COSMIC C Cross Compiler for STMicroelectronics ST7 Family

*COSMIC's C cross compiler, **cxST7** for the STMicroelectronics ST7 family of microcontrollers, incorporates over fifteen years of innovative design and development effort. Used in the field since 1997, **cxST7** is reliable, field-proven, and incorporates many features to **help ensure your embedded ST7 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 ST7 Family Microcontrollers
ANSI C Implementation
Extensions to ANSI for Embedded Systems
Global and Processor-Specific Optimizations
Optimized Function Calling
Debug Fully Optimized Code
Automatic Checksums
Real and Simulated Stack models
Smart Linker optimizes RAM usage
C support for Zero Page Data
C support for Interrupt Handlers
Three In-Line Assembly Methods
Assembler Supports C #defines and #includes
Single Precision Float Support
Absolute C and Assembly Listings
Royalty-Free Library Source Code
IEEE-695 and ELF/DWARF Debug support
Works With all Popular ST7 In-Circuit Emulators
First Year of Support Service Included
No Charge Upgrades

Microcontroller-Specific Design

cxST7, is designed specifically for the STMicroelectronics ST7 family of microcontrollers; all ST7 family processors are supported. **A special code generator and optimizer targeted for the ST7 family** eliminates the overhead and complexity of a more generic compiler. You also get header file support for many of the popular ST7 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** syntax specifications which helps you protect your software investment by aiding code portability and reliability.

Flexible User Interface

The Cosmic C compiler can be used with the included Windows IDEA or as a Windows 32-bit command line application for use with your favorite editor, make or source code control system. It's your choice!!

Automatic Checksum

The linker can automatically create and maintain a multiple segment check sum mechanism in your application. Choose either an 8 or 16-bit checksum algorithm. Call one of the included checksum verify functions from any part of your application to calculate and compare the checksums.

Function Copy (Boot loader)

Create a block of functions that is stored in ROM and copied to RAM by a included library routine. Multiple blocks may be setup and copied and/or executed independently. Separate copies of library routines may also be included in the RAM code without symbol conflicts.

Smart Linker

The compiler allocates function local variables and arguments in ST7 internal memory at fixed locations. The linker optimizes allocation of these areas by sharing those areas reserved for functions that never call each other. This minimizes RAM usage.

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
- Non-local jumps
- Formatted serial input/output
- String handling
- Memory management

The compiler package includes 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.

Optimized Function Calls

cxST7 is able to optimize calls to the most used functions in your application by placing a jump table of addresses to these functions in direct page memory of the ST7, so that the function can be called using direct addressing instead of extended addressing, which saves one byte per function call.

Optimizations

cxST7 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** debuggers,
- ***cxST7*** supports **global optimizations** which allow it to optimize whole C functions as well as C statements,
- **Peephole optimizer** further optimizes ***cxST7***'s output by replacing inefficient code sequences with optimal code sequences for the ST7,
- 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

cxST7 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 ST7 design:

- The **@near** type modifier defines data to be allocated in external memory. By default all data is allocated into the ST7 on-chip internal memory,
- 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,
- Support stack models, where the internal stack is used to store function locals and arguments, and memory models, where the stack is simulated in RAM,
- 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),
- Initialized static data can be located separately from uninitialized data,
- All function code is (by default) never self-modifying, including structure assignment and function calls, so it can be placed in ROM,
- Code is generated as a symbolic assembly language file so you can examine compiler output,
- ***cxST7*** 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.

ST7 Assembler

The COSMIC ST7 assembler, ***caST7*** supports macros, conditional assembly, 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,
- ***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 sections. 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. ***clib*** can also be used to group application object files together for a partial link.

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,
- 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 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 ***cxST7*** and/or ***caST7*** with **ZAP ST7 SIM, ZAP DVP and ZAP HDS2.** **cxST7** also supports several standard debugging formats for use with third party debuggers and logic analyzers. Supported Debug formats include **IEEE-695 and ELF/DWARF.**

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 and Windows NT4/2000/XP.

The Windows version also includes integration files for Starbase's popular **CodeWright®** Windows® code editor and project manager along with a 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 releases of the software are provided **free of charge** to support customers.

Ordering Information

cxST7 package product codes are as follows:

Host System	*Product Code*
PC MS Windows Windows 95/98/ME/NT4/2000/XP	CWSST7
PC Linux	CLXST7
SUN SPARC(SunOS/Solaris)	CSSST7
HP9000(HPUX)	CHPST7

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 Compilers/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.

COSMIC 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.

COSMIC Software

Supporting Embedded Innovation since 1983

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