

COSMIC, sarl

Version 3.32

October 1989

Cross Compiler User's Guide

for Z80/HD64180

XENIX-AT Host

Copyright (c) 1988, 1989 by COSMIC, sarl and Whitesmiths, Ltd

All rights reserved.

(

(

(

(

COSMIC, sarl

Version 3.32

October 1989

Cross Compiler User's Guide

for Z80/HD64180

MS/PC-DOS Host

Copyright (c) 1988, 1989 by COSMIC, sarl and Whitesmiths, Ltd

All rights reserved.

(

(

(

(

TABLE OF CONTENTS

PREFACE

Organization of this Manual.	- 1
--------------------------------------	-----

CHAPTER 1

INTRODUCTION

Introduction	1 - 1
Support for ROMable Code	1 - 1
Optimizations.	1 - 2
Compiler Architecture.	1 - 3
Programming Support Utilities.	1 - 3
Linking.	1 - 5
Listings	1 - 5
Host System Requirements	1 - 6

CHAPTER 2

TUTORIAL INTRODUCTION

Default Compiler Operation	2 - 2
Specifying Command Line Options.	2 - 3
Object Module Format Translation	2 - 4

CHAPTER 3

PROGRAMMING FOR HD64180/Z80 ENVIRONMENTS

Modifying the Runtime Startup.	3 - 1
Description of Runtime Startup Code.	3 - 3
Initializing data in RAM	3 - 3
Performing Input/Output in C	3 - 4
Referencing Absolute Addresses	3 - 4
The const and volatile Type Qualifiers	3 - 5
Bank Switching	3 - 6
Placing Data Objects in the bss Section.	3 - 9
Generating Inline Machine Instructions; the @builtin function.	3 - 9
Inserting Inline Assembly Instructions; the _asm() function.	3 - 9
Generating ei, di, reti and halt Instructions.	3 - 10
Writing Interrupt Handlers	3 - 11

C Interface to Assembly Language	3 - 11
Register Usage	3 - 12
Data Representation.	3 - 13

CHAPTER 4

USING THE COMPILER

Invoking the Compiler.	4 - 1
Compiler Command Line Options.	4 - 2
File Naming Conventions.	4 - 7
Generating Listings.	4 - 7
Return Status.	4 - 12
Examples	4 - 12
C Library Support.	4 - 13
How C Library Functions are Packaged	4 - 13
Inserting Assembler Code Directly.	4 - 13
Linking Libraries with Your Program.	4 - 13
Integer Library Functions.	4 - 14
Floating Point Library Functions	4 - 14
Common Input/Output Functions.	4 - 14
Functions Implemented as Macros.	4 - 15
Including Header Files	4 - 15
Descriptions of C Library Functions.	4 - 16
<u>asm()</u> generate inline assembly code	4 - 17
<u>abs</u> find absolute value	4 - 18
<u>acos</u> arccosine	4 - 19
<u>asin</u> arcsine	4 - 20
<u>atan</u> arctangent.	4 - 21
<u>atan2</u> arctangent of y/x	4 - 22
<u>atof</u> convert buffer to double.	4 - 23
<u>atoi</u> convert buffer to integer	4 - 24
<u>atol</u> convert buffer to long.	4 - 25
<u>calloc</u> allocate and clear space on the heap.	4 - 26
<u>ceil</u> round to next higher integer.	4 - 27
<u>cos</u> cosine.	4 - 28
<u>cosh</u> hyperbolic cosine	4 - 29
<u>exp</u> exponential	4 - 30
<u>fabs</u> find double absolute value.	4 - 31
<u>floor</u> round to next lower integer	4 - 32
<u>free</u> free space on the heap.	4 - 33
<u>getchar</u> get character from input stream	4 - 34
<u>gets</u> get a text line from input stream	4 - 35
<u>isalnum</u> test for alphabetic or numeric character.	4 - 36
<u>isalpha</u> test for alphabetic character	4 - 37
<u>iscntrl</u> test for control character.	4 - 38
<u>isdigit</u> test for digit.	4 - 39
<u>isgraph</u> test for graphic character.	4 - 40
<u>islower</u> test for lowercase character.	4 - 41
<u>isprint</u> test for printing character	4 - 42

ispunct	test for punctuation character.	4 - 43
isspace	test for whitespace character	4 - 44
isupper	test for uppercase character.	4 - 45
isxdigit	test for hexadecimal digit.	4 - 46
log	natural logarithm	4 - 47
log10	common logarithm.	4 - 48
longjmp	restore calling environment	4 - 49
malloc	allocate space on the heap.	4 - 50
max	test for maximum.	4 - 51
memchr	scan buffer for character	4 - 52
memcmp	compare two buffers for lexical order	4 - 53
memcpy	copy one buffer to another.	4 - 54
memset	propagate fill character throughout buffer.	4 - 55
min	test for minimum.	4 - 56
pow	raise x to the y power.	4 - 57
printf	output formatted arguments to stdout.	4 - 58
putchar	put a character to output stream.	4 - 63
puts	put a text line to output stream.	4 - 64
rand	generate pseudo-random number	4 - 65
realloc	reallocate space on the heap.	4 - 66
sbreak	allocate new memory	4 - 67
scanf	read formatted input.	4 - 68
setjmp	save calling environment.	4 - 72
sin	sine.	4 - 74
sinh	hyperbolic sine	4 - 75
sprintf	output arguments formatted to buffer.	4 - 76
sqrt	real square root.	4 - 77
srand	seed pseudo-random number generator	4 - 78
sscanf	read formatted input from a string.	4 - 79
strcat	concatenate strings	4 - 80
strchr	scan string for first occurrence of character	4 - 81
strcmp	compare two strings for lexical order	4 - 82
strcpy	copy one string to another.	4 - 83
strcspn	find the end of a span of characters in a set	4 - 84
strlen	find length of a string	4 - 85
strncat	concatenate strings of length n	4 - 86
strncmp	compare two n length strings for lexical order.	4 - 87
strncpy	copy n length string.	4 - 88
strpbrk	find occurrence in string of character in set	4 - 89
strrchr	scan string for last occurrence of character.	4 - 90
strspn	find the end of a span of characters not in set	4 - 91
tan	tangent	4 - 92
tanh	hyperbolic tangent.	4 - 93
tolower	convert character to lowercase if necessary	4 - 94
toupper	convert character to uppercase if necessary	4 - 95
va_arg	get pointer to next argument in list.	4 - 96
va_end	stop accessing values in an argument list	4 - 98
va_start	start accessing values in an argument list.	4 - 100

CHAPTER 5

USING THE ASSEMBLER

The Programming Environment.	5 - 1
Library Support.	5 - 1
Linker Support	5 - 1
Other Related Utilities.	5 - 2
Assembly Language Source Code Format	5 - 2
Labels	5 - 3
Instruction Mnemonics.	5 - 3
Assembler Directives	5 - 3
Operands	5 - 3
Comments	5 - 4
Output from the Assembler.	5 - 4
Object Code Output	5 - 4
Symbol Table	5 - 5
Listings	5 - 5
Error Messages	5 - 5
Internal Organization of x80	5 - 6
Rules for Assembly Language Source	5 - 6
Identifier Names	5 - 6
Floating Point Numbers	5 - 7
Characters	5 - 8
ASCII Character Set.	5 - 8
Numeric Labels	5 - 10
Expressions.	5 - 11
Processor Instruction Mnemonics.	5 - 14
Extra Instruction Mnemonics.	5 - 14
Assembler Directives	5 - 14
Text Stream Formats.	5 - 16
Error Messages	5 - 16
The Symbol Table	5 - 17
Listings	5 - 17
Invoking x80	5 - 18
Command Line Options	5 - 19
x80 Assembler Directives	5 - 20
.ADDR	allocate storage for addresses. 5 - 22
.ASCII	modify parity bit 5 - 23
.BYTE	allocate an 8-bit sized variable space. 5 - 24
.DEFINE	give a permanent value to a symbol. 5 - 25
.DOUBLE	allocate a double word four byte sized variable area. 5 - 26
.ELSE	conditionally assemble code sections. 5 - 27
.END	stop the assembly 5 - 28
.ENDIF	end conditional assembly of code sections 5 - 29
.ENDM	end macro definition. 5 - 30
.ENDR	end a repeat section. 5 - 31
.EVEN	assemble next byte to an even address 5 - 32
.EXITM	terminate a macro definition. 5 - 33
.EXTERNAL	declare symbol as being defined elsewhere 5 - 34
.FLOAT	allocate a floating point four byte sized variable area 5 - 35
.IF	conditionally assemble code sections. 5 - 36

.INCLUDE	include text from another text file	5 - 38
.LFLOAT	allocate a long floating point eight byte sized	5 - 39
.LIST	select sections to be listed.	5 - 40
.MACRO	define a macro.	5 - 41
.PAGE	start a new page in the listing file.	5 - 44
.PSECT	place code in a program section	5 - 45
.PUBLIC	declare a variable to be visible.	5 - 46
.REPEAT	reread the following text a number of times	5 - 47
.SET	give a resetable value to a symbol.	5 - 48
.TEXT	place text into memory.	5 - 49
.TITLE	define default header	5 - 50
.WORD	allocate storage for a word two byte sized variable	5 - 51
	Assembler Error Messages.	5 - 52

CHAPTER 6

USING THE LINKER

Overview	6 - 2
Linker Command Line Processing	6 - 4
Passing Options from STDIN	6 - 4
Inserting comments in Linker commands.	6 - 5
Linker Command Line Options.	6 - 5
Global Command Line Options.	6 - 6
Section Control Options.	6 - 7
Object Formats	6 - 10
Input Object Module Format	6 - 10
Standard Object Format	6 - 10
Linking Standard Objects	6 - 13
MARKs, DEFs and REFs	6 - 14
Multisection Object Format	6 - 15
Linking Multisection Objects	6 - 16
Section Relocation	6 - 17
Address Arithmetic	6 - 17
Setting Bias and Offset.	6 - 18
Setting the Bias	6 - 18
Setting the Offset	6 - 18
Completing the Section	6 - 19
User Control of Stack and Heap	6 - 19
Return Values.	6 - 20
Special Topics	6 - 20
Private Name Regions	6 - 20
Renaming Symbols	6 - 21
Example Linker Command Lines	6 - 24

CHAPTER 7

DEBUGGING SUPPORT

Generating Debugging Information	7 - 1
Generating Line Number Information	7 - 2
Generating Line Number and Data Object Information	7 - 3
Generating Source Listings for Object Files.	7 - 3
The lines Utility.	7 - 3
The prdbg Utility.	7 - 5
Examples	7 - 5
The clist utility.	7 - 6
clist Options.	7 - 7

CHAPTER 8

USING THE PROGRAMMING SUPPORT UTILITIES

The hex80 Utility.	8 - 2
Command Line Options	8 - 2
Intel Standard Hex Format.	8 - 4
Motorola S-Record Format	8 - 5
Tektronix Standard Hex Format.	8 - 6
Tektronix Extended Hex Format.	8 - 7
Return Status.	8 - 8
Examples	8 - 8
The lby Utility.	8 - 10
Command Line Options	8 - 10
Return Status.	8 - 12
Examples	8 - 12
Special Usage Considerations	8 - 12
The lm Utility	8 - 13
Command Line Options	8 - 14
Return Status.	8 - 14
Examples	8 - 15
Special Usage Considerations	8 - 15
The lord80 Utility	8 - 16
Command Line Options	8 - 16
Examples	8 - 18
The pr Utility	8 - 19
Standard Output Format	8 - 19
Command Line Options	8 - 20
Return Status.	8 - 21
Examples	8 - 21
The rel80 Utility.	8 - 23
Command Line Options	8 - 23
Return Status.	8 - 25
Examples	8 - 25
The toprom Utility page.	8 - 26
Command Line Options page.	8 - 26

Descriptor Format page	8 - 26
The unhex Utility page	8 - 28
Command Line Options page.	8 - 28
Return Status page	8 - 29
Examples page.	8 - 29

APPENDIX A

COMPILER ERROR MESSAGES

Preprocessor cpp80 Error Messages.	A - 1
Parser cp180 Error Messages.	A - 3
Code Generator cp280 Error Messages.	A - 9

APPENDIX B

MODIFYING COMPILER OPERATION

The Prototype File	B - 1
Changing the Combination of Options Passed to Programs	B - 3
Changing the Default Location of Libraries	B - 3
Changing Input File Name Conventions	B - 3
Creating Your Own Programmable Options	B - 3

APPENDIX C

HD64180/Z80 MACHINE LIBRARY

Conventions using the Z80/HD64180 Machine Support Library	C - 2
c.bbtou unpack bits in byte bitfield to unsigned.	C - 4
c.btou unpack bits to unsigned	C - 5
c.butob pack unsigned into byte bitfield bits	C - 6
c.dadd add double into double.	C - 7
c.dcmp compare two doubles	C - 8
c.dcpy copy double to double	C - 9
c.ddiv divide double into double	C - 10
c.dmul multiply double into double	C - 11
c.dneg negate double	C - 12
c.dsub subtract double from double	C - 13
c.dtd move double to double	C - 14
c.dtf convert double to float	C - 15
c.dti convert double to int	C - 16
c.dtl convert double to long.	C - 17
c.dtr convert double to int on stack.	C - 18
c.fadd add float into float.	C - 19

c.fcmp	compare two floats.	C - 20
c.fcpy	copy float to float	C - 21
c.fdiv	divide float into float	C - 22
c.fmul	multiply float into float	C - 23
c.fmvd	copy double to double	C - 24
c.fmv1	copy float to float	C - 25
c.fneg	negate float.	C - 26
c.frepk	repack a float number	C - 27
c.fsub	subtract float from float	C - 28
c.ftd	convert float to double	C - 29
c.fti	convert float to int.	C - 30
c.ftl	convert float to long	C - 31
c.ftr	convert float to int on stack	C - 32
c.funpk	unpack a float number	C - 33
c.ibr	jump on bc.	C - 34
c.idiv	divide integer by integer	C - 35
c.ihr	jump on hl.	C - 36
c.ilsh	integer left shift.	C - 37
c.imod	remainder of integer divided by integer	C - 38
c.imul	multiply integer by integer	C - 39
c.irsh	integer right shift	C - 40
c.itd	convert integer to double	C - 41
c.itf	convert integer to float.	C - 42
c.jltab	perform C switch statement for long value	C - 43
c.jtab	perform C switch statement.	C - 44
c.ladd	add long to long.	C - 45
c.land	and long into long.	C - 46
c.lclt	compare long to long, set NC.	C - 47
c.lcmp	compare long to long, set Z	C - 48
c.lcom	complement long	C - 49
c.lcpy	copy long to long	C - 50
c.ldiv	divide long by long	C - 51
c.libc	perform a far call bank-switching	C - 52
c.llsh	long left shift	C - 53
c.lmod	remainder of long divided by long	C - 54
c.lmul	multiply long by long	C - 55
c.lneg	negate long	C - 56
c.lor	or long into long	C - 57
c.lret	return from runtime function.	C - 58
c.lrsh	long right shift.	C - 59
c.lsub	subtract long from long	C - 60
c.ltd	convert long to double.	C - 61
c.ltf	convert long to float	C - 62
c.lxor	exclusive or long into long	C - 63
c.movestr	copy a structure to another	C - 64
c.pushstr	push a structure.	C - 65
c.0mvd	copy double to in-core register c.r0.	C - 66
c.1mvd	copy double to in-core register c.r1.	C - 67
c.0mvf	copy float to in-core register c.r0	C - 68
c.1mvf	copy float to in-core register c.r1	C - 69
c.r0	the double accumulator and other pseudo registers	C - 70
c.repk	repack a double number.	C - 71
c.ret	return from a C function.	C - 72
c.ret0	return from a C function.	C - 73

c.rets	return from a C function.	C - 74
c.rets0	return from a C function.	C - 75
c.sav	enter a C function with one or more arguments and	C - 76
c.sav0	enter a C function with no arguments and save registers	C - 77
c.savs	enter a C function with one or more arguments	C - 78
c.savs0	enter a C function with no arguments.	C - 79
c.udiv	divide unsigned by unsigned	C - 80
c.uldiv	unsigned divide long by long.	C - 81
c.ulmod	remainder of unsigned long divided by long.	C - 82
c.ulrsh	unsigned long right shift	C - 83
c.ultd	convert unsigned long to double	C - 84
c.ultf	convert unsigned long to float.	C - 85
c.umod	remainder of unsigned divided by unsigned	C - 86
c.unpk	unpack a double number.	C - 87
c.ursh	unsigned right shift.	C - 88
c.utd	convert unsigned to double.	C - 89
c.utf	convert unsigned to float	C - 90
c.utob	pack unsigned into bits	C - 91
c.zret	return from runtime compare function.	C - 92

APPENDIX D

COMPILER PASSES

The cpp80 Preprocessor	D - 1
Command Line Options	D - 1
Preprocessor Control Character	D - 3
Return Status.	D - 3
Examples	D - 3
Special Usage Considerations	D - 3
The cp180 Parser	D - 4
Command Line Options	D - 4
Return Status.	D - 7
Examples	D - 7
Special Usage Considerations	D - 7
The cp280 Code Generator	D - 7
Return Status.	D - 9
Examples	D - 9
The cp380 Assembly Language Optimizer.	D - 9
Command Line Options	D - 9
Return Status.	D - 10
Examples	D - 10

PREFACE

The Cross Compiler User's Guide for Z80/HD64180 is a reference guide for programmers writing C programs for Z80/HD64180 environments. It provides an overview of how the cross compiler works, and explains how to compile, assemble, link and debug programs on your host system for execution on your target system. It also describes the programming support utilities included with the cross compiler and provides tutorial and reference information to help you configure executable images to meet specific requirements. This manual assumes that you are familiar with your host operating system and with your specific target environment.

You can find information about Whitesmiths' implementation of the C programming language in your C Language Specification for Microcontroller Environments.

Organization of this Manual

This manual is divided into eight chapters and four appendixes.

Chapter 1, "Introduction," describes the basic organization of the C compiler and programming support utilities.

Chapter 2, "Tutorial Introduction," is a series of examples that show you step by step how to compile, assemble and link a simple C program using the C cross compiler.

Chapter 3, "Programming for Z80/HD64180 Environments," tells you how to use the features of C for your target processor to meet the requirements of your particular application. It explains how to create a runtime startup for your application, and how to write C routines that perform special tasks such as serial I/O, making direct references to hardware addresses, handling interrupts, and calling assembly language programs.

Chapter 4, "Using the C Compiler," describes compiler command line options and the action each option per-

forms. This chapter also describes the functions in the C runtime library.

Chapter 5, "Using the Assembler," describes the assembler and its use, as well as the command line options it accepts and what they do. It explains the rules that your assembly language source code must follow, and documents all directives that the assembler supports.

Chapter 6, "Using the Linker," describes the linker command line options and the action each option performs. This chapter also describes in detail all the features of the linker and how to use it to link your C and assembly language source files and object images.

Chapter 7, "Debugging Support," describes the support available for **cxdb**, the C source level cross debugger and for other debuggers or in-circuit emulators.

Chapter 8, "Using the Programming Support Utilities," describes each of the programming support utilities and the command line options each accepts. Examples of how to use these utilities both together and separately are also included.

Appendix A, "Compiler Error Messages," is a list of compile time error messages that the C compiler may generate.

Appendix B, "Modifying Compiler Operation," describes the "prototype file" that serves as input to the compiler command driver. It explains how to change the default operation of the compiler to suit your needs and how to write your own "programmable" command line options to the compiler driver. It also describes the compiler driver program, called **c**.

Appendix C, "Z80/HD64180 Machine Library," describes the assembly language routines that provide support for the C runtime library.

Appendix D, "Compiler Passes," describes the specifics of the preprocessor, parser, code generator and assembly language optimizer and the command line options that each accepts.

This manual also contains an **Index**.