

HIGHLIGHTS

- Support for Altera's Eclipse-based Nios II IDE
- Incorporates next-generation Viper compiler technology
 - Excellent code performance
 - GCC language extensions
 - ISO C'99 compliant
 - MISRA support
 - Code profiling
 - Run-time error checking
- Available for PC/Windows and SUN/Solaris

INTRODUCING THE TASKING VX-TOOLSET FOR NIOS® II

The TASKING VX-toolset for the Nios® II brings to developers the power of Altium's sophisticated, next-generation Viper C compiler technology framework, allowing them to take full advantage of the popular FPGA-based Nios II embedded processor from Altera. With its Viper technology, the TASKING VX-toolset for Nios II is able to generate code with the level of execution speed and code density needed for tomorrow's automotive, industrial and communications applications. The Nios II compiler features state-of-the-art capabilities such as MISRA C code checking, profiling through code instrumentation and run-time error checking.

The TASKING VX-toolset for Nios II comprises:

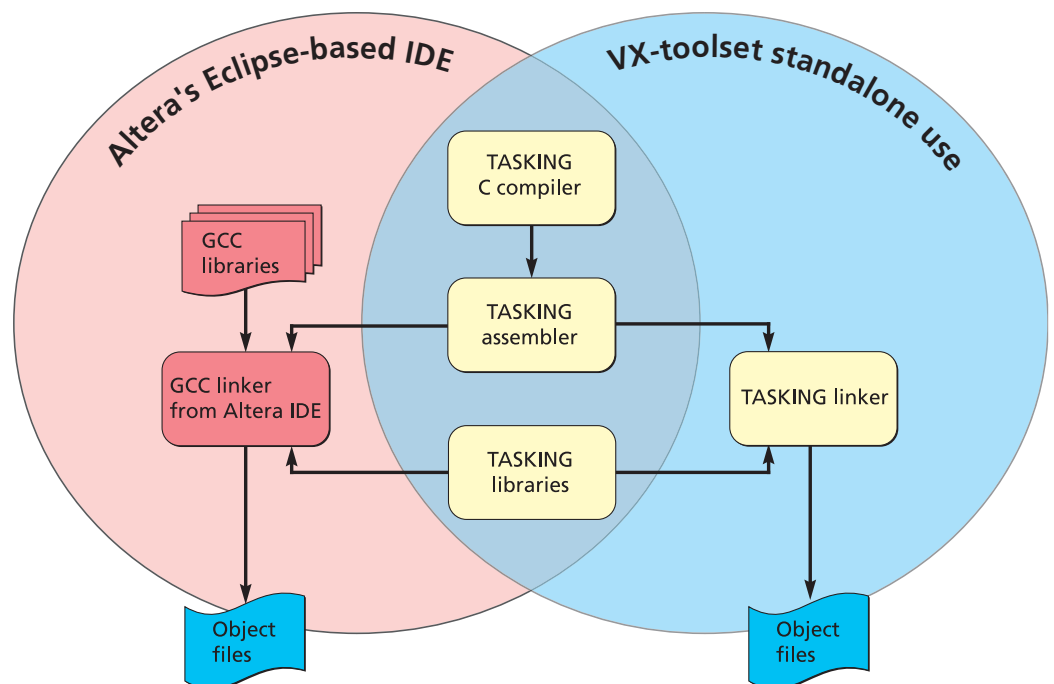
- Plug-in for Eclipse based IDE from Altera®
- C Compiler, ISO C'99 compliant, with integrated 'MISRA' enhanced code checking
- Assembler with macro-preprocessor
- C libraries, run-time libraries, floating-point libraries
- Linker and locator

The toolset provides a perfect 'drop in' replacement for the GNU based compiler provided with Altera's Nios II Integrated Development Environment, offering significantly better performance on code speed and density.

The C compiler included in the TASKING VX-toolset for Nios II is also included as part of Altium Designer, available separately from Altium. Altium Designer is the industry's first and only unified design system that incorporates all the technologies and capabilities necessary for complete electronic product development. Altium Designer brings together hardware, programmable hardware and software design within a single, unified design environment, and can significantly speed application development by fully harnessing the potential of large-scale programmable devices.

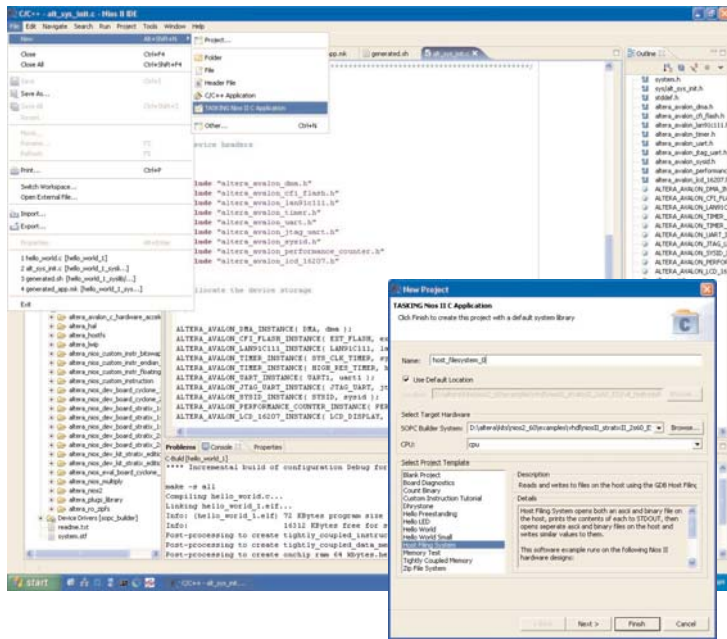
SEAMLESS INTEGRATION IN ALTERA IDE

The TASKING VX-toolset can be fully integrated as a plug-in to Altera's Eclipse based IDE. The developer then has the choice to use the C compiler, assembler and libraries from the TASKING VX-toolset instead of the GCC variants that come with Altera's IDE. Upon need, the original GCC compiler, assembler or libraries may also still be used. In both situations the GCC based linker ensures interoperability with existing libraries and third party solutions. Alternatively, the TASKING VX-toolset can also be used in a fully standalone setup, in which the full chain Compiler - Assembler - Linker from TASKING is used.



A TASKING project wizard and Compiler properties view tab will be added to the Altera IDE when installed as a plug-in to the Altera IDE. The user-interface is very similar to the corresponding Altera screens.

(New) Projects can be created and maintained using the TASKING toolset to obtain higher code density and/or execution speed.



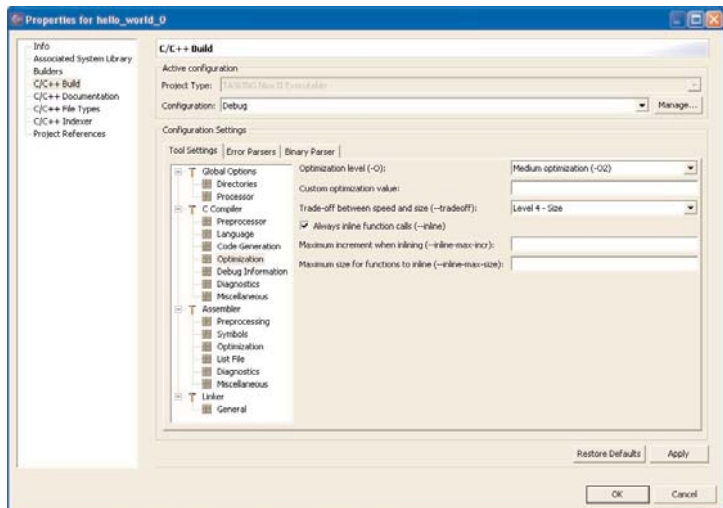
C COMPILER

Based upon Altium's latest DSP-C compiler technologies, the VX-toolset C compiler is reliable, compliant, competitive, complete, easy to use and generates the most optimal code possible to allow you to take full advantage of the Nios II core.

The TASKING VX-compiler for Nios II is tested for ISO C'99 conformity against authoritative validation suites, such as Perennial and Plum Hall. In addition, the optimization techniques of the compilers are tested with various large real-world applications (e.g. audio/GSM codec suites), as well as industry benchmark standards such as Nullstone.

Fast and compact

Altium understands that you expect your Nios II compiler to produce the most optimal code possible with no fuss. With its Viper compiler technology, the TASKING VX-toolset for Nios II, in its default configuration, generates code with the smallest footprint and fastest execution possible. Depending on the specific requirements of your Nios II application, optimizations can then be further tweaked for smaller code size or higher execution speed.



Compiler optimizations include:

- **Partial Redundancy Elimination (PRE) detects and eliminates repeating (sub-) expressions**
- **Various Loop and Jump optimizations speed up execution and reduce code size**
- **Control-flow and code-reduction optimizations remove dead code and perform transformations to minimize jumps**
- **Function inlining replaces calls to small functions with inlined copies of the function code**
- **Peephole optimizations replace instruction sequences with equivalent but faster and/or shorter sequences, or remove obsolete instructions**
- **Inter-procedural register allocation**

Code profiling

The compiler is equipped with a profiler that uses code instrumentation. Code profiling can be used to determine which pieces of your code execute slower than expected and which functions contribute most to the overall execution time of a program. A profile can also tell you which functions are called more or less often than expected. The advantage of this code profiling option in the compiler is that it can give a complete call graph of the application annotated with the time spent in each function and basic block.

Several forms of profiling output can be obtained:

- **Flat profile – shows how much time is spent in each function, how many times that function has been called, and optionally how often each lexical block within the function is executed. This is very useful if you want to know which functions or lexical blocks consume most cycles**
- **Call graph profile – shows, for each function, which functions called it, which other functions it called, and how many times. There is also an estimate of how much time was spent in the subroutines of each function**

Syntax and semantic checks

The compiler offers a vast array of syntax and semantic checks that warn about potential undesirable effects or bugs in your program. Early fixing of source code problems when reported by the compiler generally only takes minutes compared to hours, or days, when the problem is discovered at run time.

Examples of compile-time checks include:

- **Validating printf and scanf format strings against the type of the actual arguments**
- **Using uninitialized memory locations**
- **Detecting unused variables**
- **Value tracking, which is used to detect errors such as**
 - **array subscript out of bounds**
 - **division by zero**
 - **constant conditions**

Run-time error checking

TASKING's run-time error checking capabilities in the compiler offer a wealth of checks that reveal run-time errors when they first occur. The kind of errors found by run-time error checking are typically hard to find since they manifest themselves through secondary effects or, in the worst case, will not manifest at all prior to your product being shipped. By identifying the source line where the error first occurs, the run-time error checking facilities reduce the time spent in the debugger, and increase the quality of your software. You can specify whether the application will terminate or continue when an error is detected. These optional checks are implemented by generating additional code and/or enabling additional code in the standard C library. Run-time error checking has a nominal effect on code size and execution speed and can be enabled on a module by module basis, making it practical for use in debugging large applications.

The following types of checks are provided:

- **Bounds checking verifies all pointer operations to detect buffer overflows and other illegal operations such as:**
 - accessing uninitialized or null pointers
 - accessing objects outside their declared bounds
 - illegal pointer arithmetic
- **Malloc / free checks uncover dynamic memory allocation errors such as:**
 - buffer overflow
 - write to freed memory
 - multiple calls to free
 - passing an invalid pointer to free
- **Report an unhandled case value in a switch without a default part**
- **Stack overflow detects when the stack grows beyond its allocated size**
- **Divide by zero issues a message when a division by zero is attempted**

MISRA C

Altium was the first company to fully integrate MISRA C support into C compilers for embedded development purposes. MISRA C guides programmers in writing more robust C-code by defining selectable C-usage restriction rules. Through a system of strict error checking, the use of error-prone C-constructs can be prevented. The latest step in this innovation is configurability of the compliancy checking. The MISRA rules, which the application's source code should be compliant with, can be set as 'required' or 'advisory' and the diagnostic level of the generated messages by the compiler can be defined as either 'warning' or 'error'. This allows you to configure the individual rules of the MISRA C compliancy validation according to the quality standards set by your company.

The Nios II VX-toolset supports the new MISRA-C:2004 standard as well as the original MISRA-C:1998 guidelines.

CUSTOMER SUPPORT

When you purchase a TASKING product, it is the beginning of a long-term relationship. Altium is dedicated to providing quality products and support worldwide. This support includes program quality control, product update service, and support personnel ready to answer your questions by telephone, fax or email.

A maintenance period is included with the purchase of TASKING products and entitles you to enhancements and improvements as well as individual response to problems. Annual maintenance agreements are available to prolong this initial support period.

LICENSE MANAGEMENT

The TASKING toolset includes the industry standard FLEXlm license manager, offering stability as well as flexibility. Its license 'borrowing' functionality is a popular feature, allowing laptop users to take a license from the floating license pool for the period of time they are off-site, saving on cost for individual hardware-locked licenses.

PRODUCT PACKAGING & ORDERING CODES

This TASKING product comes with full documentation in easy-to-use paperbacks. This documentation is also available online as PDF files.

Product code	Package contents
07-200-113-022	C compiler, assembler, linker/locator and Eclipse plug-in for Altera Embedded Design Suite

A trial version of the TASKING VX-toolset for Nios II is available on CD-ROM or downloadable from our website at

www.tasking.com/niosii

ALTium SALES OFFICES

North America

Altium Inc

17140 Bernardo Center Drive, Suite 100
San Diego, CA 92128
Toll Free: 1-877-TASKING
Tel: 1-858-521-4280
Fax: 1-858-485-4610
Email: tasking.sales.na@altium.com

Asia – Pacific

Japan – Altium Japan K.K.

Resona Gotanda Building 7F
1-23-9, Nishi-Gotanda
Shinagawa-ku Tokyo 141-0031
Tel: (03) 5436 2501
Fax: (03) 5436 2505
Email: tasking.sales.jp@altium.com

China – Altium Information Technology (Shanghai) Co., Ltd.

Suite A&J, Floor 9, Hua Du Mansion
No 838 Zhang Yang Road
Pudong, Shanghai 200122
Tel: (021) 6876 4016
Fax: (021) 6876 4015
Email: info@altium.com.cn

Australia – Altium Limited

Level 3, 12a Rodborough Road
Frenchs Forest NSW 2086
Free Call: 1800 030 949
Fax: (02) 9975 7720
Email: sales.au@altium.com

Europe

Germany – Altium Europe GmbH

Albert-Nestler-Straße 7
76131 Karlsruhe
Free Call: 0800 0 258486 (0800 0 ALTium)
Fax: (0)721 82 44 320
Email: tasking.sales.de@altium.com

Switzerland – Protel AG

(A subsidiary of Altium Limited)
Clarastrasse 12
4058 Basel
Tel: 061 666 68 68
Fax: 061 666 68 69
Email: info.ch@altium.com

France – Protel AG

(A subsidiary of Altium Limited)
121 rue d'Aguesseau
F-92100 Boulogne-Billancourt
Ph: 0800 88 05 06
Fax: 0800 82 85 92
Email: info.fr@altium.com