COSMIC C Cross Compiler
Add-on for
XGATE

COSMIC's C cross compiler for the Freescale XGATE co-processor 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 XGATE Compiler is an add-on to the standard HCS12X Compiler and is not available as a standalone product.

The **C Compiler** package includes: an optimizing C cross compiler, relocatable macro-assembler, run-time library source code, and a multi-pass compiler command driver. All the other standard tools are shared with the HCS12X Compiler utilities.

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

Royalty-Free Library Source Code

Intrinsic functions for special instructions

In-line Assembly

C support for Interrupt Handlers

Motorola Compatible Assembler

Host Independent Relocatable Object Format

Object directly linkable with HCS12X code

Extensions for Embedded Systems

Absolute C and Assembly Listings

First Year of Support Service Included

Microcontroller-Specific Design

The COSMIC XGATE C cross compiler is designed specifically for the Freescale XGATE co-processor. A special code generator and optimizer targeted specifically for the XGATE processor eliminates the overhead and complexity of a more generic compiler. The C code is direclty compatible with the native code written for HCS12X applications, including sharing of IO registers include files.

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. Support includes :

- Character handling
- Non-local jumps
- String handling

Optimizations

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

- 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).
- 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:

- The Compiler defines several intrinsic functions producing directly the special XGATE instructions such as CSEM, SSEM, SIF, …
- 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. The Compiler provides the proper interface setting the R1 register either as a single argument or as a frame pointer if several arguments are passed.
- Char, int and long 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).
- Initialized static data can be located separately in Random Access Memory (RAM). Uninitialized data can be placed in the BSS section. All data variables can be shared with HCS12X variables.
- Code is generated as a symbolic assembly language file which is fed to the COSMIC XGATE assembler.
- 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, *caxgate*, conforms to the standard Motorola syntax as described in the document *Assembly Language Input Standard*. The assembler for the XGATE implements all of the instructions and addressing modes using standard Motorola (MASM) syntax and directives. 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.

Startup

The XGATE code is executed from the HCS12X ram and then must be initialized at the application startup. The XGATE Compiler is provided with an extended startup file allowing the XGATE code and initialized data to be copied in ram before entering the main routine.

The XGATE object files are direclty compatible with HCS12X object files so applications using both HCS12X and XGATE can be built from a single linker command file.