COSMIC C Cross Compiler
for Melexis
MLX16
Family

COSMIC's C cross compiler for the Melexis MLX16 family of microcontrollers is part of an enhanced compiler product line incorporating over ten years of innovative design and development. The COSMIC C cross compiler is field tested, robust, and reliable. It incorporates many features tailored specifically for the embedded systems developer.

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

All COSMIC compilers support non-intrusive C source-level debugging with ZAP for Windows and OSF Motif. COSMIC Cross developement tools are available for several popular host development systems including PC, SUN SPARC and HP9000/700 UNIX workstations.

Key Features

Microcontroller Specific Design
ANSI C Implementation
Full Reentrancy and Recursion
Single Precision Float Support
Royalty-Free Library Source Code
C support for direct pages
In-line Assembly
C support for Interrupt Handlers
Host Independent Relocatable Object Format
Extensions for Embedded Systems
Absolute C and Assembly Listings

Microcontroller-Specific Design

The COSMIC MLX16 C cross compiler is designed specifically for the Melexis MLX16 family of micro-controllers. A special code generator and optimizer targeted specifically for the MLX16 family eliminates the overhead and complexity of a more generic compiler. We've also added header file support for many of the popular MLX16 peripherals, so you can access their memory mapped objects by name.

ANSI C

This implementation conforms with the ANSI and ISO Standard C specifications. Standard C is upward compatible with ANSI C but provides additional reliability features and aids for the embedded systems developer.

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. Support includes :

- 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 library. This allows you to select the smaller and faster integer-only functions, if your application does not require floating point support.

Optimizations

The COSMIC compiler for the Melexis MLX16 family includes many processor-specific optimizations which lead to more compact, faster programs. For example:

- Commonly used static data can be selectively, or globally, placed into direct pages (dp, ep) to take advantage of the fast direct addressing mode.
- The compiler allows you to disable the widening of integer types in an arithmetic expression whenever possible. As a result, the compiler will perform arithmetic operations in character precision if the types are 8-bit.
- Optimized function calling sequence for functions with arguments (*i.e.* the compiler passes the first argument to a function and the return value in a register).
- Character-sized function arguments can also be passed without widening.
- Extremely efficient floating point arithmetic support.
- 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

The COSMIC C compiler includes several extensions to the ANSI standard designed specifically for embedded systems programmers. Optional Extensions to the ANSI Standard include:

- 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.
- You can define C functions as interrupt handlers using the @interrupt keyword. Compiler saves volatile registers for handling exceptions and interrupts.
- char and int sized bitfields, with the ability to select bit numbering from right-to-left or left-to-right.
- You can define a C object or C function to have an absolute address at the C-level, using the @<address> syntax appended to your data definition; this is useful for interrupt handlers written in C and for defining memory mapped I/O.

Additional Compiler Features

- A compile-time option lets you include C source line number information (as well as the name, type, address, and storage class of program data) in the object file format for processing by either ZAP C source-level debugger, or some other debugging tool such as an in-circuit emulator (see Debugging Utilities).
- Function code, switch tables, and const data can be located separately in ROM.
- Initialized static data can be located separately in Random Access Memory (RAM). Uninitialized data can be placed in the BSS section.
- All function code is 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 which is fed to the COSMIC MLX16 assembler.
- Floating point numbers are represented as in the IEEE 754 Floating Point Standard. The compiler supports single precision software floating point operations and math functions.
- The compiler creates all its tables dynamically on the heap, allowing large source files to be compiled.
- Common string manipulation routines are implemented in assembly language for fast execution.

Assembler

The compiler package includes a complete relocatable macro assembler, *camlx16*. The assembler for the MLX16 implements all of the MLX16 instructions and addressing modes using standard Melexis syntax. The assembler also supports the following:

- Automatic branch optimization (optional).
- Command line defines and include files.
- Conditional assembly with if and else directives.
- Nested include files.
- Nested macros with multiple arguments.
- Relocatable and constant expression evaluation.
- Produces Assembly listings.
- Cross reference table lists each symbol with its value, attributes, line number of definition and a list of functions and line numbers where it is referenced.

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

- Flexible and extensive user-control over the linking process and selective placement of code, data and BSS program sections.
- Multi-segment images can be constructed, with user control over the address for each text, data, and BSS section. The 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.
- Symbols can be defined, or aliased, from the linker command File.

- 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.
- Command line input can come from the command line or from console input or from any combination of both. The command line can also be input from a file.
- Input filenames and the names of files loaded from library files can be entered into the symbol table. This allows the Object Module Inspector to provide better cross-referencing information.
- Automatic data initialization. The linker creates and locates a ROM image of the initialized data.

Librarian

The librarian, *clib*, is a development aid which allows you to collect related files into one named library file, for more convenient storage. It provides the functions necessary to build and maintain object module libraries. The most obvious use of the librarian 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. The librarian can also be used to collect arbitrary binary files into one place.

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. The *Clabs* utility creates absolute C and Assembly listings from the linked object. All addresses, offsets and object code are resolved and displayed with the corresponding C and Asssembly source.

Object Module Inspector

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

- Symbol table information printing: one entry per line, including symbol address, symbol name and symbol status. Symbol status indicates whether the symbol is defined or undefined and if defined, which program section it is defined in.
- Symbols can be sorted alphabetically by symbol name, or numerically by address.
- 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 hex file generator, *chex*, translates executable images produced by the linker to standard Melexis S-record and S2 record format for use with most common In-Circuit Emulators and PROM programmers.

- Motorola S-record and S2 record format.
- Standard Intel hex format.

Debugging Utilities

The cross compiler package includes a number of utilities which provide listings for all debug and map file information to allow both host and target C level cross debugging. A compile-time option lets you include full C source-level debug information including local variable support. The debug format is directly compatible with COSMIC's source-level cross debuggers (see ZAP product descriptions).

Clst prints out the contents of C source files, with line number information and the absolute addresses of the start of each source line after linking.

Cprd extracts and prints information on the names, types, storage class, and address (absolute or offset) of program static data and the arguments and automatics belonging to program functions.

Multi-Pass Command Driver

A multi-pass command driver is a standard feature. It reads a user-modifiable configuration file to control the entire compilation process. The driver supports command line C and/or assembly defines and include files. You can specify a one-line command (which includes user-specified options) to compile and assemble your code. This makes converting your source code to object format a lot simpler.

Interfacing C and Assembly Code

Choose the level of coding suitable to each part of your application. Call assembly language routines from C and vice-versa. The package documentation provides information on function calling conventions, register usage, stack frame layout, and data representation.

Debugging Support

COSMIC Software also supplies a complete line of C Source level debuggers for the MLX16, ZAP for Windows and OSF Motif. ZAP provides an exceptional graphical user interface and is available in simulation and emulation versions. Ask for the ZAP product description for details. The compiler also support several other debugging formats including IEEE695, P&E map file and HP64000 format.

Packaging

The compiler comes with complete user documentation and is available on standard distribution media. Integration files for the Codewright editor are included on a separate disk.

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

cxmlx16 package product codes are as follows :

Host System Product Code :

PC (DOS/Windows)	CXMLX16-PC
SUN SPARC (SunOS/Solaris)	CXMLX16-SUN
HP9000(HPUX)	CXMLX16-HP

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

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 More 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

COSMIC Software

Supporting Embedded Innovation since 1983