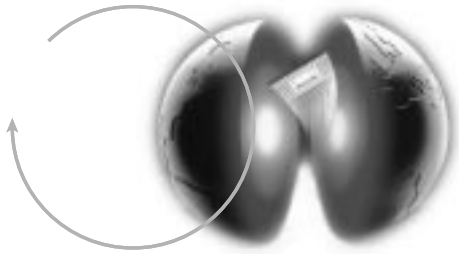


The 251 and 8x930 USB Development Solution



Plug into the power of 8x930 USB

The 251 is the second generation 8051 microcontroller that brings high performance, an increased memory mix and addressing with low power. The TASKING software development tools for the 251 take full advantage of the enhanced instruction set and the efficient high level language support that the microcontroller offers.

The toolset delivers compatibility with existing 8051 software and provides an additional level of performance that cannot be achieved by the hardware alone. The 251 toolset supports all members of the family including the Intel 8x930 USB controller and the Temic TSC80251 family.

The 251 Toolset

The TASKING software development toolset for the 251 provides a complete and cost-effective solution for programming the 251 family of microcontrollers. The complete toolset includes two C compilers, 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

The Embedded Development Environment (EDE), voted by EETimes 'Best Technology', is a single, easy to use interface, that integrates the members of the 251 toolset enabling you to edit, build and debug your embedded applications.

EDE provides the following:

- Integrated tools delivering a

rapid edit-compile-debug process

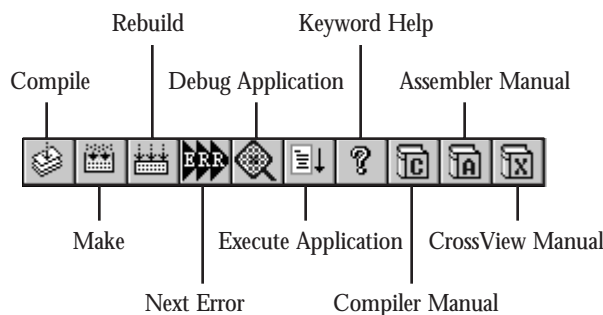
- Easy project setup with generation of Makefiles
- Push button control over a variety of development tasks
- Language sensitive editor with error parser
- Access to third party tools

EDE is project oriented helping you organize your work by grouping together all the files associated with your project, handling dependencies, and identifying the 251 as well as the tool options.

C Compilers

The complete toolset consists of two ANSI C compilers: the Starter C compiler and the Power C compiler. The Starter compiler is designed for compatibility with C 51 programming models, memory spaces and assembly calling conventions. When using this C compiler you can share C and assembly code between 251

and 8051 projects or 8x930 and 8x931 USB projects. The Power compiler is designed and built specifically for the 251 architecture: new memory models, other register usage, other calling conventions and a new software floating point implementation.



Features

- Total integrated development environment
- Tight and seamless navigation
- Tailorable to your working environment
- Visual point & click interface
- Highly optimized Compiler
- Kernel aware debugging
- Available on Windows 3.1, Windows 95, Windows NT, UNIX (SUN and HP9000)
- Proven and stable technology

The Power compiler gives you the ultimate performance for your 251 project. Both compilers allow your application to run in either source or binary mode and take full advantage of the microcontroller's architecture.

This means you can access all the special features of the 251 in C without violating the ANSI standard, such as Multiple addressing modes (with full pointer support), Extended bit memory, Special function registers (I/O ports), Interrupt functions, User in-line C functions (create your own in-line functions) and much more.

General features of our C compiler include:

- Full ANSI C to ensure early error detection
- Built-in functions (251 intrinsics)
- IEEE-754 single and double precision floating point
- Generates Intel compatible assembly source
- Complete ANSI-C libraries in C source
- Generates reentrant and relocatable code and data
- Full OMF-251 and IEEE-695 object formats to ensure interoperability with third party debuggers and emulators
- Intelligent configuration of system parameters: CPU mode, wait states, external memory configuration, startup code etc.
- Extensive user controlled mapping of memory areas, code and data.

Data Types

All ANSI types are supported.

In addition to these types, *_bit*, *_sfrbyte* and *_sfrbit* are added.

The keywords *_sfrbyte* and *_sfrbit* are available to access special function registers which deal with I/O. SFR's are treated like memory mapped variables declared with the volatile type qualifier.

Memory Models

The C compiler supports 2 data models and 4 program models. The data model determines the default memory type (*_near* or *_far*) for pointers and global variables (declared with no explicit memory type), strings and floating point constants. With the program model you tell the compiler which jump and call (2K, 64K or 16M range) instructions can be used.

Parameter Passing

The C compiler passes the initial arguments via 8 CPU registers. Passing parameters in CPU registers significantly improves system performance.

If additional parameters are involved, either the stack

(*_reentrant* qualifier) or overlayable fixed memory locations (*_dataparm* or *_nearparm* qualifier) are used. All functions and libraries are reentrant by default. Non-reentrant code uses directly addressable memory and yields the fastest program execution.

Switch Statement

The C compiler supports three different methods for implementing a switch statement. You can control how the compiler should optimize the switch statement with a #pragma (jump chain, jump table or lookup table).

Libraries

The libraries include ANSI C libraries, run time libraries including I/O calls (+ printf), memory management, arithmetic functions and floating point.

You can choose between a double precision implementation (full ANSI-C) or a faster single precision one and between full IEEE-754 exception handling or a faster version without the trapping.

Assembler

The TASKING assembler accepts Intel compatible assembler source programs and produces relocatable (.obj) object files.

Features of the assembler include:

- Control to optimize generic 251 instructions
- Control to specify Source or Binary mode
- Supports segment overlay at the assembly level
- Extensive segment directives and listing files
- Error file with textual error reporting
- Full debugging support

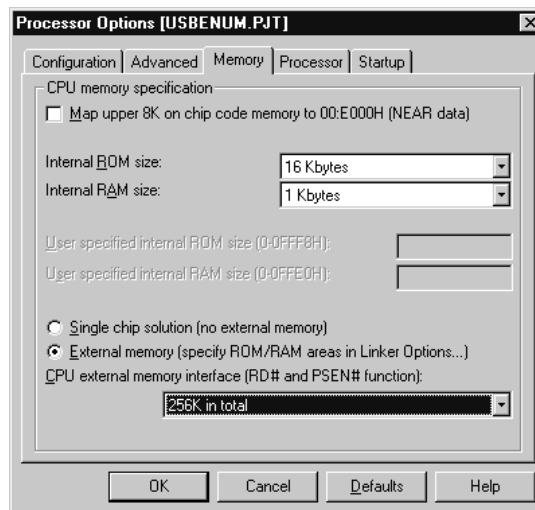
Linker/Locator

The linker/locator is an essential part of the software building process that enables you to configure the code to match your target environment. Features include:

- Automatic segment overlaying using call graph information from the compiler and the assembler

Efficiency

- Very efficient pointer arithmetic and dereferencing
- In-line assembly
- Extremely fast parameter passing and automatic variables using 8, 16 and 32 bit CPU registers
- Fast 8 bit arithmetic, including true char and *_bit* type parameters



- Generation of listfile including function call graph of whole application
- Automatic inclusion of corresponding C and floating point libraries
- Extensive map files and diagnostic messages

CrossView Pro

- Multi-window interface
- Code coverage
- Profiling
- Code and data breakpoints with optional macro execution
- Kernel awareness
- Single stepping
- Stack trace
- Simulated I/O
- C expression evaluation

CrossView Pro Debugger

An easy-to-use interface with powerful and

extensive debugging features help you debug your applications faster. CrossView Pro 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:

- 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, assembly level or mixed level, where you have your C code intermixed with the 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. This gives you immediate access to the source line where you have found a problem, that you want to correct.

Multiple Data Windows

Data windows enable you to watch or show data, browse for locals or globals, double-click to modify values or to

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, setup level breakpoints, display source for function calls, and display local variables for selected functions.

Multiple Memory Windows

Memory windows with hex and ASCII display enable you to monitor and modify any memory location, having complete control over the size and format of the data, as well as view coverage of the memory range.

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. CrossView Pro also supports coverage of data



Assembler

- Intel compatible macro assembler
- Incremental linking
- ANSI-C intermodule type checking
- Supports absolute OMF-251, IEEE-695, Intel Hex and Motorola S-record object format

regions. Profiling allows you to analyse 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.

C-Like Macro Language

With C-like macro language you can read and/or modify application variables and associate macros with breakpoints. Macros have full C expression syntax and can be assigned to user toolbox buttons.

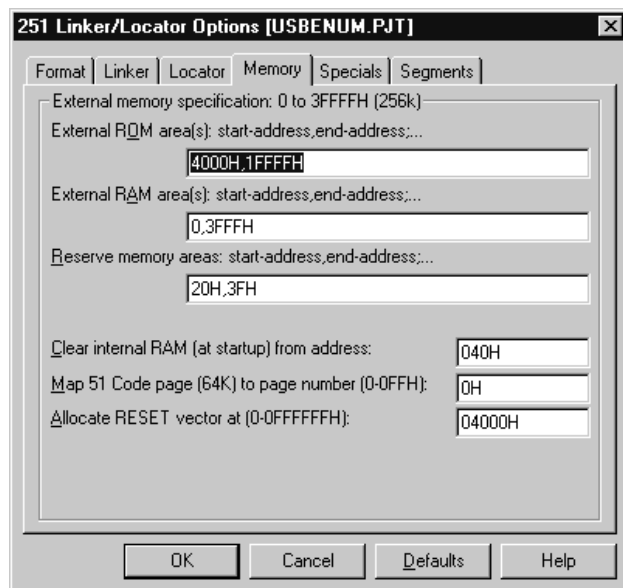
Multiple Execution Environments

CrossView Pro supports multiple execution environments with a standard user interface.

The execution environments available are Simulator and ROM Monitor.

ROM Monitor Environment

CrossView Pro ROM debugger can be used with any commercial off the shelf evaluation board (including the Intel 8x930 USB and Temic 251 A1/G1) or target



application running RISM (Reduced Instruction Set Monitor). CrossView Pro, 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

- 16 bytes of internal RAM
- timer and serial interrupt (if external UART is not used).

All files required to build the monitor are shipped as source with CrossView Pro ROM.

Simulator

The simulator uses the same control 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 Coverage, Profiling, and unlimited amount of code/data breakpoints are available to you.

Cooperation with 3rd Parties

Working with other suppliers of products for the 251 gives us the opportunity to improve the tools that we deliver. We cooperate with different hardware manufacturers for Emulators and Evaluation boards, and with software manufacturers for Real Time Kernels.

Our extensive cooperation ensures you have access to the tools you need to be your most productive.

For more information contact your local sales office or distributor. The 251 toolset is compatible with the realtime kernels from CMX (CMX) and Embedded System Products (RTXC). More kernel support is to be expected. Please contact us for availability or visit our web site for up-to-date information.

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.

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 at the end of the maintenance period.

These maintenance agreements provide you with all program enhancements released during the period of the contract, and assures a response to all problem reports submitted.

Availability

The 251 development solution is available for PC (Windows 3.1, 95 and NT), Sun (Solaris) and HP9000/700 (HPUX). We have a policy of continued improvement for our products. For the latest information contact your local sales office or distributor.

Quick Reference - Language Implementation

TASKING C Language Extensions

<i>_bit</i>	You can use data type <i>_bit</i> for the type definition of scalars in the 8051 bit area (020H-02FH) and for the return type of functions.
<i>_ebit</i>	You can use the data type <i>_ebit</i> for the type definition of scalars in the 251 extended bit area (020H-07FH).
<i>_sfrbit</i>	Data type for the declaration of specific, absolute bits in special function registers or special absolute bits in the SFR address space.
<i>_sfrbyte</i>	Data type for the declaration of Special Function Registers.
<i>_at</i>	You can specify a variable to be at an absolute address.
<i>_atbit</i>	You can specify a variable to be at a bit offset within any (extended) bit addressable variable.
<i>storage</i>	You can specify a memory type in each declaration: <i>_data</i> , <i>_bdat</i> , <i>_ebdat</i> , <i>_idat</i> , <i>_pdat</i> , <i>_xdat</i> , <i>_rom</i> , <i>_near</i> , <i>_far</i> , <i>_huge</i> .
<i>memory specific pointers</i>	You can define pointers to a specific target memory. These types of pointers are very efficient and require 1, 2 or 4 bytes of memory space.
<i>parameter passing</i>	You can specify where to allocate the non- register parameters of a function with a function qualifier: <i>_dataparm</i> , <i>_nearparm</i> , or <i>_reentrant</i> (default).
<i>reentrant functions</i>	Reentrant functions are using the stack and can be invoked recursively or by interrupt functions.
<i>interrupt functions</i>	You can specify interrupt functions directly through interrupt vectors in the C language (<i>_interrupt</i>).

Pragma	Method
<i>linear_switch</i>	Use a jump chain (if/else-if/else-if/else)
<i>jump_switch</i>	Use a jump table (fastest)
<i>binary_switch</i>	Use a lookup table (many case statements)
<i>smart_switch</i>	Let the computer decide the method

Data Model	Default Memory Type
small <i>_near</i>	(64 Kbyte direct and indirect memory addressing, 00:0000- 00:FFFF)
large <i>_far</i>	(16 Mbyte indirect memory addressing, object size < 64K, 00:0000-FF:FFFF)

Pointers

<i>_idat</i>	1 byte pointer (256 bytes) to indirectly address able on chip RAM (same as pointer to <i>_data</i> , <i>_bdat</i> and <i>_ebdat</i>)
<i>_near</i>	2 byte pointer (64 Kbytes), dereferenced via @WRJ operand (same as pointer to <i>_xdat</i> and <i>_rom</i> , when using small data model)
<i>_far</i>	4 byte pointer (16 Mbytes) dereferenced via @DRK operand, fast 16-bit pointer arithmetic (same as pointer to <i>_xdat</i> and <i>_rom</i> when using large data model)
<i>_huge</i>	4 byte pointer (16 Mbytes) dereferenced via @DRK operand, full 32-bit pointer arithmetic

Memory Types

<i>_data</i>	128 byte directly addressable on chip RAM, fast access (00:0000-00:007F).
<i>_bdat</i>	16 byte bit-addressable on chip RAM, allows bit and byte access (00:0020-00:002F).
<i>_ebdat</i>	96 byte extended bit-addressable on chip RAM (00:0020-00:007F).
<i>_idat</i>	256 byte indirectly addressable on chip RAM (00:0000-00:00FF).
<i>_near</i>	64 Kbyte direct and indirect memory addressing (00:0000- 00:FFFF).
<i>_pdat</i>	256 byte area in external RAM (01:xx00-01:xxFF)
<i>_xdat</i>	64 Kbyte external RAM (8051 XDAT page, 01:0000-01:FFFF).
<i>_rom</i>	64 Kbyte internal/external ROM (8051 CODE page, FF:0000- FF:FFFF).
<i>_far</i>	16 Mbyte indirect memory addressing, object size < 64K (00:0000-FF:FFFF)
<i>_huge</i>	16 Mbyte indirect memory addressing, object any size (00:0000-FF:FFFF)

Function	Method
<i>_testclear</i>	Test and clear bit (JBC instruction)
<i>_nop</i>	Generate NOP instruction
<i>_rol</i>	Rotate character left (RL instruction)
<i>_ror</i>	Rotate character right (RR instruction)

Program Model	C module code size	Application code size
small	<2K	<2K
medium	<2K	<64K
large	<64K	<64K
huge	<64K	<16M

Product Packaging and Ordering Codes

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

Product Code	Package contents
TK035-022	EDE, Compiler, Assembler/Linker, CrossView Pro Simulator
TK035-041	CrossView Pro ROM Monitor Debugger and Simulator
TK035-024	Special combination package: TK035-22 and TK035-041

Contact TASKING for availability.

Demonstration versions of the 251 and 8x930 USB tools are available on CD-ROM or downloadable from our web site at:
<http://www.tasking.com>

Your Distributor

TASKING assumes no responsibility for any errors which may appear in this document. TASKING retains the right to make changes to the specification at any time without notice. Contact your local sales office to obtain the latest information.

UNITED STATES (International Headquarters)

TASKING, Inc.
Norfolk Place, 333 Elm Street
Dedham, MA 02026-4530. USA
Phone: 800-458-8276 (within the USA)
781-320-9400 (outside the USA)
Fax: 781-320-9212
Email: sales_us@tasking.nl

THE NETHERLANDS (European Headquarters)

TASKING Software BV
P.O. Box 899
3800 AW Amersfoort. The Netherlands
Phone: +31 33 4 558584
Fax: +31 33 4 550033
Email: sales_nl@tasking.com

GERMANY

TASKING GmbH
Brennerstraße 5
71229 Leonberg, Germany
Phone: +49 7152 97991 0
Fax: +49 7152 97991 20
Email: sales_de@tasking.com

ITALY

TASKING Srl
Via Napo Torriani 29
20124 Milano. Italy
Phone: +39 2 6698 2207
Fax: +39 2 6698 2189
Email: sales_it@tasking.com

JAPAN

Nihon TASKING K.K.
Shiba-ST Building, 6th floor
1-15-13 Shiba
Minato-ku
Tokyo 105, Japan
Phone: +81-3-3457-6831
Fax: +81-3-3457-6834
Email: sales@tasking.co.jp

UNITED KINGDOM

TASKING Ltd
Enterprise House
Ocean Village
Southampton, Hampshire SO14 3XB
United Kingdom
Phone: +44 1703-334774
Fax: +44 1703-334772
Email: sales_uk@tasking.com

INTERNET

<http://www.tasking.com>