COSMIC C Cross Compiler
for Motorola 68HC08 Family

COSMIC's C cross compiler, ***cx6808*** *for the Motorola 68HC08 family of microcontrollers, incorporates over fifteen years of innovative design and development effort. In the field since 1995,* ***cx6808*** *is reliable, field-proven and incorporates many features to* ***help ensure your embedded 68HC08 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. ***cx6808*** includes integration files for Premia's **Codewright®** Windows editor to provide a complete Integrated Development Environment under Windows® 3.1, Windows 95 or Windows NT. ***cx6808*** can be combined with **ZAP/6808** for PC/Windows or **ZAP/6808** for OSF Motif® to provide a complete C language development and debugging environment for the 68HC08, on a PC or UNIX® workstation.

Key Features

Supports All 68HC08 Family Microcontrollers,
ANSI C Implementation,
Extensions to ANSI for Embedded Systems,
Global and Processor-Specific Optimizations,
Fully Optimized Code can be Debugged Without Any Changes,
C Support for Internal EEPROM,
C Support for Direct Page Data,
C Support for Code Bank Switching,
C Support for Interrupt Handlers,
User-defined Code/Data Program Sections,
Three In-line Assembly Methods,
Full User Control Over Stack Use,
Single Precision Float Support,
Motorola MCUasm™ Compatible Assembler,
Absolute C and Assembly Listings,
32-bit Version Supports Long Filenames on Windows 95 and Windows NT,
Static Analysis of Stack Usage,
Royalty-Free Library Source Code,
Works With Popular 68HC08 In-Circuit Emulators,
First Year of Support Service Included,
No Charge Upgrades.

Microcontroller-Specific Design

cx6808, is designed specifically for the Motorola 68HC08 family of microcontrollers; all 68HC08 family processors are supported. A **special code generator and optimizer** targeted for the 68HC08 family eliminates the overhead and complexity of a more generic compiler. You also get header file support for all the popular 68HC08 peripherals, so you can access their memory mapped objects by name either at the C or assembly language levels.

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 and reentrant**. Runtime library functions include:

- Character handling
- Mathematical functions
- Non-local jumps
- 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.

Optimizations

cx6808 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:

- fully optimized code can be debugged, without change, using COSMIC's **ZAP/MMDS08** or **ZAP/SIM08** debuggers,
- ***cx6808*** supports global optimizations which allow it to optimize whole C functions as well as C statements,
- Peephole optimizer further optimizes ***cx6808***'s output by replacing inefficient code sequences with optimal code sequences for the 68HC08,
- C functions can be declared to not use a stack using the **@nostack** keyword, or as a compile-time option; function arguments and local data are then placed in private or shared static memory in either direct page (.bsct) or the .bss data section and stack usage is minimized. When local data is placed in direct page memory, application execution times and code sizes can be reduced by up to 10% compared to the default stack model,
- Function arguments are passed in registers when possible, and *char*-sized data can be passed without widening to *int*,
- Commonly used static data can be selectively, using the **@tiny** keyword, or globally, using a compile-time option, placed into direct page memory (the first 256 bytes of memory) which reduces access times and byte counts,
- The 68HC08 bit instructions (*bclr,bit,bset,brclr,brset*) are used extensively for bit operations,
- Arithmetic operations are performed in *char* precision if the types are 8-bit,
- Strict single-precision (32-bit) floating point arithmetic and math functions. Floating point numbers are represented as in the IEEE 754 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.

Extensions to ANSI C

cx6808 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 68HC08 design:

- The **@far** keyword can be attached to a C function declaration to instruct the compiler to generate special function call/return sequences to allow bank-switching of C code. The 68HC08 EBI-style bank-switching scheme is supported by default, but all support routines are provided in source form and can be customized to custom hardware schemes,
- The **@eeprom** modifier can be attached to a C data declaration to inform the compiler that the data object resides in 68HC08 EEPROM space; the compiler will automatically generate the required code sequence when writing to the EEPROM location,
- The **_asm()** statement can be used to insert assembly instructions directly in your C code to avoid the overhead of calling assembly language functions. **_asm()** statements can only be used within C function code and can be used in C expressions,
- Arguments can be passed into **_asm()** assembly language statements to allow access to C local variables from assembly language code,
- Assembly language statements can be inserted inside or outside of C functions using **#pragma asm .. #pragma endasm** or the alias **#asm .. #endasm**,
- User-defined program sections are supported at the C and assembler levels: **#pragma section <*name*>** declares a new program section *name* for code or data which can be located separately at link time,
- The **@interrupt** keyword can be attached to a C function definition to declare the function as an interrupt service routine. The compiler preserves volatile registers not already saved by the processor,
- **@<address>** syntax allows an absolute address to be attached to a data or function definition; this is useful for interrupt handlers written in C and for defining memory mapped I/O,
- *char*- and *int*-sized bitfields can be defined, and bit numbering from right-to-left or left-to-right can be selected,

Additional Compiler Features

- **Full C and assembly source-level debugging** support. There is no limit on the size of the debug section,
- 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 *.const* program section,
- Initialized static data can be located separately from uninitialized data or data initialized to zero,
- All function code is (by default) reentrant, 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,
- ***cx6808*** 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.

68HC08 Assembler

The COSMIC 68HC08 assembler, ***ca6808***, **conforms to the standard Motorola syntax** as described in the document *Assembly Language Input Standard*; ***ca6808*** 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. Listing files can optionally include cycle counts. 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,
- ***clnk*** analyzes stack usage for local variables and function arguments and the function calling hierarchy to provide a **static analysis of stack usage** which helps you understand how much stack space your application needs,
- 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,
- 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,
- Tektronix standard and extended hex formats,
- 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.

Bank Packing Utility

For users using bank-switching, the ***cbank*** utilty program takes a user-specified list of object files, and creates an output file, which can be input to the clnk linker, that contains an ordered list of object files so as to minimize memory holes at bank boundaries. This utility saves the user from having to figure out which modules can fit into which bank.

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 ***cx6808*** or ***ca6808*** with **ZAP/SIM08 and ZAP/MMDS08** or with the debuggers provided by most popular In-Circuit Emulator manufacturers. ***cx6808*** also supports several additional debugging formats including HP64700, **IEEE-695** and P&E map file format.

Packaging

The compiler comes with complete user documentation and is available on standard distribution media. For PC-host development systems, two compiler packages are available: (1) for 16-bit DOS/Windows 3.1 systems and (2) for 32-bit DOS/Windows 95/Windows NT systems. Package (2) compiles code faster than package (1) and provides support for long filenames.
For PC hosts, files to integrate the C compiler with the Codewright code/project editor and an error parser are included on a separate disk. Once these files are installed, the user can compile/assemble/link from within Codewright and automatically detect and fix errors with a simple mouse-click.

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 E.S.T. 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

cx6808 package product codes are as follows:

Host System	*Product Code*
PC (16-bit DOS/Windows 3.1)	CDSH08
PC (32-bit DOS/Windows 95/NT)	CWSH08
SUN SPARC(SunOS/Solaris)	CSSH08
HP9000(HPUX)	CHSH08

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 Motorola 8,16 and 32-bit microcontrollers. C compiler/debugger support is available for **68HC05, 68HC08, 6809, 68HC11, 68HC12, 68HC16, 683XX and 680X0**. For more information on the **ZAP** C and assembler source-level debugger, ask for the **ZAP Product Description** and demo disk.

The Motorola Microcontroller Specialists.

COSMIC Software

For Sales Information please contact:

COSMIC Software, Inc USA
(Eastern & Central time zones)
400 West Cummings Park, Suite 6000
Woburn, MA 01801-6512 USA
Phone: (781) 932-2556 Fax: (781) 932-2557
Email: sales@cosmic-us.com
web: http://www.cosmic-software.com

(**Pacific & Mountain time zones**)
159 Sage Hills Road
Orange, CA 92869
Phone: (714) 289-0693 Fax: (714) 289-0694
Email: laird@cosmic-us.com

COSMIC Software France
33 Rue Le Corbusier, Europarc Creteil
94035 Creteil Cedex France
Phone: + 33 4399 5390 Fax: + 33 4399 1483
Email: sales@cosmic.cosmic.fr

COSMIC Software UK
Worting House Basingstoke
Hampshire, United Kingdom RG23 8PY
Phone: + 44 01256 843400 Fax: + 44 01256 843773
Email: sales@cosmic.co.uk

COSMIC Software GmbH
Rohrackerstr 68 D-70329 Stuttgart Germany
Tel.+ 49 (0)711 4204062 Fax + 49 (0)711 4204068
Email: aw.COSMIC@t-online.de

COSMIC Software Scandinavia
Sgbergsliden 5b S-414 63 Goteborg Sweden
Tel: 46 31 704 3920 Fax: 43 31 704 3921
Email: cosmic@cosmic.cybertrans.se