DeSmet C

Development Package

Published and Distributed by

C WARE CORPORATION

Sunnyvale, California

DeSmet C

Development Package

Mark DeSmet

Published and Distributed by

C Ware Corporation

Sunnyvale, California

DeSmet C Development Package

Version 2.5 — October, 1985
Version 2.4 — October, 1984
Version 2.3 — April, 1984

Published by: C Ware Corporation
P.O. Box C
Sunnyvale, CA 94087
USA
(408) 720-9696
Telex 358185 C WARE SNVLE

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.

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 available. 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 Mike Ouye (pronounced Oh´Way) for converting the manual into MacWrite format, and both Glen Emigh and Pacific Data Works for proofreading its many revisions.

We especially thank the following for repeatedly throwing their bodies in front of and on top of the more explosive parts of the Version 2.5 Beta release — Clopper Almon, Dave Auslander, Glen Emigh, Panos Galidas, Scott Guthery, Bill Hunt, George O'Neal Keyes II, Scott Lewis, Greg Mansfield, Bill Morrison, and Jim Rome, as well as the folks at Kris Jamsa Software (Las Vegas, NV), Nanosoft (Ft. Belvoir, VA), and Pacific Data Works (Santa Monica, CA).

Table of Contents

5. The ASM88 8088/8086 Assembler

6. The BIND Object File Linker

7. The LIB88 Object File Librarian

8. The D88 C Language Debugger

9. Utility Programs

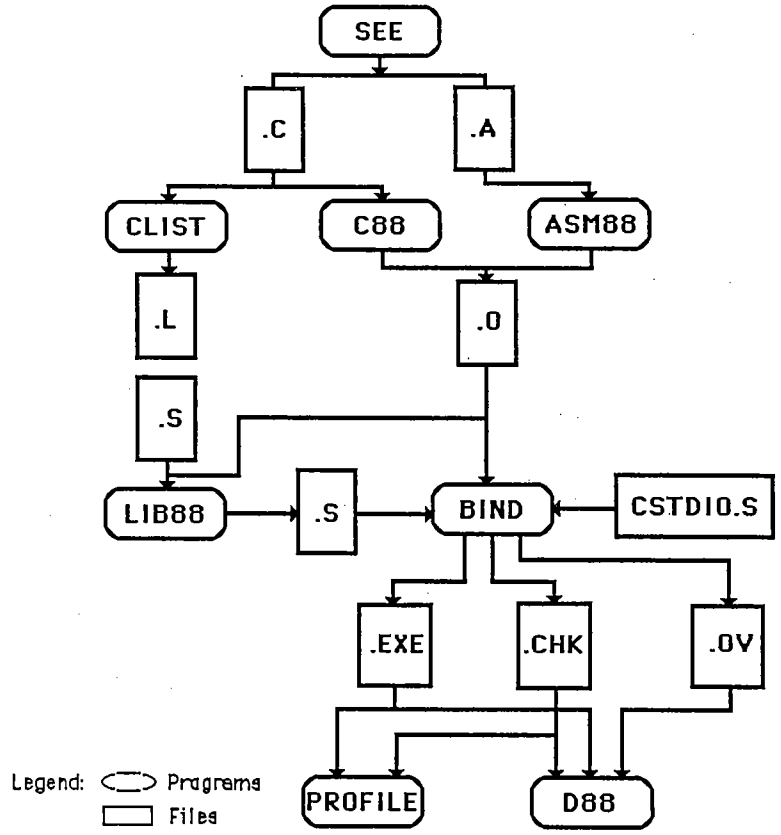
10. The CSTDIO.S Standard Library

Appendix A: Messages

Appendix B: The ASM88 Assembly Language

1. Introduction

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 MS-DOS, 1.xx, 2.xx, and 3.xx. The program execution profiler requires the use of MS-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]

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.

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].

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).

2. Getting Started

First things first. Copy all of the files from the distribution disks onto a set of working floppy diskettes or hard disk. The package is not copy-protected so the MS-DOS copy command can be used to copy the files. The package is distributed on two DOS 2 double-sided (360KB) diskettes. If your machine has single-sided drives, or only supports DOS 1, the package is available on "flippies" (each side of each diskette is formatted as a single-sided diskette and the diskette must be physically flipped over to access the other side). The distribution diskettes should never be used, they should be kept as the backup copy of the package.

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 and D88 must be configured before they can be used. See the notes in the file CONFIG.C on Disk #2 for configuration details.

2.1 Distribution Disks

The package is distributed on two 5 1/4 inch floppy diskettes, labeled Disk #1 and Disk #2. If you requested "flippies", Disks #3 and #4 are on the back sides of Disk #1 and #2, respectively. Disk #3 contains the rest of Disk #1, while Disk #4 completes Disk #2.

Disk #1:

C88.EXE:	The first pass of the C compiler.
GEN.EXE:	The second pass of the C compiler.
ASM88.EXE:	The assembler and third pass of the C compiler.
SEE.EXE:	The full-screen editor.
STDIO.H:	Include file for the Standard I/O package.
MATH.H:	Include file for the Standard Math package.
BIND.EXE:	The object file linker.
LIB88.EXE:	The object file librarian.
EXEC.O:	The Exec() and Chain() functions.
CSTDIO.S:	The standard C function library with software floating-point support.
CSTDIO7.S:	The standard C function library with 8087 support. To use this library, rename it to CSTDIO.S.

Disk #2:

RAM.COM:	RAM Disk driver for DOS 2.0 and later operating systems.
PCIO.A:	Source code for the PC screen functions.
BUF128.EXE:	128 byte type-ahead buffer program.
BUF128.A:	Source code for BUF128.EXE.
DUMP.EXE:	The hex file display utility.
DUMP.C:	Source code for DUMP.EXE.
LIFE.EXE:	Full screen game of Life.
LIFE.C:	Source code for LIFE.EXE.
BUGS!.EXE:	Arcade game (use 'BUGS! c' for color displays).
LATER.C:	Source code for a file modification date checking program.
CB.C:	Source code for a brace matching program.
CLIST.EXE	The C listing and cross-reference utility.
PROFILE.EXE:	The program execution profiler.
PROFSTAR.EXE:	Used by PROFILE.EXE.
PROFEND.EXE:	Used by PROFILE.EXE.
COMPARE.EXE:	The source code comparison utility.
D88.EXE:	The C source-level symbolic debugger.

Disk #2 also contains various object files and configuration files for generating the SEE editor and D88 debugger for systems with displays other than the IBM PC. CONFIG.C contains the display and keyboard dependent routines. See the comments in the CONFIG.C file for instructions on customizing SEE and D88.

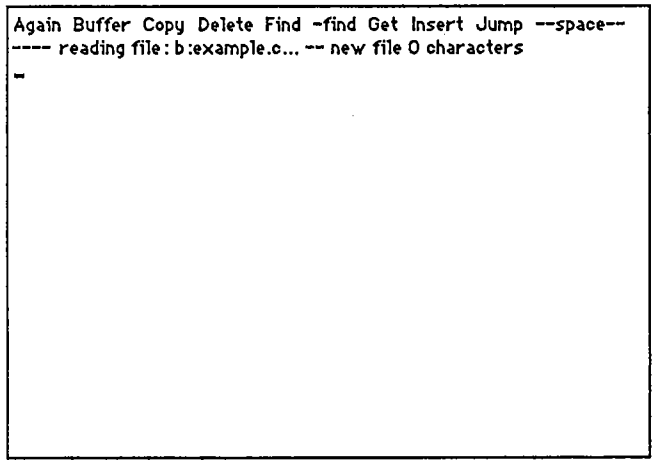
2. 2 A Short Example

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:



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:

```
Again Buffer Copy Delete Find -find Get Insert Jump --space--
----
main() {
    printf("%d plus %d is %d\n",2,2,2+2);
    }
-
```

Now that the program is entered, it must be saved to the disk. Since the name of the file was specified on the command line, just type the sequence of characters, 'Q' for Quit and 'S' for Save-exit. This will store the copy of the file in memory onto the disk and exit from the editor.

To compile the program just entered, type:

c88 b:example

and the computer will display:

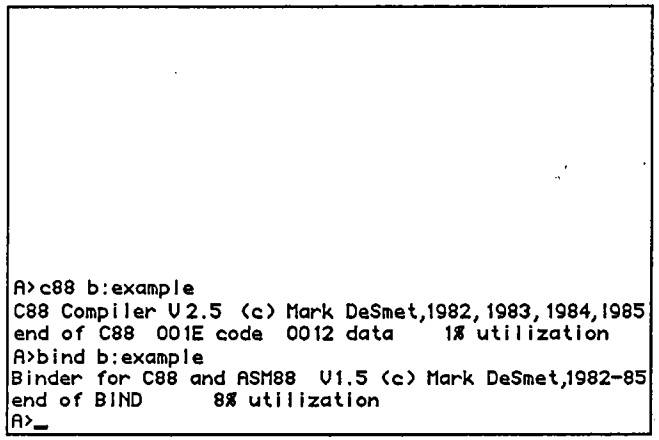
```
    }



A>c88 b:example
C88 Compiler V2.5 (c) Mark DeSmet,1982,1983,1984,1985
end of C88  001E code  0012 data    1% utilization
A>_
```

If there were errors during the compilation, go back to the editor and make sure that the program was entered correctly. (Note that the .C extension was not needed on the filename given to the compiler. When no extension is given, the compiler automatically assumes .C as the extension, just as the assembler assumes .A and the binder assumes .O).

Now it's time to bind (link) the program to the standard library and create the application program. Do this by typing:

bind b:example

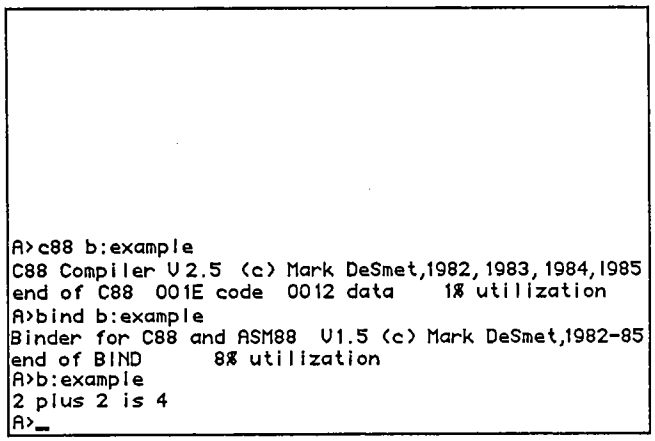
The binder will automatically search the CSTDIO.S library so it should not be included in the command line. When the binder successfully links the program, the screen should appear as follows:



The application is now ready to run. Type:

b:example

and the computer should respond with:



Don't worry if some of the utilization numbers are different from those shown in the example. These numbers will vary depending on the system being used (i.e. amount of memory, RAM disk installed, etc.).

2.3 Setting Up DOS 2.xx, 3.xx, ...

For systems utilizing DOS 2.x or later versions of the operating systems, make sure that the ASCII text file CONFIG.SYS exists on your boot disk. If it doesn't exist, you can create it with SEE. The file must contain the line:

FILES=20

since C88 supports 20 open files — stdin, stdout, stderr, and 17 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.

2.3.1 RAM DISK

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

copy the RAM.COM file from the distribution diskette to the system disk and add the line:

DEVICE=RAM.COM n

to the CONFIG.SYS file. The parameter, n, is a decimal number from 32 to 650 indicating the size of the Ram Disk. The value is specified in units of one kilobyte (1024).

Re-boot the system to install the Ram Disk. The drive letter used for this 'disk drive' is dependent on the configuration of the system. DOS will install the Ram Disk at the first free device "slot". For an IBM PC with two floppies, this will probably be drive C:. For an XT, it will probably be drive D:. Sanyo 550/5 reserves the first four slots for its floppies, so the Ram Disk is drive E:. To find where DOS has installed the Ram Disk, use

chkdsk x:

where x takes on the values c, d, ... You will get either a disk error, or a return showing the size of the Ram Disk. Once you find it, the Ram Disk will always be the same until you add other device drivers before it in the CONFIG.SYS file.

2.3.2 Completion Codes

The C88, ASM88, BIND and LIB88 programs set the completion code to:
zero if no warnings or errors occurred,
one if warnings were issued, and
two 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.

3. The SEE™ Text Editor

3.1 Introduction

SEE is a general purpose full-screen text editor designed for program entry rather than word processing. It handles files larger than memory and can edit two files simultaneously. Its macro facility allows you to capture a series of keystrokes and replay them to ease repetitive tasks. It also features automatic indentation and 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 MS-DOS but have different keyboard and/or screen interfaces than the IBM-PC (see Section 3.6).

3.2. Getting Started

3.2.1 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.

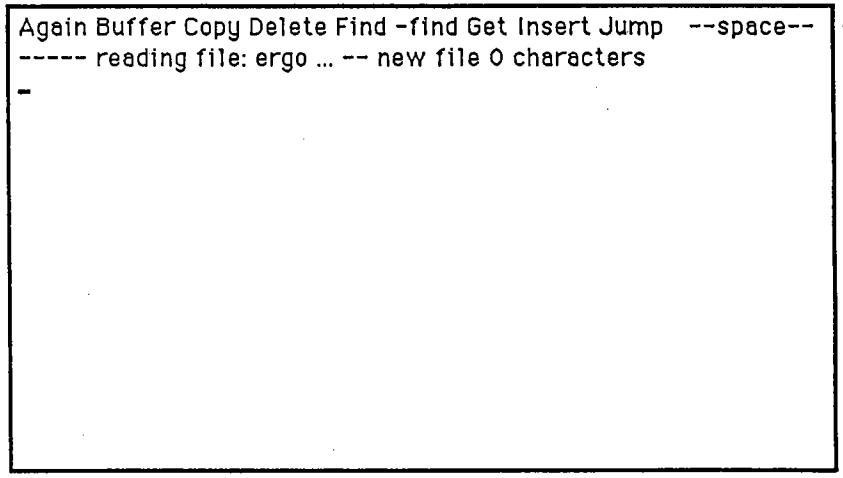
The cursor indicates the location where all action will occur. It will be in one of three states: a double-bar cursor indicating command mode, a single-bar cursor indicating Insert mode or a block cursor indicating Exchange mode. The cursor is always visible on the screen. As the cursor is moved to an 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.

3.2.2 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:



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--

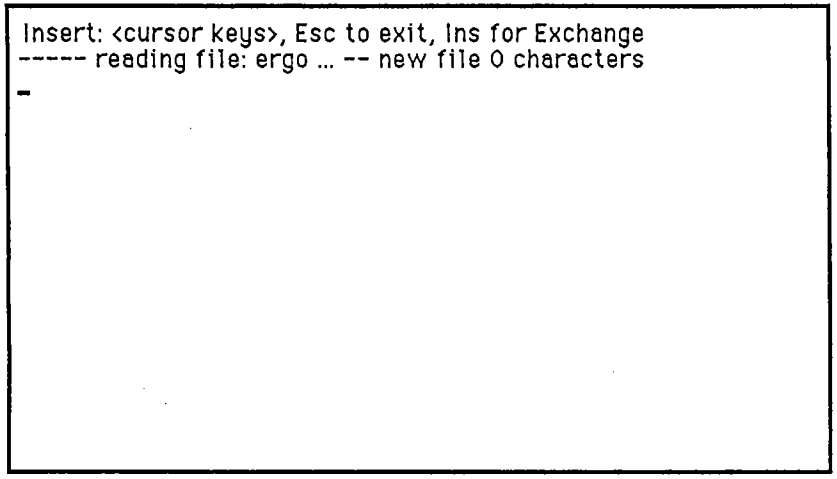
Hitting the space bar again will redisplay the first set of commands. The commands are fully described in Section 3.5 of this manual. Each command may be executed by typing the first letter of its menu item; for example, A for Again, B for Buffer, etc. The case of the command letter is not important.

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.

3.2.3 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:

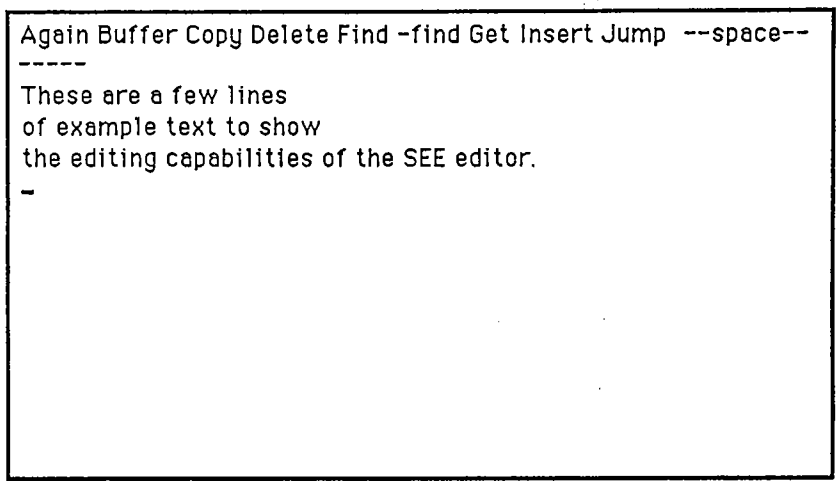


Note that the menu line has changed to indicate the types of actions, other than inserting text, that may be performed. Any character now typed, except for one of the special keys described in Section 4, will now be inserted into the text at the cursor location, just prior to the character that the cursor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

Now type in the lines:

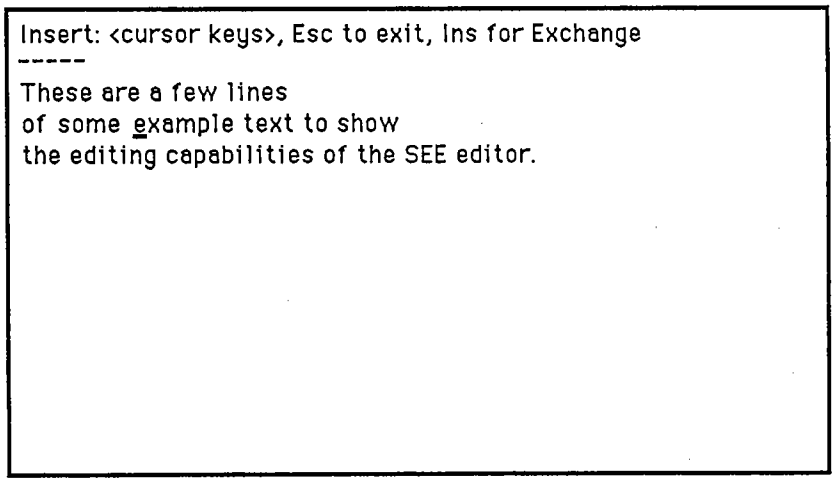
These are a few lines <Return>
of example text to shoe<Backspace>w<Return>
the editing capabilities of the SEE editor. <Return>
<Esc>

The screen should now look as follows:



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.
```

To find the first occurrence of the word 'display', press the letter 'F' to invoke the Find command. Type in the word 'display' (without the quotes) and type either Esc or Return to begin the search. The cursor should now be positioned after the word 'display' on the second line. To replace the next occurrence of the word 'display' with the word 'show', press the letter 'R' to invoke the Replace command. Notice that the previous search string 'display' now appears on the message line. Since this is the string to be replaced, simply press the Esc or Return key to select the string (rather than retyping the string). Type in the string 'show' and hit the Esc or Return key to execute the command. Press the Home key twice to move the cursor to the top of the screen. The screen should now appear 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 show
the editing capabilities of the SEE editor.
```

Another useful feature in SEE is its ability to record a series of keystrokes, command, cursor keys, etc., and replay them on command. These recordings are called **macros**. To create a macro, type 'M' to invoke the Macro command, type 'R' to indicate that a recording is to be made, and select the function key (F1 through F8) that is to be used to invoke the macro. In this example, press the F1 key. The message line now displays the line:

recording Macro F1, use Macro key to complete recording

This message will be displayed after every command to indicate that a macro recording is in progress. Now, any commands or special keys typed will be

recorded into the macro until the Macro command is executed once again. For this example, execute the following commands:

I@<Esc><control right-arrow>M

Macro F1 is now defined to insert the '@' character in front of each word. To execute the macro, press the F1 key. To execute the macro a fixed number of times, say five times, type the number 5 and then the function key F1. The macro is executed five times. To execute the macro for the rest of the words in the file, type in a large number or use the more convenient '/' character to indicate the number 32767, the largest number. Type '/' and press the F1 key. The screen should now appear 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 @show
@the @editing @capabilities @of @the @SEE @editor.
@-
```

3.2.4 Saving the File

Recall that all of the editing was performed on the file in memory. This copy of the file must be written out to the disk. Type the letter 'Q' to enter the Quit menu. The choices under the Quit menu are:

BAKup Exit Initialize Save-exit Update Write

Each menu item is explained in detail in Section 5 under the Quit command. Press the letter 'S' to save the memory copy of the file to the disk file named 'ergo' which was entered at the beginning of this example. This selection will also terminate the editor.

3.2.5 Editing Existing Files

Now to edit the file ergo again, simply type the line:

see ergo

The editor will be loaded and will attempt to load the file ergo. If the file was loaded correctly, the screen should appear as follows:

```
Again Buffer Copy Delete Find -find Get Insert Jump  --space--
----- reading file: ergo ... 156 characters
@These @are @a @few @lines
@of @some @example @text @to @display
@of @some @example @text @to @show
@the @editing @capabilities @of @the @SEE @editor.
@
```

Type 'Q' to select the Quit command and then type 'E' to exit from the editor without writing the file out, since nothing has changed.

You now have a basic understanding of how to edit files with the SEE editor. Practice editing other files using the skills developed in this example. Don't be afraid to experiment. Remember that as long as you don't write the file back out to the disk, the old copy is safe. When you are comfortable with these editing features, look through the rest of the manual to see what else can be done and experiment with some new features.

3.3. The Invocation Line

There are a few different options available when starting the SEE editor. Invoking SEE with the command line:

see

will bring up the editor with an empty buffer and no filename specified. To save the file to disk, use the Write option under the Quit command described in Section 5.

Invoking SEE with the command line:

see <filename>

will have the editor load the file <filename> if it exists. <filename> will be used by the Update and BAKup options in the Quit command. If the file doesn't exist, SEE will act as if it existed but was a zero length file. Note that the file is not created until it is written out to disk.

Invoking the editor with the command line:

see <filename1> <filename2>

will have the editor load the text from <filename1> but will write out the text to <filename2>. <filename1> will not be altered by the edit session.

3.4. The Keyboard

This section describes the special keys used by the SEE editor as defined for the IBM-PC keyboard. If the editor has been reconfigured for a different keyboard, you will have to map the reconfigured keys to the IBM-PC keys to understand the following documentation.

3.4.1 Cursor Movement Keys

In the following descriptions, the up-arrow (^) character preceding the name of the key implies that the control (Ctrl) key must be held down while the key is pressed.

Home: When the Home key is pressed once, the cursor will move to the beginning of the current line (the line that the cursor is currently on). If the Home key is pressed twice in succession, the cursor will move to the beginning of the first line on the screen.

^Home: When the control key is held down as the Home key is pressed, the cursor will be moved to the beginning of the first line of the file.

End: When the End key is pressed once, the cursor will move to the end of the current line (positioned just before the CRLF end of line sequence). If the End key is pressed twice in succession, the cursor will move to the beginning of the last line on the screen.

^End: With the control key held down, the End key will move the cursor to the end of the file.

PgUp: Moves the cursor to the fourth line of the previous screenful of text. The next screen starts from the current screens fourth line from the bottom.

PgDn: Moves the cursor to the fourth line of the next screenful of text. The previous screen overlaps the current screen with the first four lines of the current screen.

UpArrow: The up-arrow key moves the cursor up one line. The column that the cursor is in remains the same. If the cursor is positioned beyond the end of a line because of this action, the visible cursor is shown beyond the end of the line but is logically located just before the CRLF

sequence (The cursor is moved to this location when some other operation is performed.) If the cursor is already on the top line of the screen, the screen is scrolled down one line to show the new line.

DownArrow: The down-arrow key moves the cursor down one line. Again, the visible cursor remains in the same column as described above. If the cursor is already on the last line of the screen, the screen is scrolled up one line to show the new line.

LeftArrow: The left-arrow key moves the cursor one character to the left. If the cursor is at the left edge of the screen, and the screen has been scrolled to the right, the screen will scroll back to the left by 15 character locations to show the new cursor position. If the screen had not been scrolled implying that the cursor was on the first character of the line, the cursor moves to the end of the previous line.

^LeftArrow: With the control key held down, the cursor will move to the left in word increments rather than character increments. Each time this combination is pressed, the cursor will move to the last character of the previous word where word is defined as a sequence of letters or digits. Any other character separates the words.

RightArrow: The right-arrow key moves the cursor one character to the right. If the cursor is at the right edge of the screen and more text exists in the current line, the screen is scrolled to the right by 15 characters to show the new location. If the cursor was positioned at the end of the line, then the cursor is moved to the beginning of the next line.

^RightArrow: This combination moves the cursor to the beginning of the next word.

Return: The return key is normally used to insert a CRLF end of line sequence into the text, thereby positioning the cursor at the beginning of the next line. If the return key is pressed while in command mode, the cursor will simply move to the beginning of the next line.

3.4.2 Editing Keys

Backspace: The backspace key deletes the character to the left of the cursor. If the cursor is positioned at the beginning of a line, the CRLF end of line sequence is removed and the two lines are joined to form a single line.

Del: The delete key deletes the character under the cursor. If the cursor is positioned on the CRLF end of line sequence, then the next line is joined with the current line.

Ins: The Ins key is used to toggle between Insert and Exchange modes. At the command level, it will place the editor into Insert mode.

F1-F8: The function keys F1 through F8 are available for user-defined macros. Macros may be saved with the Macro-Save command.

^C or ^Break: Holding down the control (Ctrl) key and hitting the letter 'C' or the Break key (Scroll Lock) will normally stop the execution of a command (where reasonable). This is useful when you decide not to execute the Find command and are in the middle of typing in the search string. Typing control-C will abort the Find command without modifying the old search string. This key combination will also stop an executing macro.

3.4.3 The DOS Key

Under MS-DOS 2.0 and later versions of the operating system, the F9 function key allows another command shell to be executed while the editor and text remain in memory. When the F9 key is pressed, the screen will display the DOS copyright message and will prompt for a command. You can execute any command, even another copy of the editor (although this is not recommended because of conflicts with the spill files). When you want to return to the editor, type the DOS command

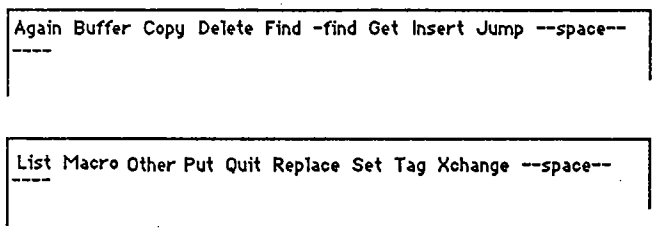
exit

and the text will be redisplayed as if the F9 key never had been pressed.

DOS, SEE, and your text occupy about 128K. You must have at least an additional 64K of unused memory in your machine to use the DOS feature.

3.5. Commands

In command mode, the menu line displays the commands available for editing and manipulating the text. Since the names of the commands are too long for a single menu line, the menu is broken into two parts. To toggle between each part of the command menu, press the space bar.



Command Menus

To invoke a command, type the first letter of the command. To terminate a command, press the escape <Esc> key. A command may be aborted by holding down the control key, Ctrl, and typing the letter C (control-C).

Many commands will take a repetition count to execute the command multiple times before completing. The repetition count takes the form of a decimal number or a slash (indicating a very large number). It is entered prior to typing the first letter of the command. Some commands — Find, -find, and Replace — may be given a question mark (?) repetition count indicating that the editor should prompt after each string is found. Note that at the command level, the cursor movement keys may also be repeated by using a repetition count. This also means that if a mistake is made in the repetition count, the Backspace key cannot be used to correct the mistake. The command must be aborted .

In the following descriptions of the commands, <rep> indicates that the command takes a repetition count and <rep | ?> indicates that it will take a repetition count or question mark repetition count.