COSMIC C Cross Compiler for ST Microelectronics STM8

COSMIC's C cross compiler, cxstm8 for the ST Microelectronics STM8 family of microcontrollers, incorporates over twenty years of innovative design and development effort.

cxstm8 is field tested, reliable, and incorporates many features that help ensure your embedded STM8 design meets and exceeds performance specifications.

The **C Compiler** package includes : an optimizing C cross compiler, macro assembler, linker, librarian, object inspector, hex file generator, object format converters, debugging support utilities, run-time library binary and source code, and a compiler command driver.

cxstm8 includes I.D.E.A. Windows editor to provide a complete Integrated Development Environment under Windows. It offers instant access to a fully-integrated editor, project manager, compiler, linker, utilities, and the highly intuitive source-level debugger ZAP.

cxstm8 can be combined with ZAP/STM8 for PC/Windows to provide a complete C language development and debugging environment for the STM8.

Key Features

Supports All STM8 Family Microcontrollers
ANSI C Implementation
Extensions for Embedded Systems
Global and Processor-Specific Optimizations
Optimized Function Calling
C support for Zero Page Data
C support for Bit Variables
C support for Interrupt Handlers
Memory Models optimized for small (<64k) or big (>64k) applications
In-Line Assembly
Single Precision Float Support
Absolute C and Assembly Listings
Royalty-Free Library Source Code
First Year of Support Service Included
No Charge Upgrades

Microcontroller Specific Design

cxstm8 is designed specifically for the ST Microelectronics STM8 family of microcontrollers; all STM8 family processors are supported. A special code generator and optimizer targeted for the STM8 family eliminates the overhead and complexity of a more generic compiler. You also get header file support for many of the popular STM8 peripherals, so you can access their memory mapped objects by name either at the C or assembly language levels. C level support is provided for Short Addressing and Bit Variables.

ANSI / ISO Standard C

This implementation conforms with the ANSI and ISO Standard C specifications which helps you protect your software investment by aiding code portability and reliability.

C Runtime Support

C runtime support consists of a subset of the standard ANSI library, and is provided in C source form with the binary package so you are free to modify library routines to match your needs. The basic library set includes the support functions required by a typical embedded system application. All runtime library functions are ROMable.

Runtime library functions include :

- Character handling.
- Mathematical functions.

- Formatted serial input/output.
- String handling.
- Memory management.

The package provides both an integer-only library as well as the standard single precision floating point library. This allows you to select the smaller and faster integer-only functions, if your application does not require floating point support.

Memory Models for different applications

cxstm8 provides 2 different memory models depending on the size of the application. For applications smaller that 64k, the “section 0” memory model provides the best code density by defaulting function calls and pointers to 2 bytes. For applications bigger than 64k, the standard memory model provides the best flexibility for using easily the linear addressing space. Each model comes with its own set of libraries.

Optimizations

cxstm8 includes global and microcontroller specific optimizations to give your application maximum chance of meeting and exceeding its performance specifications. You retain control over optimizations via compile-time options and keyword extensions to ANSI C, so you can fine tune your application code to match your design specification:

- *cxstm8* supports global optimizations which allow it to optimize whole C functions as well as C statements.
- Peephole optimizer further optimizes *cxstm8's* output by replacing inefficient code sequences with optimal code sequences for the STM8.
- Function arguments can be passed char-sized without widening to *int*.
- Commonly used static data can be selectively, using the @tiny keyword, or globally, using a compile-time option, placed into zero page memory (the first 256 bytes of memory) to decrease access time.
- The Code Factorization optimization replaces duplicated chunks of code by grouping them into subroutines.
- Assembler instructions rearrangement allows to avoid pipeline stalls for the best performance.
- Strict single-precision (32-bit) floating point arithmetic and math functions. Floating point numbers are represented as in the IEEE754 Floating Point Standard.
- Other optimizations include: branch shortening logic, jump-to-jump elimination, constant folding, elimination of unreachable code, removal of redundant loads/stores, and switch statement optimizations.

Controller Specific Extensions to ANSI C

cxstm8 includes several extensions to the ANSI C standard which have been designed specifically to give you maximum control of your application at the C level and to simplify the job of writing C code for your embedded STM8 design :

- You can define in-line assembly using _asm() to insert assembly instructions directly in your C code to avoid the overhead of calling assembly language subroutines.
- Also you can use #asm/#endasm to insert assembly instructions directly in your C code.
- You can declare bit variables (_Bool type) packed into bytes by the compiler for global and local variables. The bit instruction will be used wherever possible.
- You can define C functions as interrupt handlers using the @interrupt keyword. Compiler saves volatile registers for handling exceptions and interrupts.
- You can define a C object or C function to have an absolute address at the C-level, using the @<address> syntax appended to you data definition; this is useful for interrupt handlers written in C and for defining memory mapped I/O.
- You can define *char*- and *int*-sized bitfields, and select bit numbering from right-to-left or left-to-right.

Additional Compiler Features

- Full C and assembly source-level debugging support.
- Absolute and relocatable listing file output, with interspersed C, assembly language and object code; optionally, you can include compiler errors and compiler optimization comments.
- Extensive and useful compile-time error diagnostics.
- Fast compile and assemble time.
- Full user control over include file path(s), and placement of output object, listing and error file(s).
- All objects are relocatable and host independent. (i.e. files can be compiled on a workstation and debugged on a PC).
- Function code and switch tables are generated into the code *(.text)* section. Constant data such as string constants and *const* data are generated into a separate program *(.const)* section.
- Initialized static data can be located separately from uninitialized data or data initialized to zero.
- All function code is (by default) never self-modifying, including structure assignment and function calls, so it can be shared and placed in ROM.
- Code is generated as a symbolic assembly language file so you can examine compiler output.

- cxstm8 creates all its tables dynamically on the heap, allowing very large source files to be compiled.
- Common string manipulation routines are implemented in assembly language for fast execution.

Assembler

The COSMIC STM8 assembler, castm8, supports macros, conditional assembly, up to 255 named program sections, includes, branch optimizations, expression evaluation, relocatable or absolute output, relocatable arithmetic, listing files and cross references. The assembler also passes through line number information, so that COSMIC's ZAP debugger can perform full source-level debug at the assembly language level.

Linker

The COSMIC linker, clnk, combines relocatable object files created by the assembler, selectively loading from libraries of object files made with the librarian, clib, to create an executable format file.

clnk features:

- Flexible and extensive user-control over the linking process and selective placement of user application code and data program sections.
- Multi-segment image construction, with user control over the address for each code and data section. Specified addresses can cover the full logical address space of the target processor with up to 255 separate segments. This feature is useful for creating an image which resides in a target memory configuration consisting of scattered areas of ROM and RAM.
- Generation of memory map information to assist debugging, including the full call tree.
- All symbols and relocation items can be made absolute to prelocate code that will be linked in elsewhere.
- Symbols can be defined, or aliased, from the Linker command File.

Librarian

The COSMIC librarian, clib, is a development aid which allows you to collect related files into one named library file, for more convenient storage. clib provides the functions necessary to build and maintain object module libraries. The most obvious use for clib is to collect related object files into separate named library files, for scanning by the linker. The linker loads from a library only those modules needed to satisfy outstanding references.

Object Module Inspector

The COSMIC object module inspector, cobj, allows you to examine library and relocatable object files for symbol table and file information. This information is an essential aid to program debugging.

- Symbol table cross referencing.
- Section sizes of the individual program sections can be printed for object and library files.
- Program segment map: lists all program segments, their sizes, absolute addresses and offsets.

Absolute Hex File Generator

The COSMIC hex file generator, chex, translates executable images produced by the linker to one of several hexadecimal interchange formats for use with most common In-Circuit

Emulators and PROM programmers :

- Standard Intel hex format.
- Motorola S-record and S2 record format.
- Rebiasing of text and data section load addresses. Allows you to generate hex files to load anywhere and execute anywhere in the target system address space.

Absolute C and Assembly Language Listings

Paginated listings can be produced to assist program understanding. Listings can include original C source code with interspersed assembly code and absolute object code. Optionally, you can include compiler errors and optimization comments.

Debugging Utilities

The cross compiler package includes utility programs which provide listings for all debug and map file information. The clst utility creates listings showing the C source files that were compiled to obtain the relocatable or executable files. The cprd utility extracts and prints information on the name, type, storage class and address of program static data, function arguments and function automatic data.

Third Party Debugging Support

You can use cxstm8 or castm8 with ZAP/SIM or with the debuggers provided by most popular In-Circuit Emulator manufacturers. cxstm8 also supports several additional debugging formats including ELF/DWARF, IEEE-695 format.

Packaging

The compiler comes with complete user documentation and is available on standard distribution media.

Support Services

All COSMIC Software products come with the first year of support included in the price. You will receive a courteous and prompt service from our technical support staff and you retain control of the severity of the problem i.e. if it's a problem that is critical to your project we guarantee you a response time of one to three business days depending on the severity of the problem. Service is provided during normal business hours via email, fax or telephone and is unlimited while you have a valid annual support agreement. New releases of the software are provided free of charge to support customers.

Ordering Information

cxstm8 package product codes are as follows :

Host System Product Code :

PC (DOS/Windows)	CXSTM8-PC
SUN SPARC (SunOS/Solaris)	CXSTM8-SUN
LINUX	CXSTM8-LIN

Orders are shipped within one week of receipt of hard copy purchase order. Call your sales contact for license fees and multiple copy discounts.

Other COSMIC Software Products

COSMIC Software products focus on 8, 16 and 32-bit microcontrollers. C-Compiler/debugger support is available for a wide range of target processors. For more information on the ZAP C and assembler source-level debugger, ask for the ZAP Product Description and demo disk.

Tool Customization Services

Some customers have special tool needs and through COSMIC's tool customization service, you have the ability to control the core tool technology to help solve your technical and/or business problems. COSMIC works closely with you to understand, define and implement technical solutions according to your needs and schedule.

For Sales Information, please contact:

Europe and Other International

COSMIC Software FRANCE
33 Rue Le Corbusier, Europarc
94035 Creteil Cedex, FRANCE
Tel. : +33 (0) 1 43 99 53 90
Fax : +33 (0) 1 43 99 14 83
E-mail: sales@cosmic.fr
Web Site: http://www.cosmicsoftware.com

COSMIC Software UK
Oakwood House
Wield Road
Medstead
Alton
Hampshire GU34 5NJ, ENGLAND
Tel. : +44 (0) 1420 563498
Fax : +44 (0) 1420 561946
E-mail: sales@cosmic.co.uk
Web Site: http://www.cosmicsoftware.com

COSMIC Software GmbH
Rohrackerstr. 68
D-70329 STUTTGART, GERMANY
Tel. : +49 (0) 711 4204062
Fax : +49 (0) 711 4204068
E-mail: sales@cosmic-software.de
Web Site: http://www.cosmicsoftware.com

USA and Canada

COSMIC Software, Inc.
400 West Cummings Park, STE 6000
Woburn MA 01801-6512
Tel. : +1 781-932-2556
Fax : +1 781-932-2557
E-mail: sales@cosmic-us.com
Web Site: http://www.cosmic-software.com