COSMIC C Cross Compiler for Motorola 68HC11 Family

*COSMIC's C cross compiler, **cx6811** for the Motorola 68HC11 family of microcontrollers, incorporates over twenty years of innovative design and development effort. In the field since 1986 and previously sold under the Whitesmiths brand name, **cx6811** is reliable, field-tested and incorporates many features to **help ensure your embedded 68HC11 design meets and exceeds performance specifications.***

The **C Compiler** package for Windows includes: COSMIC integrated development environment (IDEA), optimizing C cross compiler, macro assembler, linker, librarian, object inspector, hex file generator, object format converters, debugging support utilities, run-time libraries and a compiler command driver. The PC compiler package runs under Windows 95/98/ME/NT4/2000 and XP.

Key Features

Supports All 68HC11 Family Microcontrollers
ANSI C Implementation
Extensions to ANSI for Embedded Systems
Global and Processor-Specific Optimizations
Optimized Function Calling
C support for Internal EEPROM
C support for Direct Page Data
C support for Code Bank Switching
C support for Interrupt Handlers
Three In-Line Assembly Methods
User-defined Code/Data Program Sections
User-defined Control Over Stack Usage
Single and Double Precision Float Support
Motorola MCUasm™ Compatible Assembler
Assembler supports C #defines
Absolute C and Assembly Listings
Static Analysis of Stack Usage
Royalty-Free Library Source Code
Works With Popular 68HC11 In-Circuit Emulators
IEEE-695, ELF/DWARF and P&E Debug support
First Year of Support Service Included with
No Charge Upgrades

Microcontroller-Specific Design

cx6811, is designed specifically for the Motorola 68HC11 family of micro-controllers; all 68HC11 family processors are supported. A special **code generator and optimizer** targeted for the 68HC11 family eliminates the overhead and Complexity of a more generic compiler. You also get header file support for many of the popular 68HC11 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 **double and single precision floating point libraries**. This allows you to select the smaller and faster integer-only functions, if your application does not require floating point support.

Optimizations

cx6811 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/MON11** and **ZAP/SIM11** debuggers or third-party debuggers that read IEEE695 format files,
- ***cx6811*** supports **global optimizations** which allow it to optimize whole C functions as well as C statements,
- **Peephole optimizer** further optimizes ***cx6811***'s output by replacing inefficient code sequences with optimal code sequences for the 68HC11,
- 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,
- Functions which require a stack can be marked, using the **@fast** keyword, as requiring fast entry/exit code sequences in which case ***cx6811*** will generate entry/exit code in-line rather than by calling a machine library,
- ***cx6811*** can reorder function local data so that the most referenced locals are allocated stack space closest to the function frame pointer for fast access,
- 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 **@dir** keyword, or globally, using a compile-time option, placed into direct page memory (the first 256 bytes of memory) to decrease access time,
- ***cx6811*** makes full use of the IX and IY index registers for addressing and pointer operations,
- The 68HC11 bit instructions (bclr,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) or strict double-precision (64-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

cx6811 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 68HC11 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. 68HC11K4 and 68HC11C0 style bank-switching schemes are 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 68HC11 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 (8-bit), int (16-bit)* or *long int (32-bit)* bitfields can be defined, and bit-numbering from right-to-left or left-to-right can be selected,

Additional Compiler Features

- Full C 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,
- 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,
- ***cx6811*** creates all its tables dynamically on the heap, allowing very large source files to be compiled,
- Unused variables can be flagged with an error message.
- Common string manipulation routines are implemented in assembly language for fast execution.

68HC11 Assembler

The COSMIC 68HC11 assembler, ***ca6811***, **conforms to the standard Motorola syntax** as described in the document *Assembly Language Input Standard*; ***ca6811*** supports macros, conditional assembly, includes, branch optimizations, expression evaluation, relocatable or absolute output, relocatable arithmetic, listing files and cross references. **Assembler accepts C syntax for #includes and #defines** so include files can be shared between C and Assembly modules. The assembler also creates full debug information, so that debuggers 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 ***cx6811*** or **ca6811** with **ZAP/SIM and ZAP/ROM. CX*6811*** also supports several standard debugging formats for use with third party debuggers and logic analyzers. Supported Debug formats include **IEEE-695, ELF/DWARF** and P&E map file format.

Packaging

All compiler packages are provided on standard CD-ROM with complete on-line user documentation in Adobe PDF format.

The **C Compiler** package for Windows includes: An integrated development environment (IDEA), optimizing C cross compiler, macro assembler, linker, librarian, object inspector, hex file generator, object format converters, debugging support utilities, run-time libraries and a compiler command driver. The PC compiler package runs under Windows 95/98/ME/NT4/2000 and XP.

The Windows version also includes integration files for Starbase's popular **CodeWright®** Windows® code and project editor and GNU make utility.

The **C Compiler** package for UNIX includes: *An* optimizing C cross compiler, macro assembler, linker, librarian, object inspector, hex file generator, object format converters, debugging support utilities, run-time libraries and a compiler command driver. The UNIX compiler package is available for SUN Solaris, HP-UX and PC Linux.

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 software releases are provided free of charge to support customers.

Ordering Information

cx6811 package product codes are as follows:

Host System	*Product Code*
PC MS Windows	CWSH11
- Windows 95/98/ME/NT4/2000/XP	
SUN SPARC(SunOS/Solaris)	CSSH11
HP9000(HPUX)	CHSH11
PC Linux	CLXH11

Orders are shipped within one week of receipt of hard copy purchase order. Call our sales department 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.**

For Sales Information please contact:

COSMIC Software USA
COSMIC Software, Inc.
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: www.cosmic-software.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
web: www.cosmic.fr

COSMIC Software UK
Oakwood House
Wield Road, Medstead
Alton, Hampshire
GU34 5NJ, U.K.
Phone: +44 (0)1420 563498
Fax: +44 (0)1420 561946
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: sales@cosmic-software.de
web: www.cosmic-software.de