CREATING PROGRAMS

for

ABLE SERIES COMPUTERS

October 1978

NEW ENGLAND DIGITAL CORPORATION
Box 305
Norwich, Vermont 05055

TABLE OF CONTENTS

Creating Programs for Able Series Computers

New England Digital's Scientific XPL Operating System is an easy to use Operating System that is used to create real-time programs for an ABLE computer. The system operates in an interactive mode using commands from the terminal to perform a wide variety of editing functions. Features are included in the operating system that allow a novice user to easily enter and debug short real-time programs, while a computer professional can quickly and efficiently develop complete system modules.

The Operating System includes an Editor, a 3-pass Optimizing Compiler, and several utility programs that perform many important functions. Programs are entered into the system on a line by line basis using line numbers. Since the Editor is an integral part of the Operating System, a special edit command is not required and new lines may be entered at any time. This feature greatly simplifies the interactive task of writing short test programs.

Commands such as NEW, OLD, SAVE, and REPLACE are used to direct the operation of the system. Once a program has been entered into the system and saved on diskette, it may easily be compiled and automatically executed by using RUN. Most compilations are performed in 7 to 15 seconds, depending on system configuration. Large programs are compiled at a rate of over 1000 lines per minute. Since compiled programs are stored in memory image format, pre-compiled programs may be executed with essentially instantaneous response times.

Video editing is accomplished by using the CRT command. When the system is in CRT mode, terminal output is presented on a page by page basis. Single lines or complete pages may be presented on the screen by pressing a single key on the terminal. The position of the cursor is automatically controlled by using control-Z, to simplify character by character editing. Additionally, a control-S may be used at any time to freeze the display when using a high speed video terminal.

Edit commands are used to process entire blocks of information by specifying groups of lines. Functions such as EXTRACT, DELETE, MOVE, and APPEND may easily be performed on segments of a file. The CHANGE and LOCATE commands provide string editing functions that operate on a single line, a group of lines, or several groups of lines at the same time. The PRINT command is available for directing output to a remote hard copy terminal.

System Features

The Scientific XPL Operating System is the most efficient software package available for scientific and industrial control applications. Scientific XPL is a structured programming language that provides the programmer

with the programming ease of a modern high level language, yet also provides the capability to perform high speed data collection, bit manipulation, logical expression processing, and many other real-time functions with an efficiency that on other systems is only available with assembly language programs. The Scientific XPL Compiler forms the heart of the Scientific XPL Operating System.

The efficiency of Scientific XPL derives from the three pass architecture of the compiler. The three pass approach has many advantages when compared with the interpretive or single pass approach used on other systems. Since all of the symbol table processing is performed during pass one, a great deal of memory is made available during pass two for a powerful optimization routine. This routine creates a tree structure for every arithmetic expression which is then carefully inspected for common sub-expressions, use of the multiply/divide, and so forth. Many Scientific XPL statements are compiled into a single, directly executable machine instruction.

This manual explains how the Scientific XPL Operating System is used to enter, save, compile, and execute programs for an ABLE computer. The features available in the Scientific XPL language are explained in a separate manual called the Scientific XPL Reference Manual.

Starting the System

1. Turn on the power to the computer, terminal, and any other equipment you will be using.

2. Insert a System Disk in the left-most drive. Be sure to observe the proper orientation when inserting the disk; the label should be up and the slot should face forward (away from you). Also, be sure to use a System Disk that matches the configuration of the system you are using. The configuration information is always recorded on the diskette label.

3. Insert a User Disk in the right-most drive.

4. Press the Load button on the front of the computer.

The computer will respond with a READY on the terminal. The system is now ready to accept a command, or list of commands, instructing the system what to do next.

If the terminal "beeps" when you press Load, either the System Disk has been corrupted by exposure to a stray magnetic field or you have obtained the wrong diskette. You should try a different System Disk to alleviate this problem.

If the terminal does not respond with READY after loading, perhaps the System Disk has been corrupted by incorrect use of DISKWRITE.

Appendix I lists several precautions that should always be followed when handling diskettes. Diskettes can be damaged by stray magnetic fields, dust, or even fingerprints, so follow these precautions carefully.

Appendix II provides introductory information on Using the Terminal. First-time users should consult this section for further information.

Be sure to remove both the System and User diskette whenever the power to the computer is turned on or off. Diskettes can be corrupted by unfavorable power transients if this precaution is not followed.

When entering information into the computer from the terminal, a control-Z or rub-out is used to delete the previously typed character to correct typing errors (a control-Z is generated by holding the CRTL key down and then typing Z). On most CRT terminals, the cursor will backspace, overprint the deleted character with a blank, and then backspace the cursor again to indicate where the next character will appear.

To delete an entire line of input, a control-X is used. The word DELETED will appear on the terminal and the new line may then be entered.

If a control-S character is typed while the computer is printing on the terminal, the computer will halt until any other character is typed. This is

used to freeze the display when using a high speed video CRT.

Description of Monitor Commands

User commands are typed in from the terminal to direct the operation of the system. Each command may be abbreviated to its first three letters; for example, the LIST command may be typed as LIS. Several commands may be typed to the computer on one line. To do this, type a slash (/) or period (.) at the start of the multiple command line and use a slash or period to separate the different commands on the same line.

When you are typing commands into the computer, always remember that the computer will not process a line of input until the carriage return key is pressed, which you must do at the end of each line. On some terminals the carriage return is indicated by the word RETURN, while on other terminals it is indicated by the two letters CR.

The carriage return key is usually always located in the upper right hand corner of the terminal keyboard. Look to the right of the P key if you are having trouble.

Summary of Commands

Basic File Management Commands:

NEW (filename)	initiates entry of a new program
OLD (filename)	calls up a previously saved program
RENAME (filename)	changes name of current file
SAVE (filename)	stores a new file on a User Diskette
REPLACE (filename)	stores a revised version of an old program
UNSAVE (filename)	destroys the saved copy of a program

Performance Commands:

RUN	initiates complete compilation/loading process
COMPILE	initiates the compilation of an XPL program
CATALOG (LEN, ALL)	prints a list of SAVED programs

Auxiliary Commands:

LENGTH	prints out the number of words in a program
NAME	prints out the name of the current file
CRT	directs the system to enter the CRT editing mode
NCRT	directs the system to return to the normal editing mode

Editing Commands:

LIST (ln,ln-ln)	prints entire program (or selected lines)
PRINT (ln,ln-ln)	prints entire program (or selected lines) on the remote printer
RESEQUENCE	renumbers the lines of a program starting with 100 and incrementing by 10
SEQUENCE	renumbers the lines of a program starting with 1
DESEQUENCE	removes the line numbers from each line of a program
EXTRACT (ln,ln-ln,ln)	extracts selected lines from the current file
DELETE (ln,ln-ln,ln)	deletes selected lines from the current file
LAST	prints the number of the last line in the program
MOVE (ln-ln,ln)	moves a block of lines to alter a specified line number (also resequences the file)
APPEND (filename),filename)	joins the specified file to the end of the current file and performs a resequence
JOIN (filename),(filename)	joins the specified file to the end of the current file without changing line-numbers
LOCATE (string,ln,ln-ln)	locates specified occurrences of a given character string
CHANGE (string1,string2, ln,ln-ln)	changes all occurrences of the specified string1 to read string2
BUILD	directs the system to enter the automatic line number mode

Basic File Processing: OLD and NEW

A file is a block of information in the computer. A file may be a source file containing line numbers and XPL program statements, or a compiled file containing instructions to the computer. The commands NEW and OLD are used to tell the computer that you wish to use a file. NEW says that you wish to create a new file, while OLD lets you call up a file that already exists.

Creating a NEW file

A new file is created by typing the command NEW, followed by the new name that you wish to assign to the new file:

```
NEW TESTPROG
```

Remember that a carriage return (CR) must be typed at the end of each line before that line will be processed by the computer. The specified file name may be up to 8 letters long. Be careful to not use a space or control character in the filename since this would create confusion when using CATALOG.

The new file is typed in one line at a time, with each line beginning with a line number. You need not type the lines in increasing numerical order; the computer automatically sorts them by number. Be sure to use line numbers less than 30000 since larger line numbers cannot be processed by the Editor program. When typing line numbers, remember that zero, oh, one, and el are four distinct characters.

If you notice a mistake on a line that you have already typed in, just retype the entire line; the system will automatically retain only the last version typed. To delete an entire line, just type the line number and immediately follow it by a carriage return (CR).

Observe the following example. Underlined segments indicate those lines that must be typed while the other lines are automatically printed by the computer.

```
NEW TESTPROG
READY
100 this is line 100 in file TESTPROG
110 this is line 110
105 notice I typed line 105 after line 110
90 and here is line 90!
100
LIST
90 and here is line 90!
105 notice I typed line 105 after line 110
110 this is line 110
READY
```

In the example above, the first line is 100; notice the second line that was typed began with line number 110. It is a good practice to space the line numbers by 10 so that new lines may be inserted at a later time. Remember that each line ends with a carriage return (CR).

Line 105 was typed into the computer after line 110, but when the file was printed on the terminal in response to the LIST command, the lines were printed out in the correct order.

In the above example, the last line number typed was line number 100 followed immediately by a carriage return. As mentioned, typing a line number followed immediately by a carriage return deletes the specified line from the file.

SAVing a File

Once a source file has been typed in, the SAVE command is used to store the file on the diskette. Files are usually saved on the right-most diskette (the user disk).

```
SAVE
READY
```

Once a file is saved, it may be called up at a later time via the OLD command:

```
OLD TESTPROG
READY
```

Suppose that after saving a file, you wish to add a new line. For this change to be preserved with the rest of the file, use the command REPLACE:

```
OLD TESTPROG
READY
120 this is new line 120
LIST
90 and here is line 90!
105 notice I typed line 105 after line 110
110 this is line 110
120 this is new line 120
READY
REPLACE
READY
```

The SAVE command is used only the first time a file is saved. Any later additions or changes to a file already saved (it is then an old file) are preserved with the REPLACE command.

The REPLACE command is necessary because there is a distinction between the file you are currently using and changing at your terminal, and the version of that file which remains stored on the diskette; these are always two separate entities. The current file created by the command NEW or OLD, exists only temporarily as the file you are presently using. If you are using an OLD file, the current file is a temporary copy of the saved file. As you type information into a current file, that file is the only copy of the new information until you type the SAVE or REPLACE command. As soon as you call up another file or turn off the system, the current file you had been using disappears; it is no longer in existence.

A saved file is a file preserved on a diskette. When you type SAVE or REPLACE, you create a saved file which is a copy of your current file as it exists at that moment. Once saved, the preserved version of the file remains unchanged on the diskette until you give a command to REPLACE it or to UNSAVE it. A current file called via the command OLD is not the saved file itself but merely a temporary copy of it; the saved file is still preserved and

unchanged on the diskette. This is why any changes you type into your current file do not affect the saved version of the file in any way until you type REPLACE. Then and only then is the stored file replaced by the revised current file.

In summary, if you make entries in or changes to any current file and then leave that file without SAVING or REPLACING it, there will be no stored record of those entries. Any previously saved version, however, will still be preserved, unchanged, on the diskette.

Command UNSAVE

The command UNSAVE operates only on the saved version of a file. Typing UNSAVE destroys the saved version of what is then the current file but does not affect the current file itself. As long as you do not change the current file with a NEW or OLD command, you still have the temporary, current version of the file. That current file is not lost until you call another file or turn off the system.

Command RENAME

The RENAME command is used to assign a new name to your current file. After performing the RENAME command, the original saved version of the current file will still be unchanged. The effect of rename will only be on later SAVE or UNSAVE commands:

```
OLD TESTPROG
READY
RENAME TESTER2
READY
SAVE
READY
CATALOG
TESTPROG      TESTER2
READY
```

The CATALOG Command

The CATALOG command (abbreviated CAT) prints out a list of the names of the user programs that are saved on a diskette. It is often used to determine how many files are stored on a diskette, and to see which diskette a new program should be SAVED on.

The command CAT LEN will print out the length of each file (in 16-bit words) as well as the file names:

```
CAT
TESTER          TESTER2          TESTPROG
READY
CAT LEN
TESTER    567       TESTER2    603       TESTPROG    3756
READY
```

```
READY

CAT

NO FILES ARE SAVED.

READY

NEW TESTPROG
READY

100 THIS IS LINE 100 IN TESTPROG
110 THIS IS LINE 110 IN TESTPROG
105 NOTICE LINE 105 WAS TYPED IN AFTER LINE 110
LIST
100 THIS IS LINE 100 IN TESTPROG
105 NOTICE LINE 105 WAS TYPED IN AFTER LINE 110
110 THIS IS LINE 110 IN TESTPROG
READY

SAVE
READY

CAT

TESTPROG

READY

CAT LEN

TESTPROG 55

READY

RENAME TESTP1
READY

SAV
READY

CAT

TESTPROG TESTP1

READY

NEW TESTP2
READY

10 THIS IS LINE 10 IN TESTP2
5  THIS IS LINE 5 IN TESTP2
LIS
5  THIS IS LINE 5 IN TESTP2
10 THIS IS LINE 10 IN TESTP2
READY
```

Examples of

CATALOG	(CAT)
NEW	
SAVE	(SAV)
RENAME	(REN)
OLD	
LIST	(LIS)
REPLACE	(REP)
UNSAVE	(UNS)

underlined segments indicate lines typed by user.

```
SAV
READY

CAT

TESTPROG TESTP1    TESTP2

READY

CAT LEN

TESTPROG 55     TESTP1    55     TESTP2    28

READY

OLD TESTPROG
READY

LIST
100 THIS IS LINE 100 IN TESTPROG
105 NOTICE LINE 105 WAS TYPED IN AFTER LINE 110
110 THIS IS LINE 110 IN TESTPROG
READY

OLD TESTP2
READY

LIST
5   THIS IS LINE 5 IN TESTP2
10 THIS IS LINE 10 IN TESTP2
READY

15 THIS IS A NEW LINE IN TESTP2
25 AND NEW LINE 25 - THESE ARE NOT SAVED YET
LIST
5   THIS IS LINE 5 IN TESTP2
10 THIS IS LINE 10 IN TESTP2
15 THIS IS A NEW LINE IN TESTP2
25 AND NEW LINE 25 - THESE ARE NOT SAVED YET
READY

CAT LEN

TESTPROG 55     TESTP1    55     TESTP2    28

READY

REPLACE
READY

CAT LEN

TESTPROG 55     TESTP1    55     TESTP2    66

READY
```

```
UNSAVE TESTPROG
READY

CAT

TESTP1    TESTP2

READY

UNSAVE TESTP2
READY

CAT

TESTP1

READY

UNS TESTP1
READY

CAT

NO FILES ARE SAVED.

READY
```

Accessing Programs on Remote Devices

The Scientific XPL Operating System includes a feature that allows the user to access programs that are saved on remote peripheral devices such as magnetic tape. This feature may be used in conjunction with the OLD, SAVE, REPLACE, UNSAVE, and CAT commands.

A remote device is specified by using a slash (/) followed by a two character device identifier. The slash and identifier are always typed after the filename. The following remote devices are available in Scientific XPL:

/FØ	System Flexible Diskette - drive 0 (left-most drive)
/F1	System Flexible Diskette - drive 1 (right-most drive)
/RØ	Remote Minidiskette - drive 0
/R1	Remote Minidiskette - drive 1
/TØ thru /T7	Tape drives Ø through 7
/DØ thru /D3	Cartridge disk drives Ø through 3

Using a remote device specifier is very straightforward. Observe the following OLD command:

```
READY

OLD      TESTPROG/FØ
READY
```

When using a remote specifier with OLD, the operating system will look on the specified device for the program. A suitable error message will be printed if there is no program saved by that name on the device.

The CAT command can be used to find out which programs are saved on a remote device:

```
READY

CAT   /FØ
...
...
...
READY
```

Remote specifiers may be used with other commands as shown below:

```
SAVE/RØ
SAVE   FILE3/RØ
UNSAVE   TESTPROG/TØ
REPLACE/F1
```

Using the Scientific XPL Compiler - RUN and COMPILE

The previous section explained how line numbered source files are typed in from the terminal. In general, such line numbered files will be Scientific XPL Source Programs where each line of the program will contain an XPL program statement.

A trivial example of an XPL program is below:

```
100 PRINT 'HELLO THERE';
110 PRINT 'GOODBY';
120 EOF
```

Running a Scientific XPL Program

The simplest way to run a Scientific XPL program is by using the RUN command:

```
OLD TRIV
READY
LIST
100 PRINT 'HELLO';
110 PRINT 'GOODBY';
120 EOF
READY
RUN

HELLO
GOODBY

READY
```

When the RUN command is processed, the user's current file is compiled by the Scientific XPL Compiler, and the resulting instructions are loaded into the computer's memory and executed. In order to use the RUN command, the System diskette must be mounted in the left-most disk drive (drive zero) and the User diskette must be mounted in the right-most drive (drive one).

The compilation of a short XPL program takes approximately 7-15 seconds depending on the system configuration. Once a program has been written and will be used in the future without modification, it is possible to store the program in compiled form. By doing so, the compilation will not have to be performed at each RUN command so the response will be essentially instantaneous.

To save a program in compiled form, the COMPILE and SAVE commands are used.

The COMPILE commands operates similarly to RUN, except that the machine instructions generated by the compiler are not loaded into the computer's memory for execution. Instead, with COMPILE, the compiled object code becomes your current file after the compilation. Since compiled files are machine instructions instead of letters, compiled files may not be listed on your terminal.

An example of the COMPILE command is:

```
OLD TRIV
READY
COMPILE
MEMORY MAP:
  ENVIRONMENT STARTS AT:  000000
  OBJECT CODE STARTS AT:  000156
  OBJECT CODE ENDS AT:    000175
  VARIABLE AREA STARTS:   000175
  VARIABLE AREA ENDS AT:  000275
MEMORY REQUIRED FOR PROGRAM EXECUTION:  189 WORDS.
CURRENT FILE IS NOW CALLED TRIV.
READY
```

In response to the COMPILE command, the system translates your program into executable machine instructions. At the end of compilation, a memory map is printed out on the terminal. The memory map allows you to determine how much memory is required to run your program. Note that the numbers in the memory map are printed in octal. The total memory required for execution is printed in decimal, however, for convenience.

If the compilation is successful, as in the above example, the current file name will change - in this case from 'TRIV' to 'TRIV.' - so that you cannot REPLACE the compiled file on top of the original source file. Filenames of less than eight characters in length will have the character '.' (period) added to the end of the file name. File names that are exactly eight characters in length (such as FILENAME) will have the last character of the file name changed to a period (namely 'FILENAM.').

Once a program has been compiled, the compiled version may be saved on the diskette using the normal SAVE command.

Compiled files consist of numbers that represent machine instructions and therefore may not be modified by using line numbers or the edit commands. It is imperative that the original source file for every compiled program is saved on a back-up diskette, so that modifications to the program can be performed at a later time.

Note that the COMPILE command in no way affects the saved version of your current file. After compilation, your current file will be the compiled object code, whereas the saved version will be the original source file that was compiled.

In general, when a program has been developed and no further modifications are anticipated, the program should be compiled, and SAVED in compiled form. To RUN a compiled file, the standard RUN command is used. Since the program will be in compiled form, program execution will begin immediately after the RUN command is typed.

```
OLD TRIV.
READY
LIST
COMPILED PROGRAMS MAY NOT BE EDITED OR LISTED
READY
RUN
.........
.........   (output from program TRIV)
.........
```

A special feature may be used with RUN to direct program output to either the terminal or the remote printer. This is used to create a hard copy of output that would otherwise appear on the CRT screen. The command sequence RUN, PRINTER is used instead of the normal RUN to activate this feature.

```
READY
OLD TESTPROG
READY
RUN
...
...         (output from TESTPROG)
...
READY

RUN, PRINTER        (output will be directed to remote printer)
READY
```

```
READY

CAT

NO FILES ARE SAVED.

READY

NEW TRIV
READY

10 PRINT 'HELLO';
20 PRINT 'GOODBYE';
30 EOF
RUN

HELLO
GOODBYE

READY

SAV
READY

CAT

TRIV

READY

COMPILE
MEMORY MAP:
  ENVIRONMENT STARTS AT:000000
  OBJECT CODE STARTS AT:000303
  OBJECT CODE ENDS   AT:000323
  RAM AREA    STARTS AT:020000
  RAM AREA    ENDS   AT:020100
MEMORY REQUIRED FOR THIS PROGRAM:  08256 WORDS.
COMPILATION SUCCESSFUL - CURRENT FILE IS NOW CALLED "TRIV."

READY

SAVE
READY

CAT

TRIV      TRIV.

READY

RUN
HELLO
GOODBYE

READY

CAT LEN

TRIV      22     TRIV.     211

READY
```

Examples of RUN and COMPILE

Notice that the Editor commands operate only on the current file, and do do not affect the SAVED copy. Alterations to the current file that are made by an Editor command are written out to the diskette only in response to the REPLACE command.

Edit Commands: LIST

The LIST command is used to generate a printout of the contents of a program that is stored on a diskette. The simple command 'LIST' will print the entire contents of the current file on the terminal:

```
READY
OLD TESTPROG
READY
LIST
100 /* TEST PROGRAM #1 */
110 DECLARE I FIXED;
120 INPUT I; PRINT I* I;
130 EOF
READY
```

Optionally, the LIST command may be used to list only a section, or several sections of the file. Each section is identified by the starting and ending line number of that section. For example, the command

```
LIST 100-200
```

would only list line 100 through lines 200 on the terminal. If there were no lines in the specified group, then there is no printout on the terminal.

Several sections may be specified in one LIST command:

```
LIST 100-200,300,400-500
```

The above command would print line 100 through 200, line 300 (if there was a line 300), and lines 400-500.

A special case of the LIST command is the word LIST followed by exactly one line number:

```
LIST 200
```

If LIST is followed by just ONE line number, all the lines from the specified line through to the end of the program are listed. However, IF the line number is followed by a comma, then only the one line is printed:

```
LIST 200,
```

Auxiliary Commands: LENGTH and NAME

The LENGTH command is used to find out the length, in 16-bit words, of the current file. A source file consists of line numbers, plus the individual characters that make up the program statements on each line of the file. Each line number requires one 16-bit word of storage while each 16-bit word can store two letters.

The NAME command is used to determine what the current file name is. It is most convenient in the middle of a long editing session when you can't find the piece of paper that lists the last NEW, RENAME, or OLD command that you typed!

The Editor

The Editor refers to a section of the Monitor program that performs some very useful textual processing based upon commands typed in from the terminal. While files can be altered by retyping an incorrect line, it is sometimes necessary to change line numbers or delete a whole section of a program.

The edit commands perform many useful functions:

1. Listing a section or sections of a complete file (LIST, PRINT)
2. Changing the line-numbers in a program (RESEQUENCE, SEQUENCE, DE-SEQUENCE)
3. Deleting a section or sections of a program. (DELETE)
4. Extracting a section or sections of a program. (EXTRACT)
5. Find the last line in a program (LAST)
6. Moving a group of lines to a different section of the program. (MOVE)
7. Combining programs that are saved on a diskette (JOIN, APPEND)
8. Locating lines that contain a specified string (LOCATE)
9. Changing several occurrences of a character string to a new string (CHANGE).
10. Automatic line number generation (BUILD)
11. Perform video editing (CRT, NCRT).

Editor commands are typed in from the terminal at any time. Each command operates on the current file, or the body of textual material that would be printed on the terminal in response to the LIST command.

The word END may be used instead of a line number with all the editor commands to specify the end of the program:

```
LIST 500-END
...
...
READY
```

The following page contains examples of the LIST command.

```
READY

OLD TESTPROG
READY

LIST
10 LINE 10 IN TESTPROG
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY

LIS 50-70
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
READY

LIS 30,50-60,90
30 LINE 30 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
90 LINE 90 IN TESTPROG
READY

LIST 500
READY

LIS 20,30-40
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
READY

LIS 809
READY

LIS 80
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY

LIS 20
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY
```

Edit Commands: PRINT

The PRINT command operates similarly to LIST except that the output will appear on the remote hard copy printer. The PRINT command is most often used while editing from a Video terminal and a print-out is desired.

```
PRINT
PRINT   200-300
PRINT   300, 500, 600-END
```

Changing Line-Numbers: RESEQUENCE, SEQUENCE, DESEQUENCE

Three Editor commands are available for changing the line-numbers that are associated with each line of a program. This is often done to prepare a program for a "clean" printout with the LIST command. Other times it is required during the entry of a new program and one wishes to insert a new line in between two consecutively numbered lines.

The most often used command is RESEQUENCE. Resequence changes all the line numbers in the program. After the resequence is completed, the first line number will be 100, and each line number will be ten greater than the number on the preceding line. Examples of RESEQUENCE will be found on the following page.

The SEQUENCE command operates similarly to RESEQUENCE in that every line number in the file is changed. After a SEQUENCE command, the first line number will be 1, and each line number will be one greater than the number on the preceding line.

DESEQUENCE removes the line number from each line in a file. It is most often used for generating a printout of material, such as a letter, that does not contain line numbers. After a DESEQUENCE, be sure to SEQUENCE or RESEQUENCE the program before editing or typing new lines.

```
READY

OLD TESTPROG
READY

LIST
10 LINE 10 IN TESTPROG
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY

RESEQUENCE
READY

LIST
100 LINE 10 IN TESTPROG
110 LINE 20 IN TESTPROG
120 LINE 30 IN TESTPROG
130 LINE 40 IN TESTPROG
140 LINE 50 IN TESTPROG
150 LINE 60 IN TESTPROG
160 LINE 70 IN TESTPROG
170 LINE 80 IN TESTPROG
180 LINE 90 IN TESTPROG
READY

SEQUENCE
READY

LIST
1 LINE 10 IN TESTPROG
2 LINE 20 IN TESTPROG
3 LINE 30 IN TESTPROG
4 LINE 40 IN TESTPROG
5 LINE 50 IN TESTPROG
6 LINE 60 IN TESTPROG
7 LINE 70 IN TESTPROG
8 LINE 80 IN TESTPROG
9 LINE 90 IN TESTPROG
READY

DESEQUENCE
READY

LIST
LINE 10 IN TESTPROG
LINE 20 IN TESTPROG
LINE 30 IN TESTPROG
LINE 40 IN TESTPROG
LINE 50 IN TESTPROG
LINE 60 IN TESTPROG
LINE 70 IN TESTPROG
LINE 80 IN TESTPROG
LINE 90 IN TESTPROG
READY
```

Editor Commands DELETE and EXTRACT

The commands DELETE and EXTRACT are used to delete or extract individual lines, or groups of lines, from the current file.

The format for the DELETE command is:

```
DELETE 100,200-300,400
```

DELETE is followed by a list of numbers that indicate the lines of the program that are to be affected. Individual lines may be deleted by specifying the line number, or entire blocks of lines may be deleted by specifying the starting and ending line number separated by a hyphen (200-300 for example).

The EXTRACT command operates similarly to DELETE. The specified lines, or groups of lines, are extracted from the current file. All other lines will be lost.

For both the EXTRACT and DELETE commands, the line numbers must be specified in increasing order. For example, the command

```
EXTRACT 200,100,500-800,400
```

will not perform correctly, and instead should be typed:

```
EXTRACT 100,200,400,500-800.
```

Examples of EXTRACT and DELETE are on the following page.

```
READY

OLD TESTPROG
READY

LIST
10 LINE 10 IN TESTPROG
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY

DELETE 60-80
READY

LIST
10 LINE 10 IN TESTPROG
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
90 LINE 90 IN TESTPROG
READY

DELETE 20,40,90
READY

LIST
10 LINE 10 IN TESTPROG
30 LINE 30 IN TESTPROG
50 LINE 50 IN TESTPROG
READY

DELETE 30
READY

LIST
10 LINE 10 IN TESTPROG
50 LINE 50 IN TESTPROG
READY

DELETE 10-50
READY

LENGTH
 00000.
READY

LIST
CAN NOT PROCESS AN EMPTY FILE!!

STOP!
READY
```

```
READY

OLD TESTPROG
READY

LIS
10 LINE 10 IN TESTPROG
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY

EXTRACT 20-80
READY

LIST
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
READY

EXTRACT 20,40-60,80
READY

LIST
20 LINE 20 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
80 LINE 80 IN TESTPROG
READY

EXTRACT 50,80
READY

LIST
50 LINE 50 IN TESTPROG
80 LINE 80 IN TESTPROG
READY

EXTRACT 100
READY

LENGTH
 00000.
READY

LIST
CAN NOT PROCESS AN EMPTY FILE!!
```



```
STOP!
READY
```

Editor Commands: LAST

A particularly useful Editor command is the LAST command. While processing the LAST command, the system reads through the current file and finds the last line of the program. The line number of the last line is then printed on the terminal.

By using SEQUENCE and LAST, you can find out the number of source lines in a program. Alternatively, after performing a SEQUENCE or RESEQUENCE, by using the LAST command and the LIST command, the final section of the program can be examined.

```
READY

OLD TESTPROG
READY

LIST
10 LINE 10 IN TESTPROG
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY

LAST
LAST LINE NUMBER IS 90
READY

LIST 70
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY
```

Editor Commands: MOVE

The MOVE command is used to move one line, or a block of lines, to a different section of the program. A RESEQUENCE command is automatically performed after the MOVE.

```
MOVE 50-100,200
MOVE 90,180
```

MOVE takes the line specified by the first line number, or group of lines specified by the starting and ending line number separated by a hyphen, and moves it to a position in the file after the specified line of the file. MOVE must be followed by two numbers separated by a comma (MOVE 100,50) in the case of a one line move, or three numbers (MOV 200-300,800) in the case of a movement of a group of lines.

It is possible to move a line or group of lines to any position of the file. For example, lines 200 through 300 may be moved to after line 800 by the command

```
MOVE 200-300,800
```

Alternatively, lines 200 through 300 may be moved to an earlier position in the file (for example, after line 100) by the command

```
MOVE 200-300,100
```

Examples of MOVE will be found on the following page.

```
READY

OLD TESTPROG
READY

LIST
10 LINE 10 IN TESTPROG
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY

MOVE 20-40,80
READY

LIST
100 LINE 10 IN TESTPROG
110 LINE 50 IN TESTPROG
120 LINE 60 IN TESTPROG
130 LINE 70 IN TESTPROG
140 LINE 80 IN TESTPROG
150 LINE 20 IN TESTPROG
160 LINE 30 IN TESTPROG
170 LINE 40 IN TESTPROG
180 LINE 90 IN TESTPROG
READY

MOVE 100,180
READY

LIST
100 LINE 50 IN TESTPROG
110 LINE 60 IN TESTPROG
120 LINE 70 IN TESTPROG
130 LINE 80 IN TESTPROG
140 LINE 20 IN TESTPROG
150 LINE 30 IN TESTPROG
160 LINE 40 IN TESTPROG
170 LINE 90 IN TESTPROG
180 LINE 10 IN TESTPROG
READY

MOVE 140-150,160
READY

LIST
100 LINE 50 IN TESTPROG
110 LINE 60 IN TESTPROG
120 LINE 70 IN TESTPROG
130 LINE 80 IN TESTPROG
140 LINE 40 IN TESTPROG
150 LINE 20 IN TESTPROG
160 LINE 30 IN TESTPROG
170 LINE 90 IN TESTPROG
180 LINE 10 IN TESTPROG
READY
```

Editor Commands: JOIN and APPEND

The Editor commands JOIN and APPEND are used to combine two files that are saved on a diskette. JOIN and APPEND operate similarly, except that a RESEQUENCE is performed automatically after an APPEND.

```
READY
APPEND TESTPROG
READY
```

APPEND combines the specified file to the end of the current file. The resulting file then becomes the current file. A RESEQUENCE is automatically performed after an APPEND.

Examples of APPEND and JOIN will be found on the following page.

Several different files may be combined at the same time by using a comma to separate several filenames:

```
READY
APPEND TESTPROG, FILEA, FILEB
READY
```

```
READY

OLD TESTPROG
READY

LIST
10 LINE 10 IN TESTPROG
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY

JOIN TESTPROG
READY

LIST
10 LINE 10 IN TESTPROG
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
10 LINE 10 IN TESTPROG
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY

OLD TESTPROG
READY

LIST
10 LINE 10 IN TESTPROG
20 LINE 20 IN TESTPROG
30 LINE 30 IN TESTPROG
40 LINE 40 IN TESTPROG
50 LINE 50 IN TESTPROG
60 LINE 60 IN TESTPROG
70 LINE 70 IN TESTPROG
80 LINE 80 IN TESTPROG
90 LINE 90 IN TESTPROG
READY
```

```
APPEND TESTPROG
READY

LIST
100 LINE 10 IN TESTPROG
110 LINE 20 IN TESTPROG
120 LINE 30 IN TESTPROG
130 LINE 40 IN TESTPROG
140 LINE 50 IN TESTPROG
150 LINE 60 IN TESTPROG
160 LINE 70 IN TESTPROG
170 LINE 80 IN TESTPROG
180 LINE 90 IN TESTPROG
190 LINE 10 IN TESTPROG
200 LINE 20 IN TESTPROG
210 LINE 30 IN TESTPROG
220 LINE 40 IN TESTPROG
230 LINE 50 IN TESTPROG
240 LINE 60 IN TESTPROG
250 LINE 70 IN TESTPROG
260 LINE 80 IN TESTPROG
270 LINE 90 IN TESTPROG
READY
```

Editor Commands: LOCATE and CHANGE

Another section of the EDITOR locates, deletes, inserts, and replaces sequences of characters in a file. It is normally used to edit line-numbered files but has limited use with files that have been DESEQUENCED. The two relevant commands are LOCATE and CHANGE.

As you will see in the examples that follow, there are several command formats available for LOCATE and CHANGE. These formats vary in how the line number is specified. The acceptable formats for specifying line numbers are:

1. No line number - If no line number is specified in the command then the first occurrence of the sequence of characters is acted on according to the command.

2. Single line number - If exactly one line number is specified in the LOCATE or CHANGE command, then all occurrences of the sequence of characters in the specified line will be acted upon.

3. A block of lines - If a block of lines is specified (for example, 10-30), then all occurrences of the sequence of characters in that block are acted on according to the command.

4. ALL - if the word ALL is used instead of a line number, then all occurrences of the sequence of characters in the file will be acted upon according to the command.

LOCATE

Examples:

```
LOCATE IF
LOCATE IF,20
LOCATE IF,20-40
LOCATE IF,20-END
LOCATE IF,ALL
LOCATE IF,20-30,60-80,200
```

LOCATE prints the line(s) containing at least one occurrence of the specified sequence of characters. The first two formats listed above will never print more than one line.

Notice, as in the last example above, several groups of lines may be specified.

The following page lists some examples of LOCATE.

```
READY


OLD TESTPROG
READY

LIST
10 DECLARE (NUM,I) FLOATING;      /*DECLARE FLOATING VARIABLES*/
20 DO NUM=0 TO 5 BY 2;            /*PRINT TWO COLUMNS*/
30 PRINT;                         /*PRINT CARRIAGE RETURN/LINE FEED*/
40 DO I=NUM TO NUM+1;             /*FOR EACH COLUMN ACROSS PAGE*/
50 PRINT '    ',I,': ',EXP(I),;   /*PRINT I,EXP(I), WITH SPACES*/
60 END;
70 END;
80 PRINT;                         /*FINAL CARRIAGE RET/LINE FEED*/
90 EOF;
READY

LOCATE DO,ALL
20 DO NUM=0 TO 5 BY 2;            /*PRINT TWO COLUMNS*/
40 DO I=NUM TO NUM+1;             /*FOR EACH COLUMN ACROSS PAGE*/
READY

LOCATE DO
20 DO NUM=0 TO 5 BY 2;            /*PRINT TWO COLUMNS*/
READY

LOCATE DO,40-END
40 DO I=NUM TO NUM+1;             /*FOR EACH COLUMN ACROSS PAGE*/
READY

LOCATE DO,50-END
STRING NOT FOUND

STOP!
READY

LOCATE I,10-50,90
10 DECLARE (NUM,I) FLOATING;      /*DECLARE FLOATING VARIABLES*/
20 DO NUM=0 TO 5 BY 2;            /*PRINT TWO COLUMNS*/
30 PRINT;                         /*PRINT CARRIAGE RETURN/LINE FEED*/
40 DO I=NUM TO NUM+1;             /*FOR EACH COLUMN ACROSS PAGE*/
50 PRINT '    ',I,': ',EXP(I),;   /*PRINT I,EXP(I), WITH SPACES*/
READY

LOCATE M,10-50,90
10 DECLARE (NUM,I) FLOATING;      /*DECLARE FLOATING VARIABLES*/
20 DO NUM=0 TO 5 BY 2;            /*PRINT TWO COLUMNS*/
40 DO I=NUM TO NUM+1;             /*FOR EACH COLUMN ACROSS PAGE*/
READY
```

Examples:

```
CHANGE WILE,WHILE
CHANGE WILE,WHILE,10
CHANGE WILE,WHILE,20-40
CHANGE WILE,WHILE,50-END
CHANGE WILE,WHILE,ALL
CHANGE WILE,WHILE,20-50,200-300,500
```

CHANGE substitutes the second string, or sequence of characters, for the first string in the indicate line(s). In the above examples, the string WILE was replaced by the string WHILE in the specified lines.

Notice, as in the last example above, several groups of lines may be specified in a single CHANGE command.

The following page list some CHANGE examples.

A slash (/) is used to process text strings that contain special punctuation characters such as a comma (,) or a slash (/). If the editor encounters a slash in the character string, the slash is ignored and the next character after the slash is used as part of the character string even if it is a delimiter such as a slash or comma. For example, in order to change the string A/B to A*B, the command:

```
CHANGE A//B,A*B
```

would be used. Similarly, commas may be processed with CHANGE by preceding the comma with a slash. For example, to change the string 'A,B' to 'A-B' the command:

```
CHANGE 'A/,B','A-B'
```

would be used.

```
READY

OLD TESTPROG
READY

LIST
10 DECLARE (NUM,I) FLOATING;       /*DECLARE FLOATING VARIABLES*/
20 DO NUM=0 TO 5 BY 2;             /*PRINT TWO COLUMNS*/
30 PRINT;                          /*PRINT CARRIAGE RETURN/LINE FEED*/
40 DO I=NUM TO NUM+1;              /*FOR EACH COLUMN ACROSS PAGE*/
50 PRINT '    ',I,': ',EXP(I),;    /*PRINT I,EXP(I), WITH SPACES*/
60 END;
70 END;
80 PRINT;                          /*FINAL CARRIAGE RET/LINE FEED*/
90 EOF;
READY

CHANGE DECLARE,THIS.WAS.DECLARE
READY

LIST
10 THIS.WAS.DECLARE (NUM,I) FLOATING;      /*DECLARE FLOATING VARIABLES*/
20 DO NUM=0 TO 5 BY 2;             /*PRINT TWO COLUMNS*/
30 PRINT;                          /*PRINT CARRIAGE RETURN/LINE FEED*/
40 DO I=NUM TO NUM+1;              /*FOR EACH COLUMN ACROSS PAGE*/
50 PRINT '    ',I,': ',EXP(I),;    /*PRINT I,EXP(I), WITH SPACES*/
60 END;
70 END;
80 PRINT;                          /*FINAL CARRIAGE RET/LINE FEED*/
90 EOF;
READY

CHANGE DO,BLEEP,ALL
READY

LIS
10 THIS.WAS.DECLARE (NUM,I) FLOATING;      /*DECLARE FLOATING VARIABLES*/
20 BLEEP NUM=0 TO 5 BY 2;             /*PRINT TWO COLUMNS*/
30 PRINT;                          /*PRINT CARRIAGE RETURN/LINE FEED*/
40 BLEEP I=NUM TO NUM+1;              /*FOR EACH COLUMN ACROSS PAGE*/
50 PRINT '    ',I,': ',EXP(I),;    /*PRINT I,EXP(I), WITH SPACES*/
60 END;
70 END;
80 PRINT;                          /*FINAL CARRIAGE RET/LINE FEED*/
90 EOF;
READY

CHANGE PRINT,THIS.WAS.PRINT,50-80
READY

LIS
10 THIS.WAS.DECLARE (NUM,I) FLOATING;      /*DECLARE FLOATING VARIABLES*/
20 BLEEP NUM=0 TO 5 BY 2;             /*PRINT TWO COLUMNS*/
30 PRINT;                          /*PRINT CARRIAGE RETURN/LINE FEED*/
40 BLEEP I=NUM TO NUM+1;              /*FOR EACH COLUMN ACROSS PAGE*/
50 THIS.WAS.PRINT '    ',I,': ',EXP(I),;    /*THIS.WAS.PRINT I,EXP(I), WITH
60 END;
70 END;
80 THIS.WAS.PRINT;                          /*FINAL CARRIAGE RET/LINE FEED*/
90 EOF;
READY
```

CRT Editing

The Scientific XPL Operating System includes a special CRT command that is very useful when editing a program from a video terminal. To enter the CRT mode, just type the CRT command:

```
READY

CRT
READY
```

The CRT command is normally typed once at the beginning of a programming session. The system will stay in CRT mode until the NCRT command is typed or the power is turned off.

When the computer is in CRT mode, the LIST and LOCATE commands operate in a special manner. In response to LIST, the computer will clear the CRT screen, fill the screen with one page of output, and then halt. At this point

if a line-feed is typed, one more line will be added to the screen;

if an S is typed, the computer will stop the LIST and print READY , or

if any other character is typed, the next page of output will appear on the screen.

When using LOCATE while in CRT mode, the screen is not cleared at the start of LOCATE, however, the computer will stop after one page of output if enough lines are printed. A line-feed, S, or any other character may then be typed to continue or abort the LOCATE.

Edit Commands: BUILD

The BUILD command is used to enter textual information into the computer in an auto-line numbered mode. To enter BUILD mode, just type BUILD:

```
READY

BUILD
ENTERING BUILD MODE NOW

100
```

After typing BUILD the computer enters a special mode where the line numbers are automatically assigned and need not be typed in by the user. A line number of 100 is used when BUILDing a new file; whereas a line number that is 10 greater than the last line number in a file is used when adding to a program.

When you are through with BUILD, type a single carriage return. The computer will respond with READY and the file may now be edited, using the normal line-number techniques.

APPENDIX I

Notes on the Handling of Diskettes

Notes on the Handling of Diskettes

The flexible diskettes provided by New England Digital Corporation are of the highest quality available. To achieve the maximum possible data reliability and diskette integrity, it is imperative that the diskettes be handled carefully, stored in a protected dust-free environment, and inserted into the proper disk drive at the proper time.

The following rules should always be observed when handling flexible diskettes:

1. A protective cover is provided with each diskette. Always store the diskette inside this protective cover when it is not inserted into the diskette drive. Never place an unprotected diskette on a table top, counter, terminal, or other place or dust particles picked up by the diskette surface might render the diskette useless.

2. Never bend or otherwise physically abuse a flexible diskette. When handling a diskette, hold it by the edges gently to avoid damaging the recording surface.

3. When inserting a diskette into a disk drive, be careful to insert the diskette straight into the slot and never at an angle. The diskette should slide almost effortlessly into the drive - any other condition indicates improper insertion.

4. The diskettes are temperature sensitive and should always be kept at a temperature between 50° F and 125° F. Best results will be obtained when the diskette is always kept at room temperature.

5. Never touch the exposed magnetic surface visible in the slot on one side of the diskette. Dust or moisture on this surface may destroy the diskette.

6. Since the flexible diskette is a magnetic storage media, never expose the diskette to stray magnetic fields. Common devices such as telephones, microphones, dictation machines, and numerous other instruments contain small internal magnets. If a diskette is unreadable after exposure to a magnetic field, it must be erased and reformatted before it can be used.

7. Each diskette has a white paper label indicating the type of diskette (usually "minidiskette no." and "SA104" appear on this label). While it is possible to write on this label for diskette identification, always use a felt tip pen when labeling a diskette. The high pressures associated with a ballpoint pen may reach through the diskette case and damage the recording surface underneath.

If the above rules are followed, each diskette should give you 100% data reliability. A diskette that is physically damaged by dirt, liquid, or physical abuse should be replaced immediately before such contamination is

transferred to the internal mechanism of the diskette drive. A diskette that has been magnetically erased by exposure to stray fields, may be reformatted and returned to service.

Making Back-up Copies

To guard against the loss of important data or program material when a diskette is damaged by accidental misuse, conscientious computer users develop the habit of regularly making backup copies of such material. The procedure for making a backup copy of a program is exceedingly simple, as follows:

1. Turn on your system and load the Scientific XPL Operating System in the normal fashion. Insert the User diskette that contains the program you wish to duplicate in the right-most drive at this time.

2. Type the OLD command to call up the desired program from the User Diskette:

```
OLD PROG2
READY
```

3. At this point, you may remove the User Diskette from the right-most drive (drive one). Be sure to not remove the System Diskette from the left-most drive (drive zero). (Be exceedingly careful here - inserting the wrong diskette into the upper drive may destroy the programs stored on that diskette!)

4. Now insert a second User Diskette (one that you reserve for backup copies) into the right-most drive (drive one).

5. At this point, you may type the SAVE or REPLACE command to write a copy of the program out to your backup diskette:

```
SAVE
READY
```

APPENDIX II

Using the Terminal

Using the Terminal

A Brief Description of the Terminal

Most terminals look like typewriters. They have a modified typewriter keyboard and print on paper. Other terminals have a keyboard and a TV screen; lines are "printed" on the screen instead of on paper. This type of terminal is often called a CRT (for cathode ray tube) or scope.

When you sit down at a terminal that is unfamiliar to you, take some time to get to know it. First, you must find the ON/OFF switch. It may be on the front, top, side, back; under the terminal, or even on the wall. Also find the RETURN key, the BREAK key (which may be marked BREAK or BRK or ATTN or INT), and the CONTROL key (which may be marked CONTROL, CTRL or CTL). You will be using each of these keys often.

Typing on the Terminal

Typing accuracy is important in all interactions with the computer. You are essentially carrying on a conversation with the computer; when you type something, a special supervisory program interprets it and responds in some way. Since the computer understands only certain commands and cannot guess what you mean to say, you must type with care. Certain features of typing on the computer terminals which may differ from the customary use of a typewriter are discussed below.

Unlike some typewriters, the terminal has separate keys for zero and the letter oh and for one and the letter el. In general, the computer notices spaces, so type spaces only where they are needed. If you follow the spacing in our examples, you'll have no trouble. Also, the computer does not process what you type until you press RETURN, so remember to do so at the end of each line you type.

Since almost everyone makes typing mistakes, the system is designed to allow such mistakes to be corrected easily. To delete the preceding character, hold down the CONTROL key and press the Z key; to delete more than one preceding character, repeat this process once for each character to be deleted. To delete the entire current line of type, simultaneously press the CONTROL and X keys (the terminal will print DELETED). Control characters are not printed on the terminal and, when CONTROL Z is used, do not cause the current position on the line to change. In any case, do not try to backspace and overstrike to fix errors. The computer will simply accept each backspace as a new character and will not erase the old ones.

Some terminals type only in upper case; others operate like any upper-case and lowercase typewriter. For any system commands that you type, the

computer essentially makes no distinction between cases; if you type in lower case, it will translate to upper case. When you begin creating your own files, information will be stored just as you type it.

Interrupting the Computer

Whenever you want to interrupt the computer during a program run or listing, you may do so by pressing the NULL, ATTN, INTERRUPT, or BREAK key; use whichever key your terminal has. While the terminal is typing you can also stop the typing and the program by pressing the S key.

INDEX