-Xbitfield-compress, -Xbit-fields-compress`
`-Xdouble-...`

The following options behave differently with the new C++ compiler:

- `-Xkeywords` can enable only nonstandard keywords supported by the C++ compiler. The masks 0x01 (**extended**), 0x02 (**pascal**), and 0x10 (**interrupt**) are no longer valid, nor are the masks for the SH processor's DSP extensions. 0x04 (**inline**) is invalid, since **inline** is always enabled in C++.
- `-Xlint` is equivalent to `-Xsyntax-warning-on`.
- `-Xenum-is...` has only two forms: `-Xenum-is-int` and `-Xenum-is-best`. All others (such as `-Xenum-is-unsigned`) are invalid.

Other documented options are still supported. Check the *User's Guide* if you are in doubt about the status of an option.

The `#assert` and `#unassert` directives are supported, but with a slightly different syntax from the old compiler; see the *User's Guide* for details. The `#ident` directive is not supported.

The C++ compiler no longer supports direct assembler functions. String statements and **asm** macros are still supported.

When the `__ERROR__()` function is used in C++ code, it must be declared as follows:

```
extern "C" void __ERROR__(char *, ...);
```

(For more information, see the *Diab C/C++ Compiler User's Guide: 6. Additions to ANSI C and C++.*)

For information about other changes, see 3.2 *Diab C++ Additions*, p.11.

2.3 C++ Library Information

Full documentation for the Dinkum C++ libraries (including STL) can be found at <http://www.dinkumware.com/refxcpp.html>.



NOTE: The Dinkumware documentation should be consulted for C++ library functions only. The standard C library shipped with Diab 5.0 remains unchanged and is documented in the *Diab C/C++ Compiler User's Guide*.

Some header files have new names. For example, **exception.h** replaces **except.h**.

Exception-handling must be enabled (**-Xexceptions**, which is now the default) for projects that use the Standard Template Library.

Currently, the C++ libraries do not support locales, wide or multibyte characters, or the **long double** type. The following wide-character functions are not supported:

```
btowc, fgetwc, fgetws, fputwc, fputws, fwide, fwprintf, fwscanf,  
getwc, getwchar,  
mbrlen, mbrtowc, mbsrtowcs, mbsinit,  
putwc, putwchar,  
swprintf, swscanf,  
ungetwc,  
vwfprintf, vswprintf, vwprintf, wctomb, wprintf, wscanf,  
wcsrtombs, wcstol, wscat, wcschr, wscmp, wscoll,  
wcscpy, wcsncpy, wcslen, wcsncat, wcsncmp, wcsncpy,  
wcpbrk, wcschr, wcsncpy, wcstod, wcstoul, wcsstr,  
wcstok, wcsxfrm, wctob,  
wmemchr, wmemcmp, wmemcpy, wmemmove, wmemset, wcsftime
```

2.4 Installation and Licensing

The installation utility and license-management interface have been substantially changed for the Diab 5.0 release. See your *Getting Started* manual for complete installation instructions. This release does not include a separate manual for the FLEXlm license manager.

HP-UX and Linux users should see 2.8 *HP-UX and Linux Installation*, p.5.

2.5 Online Documentation

Users of the standalone Diab tools can access online documentation, in HTML and PDF, directly from the product CD or installation directory. To read the documentation, open the file **welcome.htm** (in a Web browser) or **welcome.pdf** (in Adobe Acrobat Reader). If your Diab tools are bundled with another product (such as Tornado), see the main help screen for that product.

The online help for the RTA Suite includes the full Diab C/C++ and FastJ documentation *from release 4.4*, with content based mostly on release 4.3. It does not reflect changes to the product in the current release.

2.6 FastJ Availability

The FastJ compiler suite is currently available for PowerPC targets only.

2.7 Tornado Integration

Diab 5.0.1 supports full integration with the Tornado 2.2 IDE. It is not compatible with earlier versions of Tornado. See the Tornado 2.2 documentation for more information.

2.8 HP-UX and Linux Installation

HP-UX and Linux hosts use a command-line installation utility that differs significantly from the graphical installer described in the *Getting Started* manual. Installation instructions for these hosts are given below.

1. Consult your system administrator for instructions on accessing the CD-ROM drive. It may be necessary to mount the drive, which requires root privileges.



NOTE: The CD drive must be mounted so that all file names are mapped from their upper-case form on the CD to lower-case for use by the file system.

Example **mount** commands:

```
HP-UX: % mount -r -F cdfs -o cdcase /dev/dsk/c0t5d0 /cdrom
```

```
Linux: % mount -t iso9660 -r /dev/cdrom /mnt/cdrom
```

The device name (`/dev/dsk/c0t5d0` or `/dev/cdrom` in the example) depends on your machine configuration.

2. Go to the root directory of the CD-ROM drive. To verify that you are in the right directory, enter `ls`; you should see the **install.sh** file. Then enter:

```
% install.sh [-d destination-path] [-s source-path] [-m html|pdf|all] [-l]
```

where

- *destination-path* is the target directory for installation of the Diab tools. This is the *install_path* described in the *Getting Started* manual and defaults to **/usr/lib/diab**.
 - *source-path* is the directory on the CD from which to install. It defaults to the current working directory (that is, the top level of the CD).
 - **-m** tells the installer to copy the documentation (in HTML, PDF, or both) to the hard drive.
 - **-l** installs the library sources.
3. The installer will ask you to select a language (C++ includes C), one or more targets, and a host on which to run the tools. (You can later install additional languages, targets, or network hosts by running **install.sh** again.) The installer will ask you to verify your selections, then begin copying files.
 4. The installer will run **dctrl -t** to select the default target. For details, see *Diab C/C++ Compiler Getting Started: 2.3.2 Selecting the target*.

5. Automatic license installation is not available for HP-UX or Linux hosts. You will need to obtain a FLEXlm license file and install it manually. Please read the material on license administration in the *Getting Started* manual.
6. For additional set-up information, see *Diab C/C++ Compiler Getting Started: 2.3 Preparing to use the tools*, especially 2.3.1 *Adjusting your path* and 2.3.3 *Setting environment variables*.

3. Additions

3.1 Diab C Additions

- **Option `-Xdebug-local-cie`: generate local CIE for each unit (5.0.1)**

The new option `-Xdebug-local-cie` causes the compiler to generate a local Common Information Entry (CIE) for each unit. This is applicable only with DWARF 2 or DWARF 3 debug information.

- **New DWARF debug options (5.0a)**

The compiler can now generate DWARF 2 or DWARF 3 debug information, enabled with `-Xdebug-dwarf2` or `-Xdebug-dwarf3`. DWARF 1.1 (`-Xdebug-dwarf1`) is still the default, but this may change in a future release.

- **Change in DWARF 1 debug info (5.0a)**

The compiler now generates `.d1_line_start` and `.d1_line_end` assembler directives to mark the start and end of a `.line` debug section. The `.d1line` and `.d1file` directives are generated even when the `-Xsection-split` option is used. (This enables `reorder` to perform all optimizations). Generation of `.d1_line_start` and `.d1_line_end` assembler directives can be disabled using `-Xdebug-mode=0x40`.

- **New command-line help options (5.0a)**

The following command-line flags display information about compiler options:

Option	Shows ...
-?	frequently used compiler options
-??	less frequently used compiler options
-?X	compiler -X options
-?W	compiler -W options

- **Option -Xvoid-ptr-arith-ok: treat void pointers as char * (5.0a)**

The new option **-Xvoid-ptr-arith-ok** tells the compiler to treat void pointers as pointers to a **char** for the purpose of arithmetic.

- **Casts of scalars to struct and union (5.0a)**

The compiler now allows explicit casts of scalar types to structure or union types, if the size of the scalar equals the size of the structure or union. The compiler issues a warning on these casts, and they are not allowed in strict-ANSI mode (**-Xdialect-strict-ansi**).

- **Option -Xincfile-missing-ignore: ignore missing include files (5.0a)**

-Xincfile-missing-ignore is effective only when used with **-Xmake-dependency**. It tells the compiler to continue preprocessing even when required files are missing. If it is used with **-Xmake-dependency=2** or **-Xmake-dependency=6**, the preprocessor issues a warning when a required system file (**#include <filename>**) is not found.

- **Generation of make rules to file (-Xmake-dependency=5, -Xmake-dependency=6) (5.0a)**

The compiler can now emit a list of header-file dependencies in a separate file. **-Xmake-dependency=5** produces the same result as **-Xmake-dependency=1**, except that the output is written to a file (*mainSourceFile.d*) instead of to the standard output, and normal compilation then continues. Similarly, **-Xmake-dependency=6** produces the same result as **-Xmake-dependency=2**, except that the output is written to a file and normal compilation continues.

- **Option `-Xmacro-undefined-warn`: warn on undefined macro in `#if` directive (5.0a)**

The option `-Xmacro-undefined-warn` causes the preprocessor to emit a warning when an undefined macro occurs in an `#if` directive.
- **Option `-Xstderr-fully-buffered` (5.0a)**

`-Xstderr-fully-buffered` buffers `stderr` using a 10kb buffer. This option can reduce network traffic; `stderr` is unbuffered by default.
- **Improved optimization of conditional structure assignments (5.0a)**

When `-Xsize-opt` is enabled, the compiler generates more efficient code for conditional structure assignments.
- **`alloca()` and `__alloca()` changed to intrinsic functions (5.0a)**

`alloca()` and `__alloca()` are now handled as intrinsic functions. `__alloca()` cannot be disabled. `alloca()` can be disabled with `-Xintrinsic-mask=0x800000`. Options related to the old `alloca()` library routine (such as `-Xalloca-inline` and `-Xalloca-preserve-call`) are no longer valid.
- **New option `-Xdebug-align` (4.4c)**

The option `-Xdebug-align[=n]` tells the compiler to align `.debug` sections on specified boundaries, where *n* is a power of 2. For example, `-Xdebug-align=3` aligns `.debug` sections on 8-byte boundaries. If `=n` is omitted, alignment defaults to 4-byte boundaries.

Without this option, `.debug` sections are aligned on byte boundaries (as specified by DWARF 1.1).

3.1.1 Diab C ARM Additions

- **Improved ARM optimizations (5.0a)**

The ARM compiler now performs conditional transformations and other new optimizations.

3.1.2 Diab C MIPS Additions

- **Change in MIPS64 register use (5.0a)**

For MIPS64, the compiler uses different registers to pass floating-point arguments. All source code should be recompiled for MIPS64 targets, since existing object files may not link correctly. See the chapter on “Calling Conventions” in the *Diab C/C++ Compiler for MIPS User’s Guide* for more information on argument passing.

- **New MIPS target options (5.0a)**

The object format for a MIPS target can now be specified as **M**, **W**, **X**, **Y**, or **Z**, in addition to the older options **E**, **F**, and **L**. The new choices allow you to select an object format and set the **-Xsmall-data** and **-Xsmall-const** options with a single command-line abbreviation. For details, see the *Diab C/C++ Compiler for MIPS User’s Guide: 4. Selecting a Target and Its Components*.

- **Changes to MIPS64 options (5.0a)**

The compiler no longer supports the **-tMIPS-64** option (MIPS32 in 64-bit mode). **-tMIPS64** is still supported.

- **Change to char type (5.0a)**

The plain **char** type is now signed. To revert to the old behavior (plain **char** unsigned), recompile *all* modules, including supplied libraries, with **-Xchar-unsigned**.

3.1.3 Diab C PowerPC Additions

- **Additional PowerPC target options (5.0a)**

The compiler supports the following new options to the **-t** switch: **PPC857**, **PPC862**, **PPC7410**, **PPC7440**, **PPC7450**, **PPC7455**, **PPC8241**, **PPC8245**, **PPC8250**, **PPC8255**, **PPC8264**, **PPC8265**, and **PPC8266**.

- **Improved AltiVec scheduling support (5.0a)**

The compiler has improved scheduling support for PowerPC 7410 and 7450 targets.

- **Changes to abs and labs (5.0a)**

For PowerPC targets, **abs** and **labs** are no longer intrinsic functions and cannot be disabled with **-Xintrinsic-mask**. Instead, **abs** and **labs** are treated as standard library functions and can be disabled with **-Xclib-optim-off**.

3.1.4 Diab C SH Additions

- **Change in register use for argument passing (5.0.1)**

In little-endian mode, single precision floating point arguments are passed in registers in the following order: **r5, r4, r7, r6, r9, r8, r11, r10**. This change was made for compatibility with the GNU toolchain on SH4 targets.

- **Change in algorithm for passing arguments (5.0a)**

The compiler uses a different algorithm to pass arguments. All source code should be recompiled for SH targets, since existing object files may not link correctly. See the *Diab C/C++ Compiler for SH User's Guide: 9. Calling Conventions* for more information on argument passing.

3.2 Diab C++ Additions

Diab 5.0 includes a new C++ compiler front end and new C++ libraries. In addition to the notes below, please see 2. *General Information*, p.2 for important details about these changes.

- **Exceptions and RTTI are enabled by default (5.0a)**

By default, exception handling and run-time type information are enabled. To disable exceptions, use **-Xexceptions-off**. To disable RTTI, use **-Xrtti-off**.



NOTE: Exceptions must be enabled for projects that use the STL.

- **COMDAT is the default for template processing (5.0a)**

The **-Xcomdat** option is enabled by default.

- **Precompiled headers (5.0a)**

To speed up compilation of projects with many header files, the C++ compiler now supports precompiled headers. Precompiled headers are enabled with the **-Xpch-...** options. For details, see the *Diab C/C++ Compiler User's Guide: 13. C++ Features and Compatibility*.

- **Control namespace for standard libraries (5.0a)**

The option **-Xusing-std-on**, enabled by default, causes the compiler to search for run-time library declarations in the **std** namespace, as if **using namespace std;** had been specified in the source code. This option allows you to use the newer C++ libraries, which are in the **std** namespace, without modifying legacy code. To search for run-time library declarations in global scope (unless an explicit **using namespace std;** is given), specify **-Xusing-std-off**.

- **Column info with DWARF 2 and 3 debug output (5.0a)**

The C++ compiler now emits column information, when available, in DWARF 2 and DWARF 3 debugging sections. This can be disabled with **-Xdebug-mode=0x100**.

- **New value for __cplusplus (5.0a)**

The **__cplusplus** preprocessor macro is now defined as 199711 (instead of 1).

- **New predefined macros (5.0a)**

The following preprocessor macros are now predefined:

Macro	Indicates ...
__ETOA__	full ANSI C++ support
__ETOA_IMPLICIT_USING_STD	library declarations searched for in std namespace
__ETOA_NAMESPACES	run-time library uses namespaces
__EXCEPTIONS	exceptions enabled
__RTTI	run-time type information enabled
__SIGNED_CHARS__	plain char is signed

For more information, see the *Diab C/C++ Compiler User's Guide: 6. Additions to ANSI C and C++*.

- **New underlying type for `wchar_t` (5.0a)**

The underlying type for `wchar_t` is changed from **long** to **unsigned short**.

- **New default inlining mode (5.0a)**

The default inlining mode for Diab C++ has changed to **-Xexplicit-inline-factor=3** and **-Xinline=10**. Previously, all C++ (explicit or implicit) inline functions were inlined. **-XO** still sets **-Xinline=40** by default.

The old behavior can be restored by specifying **-Xexplicit-inline-factor=0** and (if you are not using the **-XO** option) **-Xinline=1**.

3.3 Diab Assembler Additions

- **File extension for assembler listings (5.0a)**

The **-l** and **-Xlist-file** assembler options now generate a listing to *input-file.lst* by default. Previously, the output had been saved in *input-file.L*. The new assembler option **-Xlist-file-extension=string** changes the extension of the listing file; for example, **-Xlist-file-extension=".L"** restores the old behavior.

3.3.1 Diab Assembler ARM Additions

- **.type directive supports `@tfunc` for Thumb functions (5.0a)**

With the **.type symbol** assembler directive, *symbol* can now be **@tfunc** (in addition to the existing **@function** option). Use **@tfunc** for Thumb functions.

3.3.2 Diab Assembler MIPS Additions

- **Support for TX49 targets (5.0.1)**

The new assembler option **-Xcpu-TX49xx** enables support for the TX49 processor.

- **NOP instructions inserted to fill delay slots (5.0a)**

By default, the assembler now automatically inserts a NOP instruction to fill any delay slot (**-Xset-reorder**, **.set reorder**). To suppress this feature and restore the old default behavior, use **-Xset-noreorder** (equivalent to **.set noreorder**). If **.set reorder** or **.set noreorder** appears in the assembly source, the **.set** directive takes precedence over the **-Xset-...** option.

These options affect hand-coded assembler files only. They do not affect compiler-generated code or embedded assembly code that is processed by the compiler.

3.4 Diab Linker Additions

- **Linker ASSERT command (5.0a)**

ASSERT(*expression*, *message*) can be placed in a linker command file. It causes the linker to fail (with output *message*) if *expression* is false.

- **Extensions to -A option (5.0a)**

The **-A** linker option now supports the following variations:

- **-A1** is the same as **-A**. It links all files from a specified archive.
- **-A2** is the same as **-A1**, except that it overrides **-Xremove-unused-sections** for the specified archive files (but allows **-s** or **-ss** to remove debug info sections).
- **-A3** is the same as **-A2**, except that it also overrides **-s** and **-ss** for the specified archive files.

3.5 Library Additions

- **New standard C++ libraries (5.0a)**

Diab 5.0 includes new standard C++ libraries. See 2.3 *C++ Library Information*, p.4.

- **abs and labs (5.0a)**

For PowerPC targets, **abs** and **labs** are treated as standard library functions instead of intrinsic functions. They can be disabled with **-Xclib-optim-off**.

3.6 WindISS (RTAsim) Additions

- **RTAsim is now called WindISS (5.0a)**

The **RTAsim** simulator is now called **WindISS** (the Wind River Instruction Set Simulator). Names of executables and other files have been changed as follows:

Old Name	New Name
RTAsim	WindISS
rtasim	windiss (executable, directory, command-line option)
librtasim.a	libwindiss.a
rtasim.conf	windiss.conf
RTAsim.cfg	WindISS.cfg

On the command line, use **-t...:windiss**. For example:

```
% dplus -tPPCES:windiss file.cpp
```

For compatibility, copies of the executable, libraries, and configuration file have been maintained with the old names **rtasim**, **librtasim.a**, and **rtasim.conf**.

- **Support for additional exceptions (5.0a)**

Additional exceptions are supported by the **-X exception-mask** option for 68K, M•CORE, MIPS, and PowerPC targets. See the *Diab C/C++ Compiler User's Guide* for details.

- **New log/trace mode (5.0a)**

The **-Di** option now supports the syntax **-Di symbol,0** or **-Di trigger-address,stop-address,0**. If the final argument (the *trace count*) is 0, tracing is enabled as long as the PC is within the specified function or range; when the PC is outside of range (for example, when executing a subroutine), tracing is disabled.

4. Corrections

This section documents significant problems that have been *fixed* in the current release. Workarounds are shown where available.

4.1 Diab C Corrections

- **Problem with -Xpass-source option (fixed in 5.0a)**

The compiler sometimes generated different code when the **-Xpass-source** option was used. The problem was caused by the branch-to-adjacent-label optimization, which was generating an additional unconditional branch.

Tracking number: 66770

- **Problem with nested double quotes in options file (fixed in 5.0a)**

The driver did not handle nested double quotation marks when they were included in an options file. (An options file can be specified on the driver command line with *-@fileName*).

Tracking number: 67218

Workaround: Specify option on command line instead of options file.

- **Problem with register allocation when passing structures to functions (fixed in 5.0a)**

The compiler sometimes incorrectly allocated a scratch (temporary) register to a local variable. This occurred when passing structures as parameters, and only when the structure was large enough to be passed partially in registers and partially on the stack.

Introduced: 4.0

- **Abnormal compiler termination (fixed in 5.0a)**

The C compiler (**ctoa**) sometimes terminated abnormally when invoked without the **dcc** driver and without configuration files.

Introduced: 4.4b

Tracking number: 68860

Workaround: Use the driver (**dcc**) or use the **-Y** option to specify a default search directory for header files.

- **Infinite loop caused by asm macro with pointer to typedef (fixed in 5.0a)**

The compiler went into an infinite loop when parsing an **asm** macro that contains a parameter that is a pointer to a user-defined type.

Introduced: 4.2b?

Tracking number: 67867

Workaround: Use a void pointer instead of a pointer to a user-defined type.

- **Compiler failure on preprocessor syntax error (fixed in 5.0a)**

Under Windows, the compiler terminated abnormally when EOF was encountered before an expected **#endif** directive. Under Solaris, the compiler did not terminate, but it failed to generate the appropriate error message.

Introduced: 4.4a

Tracking number: 69171

Workaround: Use **-Xpreprocessor-old**, or fix the mistake by adding a matching **#endif** directive.

- **Reorder optimization removed initialization of local variable (fixed in 5.0a)**

When an absolute memory address was cast as a pointer and dereferenced, the compiler generated an unrecognized instruction. The instructions that followed sometimes used preserved registers whose values were not set.

Introduced: 4.4a

Tracking number: 68130

Workaround: Use **-Xkill-reorder=0x8** or **-Xkill-opt=0x40000**.

- **Problem with -C option and asterisks in comments (fixed in 5.0a)**

The **-C** option did not pass comments properly when asterisks (*) occurred within comments. For example, the comment **/**test**/** was not passed correctly.

Introduced: 4.4a

- **Problem with %lab in asm macros (fixed in 5.0a)**

When a comment precedes a **%lab** directive in an **asm** macro that does not take an argument, the compiler did not expand the macro and its instructions were not inserted into the function that called it.

Introduced: 3.7

Tracking number: 69727

Workaround: Do not use comments before **%lab** when a macro takes no arguments.

- **“non-existent block in do_switch_table_arcs” error (fixed in 5.0a)**

The **reorder** program was changing some temporary labels in **switch** tables from **.Lnumber** to **.LMnumber**, but did not know how to parse the **.LM** labels. This caused the compiler to fail with a “non-existent block in do_switch_table_arcs” error.

Tracking number: 271027

Workaround: Use **-Xkill-reorder=0x8**.

- **Optimization incorrectly marked code as invariant (fixed in 5.0a)**

During loop strength reduction, a basic block was incorrectly marked as a dominant basic block. The compiler assumed that this basic block would be executed for every iteration, and therefore moved the corresponding code out.

Tracking number: 271720

Workaround: Use **-Xkill-opt=0x2000**.

- **Preprocessor expression evaluation failure (fixed in 5.0a)**

In some cases, preprocessor expression evaluation failed because the signedness of an expression lingered and was applied to subsequent evaluations in conditionals.

Introduced: 4.4a

Tracking number: 75035

Workaround: Use **-Xpreprocessor-old**.

- **Problem with #ifndef followed by #define (fixed in 5.0a)**

When a **#define** macro immediately followed an **#ifndef**, subsequent preprocessor directives in the file were not handled correctly. Specifically, subsequent **#else**, **#elif**, or **#elseif** directives were ignored.

Introduced: 4.4a

Tracking number: 74240

Workaround: Indent **#define**.

4.1.1 Diab C ColdFire Corrections

- **Problem with short and char in switch statements (fixed in 5.0a)**

The compiler generated incorrect comparison code when the expression in a **switch** statement was a **short** or **char**.

4.1.2 Diab C M-CORE Corrections

- **Wrong offset for subi instruction (fixed in 5.0a)**

Because of a bug in the stack compression algorithm, the compiler generated **subi** instructions with an incorrect offset that resulted in “constant out of range” assembler errors.

Tracking number: 71294

- **Error copying structures during optimization (fixed in 4.4b)**

The compiler was caching structure fields in registers, then reaccessing the cached values even after the structure had been overwritten.

Introduced: 4.3g?

Tracking number: 63285

4.1.3 Diab C MIPS Corrections

- **Far-absolute code addressing generated wrong MIPS16 instruction (fixed in 5.0a)**

For MIPS16 targets, **-Xaddr-code=0x11** generated **jal** instead of **jalr**.

Tracking number: 69007

4.1.4 Diab C PowerPC Corrections

- **Alignment error caused by switch statements (fixed in 5.0.1)**

A bug in the way the compiler handled **switch** statements sometimes caused alignment errors that prevented programs from loading.

Tracking number: 73421

- **Problem restoring FPU registers (fixed in 5.0a)**

Restoring the hardware FPU registers caused a change in program flow. This occurred only when an argument list contained four or more floating-point operands.

Tracking number: 68447

Workaround: Use **-g** or **-XO**.

- **rlwinm generated with empty operand after macro expansion (fixed in 5.0a)**

When expanding preprocessor macros that contain complex expressions, the compiler generated an **rlwinm** instruction with an empty operand. This occurred when an operand in the macro contained an expression with a unary ~ operator.

Tracking number: 68649, 243895

Workaround: Use a separate variable to hold the expression with the ~ operator, then pass this variable to the macro.

- **Problem casting signed short return value to unsigned short (fixed in 5.0a)**

When a function return value of type **signed short** was cast to **unsigned short** and assigned to a variable of type **unsigned short**, the cast was lost. The **reorder** program was removing the **rlwinm** instruction that truncated the result to 16 bits.

Tracking number: 271874

Workaround: Use **-Xkill-reorder=0x8**.

- **Problem with AltiVec instruction scheduling (fixed in 5.0a)**

The instruction scheduler did not correctly mark the dependence on memory for instructions that update the stack pointer, thus allowing instructions in the epilogue to be moved. When external interrupts hit in this window the floating point registers could be scrambled.

Introduced: 4.4a

Tracking number: 67726

Workaround: Disable basic reordering (**-Xkill-reorder=0x01**).

- **Incorrect AltiVec stack frame adjustment (fixed in 5.0a)**

For AltiVec targets with Vector floating-point support (PPC74xx, V floating-point option), the compiler was adjusting the stack frame to a by-16 boundary even when the vector registers were not used. This resulted in generation of incorrect references to parameters passed on the stack.

Introduced: 4.4a

Tracking number: 70964, 244909

Workaround: If the vector registers not used, compile for another PowerPC family member.

- **Optimization removed register settings (fixed in 5.0a)**

When **asm** macros that set registers **r2** and **r13** were invoked in functions, the **reorder** program sometimes removed the settings. This occurred when the function had code that triggered a tail-merging optimization.

Tracking number: 73740

Workaround: Use **-Xkill-reorder=0x8**.

4.1.5 Diab C SH Corrections

- **Problem with conditional branch instructions, optimized loops (fixed in 5.0.1)**

The compiler was generating a load instruction in the delay slot of a conditional branch; the instruction was executed even if the branch was not taken. This caused the compiler to generate optimized loop codes that read addresses once more than the expected number of times, sometimes resulting in an invalid address access.

Tracking number: 77340

4.2 Diab C++ Corrections

In addition to the items listed below, numerous problems have been corrected because of the new C++ front end. For example, the compiler now catches illegal references to protected and private class members.

4.2.1 Diab C++ PowerPC Corrections

- **Problem with casts of vector constant data (fixed in 5.0a)**

The compiler failed when a vector constant was cast to **unsigned int** and then recast to **unsigned char**.

Introduced: 4.0

Tracking number: 68836

Workaround: Use a temporary variable to hold the value from the first cast and then cast the temporary variable before assigning.

4.3 Diab Assembler Corrections

4.3.1 Diab Assembler PowerPC Corrections

- **Incorrect hid2 register definition for PPC755 (fixed in 5.0a)**

For the PowerPC 755, the assembler was assigning 1010 to the **hid2** register, instead of 1011.

Tracking number: 221335

Workaround: Add **HID2_ .equ 1011** to beginning of the source file, then use **HID2_** instead of **hid2**.

- **Missing PPC405 instructions (fixed in 5.0a)**

The assembler did not recognize the following PPC405 instructions: **tlbre**, **tlbwe**, **tlbsx[.]**, **rfdi**, **wrteei**, **dccci**, **icbt**, **iccci**, **icread**, and **dcread**. Also, the extended mnemonics for **mtspr** and **mfspir** were not recognized.

Introduced: 4.4a

Workaround: Use **-Xcpu-all**. Instead of extended mnemonics, use **mtspr** and **mfspir** with the corresponding SPR numbers (since SPR numbers differ among PPC variants and the assembler could pick the wrong one).

- **Problems with dss, dst, and dstst instructions (fixed in 5.0a)**

Some AltiVec **dss**, **dst**, and **dstst** instructions were not accepted by the assembler.

Tracking number: 71876

Introduced: 4.4a

4.4 D-DUMP Corrections

- **Problem with formatting of output from -Rp option (fixed in 5.0a)**

The **-Rp** option was formatting output incorrectly.

Tracking number: 74542

Introduced: 4.3g

5. Documentation Errata

- *Diab C/C++ Compiler User's Guide and Getting Started*, all targets:
 - The documentation incorrectly states that the keywords `__attribute__`, `packed`, and `__packed__` are not supported by the C++ compiler. These keywords are supported in both C and C++.
 - The documentation incorrectly suggests that the option `-Xkeywords=mask` is valid only for the C compiler. Certain masks—those that represent keywords not supported by the C++ compiler—are recognized only by `ctoa`, but other masks are valid for both C and C++. See 2.2 *Compatibility with Earlier Versions*, p.2 for details.
 - The documentation incorrectly states that the directive `#pragma interrupt` is not supported by the C++ compiler. This directive is supported in both C and C++.
 - The `-l` and `-Xlist-file` assembler options now generate a listing to `input-file.lst` by default. The documentation still shows the old `.L` extension for this file.
- *Diab C/C++ Compiler Getting Started*, all targets:
 - Chapter 2 (“Installation”) of the *Getting Started* manual, under the heading “Starting and stopping the license server,” incorrectly refers to a FLEXlm license manager icon in the Windows control panel. This icon is no longer installed. To control the license server, run the `lmttools.exe` program in the `bin` subdirectory of the license-server installation.
- *Diab C/C++ Compiler User's Guide*, ARM target:
 - In chapter 5 (“Invoking the Compiler”), the table listing functional groups of `-X` options is missing from section 5.4.2.
- *Diab C/C++ Compiler User's Guide*, MIPS and SH targets:
 - Some page numbers in the HTML versions of these manuals do not match the page numbers in the printed books and PDF. This affects references to pages following chapter 31 (page 448) in the *Diab C/C++ Compiler for MIPS User's Guide* and references to pages following chapter 17 (page 288) in the *Diab C/C++ Compiler for SH User's Guide*.
- *Diab C/C++ Compiler User's Guide*, SH target:

- In chapter 9 (“Calling Conventions”), under the heading “Argument passing,” step 1 of the algorithm should read “Set ... **FREG=4** ...” (instead of “Set ... **FREG=5** ...”).

6. Known Problems and Workarounds

6.1 Installation

- **Antivirus programs cause delay (Windows only)**

If some antivirus programs are enabled, the installation may appear to hang after the initial Setup dialog reaches 99%. Setup will continue after a delay of up to several minutes.

Workaround: Disable virus protection before beginning the installation. Note that e-mail programs such as Microsoft Outlook, or other programs that directly or indirectly use antivirus protection, will also cause delay.

6.2 Diab C

- **Compiler incorrectly type-extends arguments (68K targets)**

For 68K targets, when a character or short integer is passed to a function, the argument is normally extended to 32 bits. But if the function prototype is present and the parameter is explicitly declared in the prototype as a character or short integer, this extension should not occur. In cases where the prototype is available but appears in a different file from the function call, however, the compiler (incorrectly) performs the extension anyway.

- **-Xpreprocessor-old required with -Xk-and-r**

If you use the K&R standard with the option **-Xk-and-r**, the option **-Xpreprocessor-old** is also required.

- **Problem with multiple -# options**

If two or three **-#** options are present on the command line, the compiler acts as if the **-##** or **-###** option is present, respectively. (**-##** displays subprogram invocation

lines with arguments but does not execute any subprogram; `###` is like `##` but quotes arguments.)

- **Limitations on stack checking**

Stack checking is implemented only for 68K, ColdFire, M•CORE, and PowerPC processors.

6.3 Diab Linker

- **-t option invalid in linker command files**

The `-t` option is invalid in a linker command file. The `-t` option on the command line (to invoke the linker) selects the target as usual.

7. Customer Services

Wind River is committed to meeting the needs of its customers. As part of that commitment, Wind River provides a variety of services, including training courses, contact with customer support engineers, and a website containing the latest advisories, FAQ lists, known-problem lists, and other resources.

Customer Support

For customers holding a maintenance contract, Wind River offers direct contact with a staff of experienced software engineers. A full description of the Customer Support program is provided in the *Customer Support User's Guide* available at the following website:

<http://www.windriver.com/support/>

The *Customer Support User's Guide* describes the services that Customer Support can provide, including assistance with installation, product software, documentation, and service errors.

You can reach Customer Support by e-mail or telephone:

- Send e-mail to **support@windriver.com**.
- From North America, contact Customer Support with a toll-free voice telephone call to 1-800-872-4977 (1-800-USA-4WRS). For telephone access outside North America, see the Support website.

Customer Support contact information for specific Wind River products is available from the Support website. For information about reporting problems with Diab tools, please see the *Diab C/C++ Compiler Getting Started* guide.

WindSurf

Wind River Customer Services also provides WindSurf, an online support service available under the Support website. WindSurf offers basic services to all Wind River customers, including advisories, publications such as the *Customer Support User's Guide*, and a list of training courses and schedules. For maintenance contract holders, WindSurf provides access to additional services, including lists of known problems, patches, answers to frequently asked questions, and example code.

To find WindSurf, select "WindSurf login" from the main Wind River website or go directly to:

<http://www.windriver.com/windsurf/>

For the latest information about Diab tools, go to:

<http://web1.windriver.com/windsurf/products/embeddedDevTools/diab/>

8. Acknowledgments

This product is packaged using Info-ZIP's compression utility. The installation program uses UnZip to read zip files from the installation media. Info-ZIP's software (Zip, UnZip, and related utilities) is free and can be obtained as source code or executables from various bulletin board services and anonymous ftp sites, including <ftp.uu.net/pub/archiving/zip/>.

This distribution includes a copy of **dmake**, a make utility written by Dennis Vadura. **dmake** (and in this entire distribution, only **dmake**) is distributed under the terms of the GNU General Public License. For three years following the date of this publication, the source for dmake will be available from Wind River Systems,

Inc. Wind River hereby disclaims all copyright interest in **dmake**. **dmake** is distributed WITHOUT ANY WARRANTY, INCLUDING WITHOUT ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

9. Previous Release Notes

Release notes from previous versions of the Diab compiler suite are available on WindSurf.