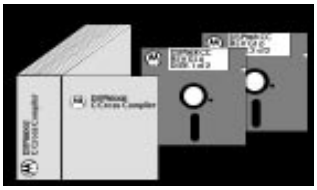
MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Order this document
by BR1126/D

DSP96KCCx



DSP96002 High-Level Tools

Software Summary
DSP96KCCx
DSP96002 C Cross Compiler

DSP96KCCx is a full ANSI C Cross Compiler supporting the development of DSP96002 applications. This software package allows the user to develop DSP96002 based application programs in a stand-alone or mixed-language programming environment (C and 96002 assembly). The software consists of an optimizing C Compiler, ANSI C Libraries, DSP96002 assembler, linker, librarian, GDB source level debugger and various utilities. The DSP96KCCx software can be run on a variety of host machines (IBM™ 386 and higher PC, SUN-3™, SUN-4™, or NeXT™ workstation) and the developed application code can be executed on the multi-DSP96002 simulator or DSP96002 Application Development System (DSP96002 ADSx).

DSP96KCCx is designed to: provide the programmer with optimized assembly code output; take advantage of the DSP96002's architecture; interface easily to the assembly language environment, C library routines, and the full-featured source level debugger; provide ANSI-portability. This product is based on the GNU C Compiler. These tools provide a powerful high-level environment to develop DSP96002 traditional DSP applications such as radar, sonar, navigation, image processing, computer graphics, and embedded control, as well as multi-processor applications.

Part Description

DSP 96K CC x

- x — Host Machine (see Ordering Information)
- CC — **C C**ross Compiler
- 96K — DSP96000 Family
- DSP — **D**igital **S**ignal **P**rocessor

Ordering Information

Host Platform	Operating System	Order Number
IBM™-386 PC/Compatible	DOS 5.x	DSP96KCCA
SUN-3™	SunOS™ 4.1.1	DSP96KCCC
SUN-4™	SunOS™ 4.1.1	DSP96KCCF
NeXT™	NeXT Mach™ 2.0	DSP96KCCG

Package Contents

Programs

g96k (optimizing C compiler)
gdb96 (symbolic debugger)
run96sim (DSP96000 simulator)
asm96000 (assembler)
dsplnk (linker)
dsplib (librarian)
COFF utilities
srec

Documentation

G96K C Cross Compiler
GDB96 Source Level 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, Inc.
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

Rev. G1.1
September 15, 1995

Optimizing C Cross Compiler

DSP96KCCx is a full-featured ANSI Standard C Compiler that compiles one or more source files containing C definitions and declarations into DSP96002 assembly code that can be assembled by the DSP96002 assembler and linked by the DSP96002 linker.

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.

Some of the optimization techniques employed are:

- Automatic Register Promotion
- Common Sub-expression Elimination
- Strength Reduction
- Code Hoisting
- Loop Rotation and Loop Invariant Elimination
- Dead Code Elimination
- Address Register Promotion For Array References
- Leaf Routine Detection
- Do Loop Optimization
- Tail Recursion Elimination

The GNU Source Level Debugger, GDB96, is included. With this debugging tool, application programs can be easily debugged and tested in the GNU Emacs (UNIX) environment. This can significantly reduce development time and effort.

The Common Object File Format (COFF) is generated by the dsp96002 asm/lnk and is used as input to the GNU Symbolic Debugger. All Motorola DSP software tools support the COFF file format 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, there.” on the screen.

```
 motorola  ls
hello.c
 motorola  cat hello.c
#include <stdio.h>
main()
{
   printf("Hello, there.\n");
}
 motorola  g96k hello.c
 motorola  ls
a.cld   hello.c
 motorola  run96sim a.cld
Hello, there.
 motorola
```

Options Available

g96k Command Line Options

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

Preprocessor Phase Options

-C - comment included to be used with -E only.
-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 - asm96000 string
-c - compile only

Link Phase Options

-crt file - use file as crt0 file
-j string - dsplnk string
-lLIBRARY - library specification
-r CTLFILE - dsplnk -R **CTLFILE**

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
-fwritable-strings - for compatibility of the older cc
-fcond-mismatch - extended type for conditional expr
-fvolatile - make all pointers volatile
-ffixed-REG - reserve REG
-g - debugging information included in output
-O - optimization (default)
-mno-dsp - no dsp optimization
-mno-do-loop-generation - no DO loop optimization
-mstack_check - stack check routine included
-mx-memory - X memory model
-my-memory - Y memory model
-ml-memory - L memory model (default)
-pedantic - ANSI warning messages
-Q - verbose mode
-S - generates assembly code
-w - all warning messages
-W - extra warning messages
-Wimplicit - warning message if a function is implicitly declared.
-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 of the four previous conditions exist
-Wshadow - warning if shadow register
-Wid-clash-LEN - warning if ambiguous id of LEN
-Wpointer-arith - warning if unusual pointer arithmetic
-Wcast-qual - warning if unusual casting
-Wwrite-strings - warning if a constant string is written

Source Level Debugger

GDB96 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.

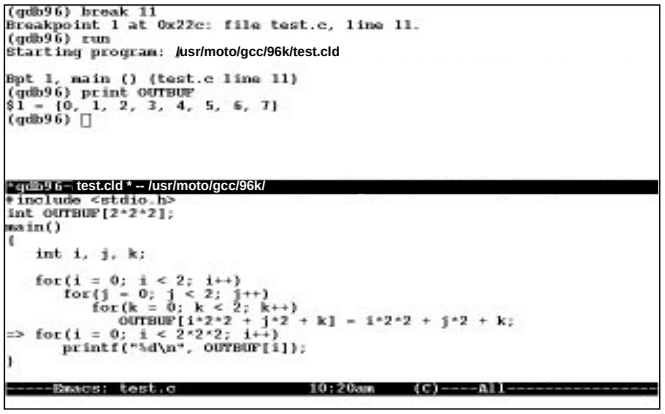
This tool also contains a stack examination feature which allows the programmer to examine the content of the top of the stack or the contents of the whole stack. The back trace feature helps the programmer to detect any bugs in passing parameters between nested function calls.

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. 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
- On-line help

The following picture shows a case where the debugger is running under GNU Emacs (UNIX) with two windows; one to execute the command line and one to examine the source program.



Running the debugger under GNU Emacs (UNIX) 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^x
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
memset	initialize memory area
modf	break a double into it's integral and fractional parts
perror	print error message
pow	raise a double to a power
putchar	write a character to standard output
puts	write a string to standard output
qsort	quick sort
raise	raise a signal
rand	pseudo- random number generator
realloc	change size of dynamically allocated storage area
setjmp	save a reference of the current calling environment
signal	set up signal handler

sin	sine
sinh	hyperbolic sine
sprintf	print to a string
sqrt	square root
srand	seed the pseudo-random number generator
strcat	concatenate two strings
strchr	find first occurrence of a character in a string
strcmp	compare two strings
strcoll	compare two strings based on current locale
strcpy	copy one string into another
strcspn	the length of the prefix of one string
strerror	map error code into an error message string
strlen	determine length of a string
strncat	concatenate a portion of one string to another
strncmp	compare a portions of two strings
strncpy	copy a portion of one string into another
strpbrk	the first occurrence of a character from one string
strrchr	the last occurrence of a character in a string
strspn	the length of the prefix of a string
strrchr	the last occurrence of a character from one string
strstr	find the first occurrence of one string in another
strtod	string to double
strtok	break string into tokens
strtol	string to long integer
strtoul	string to unsigned long integer
strxfrm	transform a string into locale-independent form
tan	tangent
tanh	hyperbolic tangent
tolower	convert uppercase character to lowercase
toupper	convert lowercase character to uppercase
wcstombs	convert wchar_t array to multibyte string
wctomb	convert wchar_t character to multibyte character

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