Philips XA Family

SOFTWARE DEVELOPMENT TOOLSET

TASKING™

HIGHLIGHTS

- Total integrated development environment
- Easy project setup and management
- Can be tailored to your own environment
- Highly optimizing compiler
- Proven and stable technology, best selling XA compiler
- Basic and advanced debugging
- Kernel aware debugging
- Seamless integration with 8051 tools
- Available on PC/Windows, PC/Linux, Sun/Solaris, and HP/UX

TAKE ADVANTAGE OF THE EXTENDED ARCHITECTURE

The Philips XA architecture is a family of true 16-bit microcontrollers that provides upward compatibility from the industry-standard 8-bit 80C51. With its extensive experience in the 8051 market, TASKING has developed a toolset that eases the migration from 8051 and enables your application to take advantage of the XA extended architecture.

The complete toolset includes a C++/EC++ compiler, ANSI C compiler, macro assembler, linker/locator, libraries, CrossView Pro debugger, and EDE, our Embedded Development Environment that provides a composite interface to the complete toolset.

EMBEDDED DEVELOPMENT ENVIRONMENT

With TASKING's EDE, you can create and maintain projects easily, so your application is always up to date. All aspects of a project are saved in a project file, such as the source files that make up your application, the tool options (compiler, assembler, linker/locator, CrossView Pro debugger), the tool directories, and the options that describe the building process. File dependencies as well as the sequence of operations required to build your application are handled automatically. EDE offers you very productive features for application and code development.

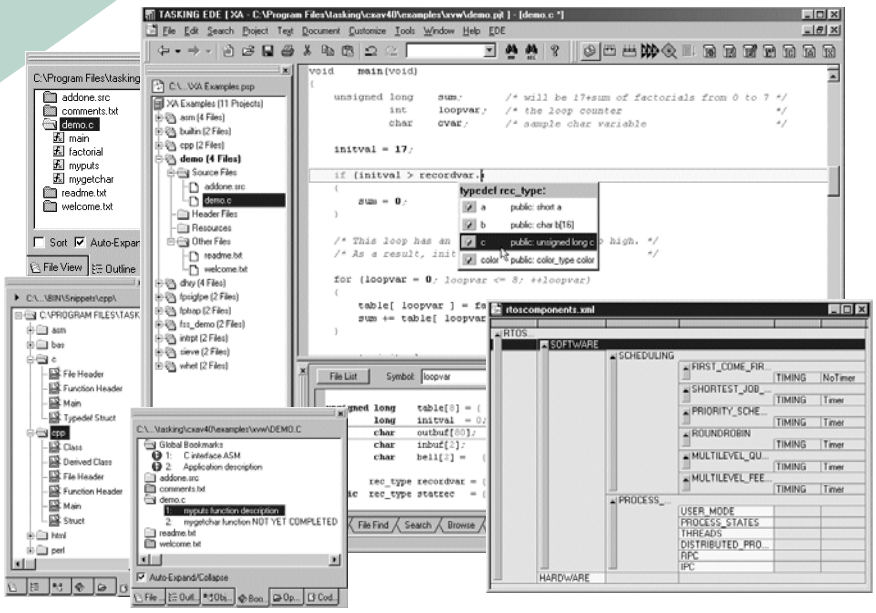
Easy/Expert modes allow simplified configuration of the TASKING tools and the XA target processor for the less experienced user. After switching to Expert mode, all advanced options become available.

Project Spaces allow grouping of multiple projects in one view, thus offering project management for more complex developments.

CodeSense advanced coding assistance offers rich type-ahead features, which help you to select the next expected function parameter or available structure members. When positioning your mouse pointer over a function name, the function prototype will be displayed.

XA DERIVATIVES SUPPORTED

- XA-C3/C37 (CAN)
- XA-G1/G2/G3x (General Purpose)
- XA-G49 (Flash)
- XA-H3/H4 (Telecom/Control)
- XA-S3 (I^2C)
- XA-SCC (Telecom)
- ArtistIC (TV Sets)
- SmartXA (Smart Cards)



Tags Browsing offers a graphical overview of the application's cross references and allows easy navigation through the available variables and functions.

CodeFolio offers easy insertion of template code, thus adding to coding efficiency and consistency. It allows macro expansion and prompted input as you insert the code.

XML Collapsible Grid Viewer displays a hierarchy of the elements and element attributes in an XML document.

HTML View Window has been integrated, to allow browsing through the product manuals, your project or code documentation or even surfing the Internet.

Make your text smart - source code is no longer text only

Insert button links in text files to perform special actions:

- **View related documents or diagrams**
- **Run macros and applications**
- **Create pop-up notes**
- **View categories and lists of bookmarks and links**

Your source code remains ASCII and compatible with other editors.

Productive editor extensions

Tailor the editor to your needs and wishes:

- **Integrated FTP utility**
- **HTML editor and viewer to browse the Internet or Intranet**
- **Develop your own extensions and integrate them through DLL's**
- **Wide range of various extensions available from fellow developers**

Right-Mouse-Button clicks expedite a variety of tasks within EDE (e.g., creating new files, adding files to a project, etc.)

Split Windows provide full control over source code, by allowing you to split your file horizontally or vertically to up to four interactive edit windows.

See the text boxes on the left and right for more EDE features.

C++/EC++ COMPILER

For several years, TASKING has been the only vendor to offer object-oriented design and coding possibilities for the XA family through an ISO C++ compliant compiler. The advantages of C++ can be incorporated into an existing C application, one module at a time, providing a graceful migration from C to C++.

Inheritance reduces the number of places where software behavior is defined and thereby speeds up development. The C++ compiler automatically includes a pre-link phase when templates are used.

POWERFUL EDITING

- **Hexadecimal editing**
- **Compile/track errors**
- **Background symbol compilation**
- **Print preview**
- **Difference editing for side by side code comparisons**
- **Syntax highlighting**
- **Smart indenting**
- **Multiple clipboards and scrap buffers. "Clip View" to display the contents**
- **Re-definable keyboard**
- **Keystroke recording and playback**
- **Run a command prompt in a buffer**
- **Configuration Wizards to walk you through several editor features**
- **Select text by stream, column, or line**
- **Drag-and-drop to load files from Explorer**
- **File size and scrap buffer up to 2GB**
- **Maximum number of buffers limited only by memory and disk size**
- **Spell check recognizes comments and string constants**
- **True soft word wrapping (without reformatting your code)**
- **Customize menu, toolbars, and pop up menus**
- **Automate your processes using macros**
- **Version control interface to various standard products**
- **CUA, BRIEF, Epsilon, and Vi key maps**

TASKING's compilers are tested for ISO or ANSI conformance against authoritative validation suites such as Plum Hall and Perennial. Additionally, the optimization techniques of the compilers are tested with various large real-world applications as well as industry benchmark standards such as Nullstone and EEMBC.

Language Extensions

In addition to full ANSI C compliance, the C compiler offers a wealth of specific language extensions for embedded XA based applications.

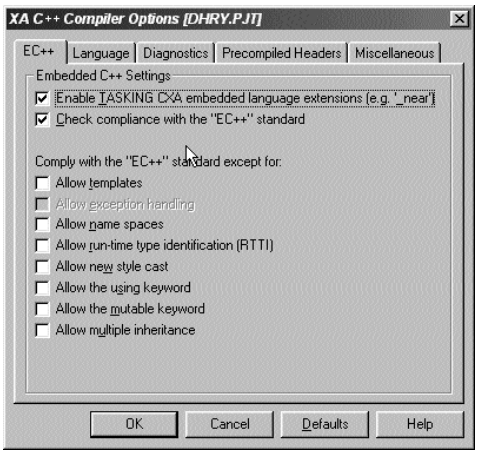
- **Additional data types, such as _sfrbit, _sfrbyte, _bit and _bitbyte**
- **The _at() function and _atbit() for easy allocation of variables at specific locations**
- **A wealth of XA specific intrinsic functions**
- **Easy C-level interrupt definition using _interrupt**
- **User definable inline functions**

Safer C

Based on the "Guidelines for the use of the C language in vehicle based software" published by the Motor Industry Software Reliability Association (MISRA®), TASKING is the first to implement the Safer C concept in a software development environment. Safer C guides programmers in writing more robust C code by defining selectable C usage restriction rules. Through a system of strict code checking, the use of error-prone C-constructs can be prevented.

Scalable C++

Compatibility with the evolving Embedded C++ (EC++) standard allows selective disabling of C++ features that may not be essential for your embedded application. By selecting (partial) compliance to the EC++ standard, codesize overhead and run-time inefficiency can be minimized.



C COMPILER

Based upon TASKING's renowned expertise in the field of C-compiler technology, the XA C compiler is highly optimized, taking full advantage of the architecture and complying fully with the ANSI standard.

A predefined configuration for compliance with the MISRA guidelines is available with a single click. It is also possible using pull-down menus to enable a custom set of Safer C rules to suit specific company requirements.

To ensure compliance with the Safer C rules throughout the entire project, the Linker/Locator can generate a Safer C Quality Assurance report. This report lists the different modules in the project with the respective Safer C configurations that were used to compile them.

So, under the guidance of Safer C in the TASKING tool chain, programmers can now

write better, more maintainable code that contains less error-prone C constructs, which will lead to more robust and safer embedded systems.

C COMPILER

- ANSI C ensures early error detection
- Complete XA support
- Intelligent configuration of system parameters
- Extensive user controlled mapping of memory, code and data
- Built-in Safer C code checker
- In-line assembly
- IEEE-754 single and double precision floating point
- Complete ANSI C and run-time libraries
- IEEE-695 object format to ensure interoperability with third party products

Powerful Optimizations

TASKING's XA C compiler tools implement a wide variety of optimizations to allow reduction of code and data size as well as execution time. Optimizations can be applied on the complete project or specific files, or they can be switched on/off at function or source line level.

Optimizations include:

- **Various loop and jump optimizations to speed up execution and/or reduce code size.**
- **Common sub expression elimination detects and eliminates repeating (sub-) expressions.**
- **Common tail merging for finding duplicate sequences of code and merging them together to reduce code size.**
- **Dead assignment, dead storage and dead code elimination removes all kinds of unreachable code or invariant data.**
- **Peephole optimizations replace instruction sequences with equivalent but faster and/or shorter sequences, or delete obsolete instructions.**

Memory Models

The C compiler supports several reentrant memory models: tiny, small, compact, medium, large, and huge. Each model uses a different default storage type for (non-register) automatic variables, (non-register) parameter passing areas, and declarations without explicit storage type. Separate versions of the C and run-time libraries are supplied for all supported models, avoiding the need to recompile or rebuild these when using a particular model.

Interrupt Functions

C functions can serve as interrupt service routines by specifying the interrupt number via the *_interrupt* function qualifier. The *_using* function qualifier can be used to define the value of the PSW placed in the interrupt vector table. The compiler emits the corresponding interrupt vector and the appropriate entry and exit code. The *_trap* function qualifier can be used for functions to define that a function is called via a software trap instruction. The keyword *_trap* is allowed with function declarations and function prototypes. Contrary to an interrupt function, a function declared with *_trap* can have parameters and can have a return value.

Intrinsic Functions

The XA Compiler has a number of built-in (intrinsic) functions that enable you to access specific XA instructions directly in C, instead of writing inline assembler. In total, the XA compiler supports 32 intrinsic functions.

Key intrinsic functions include:

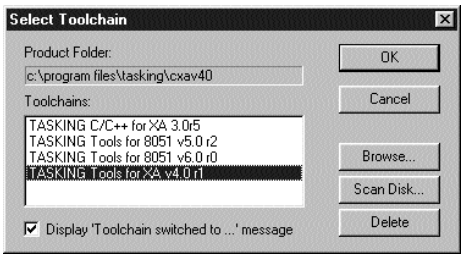
_ror/_rol	**Generate rotate instructions**
_strmovc	**Copy string from code space ROM to data space RAM**
_nop	**Generate NOP instruction**
_testclear	**Test and clear bit using the JBC instruction**
_div32	**32-bit by 16-bit signed divide using div. instruction**

Libraries

The compiler package includes ANSI C libraries, run-time libraries including I/O calls (+ printf), memory management, arithmetic functions, and floating point for all memory models. Floating point libraries are delivered in many different variants. You can choose between a double precision implementation (full ANSI-C) or the faster single precision, and you can switch between full IEEE-754 exception handling or a faster version without the trapping. Source code provided for most of the library routines allows you to tailor the libraries to your specific application.

Migration from 8051 to XA

Migrating your 8051 applications to the XA has never been so easy! For compatibility with the C51 compiler, many include files and command options and controls have been integrated. An assembly code translator is included to convert your existing 8051 code to XA code. The translator generates XA code through a "best possible match" approach, and it issues warnings in cases where architectural differences between the 8051 and the XA prevent direct translation.



The 8051 and XA tools integrate seamlessly into one environment. Switching from 8051 to XA and vice versa is hassle free and can be performed instantly.

ASSEMBLER

The assembler is an optimizing assembler that offers macro preprocessing facilities and translates XA assembly language into relocatable object code. The assembler is supplied complete with linker/locator, librarian, and object format utilities. An absolute or executable load image is obtained by using the linker/locator.

Linker/Locator

The linker/locator is an essential part of the software building process that enables linking and location of data and code into the target memory. The locator will locate a linker file to absolute addresses. The ability to accurately describe the available memory and control the behavior of the locator is crucial for successful development of embedded applications.

Linker/Locator features include:

- **Automatic or user-defined allocation of code and data in memory**
- **Advanced overlaying features allow efficient memory usage**
- **Incremental linking**
- **IEEE695 object output format with HLL debugging information**
- **SREC / Intel HEX output format for (E)PROM programmers**

Librarian and Make

The Librarian creates a library, adds object modules to a library, removes object files from a library, and lists the contents of a library. Make is a utility that automates the task of building or reconstructing your application. It prevents errors by ensuring that applications can be accurately rebuilt and saves time by recompiling only modules that have changed since the last build. Make is invoked from the EDE by clicking on the Make button in the toolbar and performs its tasks invisibly. Alternatively, Make can be called from the command line for full control in non-GUI environments.

CROSSVIEW PRO DEBUGGER

An easy-to-use interface with powerful and extensive debugging features helps you debug your applications faster. The CrossView Pro debugger is a true Windows application complete with multiple, resizable, and independently controlled windows. It combines the flexibility of the C language with the control of code execution found in assembly language, bringing functionality that reduces the time spent testing and debugging.

Functionality includes:

- **Bubble-Spy™ for quick and easy inspection of variables and functions**
- **Large Smartbuttons to maximize the viewable debugger workbench**
- **Tracking scope and monitoring locals**
- **"Intelligent" source window**
- **Double click and right mouse button functions**

You choose the windows you need to view the different aspects of your code during debugging.

Source Window

The working window is the source window. It lets you view source; set and clear breakpoints, assertions and code coverage markers; monitor and inspect variables; search for strings, functions, lines and addresses; call functions; evaluate expressions; and view performance analysis data. The source window allows you to view your code at C level or assembly level, or you can choose a mixed mode that allows you to simultaneously view your C code intermixed with its corresponding assembly code.

From the source window, you can jump directly into the editor within EDE, and you will find yourself positioned at the source line where you had your cursor in the debugger.

Multiple Data Windows

Data windows enable you to watch or show data, browse for locals or globals, double-click to modify values, or expand and contract complex data structures. Within these windows you can reformat (change display radix and type) on an element-by-element basis. You can show or watch locals from any stack level, automatically track and display locals, and easily copy any variable to a new window as show or watch.

Register Window

The register window displays and modifies CPU register values. The window is fully configurable and is updated every time the program is stopped. Highlighted registers indicate what has changed since the last stop.

Stack Window

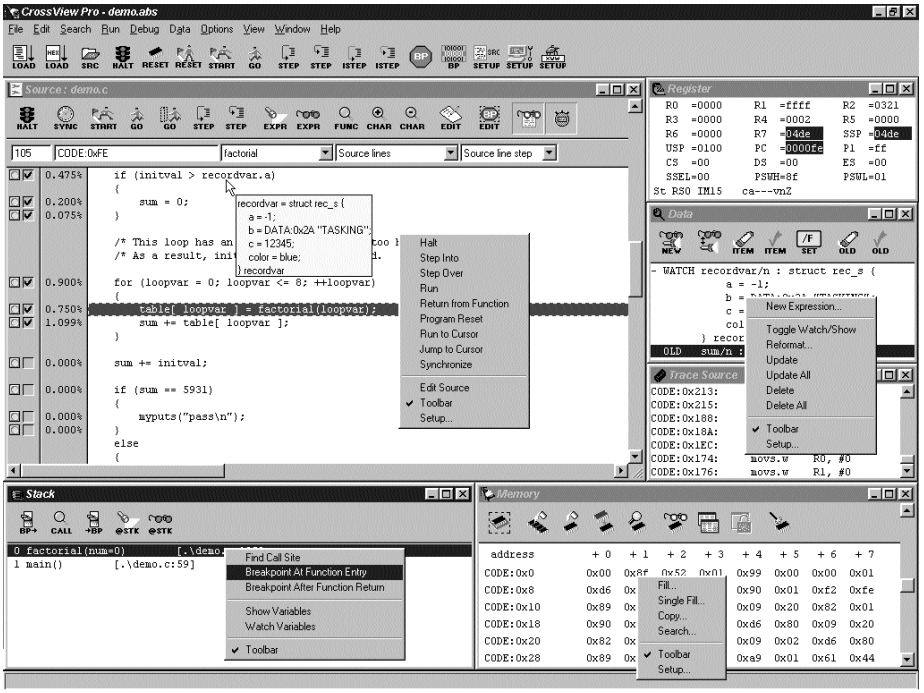
The stack window displays the state of the current stack frame. With simple point-and-click operations, you can set up level breakpoints, display source for function calls, and display local variables for selected functions.

Multiple Memory Windows

The memory window with ASCII display enables you to monitor any address change, double-click to modify, and have complete control over the size and format of data.

Coverage and Profiling

With coverage you can check whether the code of your application is reached (executed at least once) or not. Coverage helps you build a complete test suite for your product, which improves the quality of your application. The CrossView Pro debugger also supports coverage of data regions. Profiling allows you to analyze the performance of your application. You can see how the total time is divided among the C functions and which functions should be optimized for speed.



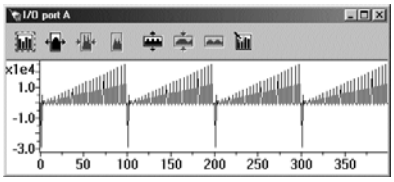
Software Assertions
Software assertions let you execute user-specified command lists after running every line of source code. This is the software equivalent of data breakpoints, and can be used to set up sophisticated error checking mechanisms that uncover the most elusive bugs.

C-Like Macro Language
With C-like macro language, you can read and/or modify application variables and call application functions from macros. It has full C expression syntax.

File System Simulation
CrossView Pro File System Simulation (FSS) allows the use of standard ANSI C system calls such as *open()*, *read()*, *printf()* and *scanf()* within your embedded application in order to interface with the host computer file I/O services.

Using FSS, you can read from and write to files on the host computer system or a CrossView Pro Virtual I/O window, directly.

Programmable Data Analysis
Programmable Data Analysis enables quick detection of gross errors in your signal processing routines by reducing large sets of data into meaningful visual diagrams. CrossView Pro can analyze the data according to pre-defined or user-defined specifications, and display the data the way you need it. This eliminates the need for reviewing or post-processing large files of raw data. You can also view the same set of data in several ways at the same time (e.g., in the time and the frequency domains).



Multiple Execution Environments
The CrossView Pro debugger supports two execution environments with a standard user interface: a ROM Monitor and a Simulator.

ROM Monitor Environment
The CrossView Pro ROM debugger can be used with any commercial off-the-shelf evaluation board or target application. The CrossView Pro debugger, running on a host computer system, communicates with the monitor on the target board via an RS232 interface using a very efficient protocol. The resources used by the monitor program are kept to a minimum.

The monitor uses:

- **Approximately 3Kbytes code space**
- **Less than 25 bytes of data space**
- **Less than 24 bytes of system stack**

Simulator
The simulator environment allows you to test, debug, and monitor the performance of code in a known and repeatable environment independent of target hardware. It uses the same description file as the linker/locator when locating your application and therefore knows exactly where and how memory is mapped.

All CrossView Pro features, including C level trace, Code/Data Coverage, performance analysis (profiling), and unlimited amount of code and data breakpoints, are available to you, so you can test code before target hardware is available.

Open Architecture
The CrossView Pro debugger supports a truly open architecture by providing public interfaces and supporting industry standards. Public interfaces such as the Kernel Debug Interface (KDI) and Generic Debug Instrument Interface (GDI) provide third party vendors easy access and interface to the CrossView Pro debugger framework. The KDI can also be used to provide kernel aware debugging for your "in-house" kernel! Visit our website for more information.

COOPERATION WITH THIRD PARTIES
Working with other suppliers of products for the Philips XA architecture gives us the opportunity to improve the tools that we deliver and to create a total solution concept for XA application development.

TASKING's XA tools are supported by emulator manufacturers such as Ashling, Ceibo, Lauterbach, and Nohau.

Future Designs, Inc. (FDI) and TASKING have teamed up to offer plug-and-play support for FDI's XTEND evaluation boards. FDI provides its XTEND boards with the TASKING monitor programmed in EPROM or on-chip. This allows the user to connect to the CrossView Pro ROM monitor debugger via a serial connection to the XTEND board and run the application monitored through CrossView Pro. Evaluation boards from Phytec are supported in a similar manner.

The XA toolset is compatible with the realtime kernels from CMX (CMX), Embedded Power Corporation (RTXC™), and Micrium (µC/OS-II).

With MicroNet™ from CMX and EMIT® from emWare we offer facilities to connect your XA application to the Internet.

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

TASKING™

PRODUCT PACKAGING & ORDERING CODES

Each TASKING product comes with full documentation in comfortable binders. The documentation is available on-line as well and provides full-text search capabilities for quick and easy lookup of topics.

The XA Development Toolset is available for PC/Windows, PC/Linux, Sun/Solaris and HP/UX.

Product Code	Package contents
07-200-012-002	EDE, C Compiler, Assembler/Linker, CrossView Pro Simulator
07-200-012-024	EDE, C/C++ Compiler, Assembler/Linker, CrossView Pro ROM Monitor and Simulator Debugger

Demonstration versions of the XA tools are downloadable from our web site at: www.tasking.com/XA

Developers forum : www.tasking.com/forum

ALTIUM OFFICES WORLDWIDE

North America

Altium Inc.
3207 Grey Hawk Court
Suite 100
Carlsbad, CA 92010
Ph: +1 760-231-0760
Fax: +1 760-231-0761
sales.na@altium.com
support.na@altium.com

Germany

Altium Europe GmbH
Philipp-Reis-Straße 3
76137 Karlsruhe
Ph: +49 721 8244 300
Fax: +49 721 8244 320
sales.de@altium.com
support.eu@altium.com

China

Altium Information Technology
(Shanghai) Co., Ltd
9C, East Hope Plaza
No.1777 Century Avenue
Shanghai 200122
Ph: +86 21 6182 3900
Fax: +86 21 6876 4015
sales.cn@altium.com
support.cn@altium.com

Japan

Altium Japan K.K.
Nomura Fudosan Yotsuya Bldg 7F
2-12-1 Yotsuya
Shinjuku-ku, Tokyo
160-0004
Phone: +81 3 6672 6155
Facsimile: +81 3 6672 6159
sales.japan@altium.com
support.japan@altium.com

The Netherlands

Altium Technology Centre
Altium BV
Saturnus 2
3824 ME Amersfoort
Ph: +31 33 4558584
Fax: +31 33 4550033
tasking@altium.com

www.tasking.com