DeSMET DC88
C COMPILER

MARK DeSMET

Published and Distributed by

C WARE CORPORATION
PASO ROBLES, CALIFORNIA

DeSmet C Development Package

Version 3.1 — May,1988

Version 3.03 — February, 1988 (DC88)
Version 3.0 — April, 1987
Version 2.5 — October, 1985
Version 2.4 — October, 1984
Version 2.3 — April, 1984

Published by: C Ware Corporation
P.O. Box 428
Paso Robles, CA 93447
USA

(805) 239-4620 (Tech Support/Sales)

(805) 239-4834 (Tech BBS)

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

The author has taken due care in preparing this book and the programs and data on the electronic media accompanying this book including research, development, and testing to ascertain their effectiveness. The author and the publisher make no expressed or implied warranty of any kind with regard to these programs nor the supplemental documentation in this book. In no event shall the author or C Ware Corporation be liable for incidental or consequential damages in connection with or arising out of the furnishing, performance or use of any of these programs.

Table of Contents

Introduction
- Overview 1.1
- Large Case Option 1.3

Getting Started
- Backing Up 2.1
- Installing The Software 2.1
  - Installing DC88 2.1
    - Installing DC88 on a Hard Disk 2.4
    - Installing DC88 on a Floppy Disk 2.7
  - Installing the Large Case Option 2.10
    - Installing Large Case on a Hard Disk 2.11
    - Installing Large Case on a Floppy Disk 2.12
- A Short Example 2.13
- Completion Codes 2.18

The SEE™ Text Editor
- Introduction 3.1
- Getting Started
  - Concepts 3.2
  - Starting the Editor 3.3
  - Inserting & Editing Text 3.4
  - Saving the File 3.9
  - Editing Existing Files 3.10
- The Invocation Line 3.11
- The Keyboard
  - Cursor Movement Keys 3.12
  - Editing Keys 3.13
  - The DOS Key 3.14
- Commands 3.15
- Configuration 3.31

Table of Contents

The C88 C Compiler

Introduction 4.1

Invocation 4.1
Examples 4.3

The C Language
Environment
Character Set 4.4
Trigraph Sequences 4.4
Language
Keywords 4.5
Identifiers 4.5
Floating constants 4.5
Integer constants 4.6
Character constants 4.6
String constants 4.6
Hardware data types 4.7
Enumerated type 4.8
Function prototyping 4.8
Preprocessor
Conditional compilation 4.9
Source file inclusion 4.10
Macro replacement 4.11
Line control 4.11
Error 4.11
Pragma 4.11
Null 4.12
Predefined macros 4.12
Extensions
Asm 4.12
Case range 4.13
Restrictions
Forward references 4.14
Externs 4.14
Large Case Option 4.15

The ASM88 Assembler

Introduction 5.1

Invocation 5.1

Examples 5.2

Large Case ASM88 5.3

Table of Contents

The BIND Object File Linker
- Introduction 6.1
- Invocation 6.1
- Examples 6.3
- Small Case BIND
  - Space Considerations 6.3
  - Overlays 6.4
- Large Case BIND 6.6
- Libraries 6.7

The LIB88 Object File Librarian
- Introduction 7.1
- Invocation 7.1
- Examples 7.2
- Libraries 7.2

The D88 C Language Debugger
- Introduction 8.1
- D88 Usage 8.1
- Command Input 8.3
- Expressions 8.3
- Commands 8.5

Utility Programs
- CLIST: a listing & xref utility 9.1
- DUMP: a hex and ascii display utility 9.2
- FASTSCR: a screen output enhancer 9.3
- FREE: a free space display 9.3
- GREP: a file search utility 9.3

Table of Contents

LS: a directory listing utility 9.4

MERGE: a C source and assembly language merge utility 9.5

MORE: a file screen listing utility 9.5

PCmake: a program maintainance utility 9.6

PROFILE: a performance monitor utility 9.9

RM: a file removal utility 9.11

SENSE87: an 8087/80287 sensing library 9.12

TOOLBOX.S: a library of useful tools 9.14

The CSTDIO Library

Introduction 10.1

Names 10.1

Program Initialization 10.2

Calling Conventions 10.4

Memory Management 10.9

Input/Output Library 10.11
- Directory Level Functions 10.11
- File Level Functions 10.11
- Stream Level Functions 10.12
- Handle Level Functions 10.13
- Screen Level Functions 10.13
- Console Level Functions 10.14

Math Library 10.15

System Interface 10.16

Environment 10.18-1

Table of Contents

Library 10.18-2
Headers
assert.h 10.18-2
ctype.h 10.18-2
math.h 10.18-3
setjmp.h 10.18-3
stdarg.h 10.18-3
stdio.h 10.18-4
stdlib.h 10.18-5
string.h 10.18-6
Functions & Macros
Alphabetical by name

Appendix A: Messages

ASM88 Messages
Banner and Termination Messages A.1
ASM88 Fatal Error Messages A.2
ASM88 Error Messages A.2

BIND Messages
Banner and Termination Messages A.7
BIND Fatal Error Messages A.7
BIND Warning Messages A.8

C88 Messages
Banner and Termination Messages A.9
C88 Fatal Error Messages A.10
C88 Error Messages A.12
C88 Warning Messages A.16
C88 ASM88 Messages A.17

CLIST Messages
Banner and Termination Messages A.18
CLIST Fatal Error Messages A.18

D88 Messages A.19

LIB88 Messages
Banner and Termination Messages A.21
LIB88 Fatal Error Messages A.21
LIB88 Warning Messages A.22

SEE Messages
Banner and Termination Messages A.23
SEE Error and Status Messages A.23

Table of Contents

Appendix B: The ASM88 Assembly Language

Identifiers B.1

Constants B.1

Expressions B.2

Registers
- General Registers B.2
- Byte Registers B.3
- Segment Registers B.3

Addressing Modes B.3

8086 Flags B.4

Address Expressions B.5

Address Typing B.5

Comments B.6

Assembler Direectives B.6

Reserving Storage B.7

Differences between ASM86 and ASM88 B.8

8086 Instructions
- Elements of Instructions B.9
- Instructions B.9

8087 B.23
- Control Word B.24
- Status Word B.25
- Tag Word B.25
- Condition Codes B.26

8087 Instructions B.27

Preface

This manual describes the DeSmet C Development Package for the IBM-PC personal computer and the other MS-DOS based personal computers. If you are unfamiliar with the C language or UNIX, the book *The C Programming Language* by Brian Kernighan and Dennis Ritchie is a good place to start. If you plan on coding in assembly language, it is advisable to get a manual on the Intel 8086 microprocessor. Books such as Intel's *ASM86 Language Reference Manual* or *The 8086 Family User's Guide* are good choices. These manuals fully describe the architecture and the instruction set of the 8086/8088 family of microprocessors.

We thank both the Pacific Data Works, and Scott Lewis for proofreading the many revisions of this manual.

Chapter 1

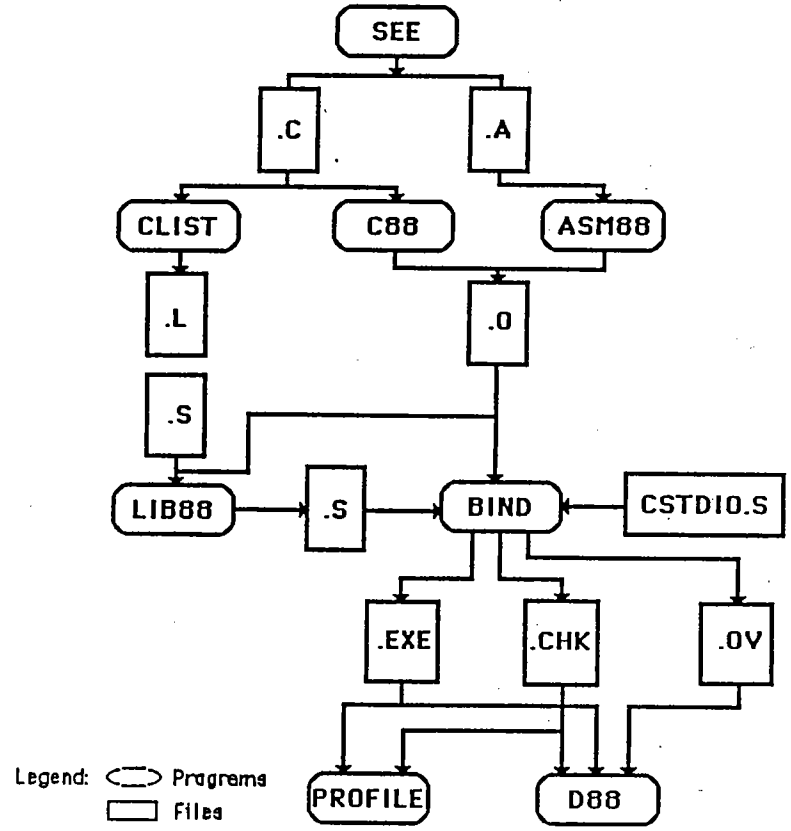
Introduction

Overview 1.1

Large Case Option 1.3

Overview

The DeSmet C Development Package is a set of programs and files for developing applications in the C programming language for the IBM-PC personal computer and its clones. The programs provided in this package require a minimum of 128K of Random Access Memory (RAM) and at least one disk drive. D88 requires 192K. Most programs will run under all versions of DOS, 1.xx, 2.xx, and 3.xx. The program execution profiler requires the use of DOS 2.x or later versions.



The diagram above outlines the interrelationships between some of the programs which are provided.

SEE is a full-screen, command oriented text editor designed for program editing rather than word processing. While SEE can edit any standand ASCII text file, its main purpose is to produce C [.C] and Assembler source files [.A]. The compiler C88 and the linker **BIND** can be invoked from SEE.

CLIST reads C source files [.C] and produces a listing file with a symbol cross-reference.

C88 is the C compiler. It reads C source files [.C] and produces either object files [.O] or assembler files [.A]. It supports the complete Kernighan and Ritchie C language plus the UNIX V7 extensions — structure assignment and parameter passing, and enumerated types. C88 supports both the Small and Large Case memory models.

ASM88 is the 8086/8088 assembler. It reads assembler source files [.A] and produces linkable object files [.O].

BIND is the object file linker. It reads object files [.O] and library files [.S] and produces an executable file [.EXE]. BIND optionally produces the debugger information file [.CHK] and overlay files [.OV]. The Large Case memory model linker is BBIND.

LIB88 is the object file librarian. It reads object files [.O] and other library files [.S] and produces library files [.S].

D88 is the C source-level symbolic debugger. It provides access to program variables by name, breakpoints by function name and line number, and special support for debugging interactive programs. Source code display and stepping by source lines are also supported.

PROFILE is the C program execution profiler. It monitors the execution of the application program and indicates where time is spent in the program.

CSTDIO.S is the Standard Library used by BIND to provide the Operating System and machine-level functions supported by the C language. Two libraries are provided in the development package, one that support the 8087 math coprocessor directly (CSTDIO7.S) and one that provides numeric support in software (CSTDIO.S). The Large Case memory model libraries are BCSTDIO.S and BCSTDIO7.S

Large Case Option

The Large Case Option addresses the needs of programs that fit neither the standard Small Case restrictions (64K of code, 64K of data *and* stack), the partitioning requirements of overlays, nor the communication limitations of the *exec* function. Its features include:

Full 1-megabyte addressability via 32-bit pointers.

Static variables combined within a single data-segment to speed access.

Large Case differs from Small Case in two aspects: pointers are four bytes long (segment:offset) rather than two bytes (offset), and function calls are inter-segment (segment:offset) instead of intra-segment (offset).

There are still some memory restrictions with Large Case. No derived data object — array or structure — may be larger than 64K. The total size of all `static` and global fundamental objects (`char, int, ...`) must be less than 64K. The restriction on `static` and global fundamental objects has to do with efficiency — they can be accessed with the same speed as Small Case.

Large Case programs are approximately 15 per-cent larger and slower than their Small Case equivalents.

WARNING: LOGIC ERRORS IN PROGRAMS USING 32-BIT POINTERS MAY BE HAZARDOUS TO YOUR COMPUTER!

Programs using 32-bit pointers can change any byte of memory via pointers. Thus, improperly initialized pointers can change critical portions of MSDOS, possibly causing corruption of, or damage to your DISKS.

In addition, corruption of the return address or function address can transfer control to an arbitrary location in memory, thereby activating code that may cause corruption of, or damage to your DISKS.

Chapter 2

Getting Started

Backing Up 2.1

Installing The Software

- Installing DC88 2.1
  - Installing DC88 on a Hard Disk 2.4
  - Installing C88 on a Floppy Disk 2.7
- Installing the Large Case Option 2.10
  - Installing Large Case on a Hard Disk 2.11
  - Installing Large Case on a Floppy Disk 2.12

A Short Example 2.13

Completion Codes 2.18

Backing Up

First things first. Copy all of the files from the distribution disks onto a set of working floppy diskettes or hard disk. The disks are not copy-protected so the DOS **copy** command can be used to copy the files. The package is distributed on three DOS 2 double-sided (360KB) or one DOS 3 quad (1.2MB) diskette.The distribution diskette(s) should never be used, they should be kept as the backup copy of the package.

Installing the Software

The following section assumes you have two drives: a floppy disk (drive A:) and either a hard disk (drive C:) or another floppy disk (drive B:). The system drive is the disk your machine "boots" from, either A: or C:. All of the relevant DeSmet C software is in the **\DC88** sub-directory on the hard disk, and in the Root Directory on the floppy disk.

Installing DC88 — There is one information and six data files in the DC88 3.1 distribution. The files, and their contents are:

BIN.EXE An archive of executable files, containing

ASM88.EXE:	The 8088 assembler.
BIND.EXE:	The object file linker.
BUF128.EXE:	128 byte type-ahead buffer program.
BUGS!.EXE:	Arcade game (use 'BUGS! c' for color displays).
C88.EXE:	The first pass of the C compiler.
CLIST.EXE	The C listing and cross-reference utility.
COMPARE.EXE:	The source code comparison utility.
D88.EXE:	The C source-level symbolic debugger.
DUMP.EXE:	The hex file display utility.
FASTSCR.EXE	Screen output speed-up.
FREE.EXE	Disk free space display
GEN.EXE:	The second pass of the C compiler.
GREP.EXE	A file search utility
LIB88.EXE:	The object file librarian.
LIFE.EXE:	Full screen game of Life.
LS.EXE	A directory listing utility
MERGE.EXE	A C source and assembly language merge utility
MORE.EXE	A file listing utility
PCMAKE.EXE	A program maintenance utility
PROFEND.EXE:	Used by PROFILE.EXE.
PROFILE.EXE:	The program execution profiler.
PROFSTAR.EXE:	Used by PROFILE.EXE.
RAM.COM:	RAM Disk driver for DOS 2 and later systems.
RM.EXE	A file deletion utility
SEE.EXE:	The full-screen editor.
TOOBJ.EXE	.O to .OBJ converter.

GRAPHICS.EXE An archive of text and library files, containing

GRAPHICS.NEW	New release information
GRAPHICS.DOC	Graphics documentation
GRAPHICS.CGA	Small-case graphics for the CGA
GRAPHICS.HGA	Small-case graphics for the Hercules Adaptor

INCLUDE.EXE An archive of text files, containing

ASSERT.H	Diagnostic include file.
CTYPE.H	Character handling include file.
DOS.H	DOS function include file.
FLOAT.H	Floating-point constants include file.
LIMITS.H	Character and numerical limits include file.
MATH.H	Mathematics include file.
SETJMP.H	Non-local jump include file.
STDARG.H	Variable argument include file.
STDIO.H	Input/output include file.
STDLIB.H	General utility include file.
STRING.H	String handling include file.

LIB.EXE An archive of library files, containing

C88.LIB	Software F/P LINK library.
C887.LIB	8087 LINK library.
CSTDIO.S	Software F/P BIND library.
CSTDIO7.S	8087 BIND library.
LLINK.BAT	LINK typical batch file.
SENSE87.S	8087-sensing upgrades to CSTDIO.S
TOOLBOX.S	Utility function library.

OBJ.EXE An archive of object files, containing

C.OBJ	LINK start-off code.
COMPARE.O	Object Code form of comparison utility.
D88.O	Object version of D88 — part 1.
D88REST.O	Object version of D88 — part 2.
EXEC.O:	The Exec() and Chain() functions.
EXEC.OBJ	Object code for `exec()` and `chain()` functions.
MSVER1.O	Object code for DOS 1 I/O functions.
SEE.O	Object code of the SEE editor

SRC.EXE An archive of source files, containing

BUF128.A:	Source code for BUF128.EXE.
C.ASM	Source code for runtime start-up function.
CB.C:	Source code for a brace matching program.
CLOCK.C	Source code to display clock face.
CONFIG.C	Source code for screen functions
DUMP.C:	Source code for DUMP.EXE.
FLIP.A	D88 screen Flip source code.
ISETUP.A	Source code for runtime start-up function.
LATER.C:	Source code for a file modification-date utility.
LIFE.C:	Source code for LIFE.EXE.
PCIO.A	INT 10H screen interface source code.
RUBRBAND.C	Line drawing source code.
STUB.ASM	LINK example source code.
TDRAW.C	Med-res drawing test.
TGETPUT.C	Screen area get/put test.
TXDRAW.C	High-res drawing test.

VERSION.DOC Contains the latest information about the release and its contents.

If you have the 1.2MB disk format, all the files will be on the one disk. If you have the 360KB disk format, the files are on the following disks:

Disk #1	BIN.EXE, INCLUDE.EXE, and VERSION.DOC
Disk #2	GRAPHICS.EXE, LIB.EXE, and OBJ.EXE
Disk #3	SRC.EXE

Each of the archive files can extract some, or all, of its contents. For example, to extract all of the SRC.EXE archive file enter

```
src
```

To extract, say, just the PCIO.A file from the SRC.EXE archive, enter

```
src pcio.a
```

If the package is to be run on a system other than an IBM PC, XT, AT, PCjr or PC-clone, the screen interface for SEE must be configured before it can be used. See the notes in the file CONFIG.C in the SRC.EXE archive for details.

Installing DC88 on a Hard Disk.

1. For systems utilizing DOS 2 or later versions of the operating systems, make sure that the ASCII text file **CONFIG.SYS** exists in the Root Directory of your system disk (C:). If it doesn't exist, you can create it with SEE (If you don't know how to use SEE, look at the example in this chapter).

```
c:
cd \
see config.sys
```

   The file must contain the line:

```
FILES=20
```

   since DC88 supports 20 open files — stdin, stdout, stderr, stdaux, stdprt and 15 other files. The default number of eight is insufficient for the BIND program. If there is enough memory available, add the line:

```
BUFFERS=20
```

   to improve file performance in the operating system. 512 bytes are allocated for each buffer specified.

   If you have a system with more than 256KB of memory, then the Ram Disk driver RAM.COM can be used to create an extremely fast disk. To add the Ram Disk, extract RAM.COM from the BIN.EXE archive

```
a:bin ram.com
```

   and add the line

```
DEVICE=RAM.COM n
```

   to CONFIG.SYS. The parameter *n* is a decimal number between 32 and 650, indicating the size of Ram Disk in KB (1024 bytes) increments.

   The Ram Disk installs as the next available drive — if the highest letter drive on your system was C:, then the Ram Disk will install as D:. Use the DOS chkdsk command to verify the drive assignment.

2. Create a sub-directory (i.e., \DC88) in the root directory of the hard disk (e.g., C:).

```
mkdir dc88
cd dc88
```

3. Unpack the BIN, INCLUDE, LIB and OBJ archives to DC88.

   - Disk #1 — 1.2MB & 360KB format.

```
a:bin c88.* gen.* asm88.* bind.* d88.* see.*
a:include
```

   - If you wish to use LINK

```
a:bin toobj.exe
```

   - If you have the 360KB format, insert Disk #2 in drive A:

   - If you wish to create programs that use only hardware F/P

```
a:lib cstdio7.s
ren cstdio7.s cstdio.s
```

     else, if you wish to create programs that use only software F/P

```
a:lib cstdio.s
```

     else, if you wish to create programs that use either F/P

```
a:lib cstdio.s sense87.s
ren *.s *.o
lib88 sense87 cstdio -ocstdio
del cstdio.o
del sense87.o
```

   - If you wish to use LINK

```
a:obj c.obj exec.obj
```

     If you wish to create programs that use only hardware F/P

```
a:lib c887.lib
```

     else, if you wish to create programs that use only software F/P

```
a:lib c88.lib
```

Be sure to change the Bind Flags in SEE (using the SET command) to invoke LINK instead of BIND, or use the LLINK.BAT file as model for linking.

- If you want your library to use only DOS 1 functions

```
a:obj msver1.o
ren cstdio.s cstdio.o
lib88 msver1 cstdio -ocstdio
del cstdio.o
del msver1.o
```

3. If you wish to use the Graphics Package, print the manual and text

```
a:graphics graphics.doc graphics.new
copy graphics.* prn
del graphics.*
```

If you have a Color Graphics Adaptor (CGA), extract its library

```
a:graphics graphics.cga
ren graphics.cga libg.s
```

If you have a Hercules Adaptor (HGA), extract its library

```
a:graphics graphics.hga
ren graphics.hga libg.s
```

4. If you have a machine other than an IBM or close clone copy.

```
a:obj see.o d88.o d88rest.o compare.o
```

If you have the 360KB format, insert Disk #3.

If your machine emmulates the IBM ROM BIOS interrupt 10H, then recreate SEE, D88, & COMPARE

```
a:src pcio.a
asm88 pcio
bind see pcio -osee
bind d88 d88rest pcio -od88
bind compare pcio -ocompare
```

otherwise modify CONFIG.C for your particular display, then recreate SEE, D88, & COMPARE

```
a:src config.c
edit config.c
c88 config
bind see config -osee
bind d88 d88rest config -od88
bind compare config -ocompare
```

Delete `see.o`, `d88.o`, `d88rest.o`, and `compare.o`.

5. Modify AUTOEXEC.BAT to specify the location of DC88 components and include files.

```
see \autoexec.bat
```

The DC88 components are specified in the PATH environment variable. Add the **c:\dc88** sub-directory to the existing PATH specification, or create a PATH specification. See your DOS manual for information on specifying the PATH variable.

The DC88 include files are specified in either the DSINC or the INCLUDE environment variable. Add either the **set DSINC=c:\dc88** or the **set INCLUDE=c:\dc88** line to the AUTOEXEC.BAT file. See Chapter 4 — The C88 C Compiler — for more information on the specifying the search path for DC88 include files.

6. Re-boot the system.

Installing DC88 on a Floppy Disk

1. Create a System Disk on drive B:

```
format b:/s
copy format.com b:
```

2. Put the System Disk in drive A: and DC88 Disk #1 in drive B: For systems utilizing DOS 2 or later versions of the operating systems, create the ASCII text file **CONFIG.SYS** in the Root Directory of your system disk (A:). You can create it with SEE (If you don't know how to use SEE, look at the example in this chapter).

```
b:see config.sys
```

The file must contain the line:

```
FILES=20
```

since DC88 supports 20 open files — stdin, stdout, stderr, stdaux, stdprt and 15 other files. The default number of eight is insufficient for the BIND program. If there is enough memory available, add the line:

```
BUFFERS=20
```

to improve file performance in the operating system. 512 bytes are allocated for each buffer specified.

If you have a system with more than 256KB of memory, then the Ram Disk driver RAM.COM can be used to create an extremely fast disk. To add the Ram Disk, extract RAM.COM from the BIN.EXE archive

```
b:bin ram.com
```

and add the line

```
DEVICE=RAM.COM n
```

to CONFIG.SYS. The parameter *n* is a decimal number between 32 and 650, indicating the size of Ram Disk in KB (1024 bytes) increments.

The Ram Disk installs as the next available drive — if the highest letter drive on your system was B:, then the Ram Disk will install as C:. Use the DOS chkdsk command to verify the drive assignment.

2. Unpack the BIN, INCLUDE, LIB and OBJ archives to the system disk.

   - Disk #1 — 1.2MB & 360KB format.

     ```
     b:bin c88.* gen.* asm88.* bind.* d88.* see.*
     b:include
     ```

   - If you wish to use LINK

     ```
     b:bin toobj.exe
     ```

   - If you have the 360KB format, insert Disk #2 in drive B:

- If you wish to create programs that use only hardware F/P

```
b:lib cstdio7.s
ren cstdio7.s cstdio.s
```

else, if you wish to create programs that use only software F/P

```
b:lib cstdio.s
```

else, if you wish to create programs that use either F/P

```
b:lib cstdio.s sense87.s
ren *.s *.o
lib88 sense87 cstdio -ocstdio
del cstdio.o
del sense87.o
```

- If you wish to use LINK

```
b:obj c.obj exec.obj
```

If you wish to create programs that use only hardware F/P

```
b:lib c887.lib
```

else, if you wish to create programs that use only software F/P

```
b:lib c88.lib
```

Be sure to change the Bind Flags in SEE (using the SET command) to invoke LINK instead of BIND, or use the LLINK.BAT file as model for linking.

- If you want your library to use only DOS 1 functions

```
b:obj msver1.o
ren cstdio.s cstdio.o
lib88 msver1 cstdio -ocstdio
del cstdio.o
del msver1.o
```

3. If you wish to use the Graphics Package, print the manual and text

```
b:graphics graphics.doc graphics.new
copy graphics.* prn
del graphics.*
```

If you have a Color Graphics Adaptor (CGA), extract its library

```
b:graphics graphics.cga
ren graphics.cga libg.s
```

If you have a Hercules Adaptor (HGA), extract its library

```
b:graphics graphics.hga
ren graphics.hga libg.s
```

4. If you have a machine other than an IBM or close clone copy.

```
b:obj see.o d88.o d88rest.o compare.o
```

If you have the 360KB format, insert Disk #3.

If your machine emmulates the IBM ROM BIOS interrupt 10H, then recreate SEE, D88, & COMPARE

```
b:src pcio.a
asm88 pcio
bind see pcio -osee
bind d88 d88rest pcio -od88
bind compare pcio -ocompare
```

otherwise modify CONFIG.C for your particular display, then recreate SEE, D88, & COMPARE

```
b:src config.c
edit config.c
c88 config
bind see config -osee
bind d88 d88rest config -od88
bind compare config -ocompare
```

Delete `see.o`, `d88.o`, `d88rest.o`, and `compare.o`.

5. Re-boot the system.

Installing the Large Case Option

The Large Case Option is distributed on a single 5 1/4 inch floppy diskettes, containing:

B88.LIB:	Large Case DOS LINK C Library (non-8087)
B887.LIB:	Large Case DOS LINK C Library (8087)
BBIND.EXE:	Large Case Binder.

BC.ASM	Large Case DOS LINK start-up source code
BC.OBJ	Large Case DOS LINK start-up object code
BCSTDIO.S	Large Case C Library (non-8087)
BCSTDIO7.S	Large Case C Library (8087)
BEXEC.O	Large Case `exec()` and `chain()` functions.
BEXEC.OBJ	Large Case DOS LINK `exec()` and `chain()` functions.
BGRAPHIC.CGA	Large-case graphics for the CGA
BGRAPHIC.HGA	Large-case graphics for the Hercules Adaptor
BLLINK.BAT	Large Case DOS LINK
BSTUB.ASM	Large Case DOS LINK MASM example

Installing Large Case on a Hard Disk

Place the Large Case Option disk in drive A:

```
copy a:*.exe c:\dc88
```

If you have a 8087 coprocessor

```
copy a:bcstdio7.s c:\dc88\bcstdio.s
```

otherwise

```
copy a:bcstdio.s c:\dc88
```

If you are using LINK

```
copy a:*.obj c:\dc88
```

If you have a 8087 coprocessor

```
copy a:bc887.lib c:\dc88\bc88.lib
```

otherwise

```
copy a:bc88.lib c:\dc88
```

Be sure to change the Bind Flags in SEE (using the SET command) to invoke BBIND or LINK instead of BIND, or use the BLLINK.BAT file as model for linking.

If you are using the Graphics Package with a Color Graphics Adaptor

```
copy a:bgraphic.cga c:\dc88\blibg.s
```

otherwise

```
copy a:bgraphic.hga c:\dc88\blibg.s
```

Installing Large Case on a Floppy Disk

Place the Large Case Option disk in drive B: and the DC88 System Disk in drive A: and copy the following files:

```
copy b:*.exe
```

If you have a 8087 coprocessor

```
copy b:bcstdio7.s bcstdio.s
```

otherwise

```
copy b:bcstdio.s
```

If you are using LINK

```
copy b:*.obj a:
```

If you have a 8087 coprocessor

```
copy b:bc887.lib bc88.lib
```

otherwise

```
copy b:bc88.lib
```

Be sure to change the Bind Flags in SEE (using the SET command) to invoke BBIND or LINK instead of BIND, or use the BLLINK.BAT file as model for linking.

If you are using the Graphics Package with a Color Graphics Adaptor

```
copy b:bgraphic.cga blibg.s
```

otherwise

```
copy b:bgraphic.hga blibg.s
```

A Short Example

This example shows the general method for creating executable programs with this package. It assumes that the disk in the default drive, in this case drive A:, contains the compiler (C88.EXE and GEN.EXE), the assembler (ASM88.EXE), the binder (BIND.EXE), the standard library (CSTDIO.S) and the text editor (SEE.EXE). The source code will reside on drive B:.

Enter the example program with the SEE text editor. To start the SEE text editor, type:

```
see  b:example.c
```

The screen will look as follows:

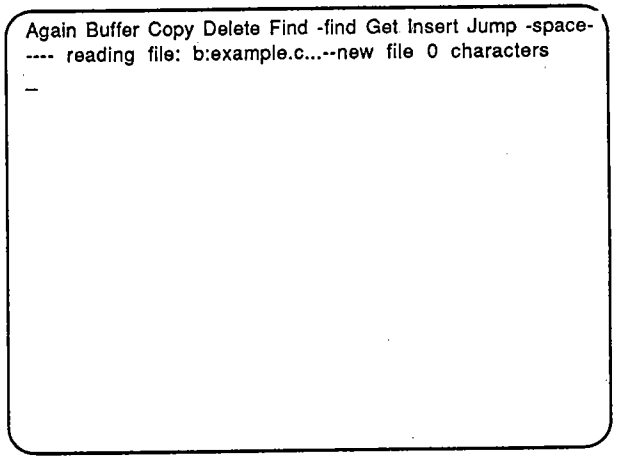


Figure 2-1
See™ Initial Screen

Other than the header, footer, and this sentence, this page is intentionally blank.

Type the letter 'I', or press the 'Ins' key, to put the editor into Insert mode. Now type in the following program:

```
main() {<Ret>
<Tab>printf("%d plus %d is %d\n", 2, 2, 2+2);<Ret>
<Tab>}<Ret>
<Esc>
```

Note that the items <Tab>, <Ret>, and <Esc> indicate the Tab, Return, and Esc keys, respectively. The <Esc> will terminate insert mode and return the editor to command mode. The screen should now appear as follows:

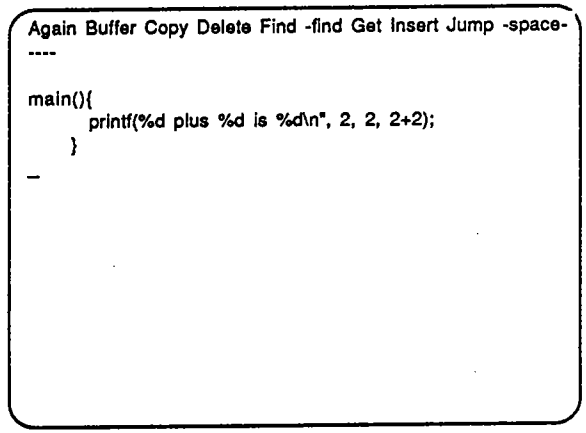


Figure 2-2
Program Display

To compile the program just entered, type the sequence of characters, 'Q' for Quit and 'C' for Compile. This will start C88 using the file in memory. The computer will display:

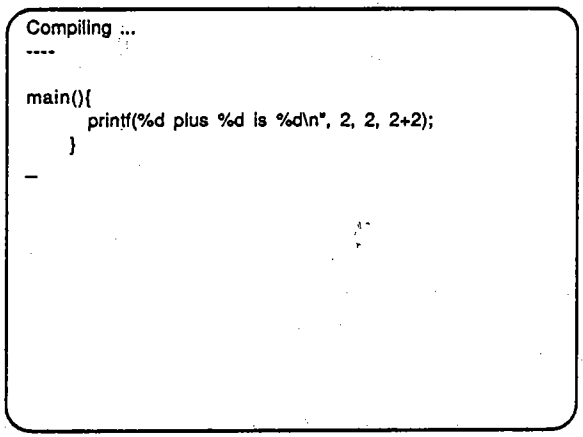


Figure 2-3
Compiling from SEE

The message "Compiling ..." replaces the first line of the display. If there are errors during the compilation, the error message will appear on the second line of the display, and the cursor will be on the error line. You can correct the error and recompile. If there are no errors BIND will be invoked. The screen appears as follows:

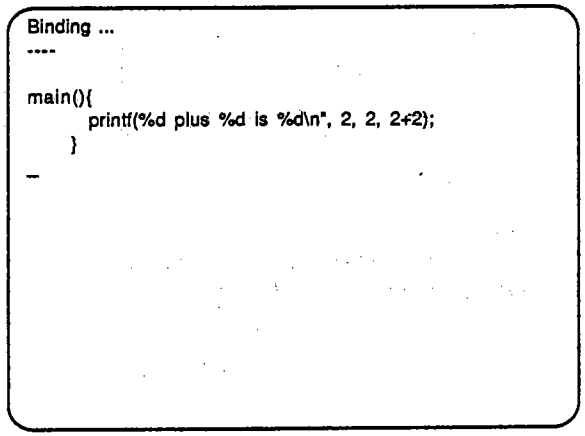


Figure 2-4
Binding from SEE

If there are any errors, they will be displayed on the message line. To run the program, press esc to escape from the Quit menu and press the F9 key to invoke a new DOS shell. At the prompt, enter

```
b:example
```

to invoke the program. The screen will look something like:

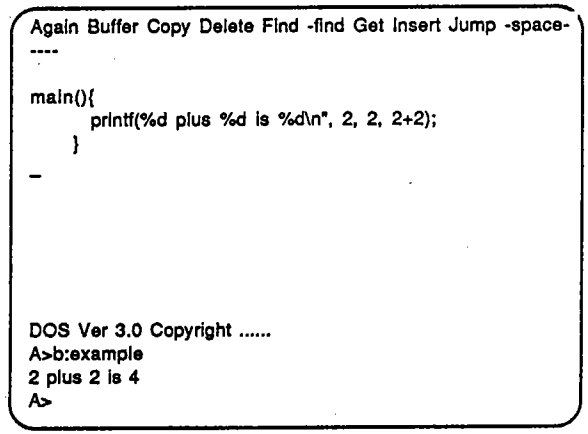


Figure 2-5
Executing example Program

To return to SEE type

```
exit
```

at the DOS propmpt. You will be returned to the SEE display.

If you wish to save the file to disk, type 'Q' (Quit) followed by an 'S' (Save-exit). The file will be saved, and control will be returned to DOS.

Completion Codes

The C88, ASM88, BIND and LIB88 programs set the completion code to:

0 if no warnings or errors occurred,
1 if warnings were issued, and
2 if errors occurred.

Batch files can take advantage of these values to stop execution or otherwise handle these exceptional cases.

The batch file CC.BAT listed below will stop if C88 or BIND reports an error:

```
c88 %1
if errorlevel 1 goto stop
bind %1
if errorlevel 1 goto stop
%1
:stop
```

More complicated development situations can be handled with the program LATER which is supplied in source form in the file LATER.C. LATER takes a list of filenames as arguments. It sets the errorlevel to one if the last file does not exist or if the last file has an earlier modification date than any other file in the list. It can only be used on systems with a battery backup clock or where users are careful about setting the date and time when the system is brought up. Assume a program is composed of the files moda.c, modb.c, modc.c and the include file mod.h. The following .BAT file can be used to regenerate the program whenever a module changes:

```
later moda.c mod.h moda.o
if errorlevel 1 c88 moda
if errorlevel 1 goto stop
later modb.c mod.h modb.o
if errorlevel 1 c88 modb
if errorlevel 1 goto stop
later modc.c mod.h modc.o
if errorlevel 1 c88 modc
if errorlevel 1 goto stop
later moda.o modb.o modc.o mod.exe
if errorlevel 1 bind moda modb modc -omod
:stop
```

This provides a service similar to the UNIX MAKE program. Only those files that need to be compiled will be compiled.

Chapter 3

The SEE™ Text Editor

Introduction 3.1

Getting Started
- Concepts 3.2
- Starting the Editor 3.3
- Inserting and Editing Text 3.4
- Saving the File 3.9
- Editing Existing Files 3.10

The Invocation Line 3.11

The Keyboard
- Cursor Movement Keys 3.12
- Editing Keys 3.13
- The DOS Key 3.14

Commands 3.15
- Again 3.16
- Buffer 3.16
- Copy 3.16
- Delete 3.16
- Find 3.17
- -Find 3.18
- Get 3.18
- Insert 3.18
- Jump 3.18
- List 3.19
- Macro 3.19
- Other 3.20
- Put 3.20
- Quit 3.20
- Replace 3.23
- Set 3.23
- Tag 3.30
- Version/View 3.30
- Wrap 3.30
- Xchange 3.30

Commands (cont.)
3.31
{}()[] 3.31
\ 3.31

Configuration 3.31

Introduction

SEE is a general purpose full-screen text editor designed for program entry rather than word processing. It features:

- invoking the compiler (C88) and the binder (BIND) from the editor — errors return control to the editor at the error line,
- invoking a copy of the shell (COMMAND.COM) to provide access to DOS functions,
- handling files larger than available memory,
- editing two files simultaneously,
- viewing the two files either on separate screens, or in two windows on the same screen,
- a macro facility which allows you to capture a series of keystrokes and replay them to ease repetitive tasks,
- automatic indentation,
- brace/bracket/parenthesis matching to ease program entry,

SEE is shipped configured for the IBM-PC and its clones. SEE may be reconfigured to run on other machines which support DOS but have different keyboard and/or screen interfaces than the IBM-PC (see Section 3.6).

Getting Started

Concepts

SEE does not directly manipulate a file on the disk. It brings a copy of the file into memory and performs all work on this internal copy. The file on the disk is not modified until the copy in memory is stored on the disk. If the file is larger than the internal buffer area, SEE will open "spill" files to swap the edited text in and out of memory. For this reason, you should not have any files named SEETMP.###, where ### is a series of three digits (currently restricted to 000, 001, 002, 003, and 004).

Commands are executed by typing the first letter of the command displayed on the menu line (the first line on the screen). For example, to execute the Delete command, simply type the letter 'D'; the case of the letter does not matter.

Whenever a block of text is deleted with the Delete command, the text is placed in a special area known as the copy buffer. Blocks selected with the Buffer command are also placed in this buffer. When the Copy command is used, the contents of this buffer is inserted into the text at the cursor location. The copy buffer is maintained as long as the editor is running and is shared by both files (if two files are being edited). This is the mechanism used to move text from one location to another or from one file to another.

The cursor indicates the location where all action will occur. It will be in one of three states: a double-bar cursor indicating command mode, a single-bar cursor indicating Insert mode or a block cursor indicating Exchange mode. The cursor is always visible on the screen. As the cursor is moved to an edge of the screen, the screen will scroll the text to keep the cursor in view, both vertically and horizontally. For example, if the cursor is moved down when it is on the last line of the screen, the screen will be scrolled up one line to show the line the cursor is on. Similarly, when the cursor is in the rightmost column of the screen and the cursor is moved to the right (assuming the line has more characters not currently displayed on the screen), the screen will be scrolled to the left by 15 columns to show the new location.

Starting the Editor

To start the editor to edit a new file named 'ergo', simply type:

see ergo

and the computer should respond with the screen:

```
Again Buffer Copy Delete Find -find Get Insert Jump  --space--
----- reading file: ergo ... -- new file 0 characters
-
```

The top line on the display is the menu line. This line displays the current mode of the editor and the commands available at any given time. In this first screen, the menu line contains the first set of commands available at the command level:

```
Again Buffer Copy Delete Find -find Get Insert Jump  --space--
```

Hitting the space bar displays the second set of commands:

```
List Macro Other Put Quit Replace Set Tag Wrap Xchange --space--
```

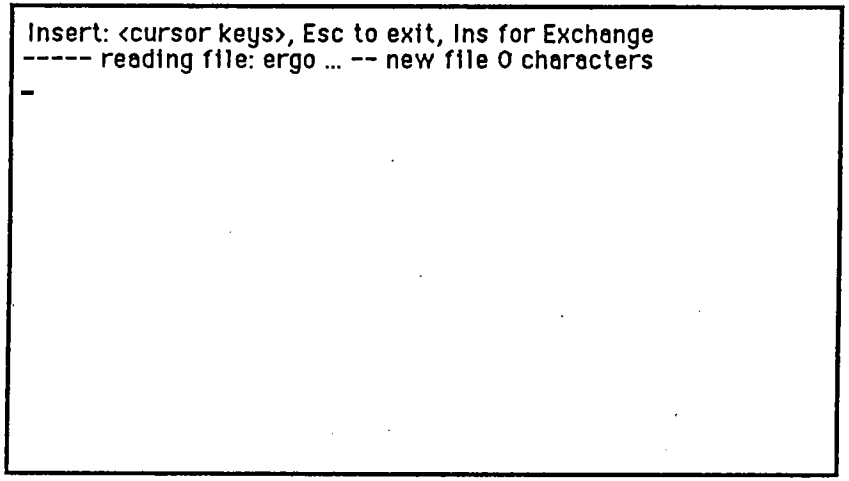
Hitting the space bar again will redisplay the first set of commands. The commands are fully described in Section 3.5 of this manual. Each command may be executed by typing the first letter of its menu item; for example, A for Again, B for Buffer, etc. The case of the command letter is not important.

The second line of the screen is used to display messages and status from the various commands and is naturally called the message line. The message "ergo ... -- new

file 0 characters" indicates that the file ergo has not been found and that the internal file buffer is empty.

Inserting and Editing Text

To insert text into the file, we must enter Insert mode. Do this by either typing the letter 'I' to execute the insert command, or by pressing the Ins key. The screen should now look as follows:



Note that the menu line has changed to indicate the types of actions, other than inserting text, that may be performed. Any character now typed, except for one of the special keys described in Section 4, will now be inserted into the text at the cursor location, just prior to the character that the cursor is on.

Now type in the lines:

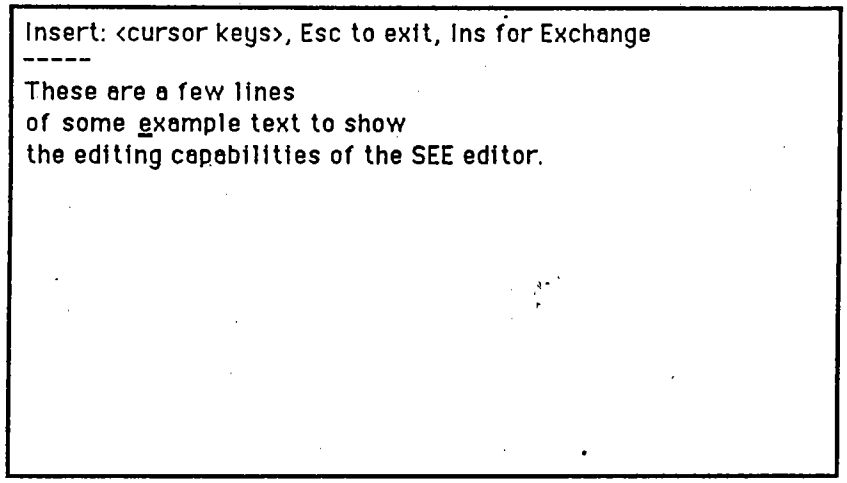
These are a few lines <Return>
of example text to shoe<Backspace>w<Return>
the editing capabilities of the SEE editor. <Return>
<Esc>

The screen should now look as follows:

```
Again Buffer Copy Delete Find -find Get Insert Jump  --space--
-----
These are a few lines
of example text to show
the editing capabilities of the SEE editor.
-
```

Note that the symbols <Return>, <Backspace>, and <Esc> represent the use of the return, backspace, tab, and Esc keys, respectively. The <Return> inserts a carriage-return, line-feed (CRLF) sequence into the file to begin a new line and the cursor moves down one line and to the left side of the screen. The <Backspace> key deletes the character preceding the cursor. The <Tab> key inserts a tab character into the file which is expanded to the next tab stop. Tab stops, by default, are located every four characters, however this value may be changed in the Set command. The <Esc> key breaks the editor out of Insert mode and places it back in command mode.

The cursor keys are used to move the cursor around the screen in small increments. Press the up-arrow key twice to move the cursor up to the beginning of the second line. Press the right-arrow key three times to move the cursor to the beginning of the word 'example'. Type the letter 'I' to put the editor into Insert mode and type the word 'some' without the quotes and add a blank. Note that as each character is typed, the rest of the line is "pushed" to the right. The screen should now look as follows:



Now hold down the control key (Ctrl) and press the right-arrow key three times. Note that the cursor jumps from one word to the next when using this combination of keys. See Section 4 for full details on all of the special keys. Also note that the editor does not have to be in command mode to use the cursor movement keys. Now hit the Ins key to change from Insert mode to Exchange mode; the menu line will display Exchange instead of Insert. In Exchange mode, the character at the cursor is overwritten by the new character rather than having the character inserted into the file. The only exception to this rule is when the cursor is positioned at the end of a line, characters are inserted rather than overwriting the CRLF end-of-line sequence. Exchange mode can also be entered from command mode by typing the letter 'X' for Xchange. Type the word 'display' and notice how the word 'show' is overwritten with the new word 'display'. Press the Esc key to go back to command mode. The screen should now look as follows:

```
Again Buffer Copy Delete Find -find Get Insert Jump  --space--
-----
These are a few lines
of some example text to display_
the editing capabilities of the SEE editor.
```

Press the Home key and note the location of the cursor. To delete this line, invoke the Delete command by typing the letter 'D', move the cursor down one line with the down-arrow key, and type the letter 'D' again to complete the deletion (the Esc key will also work). The second line has been deleted and placed in the copy buffer. Now type the letter 'C' to invoke the Copy command to retrieve the text that was deleted. Type the letter 'C' again and a second copy of the line is inserted. The copy buffer always contains the last Deleted or Buffered block of text. The screen should now look as follows:

```
Again Buffer Copy Delete Find -find Get Insert Jump  --space--
-----
These are a few lines
of some example text to display
of some example text to display
the editing capabilities of the SEE editor.
```