TORNADO 2.2

GNU Toolchain

RELEASE NOTES

WindRiver®

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

GNU Toolchain for Tornado Release Notes, 2.2

31 Jul 02
Part #: DOC-14511-ZD-01

Contents

GNU Toolchain for Tornado

Release Notes

2.2

1. Introduction

The Tornado GNU compiler in this distribution is called 2.96+. It is not related to the Red Hat compiler distributed with Red Hat Linux 7.0. The identical name is an unfortunate coincidence. This distribution is three years newer than the Tornado 2.0 C compiler, and about a year newer than the Tornado 2.0 C++ compiler. These release notes list the extensive changes and enhancements that have been made in that time.

Documentation for this Tornado GNU compiler is provided in HTML only. The *GNU Toolkit User's Guide*, the *GDB User's Guide*, and the *GNU Make User's Guide* are accessible from *installDir*/**docs/books.html** and from the Tornado Help menu. The only part of the GNU toolchain available for ColdFire is binutils.

2. General Changes

Following is a list of changes for existing Tornado 2.0 customers:

- Windows tools are built using cygwin32, rather than the Microsoft libraries. This appears to improve reliability.
- The GNU Assembler Pre-processor (GASP) has been merged into the assembler and no longer exists as a separate command. Additional C++ code compiled with the Tornado 2.0 compiler is not compatible with code produced

by the current version of the compiler. Existing C++ code must be rebuilt with the new compiler.

- Similarly, existing Tornado 2.0 projects are not compatible with the new toolchains. It is recommended that you recreate your projects after installing this release.
- The **-nostdinc** compiler flag should no longer be used. If you have pre-existing custom makefiles, you may need to remove this flag from your compile rules.
- If you get compiler errors such as "stddef.h: No such file or directory," you probably need to recreate your project or remove the **-nostdinc** compile flag from your makefile, as explained above.
- The C++ header files and some C header files (131 files in all) have been moved from *installDir*/**target/h** to various **include** directories under *installDir*/**host/***hostType*. For example, the C++ Standard Template Library (STL) headers no longer appear in **target/h**. You do not need to do anything unusual to get the compiler to find the headers in the new location; just make sure you *do not* use the **-nostdinc** flag.
- The x86 compiler is now called **ccpentium** rather than **cc386**. If you have custom makefiles and wish to generate code for a 386 (or 486), you must change your compile rules to use the flags **-mcpu=i386 -march=i386** (or **-mcpu=i486 -march=i486**).

3. An Important Note About Aggressive Alias Analysis

This version of the GNU compiler performs much stricter alias analysis than previous versions. Strictly incorrect (according to ANSI/ISO) C code that worked with older compilers may now produce unexpected results when built with the current compiler. For example, the following code is not portable ANSI/ISO C (because it attempts to access an **int** through a pointer to an **unsigned short**):

```
#include <stdio.h>

int
main ()
    {
    int a = 0x12345678;
    unsigned short *b = (unsigned short *)&a;
```

```
printf ("%x\n", a);
b[1] = 0;
printf ("%x\n", a);

return 0;
}
```

Does the above program always print:

```
a)  12345678      or      12345678       (depending on endianness)
    12340000              5678
```

or does it print:

```
b)  12345678
    12345678
```

or does it print neither?

According to the C standard, the answer is undefined. With older compilers (those without type-based aliasing) the result would always be **a**; with this version, and with newer compilers that have more powerful optimizers, it is more common to see **b** (if optimization flags are used).

You can turn off alias analysis (and allow old, "buggy" code to compile) by using the **-fno-strict-aliasing** compiler flag. However, you will then lose all the optimizations and speed improvements that come from doing aggressive alias analysis.

Please see the description of **-fstrict-aliasing** in the online manual for more information.

4. G++ Changes

This section summarizes changes and enhancements to the C++ compiler.

Code Generation and Optimization

This release reduces support for code that fails to meet the current C++ standard. In the past, that code would have generated a warning message; now it generates an error.

- For non-conforming code that can be handled, the errors can be reverted to warnings with the **-fpermissive** option.
- You can now use **-fno-implicit-inline-templates** to suppress writing out implicit instantiations of inline templates. Normally, they are written out, even with **-fno-implicit-templates**, so that optimization does not affect which instantiations are needed.
- **-fstrict-prototype** now also suppresses implicit declarations.
- On ELF systems, duplicate copies of symbols with initialized common linkage (such as template instantiations, vtables, and **extern** inlines) are now discarded by the GNU linker, so **-frepo** is not required. This support requires GNU **ld( )** from binutils 2.8 or later, which was not present in the Tornado 2.0 version.

Namespaces

Namespaces are fully supported, except that appropriate STABS or DWARF 2.0 debugging codes are not generated for similarly named data in different namespaces. You should insure for now that your data names are unique, even across namespaces. The library has not yet been converted to use the namespace **std**, however, and the old **std**-faking code is still on by default. To turn it off, you can use **-fhonor-std**.

Namespaces are supported within the limits specified below:

compilers
: Namespaces are fully supported by the compilers except for a bug; no STABS or DWARF 2.0 record is generated for **namespace** or **using** statements. All data names, regardless of namespace, appear in the global namespace.

libraries
: The C and C++ library names are not yet hidden inside the **std::** namespace. You should continue to avoid reusing documented standard library names.

debugger, StethoScope
: The debugger and StethoScope can process functions within namespaces, but not data.

Template Handling

Substantial template improvements include the following:

- Member template classes are supported.
- Template friends are supported.
- Template parameters are supported.
- Local classes in templates are supported.

The C++ Standard Template Library for this release corresponds to SGI version 3.11.

Exception Handling

Changes and improvements to exception handling include the following:

- Exception handling is now thread-safe, and supports nested exceptions and placement delete.
- The **new** operator now throws **bad_alloc** where appropriate.
- This release includes some changes to reduce static overhead for exception handling. It also includes some major changes to the **setjmp**/**longjmp**-based exception handling mechanism to make it less pessimistic.

For AltiVec-specific information, see *C++ Exception Handling and AltiVec Support*, p.11.

Other New Features

General features that are new since the last release include the following:

- String constants are now of type **const char[n]**, rather than **char[n]**. This can be reversed with **-fno-const-strings**.
- References to functions such as **&funcName(...)** are now supported.
- Lookup of class members during class definition now works in all cases.
- In overload resolution, type conversion operators are now properly treated as always coming from the most derived class.
- C9x-style restricted pointers are supported, using the **__restrict** keyword.

- Many obsolete options have been removed, including **-fall-virtual**, **-fmemoize-lookups**, **-fsave-memoized**, **+e?**, **-fenum-int-equivalence**, **-fno-nonnull-objects**.
- Protected virtual inheritance is now supported.
- For class D derived from B which has a member **int i**, **&D::i** is now of type **int B::*** instead of **int D::***.
- Loops are now optimized better; in most cases, the test now occurs at the end of the loop, as is the case with the C front-end.

5. GCC Changes

This section summarizes changes to the C compiler.

Error Handling

Error handling has been improved; code that used to compile without error may now generate errors. These errors represent previously unreported deviations from the ANSI standard.

Code Generation and Optimization

Changes and enhancements to code generation and optimization include the following:

- The compiler now implements global common sub-expression elimination (GCSE) as well as global constant/copy propagation.
- Major improvements have been made to the alias analysis code. A new option to allow front-ends to provide alias information to the optimizers has also been added (**-fstrict-aliasing**). **-fstrict-aliasing** is on by default. For more information, see *3. An Important Note About Aggressive Alias Analysis*, p.2.
- The security problem with temporary file permissions has been fixed.

- The register move optimization pass has been extensively rewritten. It now uses much more information about the target to determine the profitability of transformations.
- The compiler now recomputes register usage information immediately before register allocation. Previously, such information was not kept up to date after instruction combination, which led to poor register allocation choices by the priority-based register allocator.
- The register reloading phase of the compiler has been improved to better optimize spill code. This primarily helps targets that generate many spills (including the Pentium ports and many register-poor embedded ports).
- A few changes in the heuristics used by the register allocator and scheduler have been made that can significantly improve performance for certain applications. The compiler's branch-shortening algorithms have been significantly improved to work better on targets that align jump targets. The compiler now supports the **ADDRESSOF** optimization, which can significantly reduce the overhead for inline calls.
- The compiler now supports a code size optimization switch (**-Os**). When this switch is enabled, the compiler chooses optimizations that improve code size over those that improve code speed.
- The compiler has been improved to completely eliminate library calls that compute constant values. This is particularly useful on machines that do not have integer multiply/divide or floating point support on-chip.

Compiler Optimizations

Changes specifically to the compiler optimization include the following:

- The memory footprint for the compiler has been significantly reduced for certain pathological cases.
- The build time for the MIPS architecture family has been improved by refining the handling of scheduling parameters.
- Compile time for certain programs using large constant initializers has been improved.

New Features

General features new since the last release include the following:

- GCC now supports a **--help** option to print detailed help information.
- The DWARF 2 debugging information format is supported on ELF systems, and is the default for **-g** on those systems. It can also be used for C++.
- A new switch, **-fstack-check**, has been added to check for stack overflow on systems that do not have such a feature built into their ABI.
- The new **-Wundef** and **-Wno-undef** switches generate a warning if an undefined identifier is evaluated in an **#if** directive.
- The **-Wall** and **-Wimplicit** options now cause GCC to warn about implicit integers in declarations (for example, **register i**;), since the C Standard committee has decided to disallow this in the next revision of the standard. **-Wimplicit-function-declarations** and **-Wimplicit-int** are subsets of the **-Wimplicit** option.
- The **-Wsign-compare** option generates a warning if signed and unsigned values are compared.

Changes for the ARM Architecture Family

- The **-mlongcall** package of option, pragma, and attributes has been added. It matches the function of the PowerPC package introduced in the last Tornado release.

NOTE: -mlongcall has been tested.

- The **-march=***xxx*, **-mtune=***xxx*, **-mcpu=***xxx* options have been added.

NOTE: -mtune and **-march** are untested.

- Interworking (or 32-bit ARM code with 16-bit Thumb code) is unsupported (untested).

Changes for the MIPS Architecture Family

- New support for the MIPS4 instruction set has been added.
- The R4100, R4300 and R5000 processors are now supported.
- Multiply/Multiply-Add support has been largely rewritten to generate more efficient code.
- The **-gdwarf2** option has been reintroduced. This allows you to choose between the DWARF Version 2 and STABS debugging information formats.
- The compiler has been modified to avoid branch-likely instructions, which have proven unstable in earlier versions.
- Multiply-accumulate support has been added.

Changes for the Pentium Architecture Family

- Data in the static store is aligned to meet Intel recommendations. Jump targets are aligned, as recommended by Intel.
- Epilogue sequences have been improved.
- Back-end improvements have been made that should help register allocation on all Pentium variants.
- Support for PentiumPro conditional move instructions has been fixed and enabled.
- Several changes have been made throughout the port to make generated code more Pentium-friendly. Support for 64-bit integer operations has been improved.
- Scheduling parameters for Pentium and Pentium Pro have been added.

Changes for the PowerPC Architecture Family

- For PowerPC 604 only, the stack frame is always 16-byte aligned, rather than 8-byte aligned. This alignment does not exclude the use of third-party libraries that have been compiled for 8-byte alignment.

The following new command-line options are now supported. For more information, see the *GNU ToolKit User's Guide*.

- **-mcpu=401** (added as an alias for **-mcpu=403**)
- **-mcpu=604e**, **405**, **602**, **603e**, **620**, **801**, **823**, **505**, **821**, **860**, and **power2**
- **-meabi**
- **-memb**, **-msim**, **-mmvme**, **-myellowknife**, and **-mads**
- **-mfused-madd** and **-mno-fused-madd**
- **-mregnames**
- **-mrelocatable-lib** and **-mno-relocatable-lib**
- **-msdata**, **-msdata=none**, **-msdata=default**, **-msdata=sysv**, and **-msdata=eabi**
- **-msim**, **-mmve**, and **-memb**
- **-mtune=***xxx*
- **-mupdate** and **-mno-update**
- **-p/-pg**

The following command-line options have been changed. For more information, see the *GNU ToolKit User's Guide*.

- **-fvec** enables AltiVec instructions, but disables **vector** and **pixel** as keywords: **__vector**, **__pixel** and **bool** remain as type specifier keywords.
- **-fvec-eabi** enables AltiVec instructions, including **vector**, **pixel**, **__vector**, **__pixel**, and **bool** as type specifier keywords.
- **-mcpu=403** now implies **-mstrict-align**.
- **-mcpu=405** is simply passed to the PowerPC assembler to enable the assembly of PPC405 instructions.
- **objdumpppc** disassembles AltiVec instructions.
- **wchar_t** is now of type **long** as specified by the ABI, rather than **unsigned short**.

Unsupported Features

The following features are unsupported:

- **Prefixed Underscore**

  In the PowerPC architecture, the compiler does not prefix underscores to symbols. In other words, **symbol** is not equivalent to **_symbol** as it is in other architecture implementations.

- **Small Data Area**

  The compiler supports the small data area. However, for this release of Tornado for PowerPC, VxWorks does not support the small data area. Therefore the **-msdata** compiler flag must not be used.

C++ Exception Handling and AltiVec Support

Throwing C++ exceptions between modules that were compiled with different compiler flags may result in unexpected behavior. C++ exceptions save register state. Modules compiled with AltiVec support (using either **-fvec** or **-fvec-eabi**) save all non-volatile AltiVec registers, but modules compiled without AltiVec support do not save any AltiVec registers. If a C++ exception is thrown from an AltiVec-enabled module, caught by a non-AltiVec enabled handler, and then thrown from there to an AltiVec-enabled handler that alters the AltiVec registers, it is possible to corrupt the saved AltiVec state. In particular, the non-volatile vector registers (**v20** through **v31**) may be corrupted.

The following example illustrates the above scenario. It consists of a program comprised of two files, **file1.cpp** and **file2.cpp**. Because **file2** is compiled with the **-fvec** option, we call it AltiVec code. **file1** is compiled without a **-fvec** option, so it is called non-AltiVec code.

The example takes program flow across the two modules. It is also contrived to make intelligent guesses about the compiler's register allocation strategy. The output is incorrect when one of the files is compiled without the **-fvec** option.

Listing for **file1.cpp**:

```
extern "C" int printf (const char *fmp, ...);
extern void bar ();

void foo ()
    {
    try
        {
        bar ();
        }
```

```
    catch (...)
        {
        }
    }
```

Listing for **file2.cpp**:

```
extern "C" int printf (const char *fmp, ...);
extern void foo ();

typedef __vector signed long T;

void bar ()
    {
    // use a non-volatile vector register
    asm ( "vsplitisw 24,0" );        // v24 <- (0,0,0,0)
    }


void Start ()
    {
    // use a non-volatile vector register v24
    T local = (__vector signed long) (-1, -1, -1, -1);

    asm ( "vsplitisw 24,15" );        // v24 <- (15, 15, 15, 15)

    foo ();

    // continue using the non-volatile vector registers
    asm ( "addi 9, 31, 32" );        // local <- v24
    asm ( "stvx 24, 0, 9" );

    printf ("Finally, local = (%vld)\n", local);
    }
```

To resolve this behavior, follow the steps below.

Step 1: Reproduce the problem.

To produce a partially linked object **file2.o**, compile the two files with the following commands:

```
% ccppc -mcpu=604 -c file1.cpp
% ccppc -mcpu=604 -nostdlib -fvec -r file1.o file2.cpp
```

Download **file2.o** to a target, and execute the **Start** function.

```
-> Start
Finally, local = (0,0,0,0)
->
```

The **foo** function in **file1.cpp** is non-AltiVec code. Therefore, the **try...catch** block in **foo** does *not* save and restore the AltiVec context. Within the **try...catch** block, the

call to **bar** alters the value of vector register **v24**. Because **file1.cpp** does not save AltiVec context, the value 0 in **v24** assigned by **bar** remains unchanged when program flow returns to **Start**. The original value 15 assigned before the call to **bar** is now corrupted. This explains the incorrect output **local = (0,0,0,0)**.

Step 2: Correct the behavior.

Compile both files with the **-fvec** option:

```
% ccppc -mcpu=604 -nostdlib -fvec -r file1.cpp file2.cpp -o file2.o
```

Download **file2.o** to a target and execute the **Start** function.

```
-> Start
Finally, local = (15,15,15,15)
->
```

Since both modules now have AltiVec code (compiled with the **-fvec** option) the **try...catch** block in **foo** now saves and restores the AltiVec context. The value 15 originally assigned in **Start** is faithfully restored by **foo** when it returns.

For more information on AltiVec support, see *The only changes to GDB are extensions for AltiVex support.*, p.14 and *This section lists extensions to the WDB and WTX protocols for AltiVec support*, p.15.

Changes for the SPARC Architecture Family

- The compiler now includes V8 plus and V9 support and tuning for Ultrasparcs.
- Haifa instruction scheduling is now enabled by default.

The following new command-line options are now supported. For more information, see the *GNU ToolKit User's Guide*.

- **-mcpu=***xxx* and **-mtune=***xxx*
- **-malign-loops=***xxx*, **-malign-jumps=***xxx*, and **-malign-functions=***xxx*
- **-mimpure-text** and **-mno-impure-text**

Changes for the StrongARM Architecture Family

For information on StrongARM changes, see *Changes for the ARM Architecture Family*, p.8.

Changes for the SuperH Architecture Family

The GNU toolchain for this release uses the ELF object module format and DWARF 2.0 debug information. For very large projects, the DWARF debug information can result in slow load time in the Tornado debugger, CrossWind. For such cases, it is recommended that only a smaller number of modules be built with the **-g** option.

6. Extensions for AltiVec Support

GDB Changes

The only changes to GDB are extensions for AltiVex support.

In this release, **gdb** features a **setaltivec** command that allows users to set a particular value into a given vector register. Some typical scenarios for using the **setaltivec** commands are described below.

```
(gdb)help setaltivec

setaltivec <regname> 0x<hex>_<hex>_<hex>_<hex>
Sets the value of the specific AltiVec register.
```

To set a given value into an AltiVec register using **setaltivec**, enter the following:

```
(gdb)setaltivec v4 0x45454545_12345678_12_5A7
```

Vector register contents can be printed using the **print** command:

```
(gdb)print $v4

0x45454545123456780000012000005A7
```

WDB Changes

This section lists extensions to the WDB and WTX protocols for AltiVec support

The following new WTX and WDB API functions have been added for AltiVec support.

Table 1 **WTX API Functions for AltiVec Support**

Routine	Command Syntax	Description
wtxTargetHasAltivecGet( )	**hWtx**	Returns TRUE if the target has an AltiVec unit.

Table 2 **WDB API Functions for AltiVec Support.**

Routine	Command Syntax	Description
wdbAltivecSave( )	**void**	Saves the AltiVec registers into a buffer.
wdbAltivecRestore( )	**void**	Restores the AltiVec register values from a buffer.
wdbAltivecGet( )	**ppRegs**	Gets a pointer to the AltiVec context.
wdbAltivecSet( )	**pRegs**	Sets the AltiVec context from a buffer.

7. Binutils

Changes and enhancements to binutils include the following:

- A new command-line switch to **objdump -M** (or **--disassembler-options**) takes a parameter which can then be interpreted on a per-target basis by the disassembler. It is used by ARM targets to select register name sets, ISA, APCS, or raw versions.
- **objdump** support has been added for **-mi386:intel**, which causes disassembly to be displayed with Intel syntax.
- A new program, **readelf**, has been added. This program displays the contents of ELF format files, regardless of target machine.

- **objcopy** now takes **--change-section-lma**, **--change-section-vma**, and **--change-section-address** options. The old **--adjust-section-vma** option is equivalent to **--change-section-address**. The other **--adjust-*** options have now been renamed to **--change-***, although **--adjust-*** continues to work.
- **dlltool** now supports the **IMPORTS** command.
- **dlltool** now takes the **--export-all-symbols**, **--no-export-all-symbols**, **--exclude-symbols**, and **--no-default-excludes** options.
- **objcopy** now takes a **-j/--only-section** option to copy only the specified sections.
- The **windres** program has been added. It can be used to manipulate resources in WIN32 files as used on Windows 95 and Windows NT.
- The **objcopy --gap-fill** and **--pad-to** options operate on the LMA rather than the VMA of the sections.
- The **S** modifier has been added to the archiver to make it possible not to build a symbol table.
- The **objdump** disassembly format has been improved. Use the new **--prefix-addresses** option to get the old format. There are also new **--disassemble-zeroes** and **--no-show-raw-insn** options that affect disassembler output.
- Formats can now be specified as configuration triplets. For example, **objdump -b i386-pc-linux**. The triplets are not passed through **config.sub**, so they must be in canonical form.
- The new **addr2line** program has been added. This program uses the debugging information to convert an address into a filename and line number within a program.
- The **--change-leading-char** argument has been added to **objcopy**.
- The **--weaken** argument has been added to **objcopy**.
- **objdump --dynamic-reloc** now works on ELF executables and shared libraries.
- The **--adjust-vma** option has been added to **objdump**.
- The **-C/--demangle** option has been added to **objdump**.
- The **-p/--preserve-dates** option has been added to **strip** and **objcopy**.
  - For information on **objcopy** changes, see the reference page, which is accessible from the Tornado Help menu.

  - **objcopy** cannot be used to convert relocatable files, but is useful for absolute (fully-linked) files.
  - Documentation of **nm** type codes for ELF is found in **binutils.texi**.
- **objdump** now disassembles ppc405 instructions.

NOTE: Binutils ships with all architectures, including ColdFire, which does not include the rest of the toolchain.

8. Assembler

Changes and enhancements to the assembler include the following:

- A new pseudo-op, **.intel_syntax**, has been implemented to allow GAS to parse Pentium assembly dialect. This pseudo-op applies to the Pentium architecture only.
- GAS now assembles assembly programs with Intel syntax. This applies to the Pentium architecture only.
- This version of the assembler includes greatly improved instruction operand checking for Pentium processors. This change produces errors or warnings on incorrect assembly code that previous versions of GAS accepted. If you get unexpected messages from code that worked with older versions of GAS, please double-check the code before reporting a bug.
- The instruction to jump indirect through a register is now spelled differently. It is spelled **jmp *%eax**. This applies to the Pentium architecture only.
- A new pseudo-op, **.type**, is used (for all architectures) to provide type information so that a symbol is loaded by VxWorks. The **.type** directive explicitly declares the type of the symbol, but is only necessary for programmers who are writing their own assembly source files, and who want those symbols to be visible to the VxWorks loader.
- Two new pseudo-ops, **.func** and **.endfunc**, are provided to aid in debugging user-written assembler code.
- The **-gdwarf2** option has been added to generate DWARF 2 debugging information.

- The assembler now optimizes the exception frame information generated by EGCS and GCC 2.8. The new **--traditional-format option** disables this optimization.
- The **-a** option takes a new suboption, **m** (for example, **-alm**) to expand macros in a listing.
- The **-a** option takes a new suboption, **c** (for example, **-alc**), to skip false conditionals in listings.
- The **-MD** option has been added to print dependencies.
- MIPS16 support has been added.
- The alignment directives now take an optional third argument which specifies the maximum number of bytes to skip. If doing the alignment would require skipping more than the given number of bytes, the alignment is not done at all.
- The ELF assembler has a new pseudo-op, **.symver**, used for symbol versioning.
- A new pseudo-op, **.equiv**, has been added; it is similar to **.equ**, except that it causes an error if the symbol is already defined.
- The PowerPC assembler now allows the use of symbolic register names (**r0**, and so on) if **-mregnames** is used. Symbolic names preceded by a **%** (**%r0**, and so on) can be used any time. PowerPC 860 **mtspr** and **mfspr** (move-to and move-from SPR) instructions have been added.
- PowerPC ELF support has been added.
- Pentium and PowerPC gnu-win32 support has been added.
- GAS now directly supports macros, without requiring GASP.
- The **--defsym SYM=VALUE** option has been added.
- **-mips4** support has been added to MIPS assembler.
- **-mips32** and **-mips64** support has been added to MIPS assembler.
- PIC (position-independent code) support has been added to Solaris assembler.
- PowerPC assembler now assembles ppc405 instructions when the new command-line option **-m405** is specified.
- AltiVec instruction support has been added in GAS with **-mvec**.

9. Linker

Changes and enhancements to the linker include the following:

- **.vx** sections are passed through.
- Garbage collection of unused sections has been added, enabled by **--gc-sections**. It requires additional back-end support, which is currently implemented for **ppc-elf** and **mips-elf** only. Others ignore the option.
- The following have been added to the linker script language: **SORT** (to permit sorting sections by filename or section name), **EXTERN** (an equivalent to the **-u** command-line option), **ASSERT**, and **EXCLUDE_FILE** (for further control over wildcard filenames).
- The **-O** option has been added to optimize linker output (currently this only affects ELF shared library generation).
- The **-e** option now accepts a number as well as a symbol name.
- The **--no-undefined** option has been added to disallow undefined symbols when creating a shared library.
- The **--demangle** and **--no-demangle** options have been added.
- **SQUAD** has been added to the linker script language.
- A new option, **--no-warn-mismatch**, has been added.
- The **MEMORY** command now parses the attributes to determine where sections that are not placed in a specific memory region are placed.
- Linker scripts may contain shell wildcard characters for file and section names.
- The linker now supports symbol versions in ELF.
- The following were added to the linker script language: the **NOCROSSREFS** command, the **LOADADDR** expression, the **MAX** and **MIN** functions, and the **OVERLAY** construct.
- A new option, **--warn-section-align**, has been added to generate a warning when the address of an output section changes due to the alignment of an input section.
- The new options **--filter/-F** and **--auxiliary/-f** have been added.
- A new option, **--cref**, has been added to print out a cross-reference table.

- A new option, **--wrap SYMBOL**, has been added, which causes undefined references to **SYMBOL** to be resolved to the linker-created wrapper function **__wrap_SYMBOL**.
- A new option, **--no-whole-archive**, has been added to turn off the effect of **--whole-archive**.
- Input sections assigned to the output section **/DISCARD/** in the linker script are not included in the output file.

10. Benchmark Test Results

This section lists the results of a comparative analysis of the GNU toolchain between the Tornado 2.0 (or, where appropriate, Tornado 2.1) and Tornado 2.2. Reports were generated for compilation times and for performance times.

10.0.1 Compilation Time Reports

This section lists the reports for the compilation time comparison between Tornado 2.0 and 2.2 GNU toolchains. The BYTEmark ver. 2 (3/95) tool was used to generate these benchmarks.

Host Machine Configuration

The investigation was performed on a host machine with the following configuration:

- **Machine .** Sun Ultra-5_10
- **Physical Memory.** 256 Mb
- **Virtual memory.** 670 Mb
- **OS.** SunOS release 5.7

Command-Line Compilation Options

The compilation options applied the highest level of GNU optimization, and are listed below.

MIPS64 Options

```
-G 0 -mno-branch-likely -mips4 -ansi -EB -O3 -I. -I/vobs/wpwr/target/h
-DCPU=MIPS64 -DTOOL_FAMILY=gnu -DTOOL=gnu -DMIPSEB -D_WRS_MIPS_VR5400_ERRATA
-c  -DVXWORKS -Wp,-lang-c
```

MIPS32 Options

```
-G 0 -mno-branch-likely -mips2 -EB -ansi -O3 -I/vobs/wpwr/target/h
-DCPU=MIPS32 -DMIPSEB -DSOFT_FLOAT -msoft-float -c  -DVXWORKS -Wp,-lang-c
```

Pentium-II Options

```
-nostdlib -fno-builtin -fno-defer-pop -c -DVXWORKS -Wp,-lang-c -O3
-fomit-frame-pointer -march=pentiumpro -DCPU=PENTIUM2
```

Pentium Options

```
-nostdlib -fno-builtin -fno-defer-pop -c -DVXWORKS -Wp,-lang-c -O3
-fomit-frame-pointer -march=pentium -DCPU=PENTIUM
```

PPC604 Options

```
-nostdlib -fno-builtin -fno-defer-pop -c -DVXWORKS -Wp,-lang-c
-fomit-frame-pointer -mcpu=604 -DCPU=PPC604 -O3
```

PPC603 Options

```
-nostdlib -fno-builtin -fno-defer-pop -c -DVXWORKS -Wp,-lang-c
-fomit-frame-pointer -mcpu=603 -DCPU=PPC603 -O3
```

SH4 Options

```
-nostdlib -fno-defer-pop -c -DVXWORKS -Wp,-lang-c -fomit-frame-pointer -m4
-DCPU=SH7750 -O3
```

SH3 Options

```
-nostdlib -fno-defer-pop -c -DVXWORKS -Wp,-lang-c -fomit-frame-pointer -m3
-DCPU=SH7700 -O3
```

Results

Table 3 list the results of this investigation. Higher delta percentages indicate faster compilation times.

Table 3 **Compilation Time Comparison of Tornado 2.0 and Tornado 2.2**

Architecture	Tornado 2.0	Tornado 2.2	Delta
MIPS32	7.9	10.4	-31 %
MIPS64	7.0	9.6	-37 %
Pentium I	4.8	11.0	-129 %
Pentium II	4.8	9.5	-97 %
PPC 603	4.8	10.9	-127 %
PPC 604	4.8	10.9	-127 %
SH3	10.1*	10.1	0 %
SH4	10.5*	10.4	+1 %

* The SH tests were compiled on Tornado 2.1.

10.0.2 Performance Time Reports

This section lists the reports for the compilation time comparison between Tornado 2.0 and 2.2. The BYTEmark (tm) Native Mode Benchmark ver. 2 (3/95) tool was used to generate these reports. This benchmark generates two indexes:

- **"II" .** Integer Index (for integer operations)
- **"FPI" .** Floating-Point Index (for floating-point operations)

These tests were performed on each of the compiler front ends. Both front ends used the Tornado 2.2 (VxWorks 5.5) kernel. Tests were compiled from the command line. The command-line options are listed below with each architecture. In the result tables below, higher indices indicate better performance and higher delta percentages indicate better runtime (executable) performance.

Pentium Target Architectures

Table 4 lists the performance indices for Pentium-II boards (pcPentium2-7). The excutables were compiled with the following GNU compiler options:

```
-nostdlib -fno-builtin -fno-defer-pop -c -DVXWORKS -Wp,-lang-c -O3
-fomit-frame-pointer -march=pentiumpro -DCPU=PENTIUM2

-nostdlib -fno-builtin -fno-defer-pop -c -DVXWORKS -Wp,-lang-c -O3
-fomit-frame-pointer -march=pentium -DCPU=PENTIUM
```

Table 4 **Performance Indices for Pentium Boards**

Target Command	Tornado 2.0 Front End	Tornado 2.2 Front End	Delta
Pentium II	II = 4.965049	II = 6.126236	+23 %
	FPI = 3.710102	FPI = 4.183378	+13 %
Pentium	II = 4.970590	II = 5.397496	+8.6 %
	FPI = 3.688528	FPI = 4.154262	+12.6%

PowerPC Target Architectures

Table 5 lists the performance indices for executables run on PPC 7400 boards (mv5100). The excutables were compiled with the following GNU compiler options:

```
-nostdlib -fno-defer-pop -O3 -fomit-frame-pointer -mcpu=604 -DCPU=PPC604

-nostdlib -fno-defer-pop -O3 -fomit-frame-pointer -mcpu=603 -DCPU=PPC603
```

Table 5 **Performance Indices for PPC 7400 Boards**

Target Architecture CPU	Tornado 2.0 Front End	Tornado 2.2 Front End	Delta
PPC604	II = 7.063887	II = 7.200637	+ 2 %
	FPI = 8.696446	FPI = 8.894768	+ 2.3 %
PPC603	II = 7.196563	II = 7.280278	+ 1.1 %
	FPI = 8.628312	FPI = 8.995991	+ 4.2 %

Hitachi Target Architectures

Table 6 lists the performance indices for SH3 (ms7729se) and SH4 (ms7750se) boards. The excutables were compiled with the following GNU compiler options:

```
-nostdlib -fno-defer-pop -fomit-frame-pointer -m4 -DCPU=SH7750 -O3

-nostdlib -fno-defer-pop -fomit-frame-pointer -m3 -DCPU=SH7700 -O3
```

Table 6 **Performance Indices for SH3 and SH4 Boards**

Target Architecture CPU	Tornado 2.0 Front End	Tornado 2.2 Front End	Delta
SH7750	II = 2.036200	II = 2.121015	+ 4.17 %
	FPI = 0.920407	FPI = 0.951641	+ 3.39 %
SH7700	II = 0.852523	II = 0.842931	-1.13 %
	FPI = 0.071152	FPI = 0.071122	-0.04 %

MIPS Target Architecture

Table 7 lists the performance indexes for MIPS32 (ddb5476) and MIPS64 (ddb547-1) boards. The excutables were compiled with the following GNU compiler options for MIPS32 and MIPS64.

- **Tornado 2.0**

```
-EB -mips2 -G 0 -ansi -O3 -funroll-loops -DCPU=R3000 -msoft-float

-EB -mips4 -G 0 -O3 -funroll-loops -DCPU=R4000 -DMIPSEB
```

- **Tornado 2.2**

```
-G 0 -mno-branch-likely -mips2 -EB -O3 -DCPU=MIPS32 -DMIPSEB -DSOFT_FLOAT
-msoft-float

-G 0 -mno-branch-likely -mips4 -EB -O3 -DCPU=MIPS64 -DMIPSEB
```

Table 7 **Performance Indices for MIPS32 and MIPS64 Boards**

Target Architecture CPU	Tornado 2.0 Front End	Tornado 2.2 Front End	Delta
MIPS32	II = 2.187173	II = 2.108248	-3.61 %
	FPI = 0.032820	FPI = 0.032784	-0.11 %
MIPS64	II = 2.265500	II = 2.214943	-223 %
	FPI = 0.923565	FPI = 0.949614	+2.28 %

11. Known Problems

For up-to-date information on current and fixed software problem reports (SPRs), visit the WindSurf Web site at **www.windriver.com/windsurf/**.

12. 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 problems lists, and other valuable information resources.

Customer Support

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

http://www.windriver.com/support

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

You can reach Customer Support using either of the following methods:

- **E-mail.** You can contact Wind River Customer Support by sending e-mail to **support@windriver.com**.
- **1-800-872-4977 (1-800-USA-4WRS) .** Within North America, you can contact Customer Support with a toll-free voice telephone call. For telephone access outside North America, see the Support Web site shown above.

For Customer Support contact information specific to your products, please visit the Support Web site.

WindSurf

Wind River Customer Services also provides WindSurf, an online support service available under the Support Web site. 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 also provides access to additional services, including known problems lists, available patches, answers to frequently asked questions, and demo code.