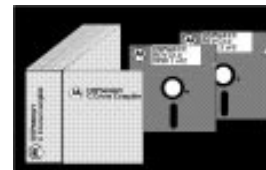


**MOTOROLA**  
**SEMICONDUCTOR**  
**TECHNICAL DATA**
**DSP56KCCx**

**DSP56000/1 High-Level Tools**
**Software Summary**
**DSP56KCCx**
**DSP56000/56001 C Cross Compiler**

The DSP56KCCx is a full ANSI C Cross Compiler supporting development of DSP56000 family applications. This software package allows the user to develop DSP56000/1 based application programs in a stand-alone or mixed-language programming environment (C and 56000 assembly). The software consists of an optimizing C Compiler, ANSI C Libraries, DSP56000/1 assembler, linker, librarian, GNU source level debugger and various utilities. The DSP56KCCx software can be run on a variety of host machines (IBM™ 386 PC, SUN-3™, SUN-4™ or NeXT™ workstation) and the developed application code can be executed on the multi-DSP56000 simulator or DSP56000/1 Application Development System (DSP56000 ADSx).

The DSP56KCCx is designed to provide the programmer with optimized assembly code output, to take full advantage of the DSP56000/1, to interface easy to the assembly language environment, C library routines, full-featured source level debugger, and to offer ANSI-portability. This product is based on the Free Software Foundation GNU C Compiler. These tools provide a powerful high-level environment to develop DSP56000/1 applications such as traditional DSP application, network control, audio and speech processing, embedded control, operating systems for the DSP56000/1, and real-time instrumentation.

**Part Description**
**DSP 56K CC x**

Host Machine (see Ordering Information)  
 C Cross Compiler  
 DSP56000 Family  
 Digital Signal Processor

**Ordering Information**

Host Platform	Operating System	Order Number
IBM™-386 PC/Compatible	DOS 3.x	DSP56KCCA
SUN-3™	SunOS™ 4.1.1	DSP56KCCC
SUN-4™	SunOS™ 4.1.1	DSP56KCCF
NeXT™	NeXT Mach™ 2.0	DSP56KCCG

**Package Contents**
**Programs**

g56k (optimizing C compiler)      dsplnk (linker)  
 gdb56 (symbolic debugger)      dsplib (librarian)  
 run56sim (DSP56000 simulator)      COFF utilities  
 asm56000(assembler)              srec

**Documentation**

G56K C Cross Compiler  
 GDB56 GNU Debugger  
 ANSI Libraries

This document contains information on a new product. Specifications and information herein are subject to change without notice.

IBM is a trademark of International Business Machines.  
 SUN-3, SUN-4 and SunOS are trademarks of Sun Microsystems, Inc.  
 NeXT is a trademark of NeXT Computer, Inc.  
 Mach is a trademark of Carnegie-Mellon University.


**MOTOROLA**

# Optimizing C Cross Compiler

DSP56KCCx is a full-featured ANSI Standard C Compiler that compiles one or more source files containing C statements into DSP56000/DSP56001 assembly code/codes that can be assembled by the DSP56000 assembler and linked by the DSP56000 linker.

Adhering to the ANSI standard increases portability of the C program and allows many existing programs to run on the DSP56000/1 architecture with minimal modification.

The mixed-language environment allows time-critical routines to be replaced with very fast assembly routines for optimum performance within a C language environment. Two distinct methods, in-line assembly and out-of-line assembly are supported by the Optimizing C Cross Compiler.

This compiler is based on the technology developed by the Free Software Foundation.

Some of the optimization techniques employed are:

- Automatic Register Promotion
- Common Sub-expression Elimination
- Strength Reduction
- Code Hoisting
- Loop Rotation & Invariant Elimination
- Dead Code Elimination
- Address Register Promotion for array reference
- Leaf Routine Detection
- Do Loop Optimization
- Tail Recursion Elimination

The GNU Source Level Debugger, GDB56, is included. With this debugging tool, application programs can be easily debugged and tested in the emacs environment. This can significantly reduce development time and effort.

The Common Object File Format (COFF) is used by the C Cross Compiler and as input to the GNU Symbolic Debugger. All Motorola DSP software tools support COFF file and will directly use the output of the C Cross Compiler.

The following illustration shows an example of how to use the C Cross Compiler to run a simple hello.c program which prints the message "Hello, DSP." on the screen.

A variety of options are supported by the C Cross Compiler including those options available with GNU C. This feature will make portability easier in the sense that the make utility can be easily used.

```
shelltool - /bin/csh
motorola ls
hello.c
motorola cat hello.c
#include <stdio.h>
main()
{
    printf("Hello, DSP.\n");
}
motorola g56k hello.c
motorola ls
a.cld hello.c
motorola run56sim a.cld
Hello, DSP.
motorola
```

# Options Available

## g56k Command Line Options

- BDirectory - utility search list
- bPREFIX - prefix the utility
- o FILE - output file
- v - verbose mode

## Preprocessor Phase Options

- C - comment included
- DMACRO - #define **MACRO**
- DMACRO=DEFN - #define **MACRO DEFN**
- E - preprocess only
- IDIR - set include-directory
- I- - set system-include-directory
- M - makefile generation
- MM - variation of the makefile generation
- nostdinc - no standard include search
- pedantic - strict ANSI standard
- P - preprocess only
- v - verbose mode
- UMACRO - undefine **MACRO**
- Wcomment - warn unusual comments
- Wtrigraphs - warn any trigraphs

## Assemble Phase Options

- asm string - % asm56000 string
- c - compile only

## Link Phase Options

- crt file - use file as crt0 file
- j string - % dsplnk string
- LIBRARY - library specification
- r MAPFILE - % dsplnk -R **MAPFILE**

## Compile Phase Options

- fno-opt - no optimization
- fno-peephole - no peephole optimization
- fno-strength-reduce - no strength reduction
- fno-defer-pop - no defer-pop optimization
- fforce-addr - force address constants into registers
- finline-functions - attempts to use inline-function
- fcaller-saves - protect registers overwritten by function calls
- fkeep-inline-functions - keep inline-function if possible
- fwritable-strings - for compatibility of the older cc
- fcond-mismatch - extended type for conditional expr
- fvolatile - make all pointer volatile
- ffixed-REG - reserve REG
- g - debugging information included in output
- O - optimization
- mno-dsp-optimization - no dsp optimization
- mno-do-loop-generation - no DO loop optimization
- mno-biv-plus-linv-promotion
- mstack\_check - stack check routine included
- mx-memory - X memory model
- my-memory - Y memory model
- ml-memory - L memory model
- pedantic - ANSI warning messages
- Q - verbose mode
- S - generates assembly code
- w - all warning messages
- W - extra warning messages
- Wimplicit - warning message if implied func. decl.
- Wreturn-type - warning message if default return type
- Wunused - warning if any unused local variable
- Wswitch - warning if unusual switch usage
- Wall - warning if any one of them above
- Wshadow - warning if shadow register
- Wid-clash-LEN - warning if ambiguous id of LEN
- Wpointer-arith - warning if unusual pointer arith

- Wcast-qual - warning if unusual casting
- Wwrite-strings - warning if a constant string is written

## Source Level Debugger

GDB56 is a source level debugger that provides the user with complete control over C or assembly code. It allows the programmer to set break points at the source level, examine the status of programs, change the environment of these programs, and debug the programs at the source level.

This tool also contains a stack examination feature which allows the programmer to examine the content of the top of the stack or the contents of the whole stack. The backtrace feature helps the programmer to detect any bugs in passing parameters between nested function calls.

The debugger provides flexible source file examination which includes line-oriented file examination and function-oriented file examination. Line specification for the line-oriented file examination can be done through relative or absolute modes.

Memory can be examined through data conversion and structure context. Data conversion is similar to the printf() conversion. The memory can also be examined automatically each time the program stops. The value history feature allows the programmer to analyze a C variable by examining the history of the contents of the variables. The following is the list of other features of the debugger.

- Flexible breakpoint assignment
- Conditional breakpoint assignment
- Commands executed on breaking
- Convenient ways to execute program
- Extensive capability to view C Language Environment
- On-line help

The following picture shows a case where the debugger is running under emacs with two windows one to execute command line and one to examine the source program.

```

shelltool - /bin/csh
(gdb56) rsn
Starting program: /usr2/choe/PROB/gcc/56k/techData/test/\
test.cld

Bpt 1, main () (test.c line 14)
(gdb56) p OUTBUF
$1 = {0, 1, 2, 3, 4, 5, 6, 7}
(gdb56) [ ]

*gdb56-test.cld*-- /usr2/moto/PROB/gcc/56k/techData/test/

    for(i = 0; i < 2; i++)
      for(j = 0; j < 2; j++)
        for(k = 0; k < 2; k++)
          OUTBUF[i*2*2 + j*2 + k] = i*2*2 + j*2 + k;
-> for(i = 0; i < 2*2*2; i++)
    printf("%d\\n", OUTBUF[i]);
}

-----Emacs: test.c          4:17pm Mail (C)-----Bot

```

Running the debugger under emacs allows user to examine both the execution and the contents of the programs at the same time. It also points out the portion of the C program which is actually running under the debugger's simulator.

## Standard C Library


The standard C Library is a collection of routines which can be called from the C program and help the programmer to reduce development time. The routines selected are based on the ANSI-Standard and all host-independent routines are supported. The portability of the resulting code is improved as a consequence of providing the standard routines.

The following is the list of the standard C library routines that the Optimizing C Cross Compiler supports.

## List of Standard C Library Routines

abort	force abnormal program termination
abs	absolute value of integer
acos	arc cosine
asin	arc sine
atan	arc tangent
atan2	arc tangent of angle defined by point y/x
atexit	register a termination function normal program
atof	string to floating point
atoi	string to integer
atol	string to long integer
bsearch	perform binary search
calloc	allocate zero-initialized storage for objects
ceil	ceiling function
cos	cosine
cosh	hyperbolic cosine
div	integer division with remainder
ldiv	long integer division with remainder
exit	terminate program normally
exp	exponential, e <sup>x</sup>
fabs	absolute value of a double
floor	floor function
fmod	floating point remainder
free	free storage
frexp	break a floating value into mantissa and exponent
isalnum	test for alphanumeric character
isalpha	test for alphabetic character
iscntrl	test for control character
isdigit	test for numeric character
isgraph	test for printing character, excluding space and tab
islower	test for lower-case alphabetic characters
isprint	test for printing character, excluding '\t'
ispunct	test for punctuation character
isspace	test for white-space character
isupper	test for upper-case alphabetic character
isxdigit	test for hexadecimal numeric character
labs	absolute value of a long integer
ldexp	multiply floating point number by 2
ldiv	long integer division with remainder
log	natural logarithm, base e
log10	base ten logarithm
longjmp	execute a non-local jump
malloc	dynamically allocate uninitialized storage
mblen	length of a multibyte character
mbstowcs	convert multibyte string to wide character string
mbtowc	convert a multibyte character to a wide character
memchr	find a character in a memory area
memcmp	compare portion of two memory areas
memcpy	copy from one area to another

memmove	copy storage	strcpy	copy one string into another
memset	initialize memory area	strcspn	the length of the prefix of one string
modf	break a double into it's integral and fractional parts	strerror	map error code into an error message string
perror	print error message	strlen	determine length of a string
pow	raise a double to a power	strncat	concatenate a portion of one string to another
putchar	write a character to standard output	strncmp	compare a portions of two strings
puts	write a string to standard output	strncpy	copy a portion of one string into another
qsort	quick sort	strpbrk	the first occurrence of a character from one string
raise	raise a signal	strrchr	the last occurrence of a character in a string
rand	pseudo- random number generator	strspn	the length of the prefix of a string
realloc	change size of dynamically allocated storage area	strchr	the last occurrence of a character from one string
setjmp	save a reference of the current calling environment	strstr	find the first occurrence of one string in another
signal	set up signal handler	strtod	string to double
sin	sine	strtok	break string into tokens
sinh	hyperbolic sine	strtol	string to long integer
sprintf	print to a string	strtoul	string to unsigned long integer
sqrt	square root	strxfrm	transform a string into locale-independent form
srand	seed the pseudo-random number generator	tan	tangent
strcat	concatenate two strings	tanh	hyperbolic tangent
strchr	find first occurrence of a character in a string	tolower	convert uppercase character to lowercase
strcmp	compare two strings	toupper	convert lowercase character to uppercase
strcoll	compare two strings based on current locale	wcstombs	convert wchar_t array to multibyte string
		wctomb	convert wchar_t character to multibyte character

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

#### Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.

EUROPE: Motorola Ltd.; European Literature Center; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.

ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbor Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.



**MOTOROLA**