

WIND RIVER

Wind River Compiler[®]

RELEASE NOTES

5.7

Copyright © 2009 Wind River Systems, Inc.

All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means without the prior written permission of Wind River Systems, Inc.

Wind River, Tornado, and VxWorks are registered trademarks of Wind River Systems, Inc. The Wind River logo is a trademark of Wind River Systems, Inc. Any third-party trademarks referenced are the property of their respective owners. For further information regarding Wind River trademarks, please see:

www.windriver.com/company/terms/trademark.html

This product may include software licensed to Wind River by third parties. Relevant notices (if any) are provided in your product installation at the following location:
installDir/product_name/3rd_party_licensor_notice.pdf.

Wind River may refer to third-party documentation by listing publications or providing links to third-party Web sites for informational purposes. Wind River accepts no responsibility for the information provided in such third-party documentation.

Corporate Headquarters

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

Toll free (U.S.A.): 800-545-WIND
Telephone: 510-748-4100
Facsimile: 510-749-2010

For additional contact information, see the Wind River Web site:

www.windriver.com

For information on how to contact Customer Support, see:

www.windriver.com/support

*Wind River Compiler
Release Notes
5.7*

16 Jan 09
Part #: DOC-16356-ND-00

Contents

1	Overview	1
1.1	Features	1
1.2	Installation and Licensing	2
1.3	Configuration	3
	Selecting a Target	3
1.4	Migration and Backward Compatibility	3
	Binary Compatibility with Previous Releases	3
1.5	Latest Release Information	3
2	Changes in This Release	4
2.1	Enhancements	4
	Architecture-Independent Compiler Enhancements	4
	Compiler Enhancements for SH and SH-2A Processors	5
	Compiler Enhancements for PowerPC-Based Processors	6
	Compiler Enhancements for Other Processors	7
	Linker Enhancements	7
	Documentation Enhancements	8
2.2	Fixed Problems	8
2.3	Deprecated Features	9
2.4	Unsupported Features	9
3	System Requirements	9
3.1	Host System Requirements	9
3.2	Target System Requirements	10
4	Usage Caveats	10
	C and C++ Compilers	10
5	Known Problems	12

6	Documentation Errata	13
	Customer Services	15

Wind River Compiler

RELEASE NOTES

5.7

1. Overview

The Wind River Compiler is a complete toolkit for embedded application development, including C and C++ compilers, assemblers, linkers, utilities, and standard libraries for a variety of target CPU architectures. The compiler version shipped with this release is 5.7.0.

1.1 Features

The Wind River Compiler provides:

- a wide range of optimizations to produce fast, compact code
- hundreds of command-line options, pragmas, preprocessor directives, and special keywords for flexible operation, precise control of output, and full utilization of architecture-specific features
- special features for device software optimization, such as
 - **-Xsize-opt**, which optimizes for code size
 - Support for user-defined linker command files
 - * Support for customized section placement via **#pragma section** or **__attribute__((section))**
 - Attributes to control alignment and packing of data structures
- compile-time checking to catch suspicious or non-portable constructs
- profiling to find bottlenecks in code and identify problems that cannot be detected by static analysis
- full ANSI C++ support, including templates, exceptions, and run-time type information
- integration with other Wind River tools and products

The Wind River Compiler includes the following tools:

- optimizing C and C++ compilers
- assembler
- linker

- standard C and C++ libraries
- archiver/librarian
- profiling block counter
- file dumper
- make utility
- instruction-set simulator and disassembler

1.2 Installation and Licensing

For information on installing the Wind River Compiler and configuring your product licenses, see the Wind River Site Configuration setup guides. There are accessible from the following URL:

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

The compiler getting started guide for your architecture also contains some information on licensing.



NOTE: Make sure that you are using the licensing software (in particular, the **wrsd** daemon) that comes with the current version of the compiler. Using mismatched licensing software—for example, by installing from a machine running an older licensing daemon—may cause compilation to fail.

Special Note: Installing on Linux Hosts

Note that some Linux distributions either do not automount media, or automount with a **noexec** option that prevents execution of files found on the media.

If **./setup_linux** fails with a permissions error, you may need to remount the DVD. To do so, use the following steps:

1. Login as root by entering the command **su** in your terminal and entering the root password.

2. Enter the following command:

```
umount dvd_mount_point
```

Where *dvd_mount_point* is the location of the DVD mount. For example, a typical Red Hat or Fedora DVD mount point might be **/media/DVD-12345-67890**.

3. Then enter the following command:

```
mount /dev/cdrom dvd_mount_point
```

Where *dvd_mount_point* is the location of the DVD mount.

4. Exit root mode with the **exit** command.
5. Set your working directory to the location of the DVD mount:

```
cd dvd_mount_point
```

6. Retry the **./setup_linux** command.

1.3 Configuration

Configuration information is available in the getting started guide for the Wind River Compiler for your architecture.

The directory containing the tools must be included in your path variable. Windows users should map a drive letter to the remote directory where the tools reside and use the drive letter when setting the path.

Selecting a Target

If you want to configure a default target and execution environment, run **dctrl -t** from the command prompt. For more information about **dctrl**, see the user's guide or getting started manual.

Using **dctrl** is optional. A target and execution environment can also be specified with the **-t** command-line option when the compiler is invoked. For information about **-t**, see the *Wind River Compiler User's Guide* for your target architecture.

1.4 Migration and Backward Compatibility

General information on migrating your installation and applications for use with this product is available in the "Converting Existing Code" section of the *Wind River Compiler User's Guide* for your architecture.

Binary Compatibility with Previous Releases

When migrating a project to a new compiler release, it is best to recompile all source-code modules—especially C++ modules. If this is not possible, legacy object and archive files can usually be linked against output from the new compiler. Note, however, the following exceptions:

- All C++ modules built with pre-5.4 releases of the compiler (release 5.3.1 or earlier) must be recompiled.
- C modules that contain packed structures and were built with pre-5.3 releases of the compiler must be recompiled.
- C and C++ modules built for pre-5.6.1 SH targets must be recompiled.

1.5 Latest Release Information

The latest information on this release can be found in the Wind River Compiler area of the Wind River Online Support Web site:

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

This site includes links to topics such as known problems, fixed problems, and patches.

For documentation updates, see

documentation.windriver.com



NOTE: Wind River strongly recommends that you visit the Online Support Web site before installing or using this product. The Online Support Web site may include important software patches or other critical information regarding this release.

For information on accessing the Wind River Online Support Web site, see [Customer Services](#), p.15.

In addition, a list of known issues and other important information is installed in `installDir/readme.txt`. Be sure to review this file before using the Wind River Compiler.

2. Changes in This Release

All components in a toolchain release (compiler, assembler, linker, and utilities) have the same version number, even when a particular component is unchanged. Changes are further organized by language and target:

- Changes to the C or C++ compiler apply to all targets unless otherwise noted.
- Changes to the C compiler also apply to C++, unless otherwise noted.
- Changes marked with a target name, such as ARM or PowerPC, apply to all languages unless otherwise noted.

2.1 Enhancements

Architecture-Independent Compiler Enhancements

Cross-Module Optimization and Whole-Program Optimization

Cross-module optimization (CMO), including options such as `-Xcmo-gen` and `-Xcmo-use`, is being deprecated in favor of another form of optimization, Whole-Program Optimization. While CMO is still supported for release 5.7, it will be removed in future releases. Therefore, it should be used with restraint, if at all.

For more information on Whole-Program Optimization, see the WindRiver online support pages.

Separate Section Classes for Data and Functions

The `-Xsection-split` option has been updated to generate separate section classes for data as well as functions.

`-Xparse-size` and `-Xparse-count`

The `-Xparse-size` option, which delayed code generation of functions until a certain memory size was reached, has been replaced by `-Xparse-count`, which does much the same thing, but delays code generation until n nodes have been used for internal tables. (By delaying generation, the compiler can perform interprocedural optimizations such as inlining and register tracking. A *node* in this case is generally

equivalent to an operator or an operand.) Using nodes instead of memory size provides more flexibility for the compiler to take advantage of greater memory availability in newer computers.

The default is 300,000 nodes with `-O`, or 600,000 nodes if `-XO` is used.

New Reaching Analysis Optimization Options

Two new options govern reaching analysis optimizations.

`-Xlimit-reaching` disables reaching optimizations when a size limit is exceeded.

`-Xreduce-reaching` *reduces* reaching optimizations when a size limit *size* is exceeded.

Compiler Enhancements for SH and SH-2A Processors

Information About `fpscr.pr` Bit State (SH-2A only)

The `-Xassume-pr-one` option tells the compiler about the initial state of the `fpscr.pr` bit. This bit in the Floating Point Status Control Register determines if instructions are single-precision (0) or double-precision (1). This option should match the state of the hardware on entry to `main()`.

`-Xassume-pr-zero` is equivalent to `-Xassume-pr-one=0`.

This option has been part of pre-5.7.0 releases but is only documented now.

Support for Multiplication and Division Instructions (SH-2A only)

The compiler now supports the `DIVS` (32-bit signed division) and `DIVU` (32-bit unsigned division).

Support for `FLDIO` and `FLDI1`, and `MULR` Instructions (SH-2A only)

The compiler now supports the `FLDIO` (floating-point load 0 immediate), `FLDI1` (floating-point load 1 immediate), and `MULR` (perform a 32x32-bit multiply) instructions.

Note: You must compile using `-Xassume-pr-zero` and `-Xdouble-avoid` for `FLDIO` and `FLDI1` to be used.

Support for `RTS/N` Instruction (SH-2A only)

Support for the `RTS/N` (Return from subroutine with no delay slot) instruction has been added.

Program Counter Value with Floating Point Exceptions (SH-2A only)

A floating-point exception that occurs after a `FDIV`, `FTRC`, or `FSRT` instruction faults may cause the PC (program counter), stored in the exception, to contain an incorrect address.

The new `-Xfp-exceptions-patch` option adds an optional post-scheduling patch to avoid problems with catching an exception when `FDIV`, `FTRC`, or `FSQRT` instructions fault.

High-Speed Performance During Interrupts (SH-2A only)

The **fast_interrupt** pragma supports the use of on-chip register banks to provide high-speed register save and restore performance during interrupt processing. It does so by removing function prologues and epilogues and by using the **resbank** instruction.

Bit Manipulation for SH-2A Supported

Bit manipulation is now supported for the SH-2A processor.

Register Preservation on Interrupt

Documentation for previous releases indicated that floating-point registers were *not* saved in interrupt functions. In fact, some, but not all, registers were preserved. The compiler behavior has been changed to reflect the documentation: as of release 5.7.0.0, floating-point registers are indeed not preserved.

However, a new option, **-Xinterrupt-saves-fp**, has been added so that users may save such registers if they choose. (Unlike previous releases, if this option is enabled, *all* floating-point registers are saved.)

Compiler Enhancements for PowerPC-Based Processors

Support for Pre-Increment and Pre-Decrement Addressing (PowerPC only)

The new **-Xupdate-slow** option directs the compiler to avoid using pre-increment and pre-decrement instructions, as in some cases running such instructions can be slower than running two separate instructions. (The compiler may still generate pre-increment and pre-decrement instructions, but will do so less often.)

SPE Instructions

The **-Xspe-fpmac** compiler option allows Signal Processing Engine (SPE) multiply-accumulate instructions to be used with e500 processors.

SPE Intrinsic Functions

When targeting SPE intrinsic functions you must now include the file **spe.h**. For example, to use the intrinsic function **__ev_abs()**:

```
#include <spe.h>
. . .
. . .
x = __ev_abs(y);
```

The compiler now needs the user to include **spe.h** to get the intrinsic function declaration of **__ev_abs()** and others.

Intrinsic Assembly

There is now a new intrinsic assembly feature available for PowerPC (including VLE and E500). By including the file **diab/asm.h**, you can access some assembly language instructions with function call syntax without using **asm** strings or **asm** macros. Example:

```
#include <diab/asm.h>

int count_leading_zeros(int i)
{
```

```

        return __cntlzw(i);
    }

int read_machine_state_register()
{
    return __mfmsr();
}

```

Support for the PWRficient PA6T Microprocessor

The Wind River compiler now supports the PWRficient PA6T from P.A. Semi. It can be specified on the command line with the **-tPA6T** option.

Compiler Enhancements for Other Processors

Support for ColdFire MCFQE51x Processors

The compiler now supports the MCFQE51x processor family from ColdFire, as well as the MCF51QE32, MCF51QE64, and MCF51QE128 processors.

Support for ARMv7

The compiler now supports the ARMv7 processor

Support for MTI MIPS 34K

Support for MTI's MIPS 34K core processors has been added through the addition of the **-MIPS34Kc** and **-MIPS34Kf** target flags. (The MIPS34Kc does not include hardware floating-point support; the MIPS34Kf does.)

Linker Enhancements

-Bt, -Bd Support with Linker Command Files

The linker options **-Bt** and **-Bd**, which specify the address of **.text** and **.data** sections, now work with linker command files. Previously, these options were ignored if a linker command file was present.

The **-Bd** and **-Bt** options provide a simple way to either

- define where to allocate the sections without having to write a linker command file, or
- change the address of sections specified in a linker command file, dynamically, from a command line

If either **-Bd** or **-Bt** is specified, the linker will use the following command specification:

```
SECTIONS {
  GROUP BIND TEXTBASE : {
    .text (TEXT) : {
      *(.text) *(.rdata) *(.rodata)
      *(.init) *(.fini)
    }
    .sdata2 (TEXT) : {}
  }
  GROUP BIND DATABASE : {
    .data (DATA) : {}
    .sdata (DATA) : {}
    .sbss (BSS) : {}
    .bss (BSS) : {}
  }
}
```

where **DATABASE** and **TEXTBASE** are replaced by the values given by **-Bd=address** and **-Bt=address**, respectively.



NOTE: If you use **-Bt** or **-Bd** with a linker command file, you must include **TEXTBASE** and/or **DATABASE** in that file; otherwise the results are unpredictable.

The default linker file (**default.ldd**) does not specify **TEXTBASE** or **DATABASE**. To use **-Bt** or **-Bd** without any linker command file, suppress the use of the default linker command file by specifying the **-W m** option with no name on the **dcc** or **dplus** command line.

If the **-N** option is given, the **.data** section is placed immediately after the **.text** section.

Here the **.text** section begins at 0x10000 and the default linker command file is ignored:

```
% dcc -Wl, -Bt=0x10000 -W -m "" foo.c
```

Documentation Enhancements

Documentation Available for Online Viewing

The Wind River Compiler user's guides (all architectures), as well as the Error Messages Reference Guide, are available as both PDF and HTML. They are located at

install_Dir/docs/extensions/eclipse/plugins/com.windriver.ide.doc.wr_compiler

where *Install_Dir* represents the directory where you install the Wind River Compiler.

- If you use Wind River Workbench and would like to view the Wind River Compiler documentation via Workbench Online Help, visit the Online Support Web site for a patch.
- If you use Eclipse (from the open-source Eclipse Foundation) and would like to view the Wind River Compiler documentation via Eclipse, visit the Online Support Web site for a patch.

2.2 Fixed Problems

For a list of problems fixed in the Wind River Compiler, visit the Online Support Web site (see [1.5 Latest Release Information](#), p.3).

2.3 Deprecated Features

Cross-Module Optimization is being deprecated and should be used sparingly if at all. See [Cross-Module Optimization and Whole-Program Optimization](#), p.4.

2.4 Unsupported Features

HP-UX

The HP-UX operating system is no longer supported.

WindISS Not Supported for Certain Targets

The Wind River Instruction Set Simulator does not support simulation of MIPS64, x86, or ARM Thumb-2 targets, nor does it support ColdFire hardware floating-point instructions.

Library Support Restrictions

While the compiler supports the `wchar_t` type, in most environments the libraries do not support locales, wide- or multibyte-character functions, or the **long double** type. (Some VxWorks files may include stubs for unsupported wide-character functions.) For user-mode (RTP) VxWorks projects, the libraries support wide-character and multibyte functions.

Intel-based Solaris Systems

Although the Wind River Compiler may run on Intel-based Solaris systems, it is only supported, for Solaris, for SPARC-based machines. No functionality is promised for Intel-based Solaris computers.

3. System Requirements

This section lists the minimum requirements for running the Wind River Compiler where the host and target are separate computers.

3.1 Host System Requirements

The host is the computer on which you do your development work. This section lists the minimum requirements for running the Wind River Compiler in the standard configuration.

These system requirements are for the Wind River Compiler only; they do not take into consideration any other software you are running on the host computer.

- Windows NT 4.0 with service pack 5 or later, Windows 2000 Professional, Windows XP Professional, or Windows Vista Professional; Red Hat Linux 7.2 or later; Red Hat Enterprise Linux (RHEL) version 4 or later; Fedora 7 Linux or later; or Solaris 8, 9, or 10.

- Intel Pentium class processor, 400 MHz or faster; Sun Ultra5/360 or higher-performance workstation
- 128 MB RAM minimum; 256 MB highly recommended
- 1GB free disk space
- Local CD-ROM drive or access to network for installation
- Active Internet connection (recommended during initial installation)

In some cases (e.g., Red Hat Linux 7.2), the compiler will run on a host system, but the installer may not. In such a case, install on a different system that supports the installer (e.g., Red Hat Linux 7.3) and transfer the installation directory to the other machine.

3.2 Target System Requirements

The target is the computer for which you are developing. The Wind River Compiler is separately licensed for a variety of target architecture families. For a list of specific target CPUs supported by the tools, use the `dctrl` utility, or see the *Wind River Compiler User's Guide* for each architecture family.

4. Usage Caveats

This section describes usage issues that you should be aware of when working with the Wind River Compiler.

C and C++ Compilers

Treatment of Invalid Command-Line Options

In most cases, passing an unrecognized option flag to the tools generates a warning. This behavior, however, is not completely consistent. In some cases, no warning is generated; in other cases, such an invalid option may cause compilation to stop.

-Xmismatch-warning Overrides -e Option

`-Xmismatch-warning` and `-Xmismatch-warning=2` override the `-e` option (used for changing the severity of a message). If either form of `-Xmismatch-warning` is used, mismatched types will only produce a warning, even if `-e` is used to increase the severity level of the diagnostic.

Cross-Module Optimization



WARNING: Cross-Module Optimization will be deprecated in future releases and should therefore be used with restraint, if at all. See [Cross-Module Optimization and Whole-Program Optimization](#), p.4.

When using cross-module optimization options such as `-Xcmo-gen=file`, avoid locating `file` on an NFS-mounted partition; that is, `file` should not be an NFS pathname. Using an NFS pathname may result in unexpected or unwanted locking behavior when accessing the database. Instead, use a local directory, such as `/tmp`, if possible.

#pragma weak Usage

Using `#pragma weak` may occasionally produce unforeseen behavior. Two instances are worth mentioning:

- `#pragma weak` is incompatible with local data area (LDA) allocation; using `#pragma weak` with `-Xlocal-data-area` or `-Xlocal-data-area-static-only` enabled will produce a warning and temporarily disable LDA.
- A global definition will override a weak definition when it is encountered. (A symbol may be defined in more than one module as long as 1) no more than one of the definitions is global *and* 2) all the other definitions are weak.)

Consider the following scenario. Function `foo()` uses `x`, which is declared weak in library 1 and global in library 2. If library 1 is searched first, the weak version of `x` will be used. On the other hand, if library 2 is subsequently linked (because, for example, another function uses it), then the global version of `x` will replace the weak version.

Using `__attribute__((section))` May Not Always Be Honored

In some cases, the compiler may not honor an attempt to use the `section` attribute to place initialized data into a section intended for uninitialized data, and vice-versa.

For example, in the following code, the compiler does not honor the attempt to put `x` into the `.bss` section:

```
__attribute__((section(".bss"))) int x = 3;
```

`x` will be assigned to the `.data` section, not `.bss`.

Far Relative Addressing and VLE

The following applies to PowerPC code using the VLE (Variable Length Encoding) instruction set.

Programs compiled to use far (32-bit) relative addressing, either for code or data (for example, programs compiled with `-Xcode-far-relative` or `-Xdata-far-relative`), must explicitly reference the symbols `_SDA_BASE` and `_SDA2_BASE`. If these symbols are not referenced anywhere in the program, the linker will generate incorrect code. Specifically, it will try to use "absolute SDA," in which `r0` is used as a base register to indicate a base location of zero. (See the user's guide section on ELF Relocation Information for more on absolute SDA.)

This is not a problem for non-VLE code, where `r0` is interpreted as zero; in contrast, in VLE mode, `r0` is interpreted as the *contents* of `r0`.

5. Known Problems

This section highlights known problems with the Wind River Compiler that may have a significant impact on installation and your experience with this product. Defect numbers, where applicable, are given in parentheses after the description of the issue.

The most current information for this release is available from the Wind River Online Support Web site (see [1.5 Latest Release Information](#), p.3).

Casting in Inline Assembly Code

The compiler fails to sign-extend certain arguments after a cast in inline assembly routines. To fix this problem, use the **-Xold-inline-asm-casting** option.

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.)

C++ Trigonometric Functions Do Not Support Complex-Number Objects

Passing instances of the complex-number class template (defined in **complex.h**) to trigonometric functions such as **cos** leads to unresolved symbols.

Using **setjmp()** and **longjmp()** in Mixed ARM and Thumb Code

If you enable interworking (**-Xinterwork**) to compile mixed ARM and Thumb code, there is a limitation to the use of **setjmp()** and **longjmp()**. The **longjmp()** routine does not support switching between 32-bit and 16-bit mode; if you call **setjmp()** in Thumb mode and make a corresponding call to **longjmp()** in ARM mode, the result will be unpredictable. (However, the reverse procedure—calling **setjmp()** in ARM mode and **longjmp()** in Thumb mode—yields correct behavior.) A workaround is to define a routine like the **__common_long_jmp()** shown below and compile it with **-Xinterwork** enabled.

```
void __common_long_jmp()
{
    return;
}
```

Incorrect Type-extension of Arguments for 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.

6. Documentation Errata

This section lists documentation errata for documents associated with the Wind River Compiler. For a detailed list of documentation errata for the Wind River Compiler, visit the Online Support Web site (see [1.5 Latest Release Information](#), p.3).

Incorrect Syntax for Specifying VxWorks Kernel Compilation Environment

In the user's guide chapter on selecting targets, the syntax for specifying a VxWorks kernel compilation environment, using the `-t` option, is erroneously given as `:vxworksx.y` (e.g., `:vxworks6.7`); it should read `:vxworksxy` (e.g., `:vxworks67`).

The section in the user's guide should read as follows:

To build VxWorks applications, specify the appropriate execution environment with the `-t` option. Usually this will be `:rtp` for user (real-time process) mode or `:vxworksxy` (where `xy` represents a VxWorks release number, e.g., "67" represents VxWorks release 6.7) for kernel mode. For example, for a PowerPC processor,

```
-tPPCEN:rtp
```

selects user mode, while

```
-tPPCEN:vxworks67
```

selects kernel mode for release 6.7 of VxWorks. For more information, see the documentation that accompanied your VxWorks development tools.

Wind River Compiler for x86 User's Guide and Wind River Compiler for x86 Getting Started

Instruction Set Simulator Not Supported for Some Targets

WindISS does not currently fully simulate the MIPS64, ARM Thumb-2, and x86 instruction sets, nor ColdFire hardware floating-point instructions. Hence, many examples given in the documentation, including the "bubble sort" example program, cannot be compiled and run as documented for x86 targets.

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 and contact with customer support engineers, along with a Web site containing the latest advisories, FAQ lists, known problem lists, and other information resources.

Customer Support

For customers holding a maintenance contract, Wind River offers direct contact with support engineers experienced in Wind River products. The Customer Support program is described in the *Standard Support User's Guide* available at:

www.windriver.com/support

The guide describes the services available, including assistance with installation problems, product software, documentation, and service errors.

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

Location	Phone	E-mail
North and South America, Asia/Pacific (outside Japan)	800-872-4977 (toll-free)	support@windriver.com
Europe, Africa, Middle East	+(00) 800-4977-4977 (toll-free)	support-EC@windriver.com
Japan	81-3-5778-6001	support-jp@windriver.com

For detailed contact information, including contact information specific to your products, see the Support Web site shown above.

Wind River Online Support

Wind River Customer Services also provides Wind River Online Support, an online service available under the Support Web site. This is a basic service to all Wind River customers and includes advisories, online manuals, and a list of training courses and schedules. For maintenance contract holders, Online Support also provides access to additional services, including known problems lists, patches, answers to frequently asked questions, and demo code.