PRO SOFT

Box 839, North Hollywood, California 91603
(213) 764-3131

September 15, 1981

Dear Customer,

Thank you very much for your order. Enclosed is documentation and a diskette for **NEWSCRIPT** (two diskettes for the Model I). These will make it easy for you to produce high-quality letters, documents, and books, and to take full advantage of your printer. We think you will find you've selected one of the finest Word Processing systems available today, regardless of price or computer used.

The manual contains a self-study tutorial intended for use by people with a variety of computer backgrounds, specifically including "none". As the preface says, you don't need to read the entire book right now, just the first two sections. Once you learn how easy it is to create and revise text, and how professional your printed results look, you'll begin to use NEWSCRIPT for most of your writing.

NEWSCRIPT has been sent to you on a ready-to-run "system" disk, including a version of the excellent DOSPLUS operating system. The first thing to do with the disk(s) is to make backup copies as described in the manual, and then work from the backups.

Along with this letter, you may find additional documentation sheets describing enhancements that were made to NEWSCRIPT after the manual was printed. Please retain those sheets for future use. We've also enclosed a summary of our current software product line. I'd especially like to call your attention to **RPM**, a disk drive diagnostic utility; and to **FASTER**, which can help speed up your BASIC programs. In the words of the staff at PERSONAL COMPUTING magazine (May, 1981, page 116)... "FASTER is effective and easy to use.!" We couldn't have said it better, and after you've cut down the running times and delays of some of your other programs, we're sure you'll agree.

Now, of course, you'll want to try some of NEWSCRIPT's formatting capabilities as soon as possible. To get you started, we've included a sample letter called "EDIT1/EX" right on the diskette. The tutorial shows you how to print that letter, and how to produce letters of your own.

Enjoy your new software, and again... thank you for choosing PROSOFT.

Sincerely,

Debbie Tesler
Customer Relations
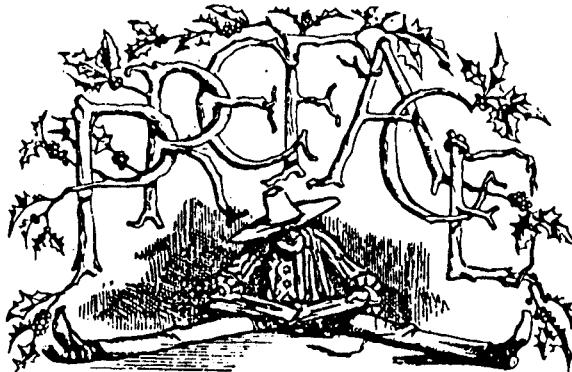
HOW TO USE THIS DOCUMENT

This isn't the kind of book you read from cover to cover.

Of course, you've probably decided that already, based on its sheer size and weight! If you wanted to read a book, you'd have bought a good novel.

However, I'm sure you want to learn enough to do some constructive Word Processing, and yet spend as little time reading as possible. That's fine, because you're actually holding two books in your lap, and only the shorter one needs to be read now. Most of these pages are reference material, to be used later on when you want to try something new or fancy.

The first thing to do is make a backup of the NEWSCRIPT distribution diskette, as explained in Section I. Then, using the Backup (never the original), read through the first two sections of the manual and practice the examples. The first section will show you how to operate NEWSCRIPT, including a hands-on tutorial for a simple letter. The second section presents "EZEDIT" and "EZSCRIPT", and is a quick, short course in the most commonly needed features of NEWSCRIPT.

The rest of the book can wait till later.

Since the fastest way to learn something is by _doing_ it, I hope you'll be sitting in front of your computer and trying things out as you go along. When something doesn't make sense, try to find an example. There are hundreds of small examples, and several medium-sized ones, in this book. The heavy-duty reference sections (III and IV) are in alphabetical order, there's a Table of Contents, a fair to middlin' Index, and even a "How To..." section. So, when you get stuck, you can look up what you need, and then keep going.

Whether you just want to write an occasional letter, or the great American novel, you've come to the right place, and you're in for a real treat.

Shall we begin?

This is a minor revision of the NEWSCRIPT 6.0 documentation and applies to PROSOFT's NEWSCRIPT release 6.1 and above.

Revision Date: September 1, 1981

THE TERMS BELOW ARE TRADEMARKS OF THE IDENTIFIED CORPORATIONS:

"Centronics" is a Registered Trademark of Centronics data computer corporation.

"EPSON" is a Registered Trademark of EPSON America.

"NEWDOS" is a Registered Trademark of Apparat, Inc.

"LDOS" is a Registered Trademark of Logical Systems, Inc.

"VTOS" is a Registered Trademark of Virtual Technologies.

"DOSPLUS" is a Registered Trademark of Micro-Systems Software, Inc.

"TRS-80" is a Registered Trademark of Tandy Corp.

"IBM" is a Registered Trademark of International Business Machines Corporation.

"PROSOFT" and "NEWSCRIPT" are Trademarks of VM-CMS Consulting Services, Inc.

TABLE OF CONTENTS

TERMS OF LICENSE AGREEMENT

ALL PROSOFT PROGRAMS AND DOCUMENTATION ARE LICENSED ON AN "AS-IS" BASIS WITHOUT WARRANTY EXPRESSED OR IMPLIED.

PROSOFT shall have no responsibility or liability for any loss or damage caused or alleged to be caused directly or indirectly by use of our computer software and/or documentation. This expressly includes, but is not limited to, loss or invalidation of customer data, programs, files, or business opportunities.

Purchase of this software conveys to the customer title to the media (diskette), but not title to the programs.

If any level of government shall enact laws taxing the possession or use of computer programs, the possessor(s) of PROSOFT computer programs shall be responsible for payment of any taxes assessed on their own copies.

All PROSOFT programs and documentation are copyrighted materials, and all rights of reproduction are reserved.

* * *

These statements are intended to protect PROSOFT from events that may be beyond our control. We hope you will find the rest of this Agreement somewhat more friendly, and indicative of how we want to do business with you.

REPORTING PROBLEMS AND OBTAINING UPDATES

If you receive a defective diskette or experience software errors during the first 30 days following purchase of NEWSCRIPT, please return the diskette to us with an explanation of the problem. We will replace the diskette at no charge.

Although all PROSOFT computer programs are sold "as-is", we welcome all properly-documented reports of programming or documentation errors, and suggestions for enhancements. While we cannot represent that all problems will be corrected, we will attempt to correct most of them.

Please describe your problem(s) as clearly as possible. Make sure you give enough information to allow us to re-create the error(s). Identify your computer, printer, Operating System, and any other programs you are using with NEWSCRIPT. Also specify the "Update Level" and Registration Numbers as they are shown on the Primay Options Menu. If possible, please send us an output listing, the original PROSOFT diskette, and a diskette containing a file that will create the problem if we follow the exact instruction you provide. All requests for updates or corrections to programs must be accompanied by an original PROSOFT diskette and a $10.00 update fee. All diskettes will be returned to you, hopefully with corrections and relevant documentation updates.

CORRECTIONS WILL BE DISTRIBUTED ONLY TO REGISTERED LICENSEES.

To register, fill in the information on the enclosed Registration Card, record the registration number for future reference, and send the card to us within 30 days of purchase. This is our only means of notifying you of corrections and updates.

Once you have registered your purchase, you may obtain the latest copy of your release of NEWSCRIPT at any time by sending us your original PROSOFT diskette. There is a $10.00 fee for this service, and it includes program and documentation updates. Note that "updates" are not the same as "upgrades", which are described next.

UPGRADE PRIVILEGES

Upgrading to enhanced or follow-on versions of PROSOFT computer programs is a privilege that we will extend only to legitimate, registered licensees who have played fair with us (by not giving away or selling bootleg copies of our products). If you learn of a major new version of a PROSOFT product you have licensed, and wish to exercise this upgrade privilege:

1. Send us the original PROSOFT diskette or tape (the one with our label on it).

2. Identify yourself, and when/where you bought the product.

3. Include payment as follows: the current retail price of the new version, minus the price you paid, plus $10.00. (Example: you paid $100 plus tax, and the price of the enhanced version is $125. You could upgrade for $35.) Please add tax in California, and shipping overseas.

4. You MUST previously have registered your purchase with us!

PROSOFT will honor upgrade or correction requests only if you have registered an original copy of the software within 30 days of purchase; we will not provide service in any form if you've given, sold, or received unauthorized copies of our products. We don't want to have to refuse your request, so please resist the temptation to pass our products around: if your friends want copies, our prices are reasonable and we offer ongoing support. To continue doing so, we have to make enough of a profit to stay in business.

* * *

Please copy the Registration Number from the Registration Card: _____

When you start to run NEWSCRIPT, an "Update Level" date will be displayed. Please make a note of it in case you need assistance or wish to report a problem:

UPDATE LEVEL DATE: _____

THANK YOU

for choosing

PROSOFT

SECTION I - INTRODUCTION AND OPERATION

INTRODUCTION

"NEWSCRIPT" is a comprehensive Word Processing system for the TRS-80 Models I and III. It consists of several programs that work together to provide text manipulation capabilities in an easy-to-use manner. Most of your time with NEWSCRIPT will be spent using the "Editor". This is a program that lets you type in your material, make corrections, save the results to diskette, and subsequently revise saved documents until they are satisfactory. The other major part of NEWSCRIPT is the "SCRIPT" program that formats and prints the documents you created with "EDIT".

The primary components of NEWSCRIPT are:

       NS/CMD   - major machine language components of NEWSCRIPT
       EDIT   - used to enter and revise text, and save it to disk
       SCRIPT   - formats and prints text from disk
       INDEX   - alphabetizes and formats an Index

Several additional utility programs and examples are also provided, and their uses will be explained as we go along.

This book explains how you can use the editor (EDIT) and text formatter (SCRIPT) to create, update, and print letters, memos, documents, and form letters. This first section will teach you how to use EDIT and SCRIPT on your TRS-80 Model I or III. Several simple examples are included for this purpose, and they will assume you're at the computer, "learning by doing."

Section II covers the essential commands of EDIT and SCRIPT. These are known as the "EZ" group, and hopefully, you will find them easy to learn and to use. The essentials of the full-screen editor are covered in this section. These first two sections are all you'll need to begin using NEWSCRIPT, so you have less reading ahead than it may seem.

Section III fully describes the editor. It explains the keyboard features, the use of the screen, and gives complete descriptions of all commands in alphabetic order. Each of these includes at least one example. As you become comfortable with EZEDIT, but find a need for such advanced features as "block moves", you'll want to read the beginning of Section III and also look up selected commands as you need them.

Section IV contains the alphabetized "control word" formats for SCRIPT (control words, also called "markups", act as commands to the text formatter). Again, each explanation includes at least one example. As your comfort level with EZSCRIPT improves, this section will show you how to do advanced, sophisticated formats and Form Letters.

Section V describes how you can use SCRIPT to create an Index to a manuscript written in NEWSCRIPT.

Printer's logo.
History of Signboards

Section VI contains "How To..." information for several common situations, such as Form Letters, Titles, and errors. The topics of this section are in the Table of Contents. When you need information on a particular method of doing something, this is a good place to look.

Section VII contains a Command and Control Word summary. This is similar to the Summary Card we've also provided.

SECTION VIII identifies the EDIT and SCRIPT commands that are implemented differently here than in Word Processors that are similar to NEWSCRIPT. These include SUBEDIT/SUBSCRIPT, and IBM's EDGAR/DCF. This section is of interest only to people with a SUBSCRIPT or EDGAR background.

SECTION IX covers special cases of tailoring NEWSCRIPT to your needs. Instructions for using NEWSCRIPT begin later in Section I, so you'll need Section IX only if you have a serial printer on the RS-232 port, or if you don't want to use the supplied Operating System (a version of DOSPLUS).

SECTION X presents a summary of the commands of the "TDOS" operating system. This is a version of DOSPLUS, and its commands are pretty much the same as the ones used in TRSDOS. Only the functions needed for NEWSCRIPT Word Processing are included, however: TDOS isn't a general-purpose Operating System.

Section XI explains the use of "FITLINE", a file re-formatting utility. If any of your text files become too large, FITLINE can be used to split them into smaller, chained pieces.

Section XII describes the optional "mailing labels" feature of NEWSCRIPT. This is an extra-cost program and was supplied only if you ordered it.

<*>

As you read this document, please note that it was written entirely in EDIT, formatted by SCRIPT, and printed on a Centronics 737. The only exceptions to this are the front cover and the art work. This means you should be able to create similar layouts yourself, within the restrictions of your printer and the support SCRIPT provides for that printer. If you have a Daisy Wheel printer, you won't be able to print double—width letters, for example. There's a table coming up that shows you what NEWSCRIPT can do on various printers.
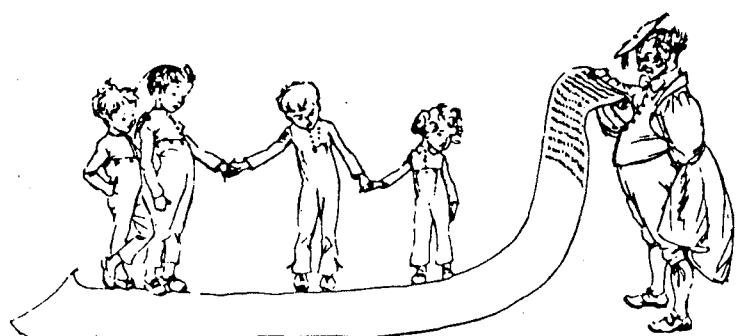
Hardware and Software Requirements

EDIT and SCRIPT run on the TRS-80 Models I and III, and require 48K of memory. EDIT can process 200-300 lines of text at a time (the equivalent of about 4-6 single-spaced printed pages), and can "chain" to unlimited additional text files as needed. SCRIPT can process an unlimited number of chained files, treating them as one document. The files can be on as many diskettes as are needed to contain your documents.

At least one disk drive is required, but at least two are recommended to facilitate the management of multiple text files.

A printer is required if hardcopy output is desired (why else would you be doing Word Processing?). Serial printers require special interface software, and further information about this is contained in Section IX.

If you have only one disk drive, it is not possible to create a Table of Contents or an Index for a document that spans multiple disks, since these features create files that must remain open and on-line throughout the formatting of the document.

A lower-case modification to the TRS-80 is not required, but text can be printed in lower-case only if your computer has the lower-case feature. Therefore, installation of this hardware capability is strongly recommended. Edit and Script both recognize mixtures of upper- and lower-case commands, but the documents you are trying to produce will look a lot better in mixed-case than in upper-case only. The lower-case hardware from Radio Shack, and the modification for "Pencil" both work with NEWSCRIPT. If your lower-case feature is switch-controlled, make sure the switch is turned "on" BEFORE initializing Word processing.

If you connect a speaker to the tape cassette "EAR" plug, then certain error conditions will cause an audible "BEEP" to occur.

All programs in this system have been tested and should work with these operating systems:

MODEL I: TRSDOS 2.3, NEWDOS, NEWDOS/80, DOSPLUS, LDOS

MODEL III: TRSDOS 1.1 and up; DOSPLUS

We intend to provide working versions of these programs for officially enhanced versions of DOSPLUS, TRSDOS, NEWDOS/80, and LDOS. However, we may not be able to support NEWSCRIPT when it is used with modified versions of these Operating Systems or with machine-language routines other than "NS/CMD" and "NSPRT", both of which have been supplied to you as parts of NEWSCRIPT.

NEWSCRIPT can display the directory of "native" diskettes for NEWDOS/80, DOSPLUS, LDOS, and Model III TRSDOS. It cannot display a directory for Model I TRSDOS or NEWDOS.

For diskette capacity purposes, a Model I diskette can hold about 30-40 pages of single-spaced text, while a Model III diskette can hold about 50-70 pages (if there is nothing else on the diskette).

Lead pointer. *K & E*

Features Supported on Various Printers

NEWSCRIPT is only as smart as your printer. Most of the modern dot-matrix printers can print double-width characters (like this), but Daisy Wheel printers cannot. The table below shows which special features of NEWSCRIPT work on which printers. Features not shown should work on all printers. NEWSCRIPT supports many other printers besides the ones listed (such as Radio Shack's other "Line Printer" series), but we've only tested special feature support for the printers listed below.

NOTE: This list is current as of August, 1981. As time goes by, additional printers and features may be added to the list.

PRINTER	UNDER LINE	OVER STRIKE	DOUBLE WIDTH	SUB/SUPER SCRIPTS	RIGHT JUSTIFIED PROPORTIONAL	TRS-80 GRAPHICS
ANADEX 9500/9501	Y	X	Y	X	X	N
ATARI 825	Y	Y	Y	Y	Y	X
CENTRONICS 737/739	Y	Y	Y	Y	Y	X/N
DIABLO 1620	Y	Y	X	Y	N	N
EPSON MX-80/FT/G	Y	Y	Y	N	X	Y
MICROLINE 80/82/83	X	X	Y	X	X	Y
NEC SPINWRITER	Y	Y	X	Y	N	N
RS LINE PRINTER IV	Y	Y	Y	Y	Y	X
RS DAISY WHEEL II	Y	Y	X	Y	Y	X
SELECTRIC	L	N	X	X	X	X

KEY: Y=supported   L=limited support
      N=not supported   X=printer can't do it

If your printer is not on this list, it probably will work with NEWSCRIPT, but not with all of the special features described above. In cases where it would be possible, but very hard, to simulate a feature on a printer that doesn't have the capability, we've indicated 'N', even though 'X' is probably more accurate. Super-scripts on the MX-80 fall in this category.
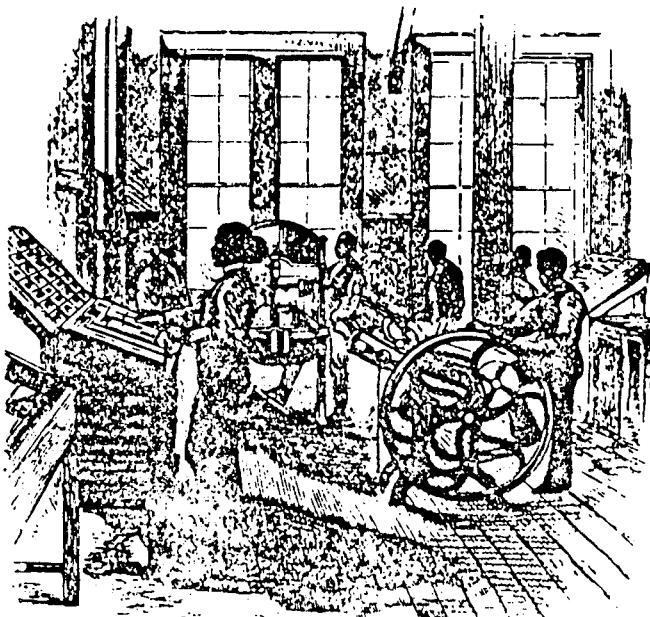
Printers. *Harper's*

What Is An Editor? What Is A Text Processor?

As used in this document, and in the system which it describes, an Editor (EDIT) is a program that will let you create, revise, and update the contents of a file. A Text Processor (SCRIPT), on the other hand, will format the contents of an existing file (created by the Editor) according to your specifications, and then print the result. If you use an Editor and Text Processor on a computer, then you have a Word Processing system.

EDIT and SCRIPT operate as two separate programs. While it would be more desirable (in many cases) to have a single program that combines all functions, the limited memory of micro-computers does not encourage such a marriage. By having two programs, it was possible to build more functions into the overall system.

Control of the computer passes back and forth between EDIT and SCRIPT, so the net effect is similar to having a single, multi-function program, but the transitions take a little longer.

Full Screen Editing

A "Full Screen Editor" is one that lets you move the cursor to any position on the video screen, and then insert, delete, or replace text at that point. EDIT's machine-language component, NS/CMD, gives you that capability. However, not all full screen editors are alike: this one, being based on mainframe editors, separates the screen into three separate areas:

* a Command line (at the very top of the screen)

* a data area (most of the rest of the screen)

* a LIne Manipulation Area ('LIMA', down the left side of the screen)

The "Command Line" lets you control the overall editing session: you can search for data, make "global changes", see and change status, save your text, etc.

The "data area" is where you compose and update your document. The data area operates in either "overlay" mode or "insert" mode. When in "overlay" mode (the default), you can type right over anything that is on the screen. When in "insert" mode, the text to the right of the cursor is pushed along in front of the cursor, and is not replaced by what you type. In either case, when words run off the end of the screen, a blank line is created just below the line on which you're typing, and those words move down onto that new line.

If you're entering text in the data area, you can type until you fill memory, if necessary. Whenever you hit <ENTER>, EDIT will process all the changes on the screen and then may generate another blank line on which you can continue typing. For the most part, you should press <ENTER> only when you want EDIT to process a Command or LIMA entry.

NEWSCRIPT has both repeating keys and typeahead, so regardless of how fast you type, you won't lose keystrokes: not at the end of a line, and not while EDIT is "thinking". The typeahead buffer can hold 128 characters, which should give you lots of leeway.

The full screen editor is designed to work naturally, and requires minimal training. Its extra features can be learned at your convenience, but you don't ever have to get beyond the basics to get satisfactory results.

The LIMA commands let you insert, delete, duplicate, and mark lines for various purposes. You don't have to use them, but if you take the time to learn how these special commands work, it'll add even more to your personal productivity. To move the cursor into the LIMA, move it to the line you need to manipulate, hold down <SHIFT>, and press the left-arrow. If the cursor wasn't at the left edge of the data area, just press SHIFT-left-arrow a second time. This method is intended to ensure that you won't get into the LIMA accidentally.

The Editing Window

Since most of your documents will be considerably larger than the size of the TRS-80's screen, you'll be able to see only a portion of a document at any one time. With EDIT, you can select the lines you want to see, and the portions of those lines. In effect, you have control over a window. The window is 15 lines high and 60 characters wide (text lines may be up to 255 characters in length, but you can only see 60 at a time). You can move the viewing window up, down, left, or right. Quite a few commands are provided to help you do this. In fact, EDIT has over forty distinct commands, although you're not likely to need more than a dozen of them in most cases.

Control Words

I'm sure you had no trouble understanding what was meant by "commands" in the discussion of the Editor. After all, you've used commands (and on occasion, expletives) when using the TRS-80 for many other purposes. However, the term "control words" may be new to you.

A "control word" is something that tells the Text Processor (SCRIPT) how it should format the document you are creating. ALL control words begin with a period (.), and the first control word on a line must begin at the extreme left-most position of a data line. Usually, you can place multiple control words on a line, but there are a few exceptions to this.

In short, control words are formatting instructions for use by SCRIPT, and they are placed where needed in the document that is to be formatted.

This would be a good time for some examples (there will be many places where examples are appropriate, and I'll try to anticipate them for you). Suppose you want to start a new page some place in your document. SCRIPT's control word is:

.PA

Century Magazine, 1894

Control words can be in upper or lower case. If you wanted to get rid of that nicely right-justified margin you could say:

.ju off

Since "right-ragged" looks hand-typed, while "right-justified" leaves the reader suspicious as to whether a human being was ever involved in getting your deathless prose on paper, it is often desirable to achieve a manual-looking effect. SCRIPT will let you do it, while still making sure that every line has as much text as it can handle without exceeding the maximum line width you've requested. (This paragraph was done with ".ju off".)

The use of control words makes it much easier for you to find and read the formatting specifications in a document. After all, they aren't internal or graphics values, but normal, readable data. You can edit and change them whenever you want to, and you can change formatting setups as often as necessary within a single document. If someone else created a document and you must update it, you'll find this built-in control word readability very helpful. By the way, one of the control words is ".CM", which stands for "comment". This lets you annotate your documents internally, without those comments being printed.

If you don't supply any control words at all to SCRIPT, it will take defaults. If you want to override these defaults, or create fancy effects such as headings and footings, then you must supply the appropriate control words.

At this point, you may (and should) be asking how you can tell SCRIPT what you want done. You do it through the Editor.

Control words start with a period. When was the last time you ran across printed material in which the first character on a line was a period? Pretty long ago, right? Since this isn't used much, it can be reserved for communication from you to SCRIPT. Just place a control word (starting with a period) at the beginning of a line in the file you are editing. Regular text (data) can go on other lines. While all control words look like data to the Editor, they look like commands to SCRIPT. Instead of printing them, SCRIPT will modify its format setups appropriately, and you will (hopefully) get the visual effects you wanted. In most cases, you can save space by placing several control words on a single line, separated by semicolons:

.ad 10;.ll 60;.pp

(Those mean: "adjust left margin to 1.0 inches; set line length to 6.0 inches; start a new paragraph.)

Machine Language Routines

NEWSCRIPT is a combination of Z-80 Machine Language and TRS-80 BASIC. There are over 2,000 Machine Language instructions hidden away in high memory when NEWSCRIPT is running. For operational convenience, all this code has been combined into a single command executed from DOS. The command is "NS". Once in control, it invokes BASIC and brings up a "menu" program that lets you select what you want to do. The menu is a BASIC program, and it's named "NSINIT" (NewScript INITialization, of course). The major programs, called "EDIT" and "SCRIPT", are in BASIC. They give control of the computer to various parts of the "NS" Machine Language program based on analysis of the commands you issue. To use NEWSCRIPT, "NS" must always be in control.
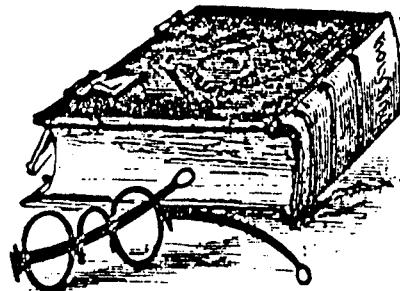
QUICK OVERVIEW OF SOME EDIT COMMANDS

Section II describes the most commonly used EDIT commands and SCRIPT control words, and should be read as you learn NEWSCRIPT. However, those common features will be previewed here just to give you a glimpse of what's ahead.

When NEWSCRIPT starts to run, it gives you a selection menu. All documents must be created and updated using "EDIT", and printed using "SCRIPT". So, to create or change something, choose "1" (EDIT) on the menu. When asked for the I.D. of the file to be edited, give the entire filename and extension. (This is called the "fileid".) If the file is on more than one disk, or if it is on a write-protected disk, make sure to also give the drive number just to be extra safe. For example:

<p align="center">THANKYOU/LET:1</p>

After confirming that you really do want to edit the specified file (by pressing <ENTER>), wait for the Edit screen to be displayed. Then, you can move the cursor in any direction with the four arrow keys until it is where you want to type or change something. At first, all your typing will be in the "data area" or on the primary command line next to the "E=>" at the top left corner of the screen. Later, you'll start to use the "LIMA" along the left edge.

Insert and Delete

"Insert" refers to the addition of text in the middle of an existing document. To insert new text on blank lines, just move the cursor to where you want to begin, and start typing. If there are no blank lines where you need them, move the cursor to the line just above that spot and press <ENTER>. A blank line will appear, and you can start typing on it.

If you want to insert some characters between some others, move the cursor to the correct position, then hold down the <CLEAR> key and press the letter "I". The cursor will become larger (like a capital "I"), and anything you type will push everything past it a bit further to the right, or onto a new blank line that will appear just below. To turn off this "insert" mode, just hold down <CLEAR> once more and press "I" again. "I" acts as a toggle switch, and <CLEAR> is NEWSCRIPT's "control key".

To delete a character, move the cursor to it, hold down <CLEAR>, and press "D". Each time you do, another character will disappear. If you hold both keys down, the delete function will repeat very slowly, or you can hit the "D" over and over again to achieve a faster repeat.

To insert several lines below an existing one, scroll that line to the top of the screen, then enter the "I" command. To delete several consecutive lines, move the first of them to the top of the screen, then enter the "DEL n" command ('n' is the number of lines to delete).

Audible Errors

If you connect a speaker or "TBEEP" (Trademark of WEB International) to the tape "AUX" plug, you will hear a "beep" when EDIT detects certain errors.

EZEDIT Command Summary

Some elementary "EZEDIT" commands are summarized below. Remember that there are about forty other commands, but you don't need them when you're first starting. After entering a command, you must press the <ENTER> key to tell EDIT to process the command. Until you do, you can change the command, wipe it out, or make other changes anyplace on the screen. Also note the "WHOOPS" command, below.

Moving up and down (scrolling) through a document.

    FORWARD moves one screen (15) lines ahead.
    BACK moves one screen back.
    DOWN moves one line ahead.
    UP moves one line back.
    Holding an up or down arrow will also cause scrolling.
    TOP takes you to the beginning of the file.
    BOTTOM takes you to the end of the file.

Searching and changing text.

    LOCATE /any text/   searches for "any text".
    CHANGE /here/there/   changes "here" to "there"
    Or, the cursor may be moved and the text just typed in.

Writing text to disk.

    SAVE writes the latest copy from memory to disk.
    END   writes the copy to disk and takes you to SCRIPT.
    QUIT discards what's in memory and takes you to SCRIPT.

    EDIT and SCRIPT work only from what you've saved onto disk. If you never issued "SAVE" or "END", you haven't made a permanent copy of your text, and it will be lost. If you had an old copy out on disk, that old copy will still be there. If you're finished editing, you can use either "SAVE" or "END": you don't need to use both. The difference is that "SAVE" writes the permanent copy and leaves you in the editor, while "END" writes the permanent copy and takes you out of the editor and into SCRIPT.

Cancelling errors.

WHOOPS refreshes the screen to how it was before <ENTER> was pressed. It doesn't
        cancel all changes to date.

QUIT cancels all errors by not saving anything.

REMOVING AN INK BLOT

H.H. COLLINS IMPROVED
EUREKA
INK ERADICATOR
27 UNION SQUARE
N.Y.
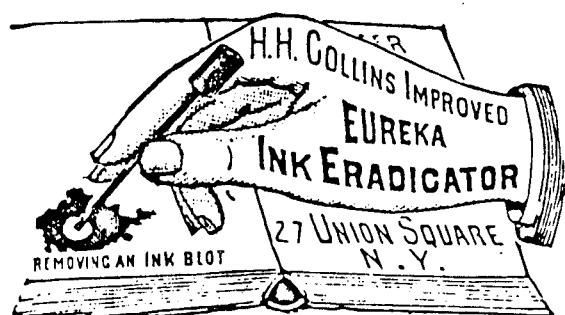
SCRIPT Format Assumptions

NEWSCRIPT gives you full control over the format, page layout, and typefaces in your printed results. However, it isn't necessary to specify all of this information unless you want to change something! NEWSCRIPT makes assumptions (takes defaults) about everything, unless you say otherwise. For example, 8-1/2 by 11 inch paper is assumed, but you can tell SCRIPT that you're using something else.

The formats you wish to use are specified through the use of control words. Some of SCRIPT's defaults are summarized below, according to category. If you want to change them, you can do so right in each document. You should avoid changing them within the program itself, since future updates from PROSOFT will just set things back to the way they began. Section IV lists all SCRIPT control words in alphabetical order, and fully describes them.

Default Page Layout and Margins

Paper size: 8-1/2" wide by 11" high.

Margins: one inch on each of the four sides. A diagram of this is shown on the very last page of Section IV.

> The Left Margin is controlled partly by how you insert the paper, and partly by the ".AD" control word.

> The Right Margin is controlled by the ".LL" control word (the Right Margin is whatever is left over after the line is printed).

> The Top Margin is controlled by the ".TM" control word, and of course the Bottom Margin is controlled by the ".BM" control word.

The pitch (or font) defaults to proportional on the Centronics 737, 739, Line Printer IV, and Daisy Wheel II, and to 10 characters per inch (10 CPI) on all other printers.

Automatic page numbering is turned on. The first page is not numbered unless you've set up a title (".TT" or ".BT"), but all subsequent pages are numbered in the upper right-hand corner unless you turn page numbering off through use of the ".PN" control word.

Right-margin justification and shifting of text from one line to another to achieve this justification occur unless you tell SCRIPT not to do so with the ".FO OFF" (Format Off) control word.

The length of each printed line defaults to 6.5 inches, specified as ".LL 65" (no decimal point in the 65).

Output will go to a line printer, without overstriking, unless you use some of the "options" SCRIPT offers you when you begin to run it. You can also cause overstriking of individual lines through use of the ".DA" (dark) control word. This is particularly useful with the EPSON printers, which offer several levels of overstriking.

Summary of Common and Default SCRIPT Values

.AD 5 – Adjustment for left margin (0.5 inches, plus positioning effect of the paper, which usually adds another 0.5 inches for a total of 1 inch).

.LL 65 – Line length (6.5 inches). Taken with the .AD value and the paper position, this leaves a Right Margin of 1 inch. There's no way to explicitly specify the Right Margin.

.TM 6 – Top Margin of 6 blank lines (1 inch at 6 LPI).

.BM 6 – Bottom Margin of 6 blank lines.

.FO ON – Formatting is on, so the right margin will be justified (that is, smooth, not ragged).

.BF 10 – 10 character-per-inch font is used on all printers except as noted above for proportional support. Most of this manual is printed in proportional.

.PN ON – Page Numbering is ON.

.SK 1 – skip down one line, unless larger value used

.CE 1 – center the next one line

.PP   – start a new paragraph

.PA   – start a new page
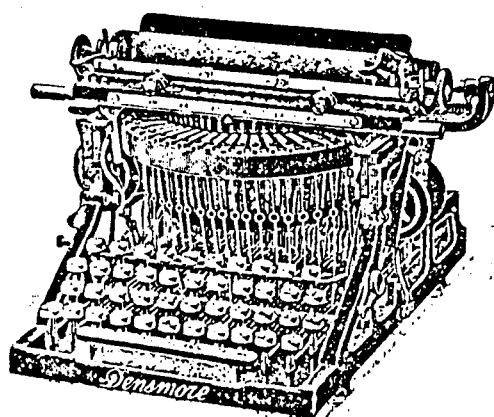
Typewriter. *Cosmo, Vol. 18*

INSTALLING NEWSCRIPT

NEWSCRIPT has been sent to you on a ready-to-run diskette, complete with a special version of DOSPLUS, called "TDOS".

(Two diskettes are provided for Model I systems, since the optional components of NEWSCRIPT exceed the capacity of a single-density 35-track diskette. A single Model III diskette contains both the required and the optional components. The first Model I diskette, "1 of 2", is all you need for most purposes, and is the one that is referred to below in the installation procedures. Naturally, you should also make a backup of the second diskette, and save both originals. The backup of the second diskette can be used if you ever need any of the optional components.)

The very first thing you should do is to make at least one backup of this diskette, and then to store the original away in case you need updates from us in the future (we only issue updates when original diskettes are returned).

To make a backup, turn on the computer, wait until the disk drives stop running, insert the NEWSCRIPT diskette in drive 0, and press the <RESET> key. You will see a message saying "DOSPLUS", which means that the operating system is ready to accept commands. Needless to say, the first command you should issue is:

<p align="center">BACKUP</p>

You will be asked these questions:

> SOURCE DRIVE NUMBER ?
> DESTINATION DRIVE NUMBER ?
> BACKUP DATE (MM/DD/YY) ?

The source drive number is '0' (no quotes). If you have at least two drives, the destination drive should be '1', and you should insert a blank diskette in drive 1 at this time. If you have only one drive, reply '0' and be prepared to follow the prompts carefully as you are asked to switch diskettes back and forth. The date, of course, should be today's date.

If anything goes wrong during backup, try again with another blank diskette. If your problems appear to be with the original NEWSCRIPT diskette, please wrap it carefully in cardboard and return it to us with a note of explanation. Inability to interchange diskettes between computers happens from time to time, and we'll just replace it (them) for you at no charge. It isn't necessary to heavily seal envelope, as it just makes it harder for us to get at the disk. Please do not use staples!

Assuming that the BACKUP succeeded, put an external label on the new disk, place it in drive 0, and press <RESET>. When you get the "DOSPLUS" message, you may wish to set the "AUTO" command to cause automatic startup of NEWSCRIPT from now on. To do this, just enter this command:

<p align="center">AUTO NS</p>

Next, format a "data diskette" to hold the documents you will be writing. If you have only one disk drive, you should make another backup of the NEWSCRIPT diskette, and then KILL all the files displayed by the "DIR" command. Disks like this will be needed to hold all your work, while the first backup will hold the programs.

If you have more than one disk drive, place a blank disk in drive 1, then type:

FORMAT :1

You'll be asked a series of questions, and if you have trouble answering any of them, turn to Section X and lookup the "FORMAT" command. Please note that on the Model I, the diskettes are in TRSDOS-compatible format, but on the Model III, they are not readable by TRSDOS, and TDOS cannot read a TRSDOS diskette except to CONVERT it.

Once you have at least one "data" diskette, you're ready to customize NEWSCRIPT and complete the installation. To do this, type "NS" (no quotes) to activate NEWSCRIPT this first time. This command must be issued from DOS, not from BASIC. A series of copyright notices will flash by, and you'll finally be presented with the

<*> PRINTER SELECTION MENU <*>

This happens automatically only the first time you use NEWSCRIPT. Identify your printer according to the instructions on the screen (choose '99' if your printer isn't on the screen).

Next, you'll be asked whether your printer is connected to the parallel or the serial (RS-232) port. Most printers use parallel, so if you aren't sure, select '1'.

Next, you'll be asked to identify the Operating System you are using. In this case, of course, it's "TDOS", so just press '1'. Later on, if you decide to move NEWSCRIPT to something else, you can change this easily.

After all the menus are answered, your answers will be saved on disk under the name, "NSINIT". This should be on the drive 0 diskette, which up to this point must not be write-protected. After the 'save' completes successfully, you should place a write-protect tab on the diskette as a precaution. All your manuscripts, letters, etc., should be stored on other disks, not on the Word Processing System diskette.

After the "save" completes, you'll be shown the "PRIMARY OPTIONS MENU". This is what you'll see each time you run NEWSCRIPT from now on:

<*> PRIMARY OPTIONS MENU <*>

PLEASE CHOOSE BY NUMBER:
        (1) EDIT
        (2) SCRIPT
        (9) CUSTOMIZE/INSTALL
        (0) EXIT TO BASIC

You've just completed option '9', of course, and NEWSCRIPT is now installed. Pretty easy, isn't it?

OPERATING INSTRUCTIONS: Getting Started

By now, you must be pretty anxious to get into the actual use of NEWSCRIPT. So, let's go through a sample session in which the Editor is used to create a letter that is subsequently printed by Script; and upon finding a need to revise that letter, Edit and then Script are used again.

This portion of the manual is intended for hands-on self-study, and you should go through it while sitting at the computer, using the customized diskette you just made, and a blank data diskette in drive 1.

1. If necessary, power up or re-boot, ending up in DOS. If you're already at the "PRIMARY OPTIONS MENU", you shouldn't do this, and you shouldn't issue the "NS" command again. However, if you're in "DOS" rather than "BASIC", begin at the beginning of this step by pressing <RESET>. The "NS" command is the only way to start NEWSCRIPT, and may be issued only once between <RESET>'s. Once in NEWSCRIPT, it isn't necessary to reset again when changing from one function to another.

2. Type: NS   (if not using DOS's "AUTO" startup)

   This will activate the Machine Language components of NEWSCRIPT, turn control over to BASIC, and run the Menu to let you select EDIT or SCRIPT. To create or revise a document, you'll want to use EDIT, so just press "1" when presented with this menu, and wait until the editor is loaded and started.

3. The Editor will ask you a series of questions. The first of these is:

ENTER I.D. OF FILE TO BE EDITED

If you are creating a new file, rather than updating an existing one, choose a name (extension and drive number are optional...just be sure to enter a valid DOS file I.D.).

If you can't remember the name of a file you need to edit, you can display the contents of the diskette Directory: just reply with a question mark ("?" ... no quotes), optionally followed by the disk drive number. If NEWSCRIPT supports reading of Directories under your Operating System, all User files on the specified drive will be displayed, and the "I.D." question will be asked again. For example:

ENTER I.D. OF FILE TO BE EDITED ? ?1

The underlined reply will display the directory if possible.

NEWSCRIPT can't display the directory if you're using Model I TRSDOS or Model I NEWDOS. It can be displayed if you're using NEWDOS/80, DOSPLUS, LDOS, Model III TRSDOS, or, of course, the "TDOS" you are probably using now. It can't be displayed from a TRSDOS data disk when you're running DOSPLUS or LDOS, but only from diskettes formatted by DOSPLUS or LDOS, respectively. Attempts to read the directory from unreadable diskettes can result in loss of what's in memory, so please be careful.

4. EDIT will scan all on-line disks for this file. If it doesn't find it (and it shouldn't, if it's a new file), this message will be displayed:

> *** fileid *** WOULD BE A NEW FILE. PRESS <ENTER> TO
> CONTINUE, OR ENTER ANOTHER FILE I.D.

Of course, if you've decided to work with our "EDIT1/EX" (which is on the second diskette if you have a Model I), you would be told that:

> *** edit1/ex *** ALREADY EXISTS. PRESS <ENTER> TO
> CONTINUE, OR ENTER ANOTHER FILE I.D.

Because reading data from a diskette on the TRS-80 is relatively slow when compared with doing it on a mainframe computer (a large IBM computer that uses CMS can read 1,000 lines from disk in about half a second), EDIT gives you an extra chance to change your mind before beginning a potentially time-consuming process. (EDIT's I/O is fairly fast in TRS-80 terms, but it just can't match a million dollar system.)

If the name you chose is acceptable, just hit <ENTER>. If not, either enter another name, or use the "?" to check the directories again.

5. The space available for text entry will be displayed:

> nnn LINES AVAILABLE

(This value will vary depending on the copy of NEWSCRIPT you've received, and the operating system you are using.) When you edit an existing file, the value will be smaller than when you are starting with a new file, since some of memory has already been used by the existing text.

The "number of lines available" is an estimate based on an assumption of an average of 35 characters per line. Since control word lines tend to be much shorter than regular lines, this estimate is, hopefully, not too far off. EDIT has its own space counter, and will let you know when there is no more room.

6. A suggestion is in order at this point: text files should be kept small (150-250 lines). The Editor needs more time to read and write a large file than a small one; small files can be made bigger without running into insufficient memory problems, and small files are usually easier to revise. SCRIPT is not subject to file size restrictions: by appending files together, it is possible to process unlimited pages of text, spanning many diskettes.

7. There will be a momentary delay while EDIT completes its initialization, and then the screen will go into EDIT mode. While in EDIT mode, the top line of the screen is always the command line, and the rest of the screen contains the text and the LIMA. Since you're working with a new file at this point, the data area will mostly be blank, except for the top of file and end of file markers, so the screen will look something like this:

```
| E=>                                                                    |   <- Command Line
| ___  xTOFx..10!....!..20!....!..30!....!..40!....!..50!....!..60! |
| ___                                                                    |
| ___                                                                    |
| ___                                                                    |
| ___                                                                    |
| ___                                                                    |
| ___                                                                    |
| ___                                                                    |   <- Data Area
LIMA -> | ___                                                             |
| ___                                                                    |
| ___                                                                    |
| ___                                                                    |
| ___                                                                    |
| ___                                                                    |
| ___                                                                    |
| ___  xEOFx..10!....!..20!....!..30!....!..40!....!..50!....!..60! |
|_____|
```
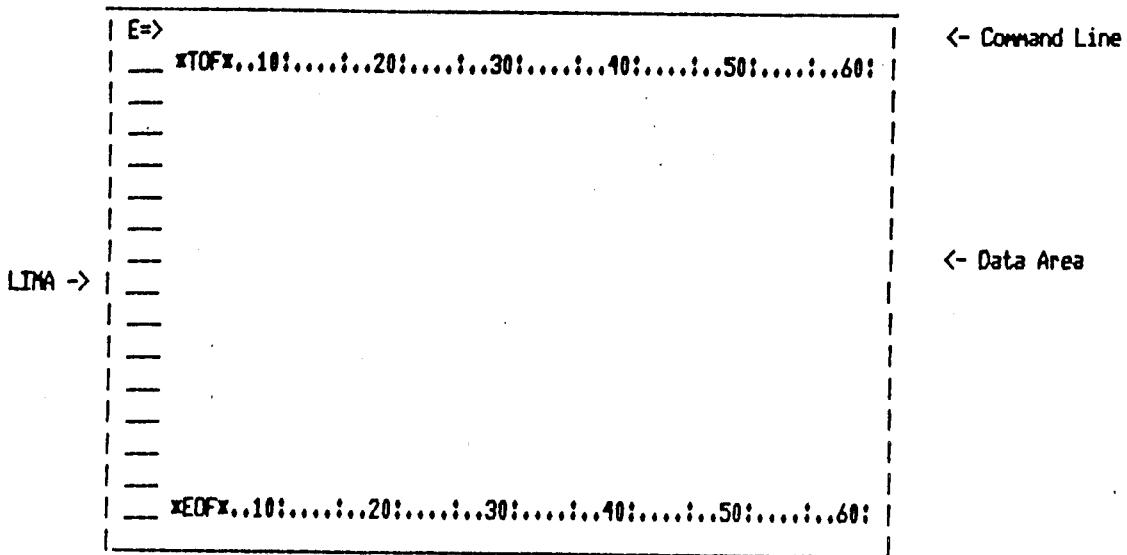
8. Although there are several "status" commands you could issue at this point, what
   you probably want to do is enter the contents of your letter. A sample letter
   (EDIT1/EX) is provided on the distribution disk, and may be edited if you wish.
   That letter is also printed a few pages later on in this manual, first in its "raw",
   unformatted form, and then in its final output form. An explanation of each of the
   control words used in the sample is also given.

9. In order to enter this letter, or any other text, you should use the arrow keys to
   move the cursor into the data area (in the diagram above, that begins just after the
   TOF grid), and begin typing. Whenever you reach the end of a line, EDIT will move
   the cursor to the next line on the screen, taking with it the word you're currently
   typing. When you reach the bottom of the screen, EDIT will begin to move the
   screen up, a line at a time, so that you'll always have a place to type. (This is
   called scrolling.)

   If you need to enter a SCRIPT control word, you'll have to get to the beginning of
   the next line on the screen, which must contain no regular text (it would be a blank
   line). One of the ways to do this is to just hit <ENTER>. Remember that normal
   text NEVER goes on the same line as a control word. Normal text should neither
   precede nor follow control words on a single line, although text associated with
   control words (such as titles, underscored material, etc.) will be placed on the
   same line as the control word that... well, controls it. (I guess that's why they're
   called control words.)

10. If you make a mistake, you can move the cursor, via the arrows, to the error, and
    overstrike a correction. If you need to delete one or more characters, hold down
    the "CLEAR" key (that's EDIT's Control Key, for those of you who care about
    terminology) and while holding it down, press "D". The character under the
    blinking cursor will go away, and the rest of the line will shift to the left.

    If you need to insert one or more characters, hold down the CLEAR key and press
    "I". The cursor will change from its square-oh shape (oh as in" overlay") to a
    tall-I-shape ("I" as in "insert", of course), and everything you type thereafter
    will push whatever is to the right of the cursor further to the right. If necessary,
    EDIT will shift text to a new blank line directly under the current one, always

breaking the text at a blank. Eventually, EDIT will automatically switch back to overlay mode, but you can switch back yourself by pressing CLEAR-I again ("I" acts as a toggle switch, and the shape of the cursor tells you which mode you're in at any moment).

If you just remember that "CLEAR" is the control key, CLEAR-D deletes a character at a time, and CLEAR-I toggles insert mode, you'll find the editor works pretty much as you would like an electric typewriter to work. You won't have to think about it, and hopefully will find it usually works naturally and predictably.

(There are several other keys that work with CLEAR, but we don't need to learn them yet.)

11. After entering your letter you'll undoubtedly want to store it on disk and then run it through SCRIPT. To store the file, you can use either of two commands: SAVE, or END. The difference is that "SAVE" lets you keep editing after the file has been written to disk, whereas "END" terminates the edit session after saving the file. For the purpose of this e... you should use "END".

Since "END" is a command, it must be entered on the command line (that's the very top one, to the right of the "E=>"). Commands don't have to be left-justified on the command line ... EDIT will shift them over and find them on its own.

There are two ways to get to the command line. One way, of course, is to press the up-arrow until the cursor gets to the top. Another, faster way, is to hold down "SHIFT" and press the up-arrow (a neat feature if I do say so myself). After typing the necessary command (which can be edited just like data, except that it can't flow over to a second line), you must press <ENTER> to signal EDIT that something special has happened.

12. EDIT displays blank lines, but doesn't normally save them to disk, since it's so easy to create extra, unwanted blank lines while editing. The normal way to specify blank lines in your documents is through the ".SK" or ".SP" control words.

If you want to save totally blank lines, you can use the EDIT command: "DBLANKS OFF", as explained in Section III.

13. To review your typing before ENDing the editor, you can move the cursor to the top or bottom of the screen, via the up or down-arrow, and press that arrow again. The screen will "scroll" one line in the desired direction until the *TOF* or *EOF* is reached. You can make any changes necessary by using CLEAR-I or CLEAR-D. There are many, many other ways to make changes to text, but we don't need to learn them yet.

14. Before going on to the use of SCRIPT, let's look at the sample letter (EDIT1/EX) to see how it might look when typed into EDIT, and how it will look after SCRIPT processes it to the printer.
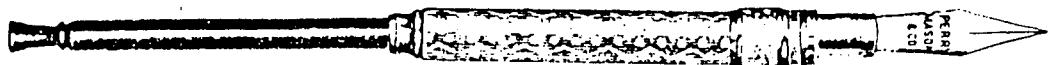
Sample <u>Letter</u> (as entered into the Editor)

```
.ce on
PROSOFT
Box 839
No. Hollywood, Ca. 91603
.ce off
.sp 3
.in 40
.fo off
August 15, 1981
.in 0
.sp 2
Mrs. Joanne Riley
55 South Hope Street
Los Angeles, Ca 90001
.sp
Dear Mrs. Riley:
.fo on
.pp
```
Pursuant to our telephone conversation of the 10th, please
send me 250 5-1/4" diskettes at a unit price of $1.65 each.
Please use UPS and the above address as the destination.
```
.sp 2
.fo off
.in 40
```
Sincerely,
```
.sp 4
```
Chuck Tesler
```
.in 0;.fo on;.sp 10
.us P.S.
```
Sorry about that, folks, just kidding. There isn't a Mrs.
Riley at that address (as far as I know), and I'm paying
about $2.65 a diskette, just like you are. The numbers
were just an attention getter.

Here is the letter as NEWSCRIPT printed it:

<div align="center">

PROSOFT
Box 839
No. Hollywood, Ca. 91603

</div>

August 15, 1981

Mrs. Joanne Riley
55 South Hope Street
Los Angeles, Ca 90001

Dear Mrs. Riley:

Pursuant to our telephone conversation of the 10th, please send me 250 5-1/4"
diskettes at a unit price of $1.65 each. Please use UPS and the above address as the
destination.

Sincerely,

Chuck Tesler

P.S. Sorry about that, folks, just kidding. There isn't a Mrs. Riley at that address (as
far as I know), and I'm paying about $2.65 a diskette, just like you are. The numbers
were just an attention getter.
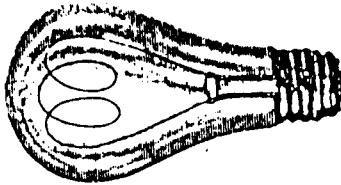
<u>Explanation</u> <u>of</u> <u>Sample</u> <u>Letter</u>

Two pages back is the original input to Script, as entered into the Editor. As you can see, "control words" (The mnemonics that begin with a period) are inter-mixed with the text of the letter. Both were merely data to the Editor, which makes it very easy ("self-documenting") to see how a given format was specified.

Now, the way something looked after you typed it into EDIT is usually <u>NOT</u> the way it gets printed by SCRIPT. The control words tell SCRIPT how to perform the formatting and printing. As a result, lines of text on the EDIT screen are connected together or split apart as needed to place as many words as possible on each printed line and still produce a smooth right-margin. Of course, there are times when you don't want this automatic formatting to occur, and that's where you'll need to use control words. So, let's review the sample letter to see what these control words do.

<u>.ce on</u> turns on Centering. Until turned off, all subsequent lines of text will be centered. If you only want to center one line, the "on" may be omitted. The text to be centered begins on the next line.

<u>.ce off</u> turns Centering off. Once you turn something on, NEWSCRIPT generally leaves it that way until you tell it to stop.

<u>.sp 3</u> causes three blank lines to be inserted in the letter. If you want just ONE blank line, ".SP" with no number after it is sufficient. If the blank lines might fall at the bottom of the page and you don't want the excess blank lines continued on the next page, use ".SK instead of ".SP". ".SP" is unconditional, whereas ".SK" terminates at the end of a page or when the requested number of blank lines have been skipped.

<u>.in 40</u> causes subsequent text lines to be "indented" 4.0 inches to the right of the normal left margin. This indentation was needed because the next few lines will be the sender's address.

<u>.fo off</u> turns "Formatting" off. By default, SCRIPT formats all text. That is, it right-justifies what it prints by taking text from several consecutive input lines, finding out how much will fit (in terms of full words), and then generating blanks or micro-spaces (dot-spaces) to produce the smooth right-margin. The left-over text is used to form the beginning of the next printed line, and so forth.

When formatting is to be suppressed, it can be turned off, as shown here. Then, the input text (date, name, and address in this case) will be printed "as-is".

<u>.in 0</u> moves the indentation back to the left margin ("indent zero").

(Now, we'll start skipping over control words whose meanings have already been described...)

<u>.fo on</u> turns Formatting back on. This is needed in order to let the body of the letter be right-justified, and is the normal setting to use.

<u>.pp</u> defines a new paragraph. By default, this causes one blank line to be skipped, and a half-inch indent to occur on the first line of the new paragraph. Also, there must be at least two lines left for printing on the current page. If there's only one (or none), a new page will be started. This avoids "widows" (single lines at the bottom of a page).

.in 0;.fo on;.sp 10 are all familiar to you by now. The only new thing is that you can place more than one control word on a line. When you do so, you must precede each control word (except the first one) by a semi-colon. Not all control words can be followed by other control words. The exceptions are identified in the detailed explanations of Section IV, but as a rule of thumb, control words that take data as operands (titles, file chains, and some others) must be the last things on a line. This is necessary because you may want to use any string of data in those operands, including things that look like control words. You should also note that if multiple control words are placed on a line, the semi-colon-period sequence, with no intervening blanks, is absolutely required.

As you can see, most of the paragraphs on this page began with an underlined control word. That was accomplished by placing the following kind of data on a line, beginning in column 1 (note the blank just before the second period):

.us .in 40

Since there was no semi-colon preceding the second period, Script understood that I wanted ".in 40" to be data. If you are wondering how the examples in this book manage to show control words, the technique is to precede them by a blank. So, the ".us .in 40" shown above was started in column 2, not column 1.

Please REMEMBER: any line having a period in column one is a control word line and may contain only control words plus their operands (no normal text). Any line not having a period in column one is considered a line of text, and will be printed according to the existing format specifications.

Formatting the Sample Letter with SCRIPT

When you finish with the Editor and type "END", your text file will be written out to disk, and then EDIT will ask you:

PASS 'fileid' TO SCRIPT (Y/N/C/A, DEFAULT=Y)

There are six possible responses to this question:

1. Press <ENTER> if you want the file to be formatted and printed by SCRIPT. This is the normal response;

2. Reply "Y" (no quotes) and press <ENTER>. This is the same as just pressing <ENTER>;

3. Reply "N" (no quotes) and press <ENTER>. This will re-run EDIT, allowing you to edit another file. If you want to do some more editing on the current file, use "C".

4. Reply "C" to continue editing the current file. The last editing window will be re-displayed and you can continue as though you had issued a "SAVE" or decided not to QUIT.

5. Reply "A" to re-run EDIT for the purpose of editing the next file in your ".APpend" chain. If you take this option, the very last line of the current file must contain, as its only control word, ".AP", followed by the file I.D. to be chained. Blank lines after the ".AP" will cause this option to be

ignored even though those lines would cause no harm during SCRIPT'ing.

6. Hit <BREAK> if you are done with EDIT, and don't want to run SCRIPT at this time.

If you just hit <ENTER>, EDIT will store the current file I.D. in a reserved high-memory table and then automatically run "SCRIPT", so the next message you will receive will be from SCRIPT. That message will be:

ENTER I.D. OF FILE TO BE SCRIPTED (DEFAULT = fileid)

(If you had begun with SCRIPT instead of EDIT, so that no file I.D. had previously been stored by either EDIT or SCRIPT, then there would be no "DEFAULT = fileid" in that message, and you would have to enter one in reply to this question.)

"fileid" should be the same as the name you were using a few minutes ago with EDIT; if it is, just press <ENTER>, and SCRIPT will use the assumed fileid. If you don't like that fileid, and have another file to be SCRIPTed, type in its full name and then press <ENTER>. If you need to look at a diskette directory, you can type a question mark, possibly followed by the drive number. As with EDIT, displaying the Directory works with some Operating Systems, but not all: the same restrictions apply here.

SCRIPT will check to make sure that "fileid" exists. If it does (and in this case, it should), then it will ask you to specify any run-time options:

ENTER OPTIONS (ST, V, ED, PG, DS, CC, ID, DA, TC)

The options will be explained shortly, but for now, let us assume you want all the defaults, including printer output. So, just press <ENTER>.

Next (and just about last), SCRIPT will ask you to set up the printer, turn it on, and place it in "READY" state. If you use the "V" (video) option, this question will not be asked, and output will go only to the screen. Since I'm sure you want to see this Word Processing system in action, just press <ENTER>. If the printer is NOT ready at this point, SCRIPT will tell you about it:

* * * PRINTER IS NOT READY * * *

and then it will repeat the original prompt:

HIT <ENTER> WHEN PRINTER AND PAPER ARE READY

If you're using a serial printer, you won't get this message, and SCRIPT will assume you've set things up. If you are using an MX-80 and it is powered off, this message will NOT appear, because that printer presents "READY" status to the computer when off.
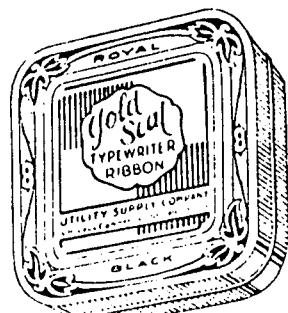
Positioning the Paper on Most Printers

When using continuous forms paper, be aware that the front and back are somewhat different, and that the front should face the ribbon. The back of the paper will have more roughness at the holes and perforations than the front, and you can detect this by sliding your fingers lightly along both sides. If you put the paper in backwards, there is increased likelihood of snagging the ribbon on the perforations between the pages.

Move the paper so that printing would occur on the very first physical line (the very top). SCRIPT will move the paper down from this point, leaving a top margin. When feeding cut sheets (via the run-time STOP option of SCRIPT), all pages after this first one should be advanced to the first physical line to be printed (SCRIPT won't leave a top margin, but will assume you've done so).

Positioning the Paper on the MX-80

Raise the top cover, pull back the paper bail (the metal bar with the column guide on it), and find the platen. (The platen runs horizontally behind the paper, exactly at the level of the print head. It is steel colored.) Notice a seam in the metal just above the platen, running the width of the printer. The top edge of the paper should be aligned at this seam. The printer must be powered off if it is necessary to move the paper downwards (backwards) through the printer.

Positioning the Paper in the 737/739/Line Printer IV

SCRIPT assumes you have placed the paper in the printer so that printing can start at the first possible line, at the very top of the page. Of course, you would rarely want to begin printing at this point, but it allows SCRIPT to calculate the line at which you do want to start printing.

To set the paper up correctly:

1. Turn the printer on, place it in "Local" mode, and move the paper up or down until the top edge is at the top of the print-head assembly. This is the horizontal blak bar above the ribbon, not the print-head itself.
2. Use the "PAPER REV" switch to move the paper DOWN three (3) lines.
3. Place the printer "ONLINE".

When SCRIPT finds the printer is ready, it will move the print-head about half way across the page, then re-position the print-head at the left-margin, on the same line as it found it. This is done in an attempt to ensure that the ribbon is not hooked over the top of the print-head itself, an event that can occur when the inter-page perforation is rough enough to carry the ribbon up with it during paper advance operations.
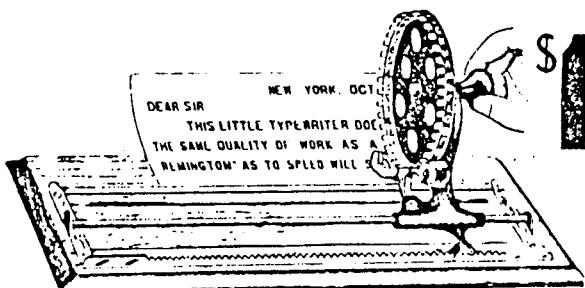
Dollar typewriter. *Cosmo, Vol. 18*

Formatting and Printing

When the paper is positioned correctly and the printer has been placed in "READY" or "ONLINE" mode, just press <ENTER>. From this point on, SCRIPT will read your file, display each line on the screen, format the text, and print it. If it encounters any errors (mostly illegal control word requests), it will display error messages. When it finishes printing, if there were any errors, it will say:

PRESS <ENTER> FOR SUMMARY OF ERRORS

Obviously, you will want to press <ENTER>. A table of error messages and the number of times those errors occurred will be displayed. The individual error lines are NOT displayed again, however. Whether or not the error table was displayed, the next messages will be:

> TOTAL NUMBER OF ERRORS = n
>
> EJECT PAGE (Y/N, DEFAULT=N)
>
> PRESS <ENTER> TO PRINT AGAIN, OR <CLEAR> TO EDIT OR IF DONE
>
> PASS 'fileid' TO EDIT (Y/N, DEFAULT = Y)

Since you probably do want the letter ejected so that the next printout can start on a new page, you should reply "Y" (or "y" ... SCRIPT and EDIT understand upper and lower case commands) to the "eject" question.

If you want to do something else with the SCRIPT processor (print another copy...although there is an option that lets you do that automatically; or format a different file), reply to the "CONTINUE" question with <ENTER>. In the case of this tutorial, we are done for now with SCRIPT, so the proper action is to press the <CLEAR> key. Doing so will cause the "PASS THIS FILE..." question to appear.

As you will see below, by replying to the "PASS THIS FILE" question with just a simple <ENTER>, you can chain along without having to type a fileid in when you get back to EDIT.

This completes the initial tutorial. What follows is a second, shorter session with EDIT, for the purpose of correcting the fantasy-error about diskettes for $1.65.

Revising the Letter

In the Sample Letter, there is a glaring price error: those disks cost $2.65, not $1.65. It also turns out that the addressee spells her name "Raugh", not "Riley". Since we're using a Word Processor, it should be a simple matter to fix those up and print another copy.

To revise the letter, we want to go back to EDIT. If you are still at the end of the "SCRIPT" session, press <CLEAR> if you haven't done so when prompted. If you did press <CLEAR>, you should press <ENTER> next to automatically return to EDIT. If you did anything else, it's safest to start all over again by hitting <RESET>, typing "NS" if you haven't set the "AUTO" command, and selecting option "1" (EDIT). You don't have to press <RESET> and re-issue the "NS" command to go back and forth between EDIT and SCRIPT, and normally shouldn't do so. Just follow the prompts and remain within NEWSCRIPT at all times.

If you have chained back to EDIT, the editor will say:

ENTER I.D. OF FILE TO BE EDITED (DEFAULT = fileid)

(If no default is shown, enter the name of the file you have been using.) Press <ENTER>, and you will be told:

fileid ALREADY EXISTS. PRESS <ENTER> TO CONTINUE OR ENTER ANOTHER FILE I.D.

If the name is O.K., just press <ENTER>. Note that EDIT tries to make things easy for you, but still gives you the chance to override its assumptions. If you aren't sure of the name, you can check the diskette directories through use of the "?" reply, within the restrictions explained earlier.

The file will be read into memory. As this happens, you will see this message on the top line of the screen:

READING FILE fileid, RECORD #   nnn

("nnn" is a counter that shows you progress is being made.)

You will be placed in EDIT mode, with the first 15 lines of the file displayed. Now, it is time to fix those two errors. The EDIT commands I've chosen to use in this tutorial are not the only way these changes could be done, and perhaps not even the best way. I'm trying to show you some of the flexibility of the Editor, so we will make these changes in two different ways.

As you learned while creating the letter in the first place, you could just move the viewing window up or down with the arrows until the text to be changed appears on the screen, then move the cursor to the text and overlay, delete and/or insert the corrections. Doing it that way won't teach you anything new, so let's make the changes by using some EDIT commands.

The "Current Line"

The "current line" is the second physical line on the screen, that is, the one just below the "E=>" command line.

Some EDIT commands move the viewing window up or down, and when the window moves, whatever is shown at the top of the screen becomes the new current line. While you can change text that isn't even on the screen, you'll probably want to see what's going on, and therefore will want to move the viewing window until it displays the text to be changed.

The reason this is so important is that all changes to text start at the current line. Changes never affect the parts of the text file that precede the current line. All searches (the "Locate" command, for instance) begin at the line following the current line, and continue down to the end of the file.
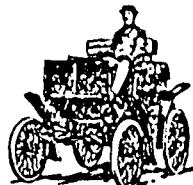
Moving Forward and Backwards Through a Text File

One way to scan through a file is to use the commands "F" (Forward) or "B" (Backward). "F" would be the first one to use, and should be placed on the command line, after the "E=>" prompt. Then, hit the <ENTER> key. When you do this, the next 15 lines of text will be displayed. You can repeat the "F" <ENTER> as often as necessary, until you find what you need. If you've gone too far, you can use "B" instead, to move the viewing window back towards the top of the document. These work quite nicely, but if you know what you want to find, you can tell EDIT to do the search for you.

Another way to scan a file is by moving the cursor to the bottom of the screen and then pressing the down arrow to see one more line at at time. If you want to go back up through the file in this manner, move the cursor to the top of the screen (the command line) and press the up arrow. Yet another way is through the "up" and "down" commands, which are described in Section II under "EZEDIT".

LOCATE and CHANGE

While first creating the letter, we learned how to move the window up and down with the arrows, how to overstrike (overlay) existing text at the cursor, how to delete characters, and how to insert them. All those facilities are still available to you, and you may choose to keep using them for most purposes. However, since there are so many other similar capabilities in the editor, let's start learning how to use some of them now. (I assure you, these things will come in handy later on.)

Let's fix the price first (no offense intended to the government's regulatory bodies). You could do this in any of several ways, but one quick way is to command the Editor to locate "1.65" and to then change the "1" to a "2". The commands are as follows:

                    locate /1.65
                    change /1/2/

Please try those two commands (yes, I know it would be much easier to just move the cursor to the "1" and overlay it, but you already know how do it that way, so let's try it another way). The first one worked, right? The Editor placed the line beginning "send me 250..." at the top of the screen, just below the command line. But, the second command changed the "1" in "5-1/4", not the "1" in "1.65"! This is what is meant by a context editor. It scans along looking for a match on whatever you tell it to scan for. And, if you followed my intentionally erroneous instructions, you told it to change the first occurrence of the character string "1" to a "2".

The error in my "change" command was that I wasn't specific enough. To do it properly, first change the "2" back to a "1" (you can move the cursor and overlay it), and then type:

                    c/1.65/2.65

This time it worked, yes? Please note that the command you used this time is still the "change" command, but some short-cuts were taken: "change" was abbreviated to "c", and since "change" is one of the more frequent commands you will use, you can see why abbreviations are nice.

Any command may be abbreviated to at most two letters; and most commands can be abbreviated to a single letter. There are two kinds of exceptions to this. The first involves ambiguity: the abbreviation for "COPY" is "CO", since "C" is the abbreviation for the more commonly used "CHANGE"; and the second is when the file itself might be adversely affected: "QU" for "QUIT", even though there are no other commands starting with "Q".

Also notice that no spaces were left after the "c" (spaces are only needed when there might be an ambiguity, or where you want spaces to be included in the text). Finally, there are only two slashes ("/"), not three.

Delimiters

The slashes are used as "delimiters". A delimiter marks the start and/or the end of a string of information. Delimiters are needed in cases where more than one string may occur in a single command, and also in cases where there might be an ambiguity. For example, if you want to change something that includes blanks, you have to be able to tell the Editor just how many blanks are included. If you precede those blanks with a delimiter, then you've removed the ambiguity.

There are three rules to follow when using delimiters with EDIT. If you learn them correctly, they will become habits and you'll never have to think about them again:

1. A delimiter must be the first non-blank after the command (there are one or two exceptions to this rule, and they are identified in SECTION II);

2. A delimiter may be any character whatsoever, as long as it DOES NOT APPEAR ANYPLACE IN THE STRING(S) IT DELIMITS. In the above examples, a comma, the letter 'x', or any of about fifty or seventy other characters could have been used as the delimiter. However, neither "1" nor "2" nor a period could have been used, since they all were in the strings being delimited.

3. In any command line that uses delimiters, the same delimiter must be used throughout the line. You can't switch in the middle, because the definition of the delimiter is the first non-blank character after the command. That definition continues in effect until the end of the command line.

   This also means that control words such as ".TT" (top title) whose operands use delimiters cannot be followed by other control words (that bit about using a semi-colon-period won't work in these cases).

Now, strictly speaking, each delimited string must be both preceded and also followed by the selected delimiter. When using the "change" command, the "1" is surrounded on both sides by the slash. Perfectly regular syntax would require that the "2" (the second string) also be followed by a delimiter. However, EDIT is smart enough to tell where your final string ends, even if you leave off the final trailing delimiter. So, the second example of the "change" command took this short-cut and omitted the third occurrence of the slash. Actually, the only time you really need that third delimiter is when you have some trailing blanks in the "new" value.

Back to revision of the letter. You've changed the price, and now you've found that "Mrs. Riley" is actually "Mrs. Raugh". The name occurs three times in the letter, and must be changed wherever it occurs. Moreover, one of those places is above where you are now. Like BASIC, EDIT has this notion of "current line", and does all its scans from the current line on downwards towards the end of the file. The current line is the one at the top of the screen, just below the command line.

Since you are in the middle of the file right now (at the price line), you could do another "locate" to get to the next occurrence of "Riley", change it, and then somehow get to the first occurrence. However, there is a better way. We will move the "current line pointer" back to the top of the letter, and then make what's called a "global change". A global change, once you know about it, is one of those things you begin to really miss when you're working with an Editor that doesn't support it. The commands to move the current line pointer and then make the change are:

                    top
                    c/Riley/Raugh/ *

(TOP could be abbreviated to "T".) The "change" command begins in what by now should be a familiar form, but after the third and final delimiter, there is an asterisk. Is this yet another violation of the rules for delimiters? Not really.

Range of Effect of Edit Commands

This brings us to the notion of the "range" of effect a command can have. In the case of the "locate" command, the range begins on the line just below the current line, and continues to the bottom of the file, or to the first occurrence of the specified string (for EDTASM users, "locate" acts like the "FIND" command in EDTASM, and the "find" command in EDIT does something a little bit different).

The range of a change command is up to you. If you just say: "c/1/2/", the change will be attempted only on the current line, and only for the first occurrence of the string to be changed. If you want it to occur on more than one line (should the string in question be found elsewhere), then you have to specify a range of lines. This specification is given after the final trailing delimiter (that's the third occurrence of the delimiter), and may be either a number or an asterisk. A number specifies a definite quantity: "c/X/YY/5" tells EDIT to change the first occurrence of "X" to "YY" on each of the five lines beginning with the current line. It's O.K. for the "X" not to occur on all of these lines. If it doesn't occur on any of them, you will be told so. On the other hand, if the string is found, each changed line will be displayed as the changes occur. Then, the screen will be re-displayed from the original current line, but reflecting any changes that occurred. The command line will contain a message telling you how many lines were changed during the search.

If you want a change to be made from the current line to the end of the text file, you can either specify a very large number, e.g., 999; or else you can use the asterisk: "c/1/2/*" will change the first occurrence of "1" to a "2" on every line from the current one until it reaches the bottom of the file.

Please note that only the first occurrence of the search string is changed when this form of "change" is used. If you want to change every occurrence on a single line (assuming there are multiple occurrences), then you would do it this way:

                    c/1/2/1*

The "1" just before the asterisk is still the line-range (one line). The asterisk is the <u>second</u> parameter after the trailing delimiter, and as such, specifies the range within each line, rather than the number of lines.

The range of changes on a single line is either the first occurrence or all occurrences; EDIT does not support anything in between.

Finally, if you wanted to change every occurrence of a string throughout a file, you could do this:

```
top
c /string1/string2/ * *
```

The "top" ensures the line range can cover everything. The first asterisk says that the line range will be through the end of the file. And the second asterisk says that every occurrence of "string1" on each line should be changed to "string2".

Due to the way in which EDIT analyses commands, if you need to specify global ranges (using asterisks) in a given command situation, you cannot use the asterisk as a delimiter. In all other cases, including a single change on a single line, or a single change on each of a definite number of lines, the asterisk is acceptable as a delimiter.

In the case of LIMA commands, the range of effect is usually just the one line for which the command is given. When a numeric range follows the command, that number defines the range.

Just to complete the second part of this tutorial: having corrected both errors, you can tell the Editor to "END", and then let it switch you back to SCRIPT. When SCRIPT issues its prompts (the same ones as you were told about before), you can again just hit <ENTER> and get the revised letter printed again. Since there is no real point in doing that in this exercise, you can stop once you've "END'ed" EDIT. Just reply "N" to the question about passing the fileid to SCRIPT, or simply hit the <BREAK> key and get back into BASIC.

END OF TUTORIAL

If you're using this manual to learn EDIT and SCRIPT, I'd recommend you skim the rest of this section and then learn "EZEDIT" and "EZSCRIPT", which are covered in SECTION II. Practice and experiment on your computer as you do so, and leave Sections III and IV for later. The "EZ" facilities use only a few commands each, and represent the features needed to get most Word Processing done. After you become comfortable with "EZ", you can learn more of NEWSCRIPT on an "as-needed" basis.

The bulk of Sections III and IV describe each EDIT and SCRIPT command or control word, showing at least one example for each. Since SCRIPT supports a number of advanced functions (Form-letters, Table of Contents, Indices, chained files, multiple headings and footings, multiple type fonts), you'll probably find ways to meet most of your requirements by browsing through Sections III and IV. The "HOW TO..." chapter will also be useful for specific topics.

EDIT Run-Time Options

    "Options" are things you can control at a level that is even higher than the
normal command level. In the case of EDIT, there are only a few such options. As a
reminder, the options are available when you're asked to enter the name of the file to
be edited (you can reply with a filespec, a question mark, or the <ENTER> key), and
when you're exiting from the Editor. At that time, you're offered these options, which
were covered earlier:

                    PASS 'fileid' TO SCRIPT (Y/N/C/A, DEFAULT=Y)

The replies are:

                    Yes No Continue Append

SCRIPT Options

    The run-time options of SCRIPT are different from the control words that you
place inside a text file. The control words tell SCRIPT how to format the document,
whereas the options let you select the output device (screen or printer), the number
of copies, and give you the ability to handle single sheets rather than continuous
forms. Please note that some SCRIPT options are similar in spelling to some SCRIPT
control words (for example, "ST" as an option means "stop at the end of each page for
cut-sheet changes"; whereas the ".ST" control word means "stop processing until the
user presses <ENTER>, usually for a disk change).

    If you want to use any run-time options, you must select them EACH AND EVERY
TIME you are asked for them. SCRIPT won't remember them for you, since it assumes
you may want to change them from one run to the next.

    Several run-time options are available. Each time you run or re-run SCRIPT, you
will be asked whether you want to use any options:

            ENTER OPTIONS (ST, V, ED, PG, DS, TS, CC, ID, DA, NU, TC)

    If you want to run with the defaults, just press <ENTER>. If you want to select
any options, enter ALL THE ONES YOU NEED at this one time, and then press
<ENTER> (it only asks once).

The options are as follows:

ST stands for "STOP" and tells SCRIPT to stop at the end of each page, and to wait
for you to press <ENTER> before starting the next page. You will want to use this
option whenever you are printing on single (cut) sheets, as opposed to continuous
form, fanfolded paper. Each time SCRIPT gets to the end of a page, it will display
this message:

                PRESS <ENTER> WHEN READY TO CONTINUE

You should remove the completed page, insert and position the new page, and then
press <ENTER>. Make sure each new page is positioned at the first line on which
printing is to occur.

    NOTE: This option is different from the ".ST" control word, which is used to
allow the operator to change disks or take other needed action.

V stands for "VIDEO" and tells SCRIPT to output to the video display (the screen) of the TRS-80 instead of to the printer. The "V" option is used for checkout purposes, and is useful in catching errors you may have made in control words and page layouts. Since it doesn't use paper, it is cheaper, quieter, and faster than running to the printer.

Output to the screen will **NOT** be formatted, since justification to the printer sometimes requires the use of special ASCII characters that appear as garbage on the screen. So, with the "V" option, SCRIPT selects the next line, sets it up for printing, determines how much can fit on that line, and then shows you what it selected. It would be possible for SCRIPT to show you right-justified video output, but this would take extra time and slow you down.

Because of the usefulness of a checkout facility, I recommend you consider using the "V" option at least the first time you format each new document. It'll save you time and setup effort.

ED stands for "EDIT" and lets you do limited editing of your previously-created document, while it is being formatted. If you select this option, each line of text, including control words, will be displayed on the screen, and you will be prompted for any changes you want to make:

                              ?* EDIT *? ¦?

If you just press <ENTER>, the line will be accepted and used. The other options are: "D" (delete), "R" (replace), "I" (insert BEFORE this line), and "T" (terminate editing).

The Delete option just throws away the current line.

The Replace option prompts you to enter a replacement line, which is immediately used.

The Insert option must be followed, on the same line, by the line of text (or control characters) you want inserted before the line that has been displayed. After both of these have been processed, SCRIPT will proceed to the next line and display it. Note that this use of Insert is different from the use of Insert within the full-blown EDIT program, where Insert goes after the specified line.

The Terminate option causes the interactive editing feature to go away. SCRIPT reverts to its normal mode of reading everything from the disk file and processing what it finds. "T" was used instead of the "Q" (quit) used elsewhere to make it a little harder for you to accidentally drop out of EDIT mode.

PG n1,n2 This lets you select a starting and an ending page for printing. By default, everything is processed and printed (or displayed). However, if you just want to print one page that didn't come out well, and happens to be page seven, this option will let you avoid the time and paper it takes to print the first six (or however many) pages. If you want to print a range of pages, this option is also the way to tell SCRIPT what you need. No printing or paper movement occurs until the specified page is encountered, but you should leave the printer in READY status since it may be necessary to issue font (typeface) changes along the way.

'n1' is the starting page number, and 'n2' is the ending page number. If you only supply 'n1', then printing begins with that page, and continues to the end of the document. If you just want to print a single page, specify it as both 'n1' and 'n2'. Here are some examples:

            PG 5,5   print page 5 only
            PG 8   print from page 8 to end of document
            PG 1,3   print the first three pages only

     The only way SCRIPT can find the starting page is to process and format the entire document from the beginning, so it will take a while for printing to begin. It takes about 1/2 as long to process to the screen as it does to process-and-print, so if you figure one-to-two minutes per page, you can estimate how long your coffee break should be. SCRIPT is smart enough to stop processing after formatting the ending page, if you specify one; so you won't have to wait for the remainder of the document to be scanned. SCRIPT doesn't eject pages while finding the starting place, but may have to issue font change commands as they are encountered. Therefore, be sure to leave the printer ON-LINE and READY during this time.

<u>DS</u>   stands for "double space" and causes the printout to be double-spaced (what else would it do?). You can specify double or single-spacing within the document also, but if you expect to be annotating and revising a document, this gives you more room on the paper to do so. You should still plan to use the ".ds" and ".ss" control words within a document when you want portions of a document to be double-spaced in the final draft, but should not use the "DS" option with documents that force single-spacing internally.

"TS" is like "DS", but causes triple-spacing. There is no control word for this: it's only a run-time option.

<u>CC n</u> This stands for "number of copies", and lets you tell SCRIPT to make 'n' copies of your document. If you want just one copy, you don't have to use this option, since '1' is the default. If you want five copies, you would specify:

                              CC 5

A total of five copies, including the original, are produced. If you are running form letters, you don't need this option, unless you want multiple copies of each letter.

<u>ID</u> tells SCRIPT to print the name of each file in the left-hand margin when it starts processing that file. It is useful in making revisions to long documents, since anything much over 150-250 lines should be broken up into several separate files. By seeing the name of the file over in the margin, you are able to tell where you should be editing your changes.
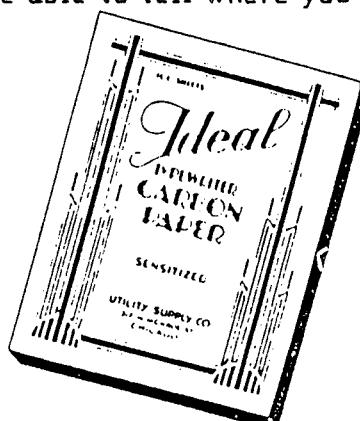
DA stands for "DARK" and causes SCRIPT to overstrike every line in the printed output. The option is useful when the ribbon is wearing out and photo-mechanical copies are to be made. It is also useful when printing in 10 character-per-inch (10 cpi) font, since this font sometimes seems lighter than the others. NEWSCRIPT does not explicitly support overstriking of individual characters or words within a line, although judicious use of the backspace sequence can produce such a result on a few printers.

When used on an Epson printer, "DA" turns on the "Emphasized" mode, and the results are very nice. Note, however, that the Epson manual states that Emphasized cannot be used with 16 cpi pitch: if you try to use both, the result will be printed at 10 cpi, but NEWSCRIPT will think it's at 16, so the results will be poor. This is a function of the printer, and is not controllable by NEWSCRIPT.

There is also a ".DA" control word that can be used within a SCRIPT file to selectively turn DARK on and off. Files containing that control word should never be printed with the "DArk" option, since the control word will override the option. The "DA" run-time opation corresponds to ".DA 1".

NU causes SCRIPT to number each line of each page. The numbering falls in the left margin, so if you haven't left enough room for it (via ".AD"), the text lines will be moved slightly to the right. This causes problems only if you've set line length (".LL) to be about as wide as the maximum width of your printer. The "NU" option is intended for use in legal briefs and theater scripts. Numbering begins anew on each page, and blank lines count even though they aren't numbered; this even applies when double-spacing. Titles aren't numbered, but are counted.

TC instructs SCRIPT to print only the Table of Contents for the current document. That Table must have been created previously by SCRIPT and still be available on a disk. This option is a time-saver, since it lets you bypass the formatting of an entire document when all you need is another copy of its Table of Contents.

There are two other run-time options you should know about. One of them allows you to stop SCRIPT in mid-flight without hitting the <BREAK> key. Just press the <ENTER> key for a moment or so, long enough for SCRIPT to notice it, and you will see this message:

DO YOU WANT TO CONTINUE PROCESSING (Y/N/E, DEFAULT=Y)

If you hit a key by mistake, just hit <ENTER> and SCRIPT will ignore the intrusion. If you really do want to stop, type "N" (or "n") and hit <ENTER> (no quotes around "N"). You will still be able to print the Table of Contents if there is one.

This feature is useful when you see a minor error, or if the printer gets hungry and chews up some of your paper. You can interrupt SCRIPT, and restart on the mangled page by using the "PG" option.

Replying "E" <ENTER> will activate SCRIPT's built-in mini-editor and continue processing as though you had taken the "ED" run-time option. All mini-edit commands are valid again, including "t" (terminate mini-edit mode).

The other useful option is the ability to enter control words and text directly from the keyboard, without using a disk file. When you do this, your work is NOT **saved on disk, but you can experiment with page layouts** and SCRIPT formatting options very easily, without having to use the Editor first. The option is also useful if you just want to write a quick letter and are confident about your typing akuracie. As you may have guessed, I don't use that option very often, but it is excellent for self-education purposes.

Oh! I forgot to tell you how to invoke this direct-entry option. Sorry about that! When you run SCRIPT, the first question you are asked is:

<p align="center">ENTER I.D. OF FILE TO BE SCRIPTED</p>

(there may be a default shown). Reply with a semi-colon ";" and press <ENTER>. Instead of looking for a file, SCRIPT will place you in "EDit" mode immediately (but will still ask for any other options you may want to use). You will thereafter receive the EDit prompts until you enter a null line (an extra <ENTER> without any preceding data). Anything you type in during EDit mode is treated as though it came from a disk file, so you have all the power of SCRIPT available.

Run-Time Delays

Because portions of NEWSCRIPT are written in BASIC, they are subject to occasional run-time delays when the BASIC Interpreter (the ROM) performs string compression (also called "garbage collection"). When this happens, the computer appears to be stalled, and it will not respond to the <BREAK> key or any other attempted intervention short of the <RESET> button. These delays can last up to two minutes in extreme cases; although they usually last only a few seconds.

Whenever string compression does occur, NEWSCRIPT will place a large graphic "C" in the upper right-hand corner of the screen, and remove it when compression ends. If you happen to be typing data during this time, you can continue typing for a few moments: the typeahead buffer can hold up to 128 keystrokes at a time.

File Format

EDIT and SCRIPT create and use plain, unadorned, vanilla ASCII files. These are compatible with files processed by BASIC, are written with the "PRINT #1" statement, and read with the "LINE INPUT #1" statement. Therefore, any other programs you use that create files this way, or read files this way, can exchange files with EDIT and SCRIPT. Each record is limited to 255 bytes in length, and the files are sequential. The only chaining from one file to another is through the SCRIPT control words, ".ap" and ".im", so private index structures are not involved. Some users of SCRIPT have processed files created by other editors through this Word Processing system without any problem. However, PROSOFT has not tested or attempted to provide compatibility with other Word Processing systems.

NOTATION

This material
may be skipped or skimmed
when first learning NEWSCRIPT

To explain how to use SCRIPT and EDIT, we have to be able to distinguish between required words (commands, control words, keywords) and optional words or numbers. (These options are called "operands" or "parameters".) For example, if you want to tell SCRIPT to leave one blank line, you can say:

.SK

However, if you want several blank lines, you must supply the quantity:

.SK 3

In this example, ".SK" is required, while the number of lines (the "parameter") is optional. In other cases, there may be more than one "parameter", and in still other cases, there may be a choice of parameters. If you think that's confusing, just imagine what it might be like to have to try to explain all the possibilities when you and I can't readily talk it over.

To reduce the confusion, we will agree to use a standard notation throughout the rest of this document. Since most other people using computers use these conventions, once you learn them, you will be able to skip sections like this in the future. Actually, as you will see, the notation is pretty simple.

First of all, every command and control word will be shown along with references to all the parameters that are allowed. The command (we'll understand "command" to also mean "control word" from now on) is ALWAYS first (otherwise, it wouldn't be a command), and the options follow it, usually in a specific order (the same order as shown in the command format).

Anything that is required (anything you HAVE to type to get the result you want) is shown in CAPITALS. This doesn't mean _you_ have to type in capitals, however. SCRIPT and EDIT understand both upper and lower-case whenver there is no real difference between them (SK, sk, sK, and Sk all mean the same thing to SCRIPT).

Most commands can be abbreviated. For example, the letter "C" is all the Editor needs to be able tell that you want to "change" something. But, if you want to type out the full word, or part of it, EDIT understands "change", "cha", etc., to all stand for the same thing as "c". The shortest valid form of each command is the _required_ part, and in the command format is shown in CAPITALS. The optional remainder of each command is shown in lower-case:

Backpage

Some EDIT commands may be abbreviated to one letter, others to two letters. However, _all_ _SCRIPT_ _control_ _words_ _MUST_ _contain_ _exactly_ _two_ _letters_. The Edit commands that cannot contain fewer than two letters are the ones that could be confused with other, more common Edit commands ("BReak" and "BOttom", as opposed to "Back"... do you see the difference? "B" always stands for "back"; if you want one of the other commands that start with "B", you must give a second letter to make the

command unique). In addition, commands that could give you grief if you use them at the wrong time ("DElete", "QUit", and some others) make you use two letters in an effort to ensure you mean what you say.

That's about it for command and control words. What about those things called "parameters" or "options"?

Some parameters are optional (no pun intended). They are shown in brackets:

<p align="center">Down <n></p>

This means that you only have to supply the number of lines 'n' if you want to move the window DOWN more than one line. (In a moment, we will show what 'n' means, and why '1' is assumed.)

EXCEPTION: the special keys <ENTER>, <BREAK>, and <CLEAR> are shown in brackets also, but when shown, even this way, they MUST be used! They aren't optional, even though they are in brackets.

If a parameter is required, it will not be in brackets.

If a parameter is a "keyword", then it will be shown in CAPITALS, and if used, must be typed exactly as shown:

<p align="center">Move n1 UP n2</p>

This tells the Editor to move some lines upward in the file. Since "DOWN" and "TO" are also options of "MOVE", you have to tell EDIT which direction, or where you want the block of lines to go.

Many parameters are numeric. The notation for these is either 'n', 'n1', or 'n2' (if three numbers are possible, 'n3', etc.):

<p align="center">.SK <n></p>

This means that you may optionally supply a number of lines to be skipped (left blank).

Many commands take defaults (make assumptions) when their parameters are omitted. In the command formats, the defaults are underlined:

<p align="center">.SK <n> | <u>1</u></p>

This is the complete command format of the conditional-skip command. You don't have to supply the number of lines, since 'n' is in brackets, and when you omit a number, '1' is used. (That vertical bar, "|", is explained later.)

Some parameters are words (called "strings") or Filenames (called "fileid"). The notation for strings is "string" or "string1" or "string2", etc. If the string can be only one character long, then "c" is used for notation. Remember that numbers are often valid characters and members of strings... it all depends on the usage. For example, the "Insert" command format is:

<p align="center">Insert <string></p>

This means that the command can be abbreviated to the single letter "I", and that a string of characters may optionally follow the command. All four of these would be valid:

```
i Dear Mr. Jefferson:
Ins Please let me know when...
i
i x
```

Each of these inserts one line of text just below the top line on the screen, and that new line becomes the current line. If no data follows the "i" (as in the third case above, a blank line is inserted.

Last, but not least, there are times when you have a choice of parameters. When you may select from any of several alternatives, there either will be a vertical bar (it stands for "or") between each of the alternatives, or they will be listed vertically:

```
MOve n1 UP | DOWN n2
     n1 TO string
```

As you can see, when the alternatives begin to get out of hand, more than one line may be needed to show the valid combinations. In this example, you are shown that you can move one or more lines ("n1") UP or DOWN one or more lines ("n2"); or you can move some lines ("n1" again) TO a specific point in the file: a line beginning with "string".

Here is an example of a very common SCRIPT control word, used to turn both right-justification and also concatentation of text either ON or OFF:

<div align="center">.FO <<u>ON</u>> | <OFF></div>

The notation used in this command format indicates the following:

1. The control word is ".FO" and must be entered that way;
2. Both of the possible parameters are optional;
3. Only one parameter (or none) may be selected;
4. If a parameter is selected, it must be spelled as shown;
5. If no parameter is supplied, the default ("ON") will be used.

This completes the description of notation conventions used here, and also marks the end of Section I, the introductory and tutorial material. After you catch your breath, I'd suggest you go through SECTION II, and leave SECTIONS III and IV for later.

SECTION II - EZEDIT and EZSCRIPT

This section is intended to help you learn enough about EDIT and SCRIPT to meet most of your common needs. If you learn how to move the cursor, use the control keys, twelve EDIT commands and ten SCRIPT control words, you will be able to create, revise, and print most normal documents. As your comfort level increases and you find a need for more information, you can browse through Sections III and IV later on.

A brief explanation of each EZ command is given here. If you want to learn their full functions, you can look them up in SECTIONS III and IV later on. Also, please remember that trying things out on the computer is the best way to learn what really happens.

As explained earlier, EDIT divides the screen into three parts: the Command Line, the Data Area, and the LIMA (Line Manipulation Area). The Data Area and the Command Line will be explained here, and the LIMA will be explained at the beginning of Section III, as an advanced (but very useful) topic.

These are the Edit commands you will most probably be using most of the time. The capitalized portions are the minimum valid abbreviations:

Locate	Up	Top	Forward	SAve
Change	Down	Bottom	Backpage	ENd
Whoops				QUit
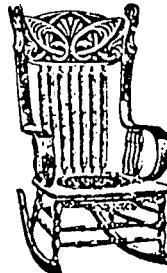
These are the SCRIPT control words you will be using the most:

.SKip	.CEnter	.FOrmat	.LLength	.APpend
.PAge	.PPara	.UScore	.BReak	.BFont

Since most of these commands are taken from normal English, you can see why they are so easy to learn and remember.

EZEDIT

Cursor Movement

The four arrows (up, down, left, right) can be used to place the blinking cursor almost anyplace on the screen. The "change signal" column following the LIMA is the only exception to this. Once the cursor is where you want it to be, you can overlay whatever may have previously been there, or by setting Insert mode, you can stick as many characters or lines of text as you wish into an existing document.

If you place the cursor on the command line (the top of the screen) and press the up arrow again, the screen will scroll towards the top of the text, one line at a time. Similarly, if you place the cursor on the last line of text and press the down arrow, the screen will scroll towards the bottom of the text, one line at a time.

The cursor moves forward every time you press a letter or number (a character). If it gets to the end of a line (except the command line), it moves to the start of the next line and takes with it the current word being typed. If it gets to the end of the screen, the screen is scrolled up one line and the next text line, if any, comes into view. If you reach "*EOF*" and keep typing, new blank lines will appear as needed just above the "*EOF*".

Since SCRIPT control words must always begin in column 1 (the first position of the data area, just to the right of the LIMA), you'll find it necessary from time to time to move the cursor to that position. The quickest way to do so is to hold down <SHIFT> and press the down-arrow. That will move the cursor to the start of the next line.

Now, if that next line is not blank, and you wanted to continue your typing on a blank line, you should just hit <ENTER> (as though on a typewriter), not SHIFT-down-arrow. The <ENTER> key acts like a typewriter's carriage return key when in the "DATA" area of the screen, and moves you to the next line for more typing. However, if that next line isn't blank, EDIT generates one for you.

Control Key

The <CLEAR> key is EDIT's control key. Before using it, position the blinking cursor where you want to make a change. Then, hold it down while you press one of these keys:

I - set or reset character INSERT mode

D - delete character at cursor

Right Arrow - splits line into two lines

Space Bar - blanks out rest of line (to spaces)

Up Arrow - enters an up arrow (prints as "[") on the screen

Commands

While the arrows and normal typing suffice for entering text and making minor corrections to it, they are not sufficient for full-scale text review and revision. For such purposes, several dozen "Commands" have been provided. Commands are always entered on the top line, next to the "E=>" prompt. Commands are used to locate and change data throughout a document, to move text around, to set or check on status (how many changes have been made, how much room is left), and to save the results of editing.

Once you've entered some text, you probably will want to make some changes to it. The "Change" command (convenient name, isn't it) can make a change on a single line of text, across a range of lines, or throughout the entire file.

If you want to look something over, and aren't sure where in your text file it is, use the "TOP" command to get to the first line of the file, and the "Locate" command to scan along looking for a phrase that is part of what you want to look at. If you don't remember a particular phrase, then you may have to read everything. You would still start at the "Top", but this time you might read each "page" on the screen (fifteen lines per page), and then "Forward" to the next "page". For example:

```
top
f
f
f
c/here/there/
```

That would get you to the top, let you look over three screenloads (45 lines), and then, if what you were looking for was at the current line (in this case, that would work out to be the 31st line of the file, and would appear at the top of the third screen you examined), cause a change of words to be made.

By the way, when you see quotation marks in this narrative, it doesn't mean you should use those marks. Neither EDIT nor SCRIPT expect to see quotation marks, and will be very confused if they are used. Unfortunately, when a computer program becomes confused, it usually doesn't do anything the way you want it done.

There are several ways to perform "global searches" in EDIT. The most common one is to use the "LOCATE" command, which can be abbreviated to the letter "L" or to a slash "/". The "L" must be followed by a delimiter (the "/" makes a good one, usually, and the "L" can be omitted if you use the "/" or the "-"), and the delimiter should be followed by whatever it is you want to find. The search takes place within the currently defined column ZONE, starts just after the top line on the screen, and continues until the search argument is found, or until end of file is encountered. Like "CHANGE", "LOCATE" can be interrupted at any time by pressing any key.

Here are some examples of "LOCATE":

```
locate /don't let this/
L Xany delimiter is okX
/if it isn't in the string
```

However, this would be invalid, since the delimiter also occurs as part of the text:

```
Locate 'isn't it there'
```

If you want to delete a line of text, just get it on the screen, move the cursor to the line, hold down <SHIFT>, and hit the left-arrow either once or twice, so that the cursor jumps to the extreme left-hand side of the LIMA (AHA! That's how you get into the LIMA ... that's the only way). Then, type the letter "d" (subtle, ain't it?) and hit <ENTER>. If you want to delete two lines in a row, you could put a "d" on each one, or you could put "d2" in the LIMA of the first one. Actually, you can delete up to 99 lines at a time this way, but for safety, you probably should delete only what you can actually see on the screen.

Something to remember is that text is stored and manipulated by lines, not by words, so you can't use "delete" to remove a few letters or words from the middle of a line; "delete" gets rid of entire lines. However, you can use CLEAR-d to delete characters anyplace at all, or you could use the "change" command. Since we've already talked about the CLEAR-d approach, let's look at how the "change" command could be used to get rid of text in the middle of a line. For example:

> While travelling with Steve and Jill in Europe,
> we found a quaint chateau in southern France.

If you wanted to remove "with Steve and Jill" from the first line, and "southern" from the second line, you could do it as follows (there may be better ways to position the text so that the "Steve and Jill" line is the "current line", but this way will work and is easy to understand):

```
top
locate /Steve and Jill/
change /Steve and Jill //
down
change /southern //
```

Note that the trailing blank should be removed as well, or you will wind up with two blanks in a row (the one before the phrase being removed and the one after it). The "change" command understands "null" strings, and that is what you are specifying as the replacement value in both these lines. The "null" string contains zero characters, and is found between the two successive delimiters (there's nothing between them, and that's what the "null" string is).

Also note that "top" can be abbreviated to "t"; "locate" to "l"; "change" to "c"; and "down" to "d".

Since you now know that EDIT has the "top" and "down" commands, you've probably figured out that it also has the "bottom" and "up" commands. "Top" gets you to the first line of your file, so "BOttom" (two letters minimum) gets you to the last line. "Down" moves the window down one or more lines ("Down 5" moves the window to the fifth line following the current line), so "Up" moves the window up. After the window is moved, the line at the top will be the new "current line".

The reason why the shortest abbreviation of "BOttom" is two letters is that "B" is the abbreviation for "Backpage". "Backpage" moves the window back one full screen (fifteen lines), and is a nice way to scan through your document. For those of you who prefer to scan forward so that you can read the text normally, EDIT provides the "Forward" command (15 lines at a time). "Backpage" and "Forward" both let you specify the number of fifteen-line pages to be "scrolled" (that's what it's called, a term probably coined by a student of ancient history), so you can skip around pretty fast.

We're almost done with EZEDIT.

The last of the EZEDIT commands are the ones that let you save your text onto diskette. The simplest way to save a file is to just type "SAve" when in EDIT mode. Whatever is currently in main memory will be written to disk under the name you supplied when you started editing; and then you will regain control, still in EDIT mode, still able to make more changes to the current file. Typing "SAVE" from time

to time is good practice, since it is annoying and sometimes costly to have a power failure, computer glitch, or (perish the thought) a bug in EDIT after spending an hour or so writing a paper. If you type "SAVE" after every few dozen lines, or every five to fifteen minutes, you can minimize the impact of a system failure.

If you have trouble getting your text files onto drive 1, just remember to include the drive number ("filename/ext:1"). Or, you can make sure there are no text files on drive 0, and always keep it write-protected by using a silvery tab over the square notch.

When you are all done editing, and want to go on to something else (such as SCRIPT), the command to use is "END". It will save your file and ask whether you want to pass the file on to SCRIPT. By pressing just the <ENTER> key in reply, you will cause SCRIPT to be run. Since "END" performs a "SAVE" automatically, it isn't necessary to use both of them when you're done: just use "END".

Finally, if you decide you don't want to save the current edited version of the file at all, you can just type "QUit" as a command. It will ask you if you mean it, and if you do, reply "Q" (no quotes). Of course, if you want to quit at any point, you could just press the <BREAK> key, and get the same result, but that's a bit sloppy.

Oh! I almost forgot to mention one of the most useful EDIT commands of all: WHOOPS (it can be abbreviated to "w" once you're on speaking terms with it). "Whoops" tells EDIT to ignore the changes that are currently on the screen, and refresh the screen to how it was before. All changes on the screen are discarded, but any changes that have scrolled off the screen, or changes made before the previous <ENTER>, are still in effect. For even better protection, use the "SAVE" command from time to time. Then, between SAVE and WHOOPS, you should be able to recover from most self-inflicted wounds with minimal anguish.

And that's EZEDIT. Easy, wasn't it? Remember that there are three dozen other EDIT commands we haven't covered here, and that there will be times some of them can be very useful. Also, I haven't told you about all the options of the EZEDIT commands, nor most of the uses of the LIMA, in an effort to keep things relatively simple. After you've become comfortable with the "EZ" commands, you'll find Section III useful in learning more about EDIT.

SECTION II

EZSCRIPT

   For simple documents such as reports, you may be able to avoid using any Script control words. However, there are many times when you will need to exercise some control over the formatting of your work, and in many cases, the control words of EZSCRIPT will be all you need.

   The first thing you will probably want to be able to do is place blank lines between your paragraphs. What you want to do is "skip" some lines, and the control word to use is ".SK". You saw this used in the "letter" example, so the use of the dot as the first character of every SCRIPT control word is not entirely new to you by now. Just remember that control words always begin at the extreme left side of a line, and that normal text cannot follow them.

   If you just want to skip one line, ".SK" is all you need. If you want to skip more than one line, you have to supply the number:

                                 .SK 7

will leave seven blank lines (unless you're at the end of a page; SCRIPT is smart enough to not carry the extra blank lines over to the next page). You don't need a "space" between ".SK" and "7", by the way: ".SK7" is fine. Again, remember that the quotation marks you see here should NOT be included when you enter control words in your own documents.

   If you want to cause a page eject, use ".PA".

   If you want to center a line of text, what do you think the control word would be? Right! The control word is ".CE"!

            To get a phone number that isn't in the directory, dial:
            .ce
            555-1212

"555-1212" would be centered on the line following the "To get a phone..." line.

If you want to center several lines in a row, just supply the number right after the control word: ".CE 3". Or, you can turn centering on ".CE ON", type the lines that are to be centered, and then turn centering back off: ".CE OFF".

   If you want to disable SCRIPT's line-formatting capabilities for a while, and have it print each line "as-is", you can turn formatting off:

                                 .FO OFF

   Since "OFF" is a word, not numbers, you have to leave a blank before it. When you decide to enable formatting again:

                                 .FO ON

   One convenient way to start a new paragraph is by preceding it with ".PP". That will generate a blank line, ensure there are at least two lines left on the page (see ".CP" or "Conditional Page" in Section IV), and indent the next line of text five characters (about a half-inch).

If you want to produce tables or other column-aligned results, you must turn format off (.FO OFF), be in a "monospace" font (.BF 10,12, or 16), and optionally use tabbing. We won't talk about tabbing right now, but will assume that you'll layout the material on the screen exactly the way you want it printed. (Tabbing in SCRIPT is definitely an advanced topic.)

.BF 10

will switch you into 10 characters-per-inch mode, and leave you there until you say:

.BF 737

which takes you back into proportional mode. If you want to print something in the dense, 16.7 cpi font:

.BF 16

(don't say "16.7"). Although SCRIPT will let you change font and pitch in the middle of a line, some printers can make the switch only between lines, so it may be safer to reserve font changes for new lines of printing (that is, place ".BF" after some other control word that <u>does</u> cause a control break). Also, ".BF" must be the last control word on a line. Finally, "16" is an approximation. On some printers, it's really "16.5", on others, "16.7", and on still others, it's "17.1". SCRIPT compromises at "16.7" in all cases, so on some printers, use of 16 pitch may lead to lines that are slightly shorter than expected. (In this manual, "pitch" and "font" are used interchangeably. They don't mean the same thing in English, and "pitch" might be closer than "font" to the feature you are selecting. However, many of the printer manuals use "font", so that term is used here as well.)

When setting up print lines, SCRIPT assumes that you want a 6.5 inch line length. On a normal printer, or in 10 cpi mode, this corresponds to 65 characters per line. In proportional font, of course, it could correspond from about 162 lower-case "j's" down to 54 upper-case "M's". When you want to change the line-length, you do so with the ".LL n" control word, where 'n' is a number between 10 and 80 (meaning: 1.0 inches to 8.0 inches). Line lengths are specified in tenth's of an inch for compatibility with Word Processing systems that only support 10 cpi printers:

.LL 60

would give you a 6-inch wide output line on the printer.

There may be times when you want to indent an entire section or paragraph. The control word to use is ".IN n", where 'n' is a number giving the INdentation in tenths of an inch:

.IN 5

causes all subsequent text to be pushed .5 inches (1/2 an inch) in from the left margin. The right margin is <u>not</u> pushed to the right by this. When you've finished with the indentation, remember to move it back to the left margin:

.IN 0

If you don't want to turn formatting off (.FO OFF), but want to separate two of your input text lines from each other, you can use the ".BR" control word. This stands for "BREAK" and causes a break in the text. Use of "BREAK" is considered an advanced topic, so for more information, please see Section IV.

Now is as good a time as any to explain that most control words cause a "break" in the text. A "break" suspends concatenation momentarily, forcing SCRIPT to format and print everything up to the "break" before looking at what comes after the "break". There are a few execptions to this rule: ".US" (which is covered in a minute, below) is used for underlining, and allows you to underline (or underscore) one or more words anyplace you want them underscored; and ".BF" (begin font, covered earlier) lets you change fonts in mid-line, even though it isn't recommended for most printers. Would you like some examples (I thought so)?

```
This might be your input text, one weighty word
after another.  SCRIPT is concatenating one line to
the next as though there was no tomorrow.
.br
Whoops! This input text line will start on a new
output line, because of that control word.
.fo off
This input text line will also start on a new output line,
because ".fo" causes a control break.
Of course, with formatting
turned
off, this intentionally ragged input will be reflected
in the output, so let's turn it back on!
.fo on
Now, suppose you wanted to underscore something for
.us emphasis.
That's how you would do it, and the result would print
as follows:
```

```
Now, suppose you wanted to underscore something
for emphasis.  That's how you would do it.
```

And now you've learned how to use the ".US" control word: whatever follows it (after a single separating blank) gets underscored. There can be several words in what is to be underscored, and SCRIPT is smart enough to maintain the underscoring across more than one printed line.

The last EZSCRIPT control word is used to chain files together. Chaining is used as a means of overcoming the memory size limitations of computers, the diskette capacity limitations, and also to permit you to develop several small Edit files instead of one great, big, unmanageable one. Chaining is accomplished by "appending" one file to another:

.AP PART2/MS

".AP" should be the very last line in a text file, since the file it identifies for appending becomes the file that is processed. Control never returns to the original file, but there is a control word (".IM"), not covered within EZSCRIPT, that does let it regain control. With "APPEND" you are essentially telling SCRIPT to switch over to another input file and treat its contents as though it was part of the first file.

The second file can append a third file, etc. If you develop a document whose several member files exceed a disk's capacity, you will want to look at the ".ST" control word (not part of EZSCRIPT, but part of SCRIPT). Between ".AP" and ".ST", you can develop documents of unlimited length (this manual spans five diskettes).

It is good programming practice to keep programs short. The statement also applies to documents. A complex programming problem is more easily solved if lots of little programs are used than if a single monster is attempted. Similarly, Editing several relatively small files that are chained together via ".AP" makes revision easier and faster. 150-200 input lines, each of screen width, seems about right to me on the TRS-80. This also makes it easy to print just one portion at a time while you are making corrections and revisions. That would come to 3-5 single-spaced printed pages.

And that's EZSCRIPT. Like EZEDIT, it covers only part of what can be done with this Word Processing package, and if you browse through SECTION IV, you'll be aware of other facilities for when you need them.

I hope this tutorial information has been of benefit to you. With a little practice, you'll find yourself becoming proficient, and then expert in the uses of EDIT and SCRIPT. In the years to come, as micro-computers get bigger and less expensive, a lot of mainframe software will become available on micros. Since EDIT and SCRIPT are based on one of the more widely-used mainframe software packages, you will find that you've had a nice head start on the future.

SECTION III - THE EDITOR

Features

NEWSCRIPT's Editor has four distinct groups of capabilities:

1. Cursor movement and text input;
2. Control key functions;
3. Line Manipulation Area (LIMA) commands;
4. Normal commands.

The ways and reasons for using each of these will be covered in this section.

Sequence of Screen Processing

Because so many things, and combinations of things, can be done on the screen, it's important to know in which sequence EDIT will process your commands and changes. Not understanding this can lead to unforeseen and possibly unwanted results, whereas a mastery of the facilities and sequences will let you edit and revise very rapidly and easily. A common error is to use a LIMA command that moves the current line, while also using a command-line command that didn't expect the viewing window to move. This error is easily corrected by "Backpaging".

Screen processing occurs in this sequence:

A. When <ENTER> is not pressed, scrolling (vertical movement of the viewing window) occurs as needed. Only the one line that moves off the screen is stored internally. The command line is not examined, the LIMA is not examined, and any LIMA commands that scroll off the screen are permanently lost without being processed.

B. When <ENTER> is pressed, processing proceeds as follows:

1. If the command line says "W" (whoops), the screen is refreshed without being processed, and control returns to the user. Otherwise...

2. The text area of the screen is copied to permanent in-memory storage (buffers).

3. If the BREAK key was pressed, BASIC is permitted to honor the BREAK (you can resume by typing "CONT"). Otherwise...

4. The LIMA is processed from the bottom of the screen up to the top of the screen. Inserts and Repeats will update the pointer to the "current line" (the line that will appear as the top one on the screen). Deletes will cancel any named points associated with the deleted lines.

5. The Command line is processed.

6. Control is returned to the user. However, if any automatic functions have been triggered (AUTOSAVE or AUTO-allocation of additional text buffers), they will complete first.

Because of the typeahead feature of NEWSCRIPT, you can keep typing while most of this processing occurs. Up to 128 character scan be entered with typeahead. Except when a file is being saved, it's unlikely that this buffer will overflow

Cursor Movement

The blinking cursor can be moved to any position on the screen except the column right after the LIMA (column 4). The four arrow keys will move the cursor in the appropriate directions. If the cursor is at the bottom of the screen and the down-arrow is pressed, the screen will move (scroll) one line towards the end of the file, so that a text line previously not on the screen will appear at the bottom of the screen (the top text line is removed and saved). If the cursor is at the top of the screen (on the command line) and the up-arrow is pressed, the screen will scroll one line towards the top of the file.

There are some limitations on scrolling: if the *TOF* marker is reached, the up-arrow will not cause any further scrolling towards the top of the file; if the *EOF* marker is reached, the down-arrow will cause brand new blank lines to appear, one at a time, at the bottom of the file.

The left arrow moves the cursor back through the text area. When the left-most column of text (column 5) is passed, the cursor moves to the right-most position of the previous line (it will not go into the LIMA). The cursor cannot be backed into the command-line prompt ("E=>") unless shift-left-arrow is pressed, and no data can be entered in the prompt area (an error results, giving a BEEP if you've connected a speaker to the tape EAR plug).

The right arrow moves the cursor forward through the text area, skipping over the LIMA. If the cursor is at the end of the screen and the right-arrow is pressed, the screen is scrolled up one line and the cursor advances to the start of the new line at the bottom of the screen. If the cursor is in the LIMA, it advances normally until it moves out of the LIMA. At that point, it skips over the "change signal" column (column 4) to the first text position.

Shift-up-arrow moves the cursor to the start of the command line.

Shift-down-arrow advances the cursor to the start of the next text line. Unlike <ENTER>, it normally does not allocate new blank line, but just advances the cursor and possibly scrolls the screen if the cursor was on the last line. If the next line is the *EOF* marker, then a new blank line will appear.

Shift-left-arrow moves the cursor back to the left margin of the current line. However, if the cursor is already at the left margin, it is moved into the LIMA. This is the only way to get into the LIMA, and is intended to avoid accidental issuing of LIMA commands.

Shift-right-arrow moves the cursor to the end of the text on the current line. If the cursor is already at the end of the line, it's moved to the start of the next line. If the cursor is in the LIMA, it's moved to the start of the current line.

Text Entry

Text and commands are entered through normal typing. When the end of the command line is reached, an error BEEP occurs (if you've connected a speaker to the tape "EAR" plug), and forward cursor movement is inhibited. When the end of a data line is reached (lines 2-16 on the screen), the cursor moves to the next line, taking with it the current "word" being typed (a word is bounded by blanks). If the next line contains any text of its own, then a blank line is supplied to avoid destroying existing text. If the next line is blank, typing merely continues on that line.

Unless turned off, every key will auto-repeat if held down for about a second. Repeat speed is normally 20-25 characters per second. If you want to change this, see the discussion of "NSINIT" in SECTION IX.

As you enter text, it normally replaces whatever was there previously. This is called "overlay" mode and is indicated by the shape of the blinking cursor: a small, square, "o". If you want to insert or delete characters, you must either use the Control key functions (described next), or the "CHANGE" command (described in EZEDIT and later on, under EDIT commands).

Control Key Functions

The <CLEAR> key is NEWSCRIPT's control key. Holding down the control key gives you access to several additional features even though the keyboard of the TRS-80 doesn't have extra keys for control purposes.

Actually, there are two control modes. The Editor uses <CLEAR> plus another key (described next), while the general-purpose keyboard routine in "NS" itself uses <SHIFT-CLEAR>. You'll want to use both, so let's cover each one in turn.

CLEAR key

When you press the <CLEAR> key, the blink rate of the cursor slows down and then stops, waiting for you to press another key WHILE STILL HOLDING DOWN <CLEAR>. If you release the <CLEAR> key, you immediately return to normal text entry mode. However, while holding it down, you can press any of the following keys as needed:

D   will delete the character at the cursor and shift the rest of the line one
     position to the left. This key will repeat slowly, giving you lots of time to
     release it when finished. NOTE: if you have a long text line (over 60
     characters), not all of it will be seen on the screen, and the portion that is not
     visible is NOT left-shifted. This makes it easy for you to line up report
     headings and tables without accidentally losing column alignment. If you do want
     to left-shift the data during character deletion, you should use the "CHANGE"
     command instead. (If you use "CHANGE", remember that if "FLOW" is ON (by
     default), long lines will be split into multiple lines. If you don't want that
     either, then turn FLOW OFF first.)

I   turns Insert mode on or off (it acts as a toggle switch). When in Insert mode, the cursor looks like a capital "I". Any text entered while in Insert mode pushes anything else to its right further to the right. If such text reaches the right side of the screen, all words after the cursor are moved to a new blank line just below the current line. Once there is no more text past the cursor, EDIT will automatically turn Insert mode off (it will wait until it gets to the end of the line, just to be sure). Also, pressing the <ENTER> key, or any scrolling, will turn Insert off for you. Insert mode works in the text area and on the command line, but not in the LIMA.

Space bar   blanks out the portion of the line to the right of the cursor. Again, this does not affect portions of a long line if they aren't on the screen. And it doesn't affect the next line on the screen. It can be used in the text or command line areas.

Right Arrow   causes a "text split". A new, blank line is created just below the line containing the cursor, then all text from the cursor to the end of the line is moved to that new blank line. The cursor stays on the original line. This is useful in moving small pieces of data around. Any text not on the screen will not be split, so if you are working with a long line, be careful about using this feature. The "BREAK" command of EDIT may be used instead. Also, see the "JOIN" command.

Up Arrow   enters an up-arrow character as normal data, rather than as a movement of the cursor. This generally prints as a left-bracket: [

BREAK Key

The BREAK key is partly disabled when EDIT is running. If you press it, EDIT will first save the contents of the screen, and then let BASIC honor the BREAK. You can issue BASIC's "CONT" command to continue, and when you do, the screen will be restored to its contents before BREAK was pressed.

CAUTION: Don't hit <BREAK> several times in a row, since it is just barely possible that you will succeed in interrupting EDIT while it's trying to save the screen. If this happens, some of the screen will be lost.

SHIFT-CLEAR

Holding down <SHIFT> and pressing <CLEAR> activates the keyboard routine's control mode. A pair of question marks will appear in the lower right-hand corner, and you can then release the two keys. While the "??" prompt is visible, any of the following can be done, after which the prompt is replaced by whatever data had been on the screen, and normal text entry mode resumes. These functions are available whenever NEWSCRIPT is active; you don't need to be using EDIT at the time:

R   turns auto-Repeat off and on. It acts as a toggle switch.

P   copies the screen to the Printer. 16 lines of 64 characters are printed. Text not on the screen (above, below, or on either side of the viewing window) is not printed. Pressing the <ENTER> key during this process terminates the printout.

<u>V</u> turns dual-routing of Video-to-printer on and off. When on, everything BASIC writes to the screen is also written to the printer. However, changes made directly on the screen will <u>not</u> be routed to the printer in this mode. "V" acts as a toggle switch to turn this on and off.

<u>J</u> toggles the Japanese character-set on the Model III. This only affects the video, not the printer.

<u>S</u> toggles the "special characters" and "space compression" on the Model III. This only affects the video, not the printer.

Any Hexadecimal value from X'01' through X'FF' may be entered by typing its two-character Hex value. For example, typing "BF" will produce the largest graphic rectangle. However, it isn't safe to use all possible values. For example, a X'0D' is a line feed/carriage return and may confuse BASIC later on (that is to say: don't use X'0D'). Also, X'5B' (up-arrow) is treated as a cursor movement. This feature is useful when you want to print some of the special characters on printers such as the Daisy Wheel II.

Some special symbols can be entered with one keystroke, without knowing their Hexadecimal values, when SHIFT-CLEAR has been activated. This table gives the hexadecimal, key, and print symbols for each of the extra characters supported:

HEX	KEYBOARD KEY	PRINTS AS
5C	Down-arrow	\
5D	Left-arrow	]
5E	Right-arrow	^
5F	Minus sign	_
7B	Shift-up-arrow	{
7C	Shift-down-arrow	|
7D	Shift-left-arrow	}
7E	Shift-right-arrow	~
7F	Shift-minus sign	non-splittable blank

Egyptian hieroglyphic

The "up arrow", which normally prints as a left bracket ("["), is entered by holding down <CLEAR> and pressing the <up arrow>.

The <u>Line</u> <u>Manipulation</u> <u>Area</u> (LIMA)

The LIMA provides a way of controlling individual lines. Whereas the data area is character oriented, and the primary command line is file-oriented, the LIMA applies only to the one line next to which a command is placed. You can place commands in several LIMA lines at once, and can also have made changes to the text area, and for that matter, also have a normal command sitting on the command line. All of these will be processed when the <ENTER> key is pressed.

To the left of each data line on the screen is either a three-position line or series of dashes. This is the LIMA. Certain commands may be placed in this area, but only one command at a time may be placed next to any line. Errors in the LIMA are ignored and no signal or message is given.

To minimize the likelihood of placing the cursor in the LIMA accidentally, the only way to move it there is to press <SHIFT> and <left-arrow> when the cursor is already at the extreme left margin of the data area. LIMA commands are processed only after the <ENTER> key is pressed, so LIMA commands that scroll off the screen due to windowing or vertical cursor movement will be lost and not processed.

The following are valid LIMA commands:

|	
D	- deletes this line
Dnn	- deletes 'nn' lines ('nn' is 1-99)
I	- inserts a blank line after this line
Inn	- inserts 'nn' lines after this line ('nn' is 1-99)
R	- repeats this line (makes a duplicate of it)
Rnn	- repeats this line 'nn' times
/	- moves the window to make this the "current line"
.A	- marks this line as named point 'A'
.B	- marks this line as named point 'B'
.C	- marks this line as named point 'C'

Named Points

Since you can see only 15 lines of a document at a time, it's sometimes hard to keep track of where you are. It's often desirable to be able to get back to a specific line (or point) in a file, and there will be times when you'll want to move or copy a block of text from one place to another. With EDIT, there are a number of ways to accomplish each of these objectives, and one of those ways is to mark certain lines as points.

EDIT allows you to specify up to three points (.A, .B, .C) at one time. PLEASE don't confuse these with SCRIPT control words, which also start with a period. Named points are for use only in the LIMA, can be used only while editing a file, and are not saved with the file when it's stored on disk. You can re-use these names as often as needed, but you must remember that there are only three of them, and they are A ,B, and C.

The first use of a named point is to place one of the three markers in the LIMA, and subsequently (immediately or much later on in the same EDIT session) use that same name on the Command line. When you do so, the named point becomes the top line on the screen. That makes it easy to find your way back to a place requiring further attention.

If a line at a named point is deleted, that point becomes unassigned. If a point is re-assigned to a second line, the first assignment is replaced by the second one.

The following examples use a mini-screen (I suppose a micro-screen would be more suitable, all things considered, but I didn't buy any that small, so we'll just have to use a wee bit of imagination... [sorry about that].)
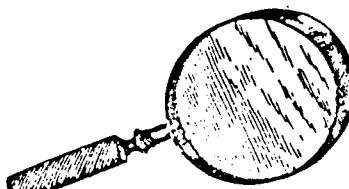
Examples of Using LIMA

```
|E=>                                      |
|-- The horizon of man is unlimited       |
|-- .ce on                                |
|-- A calm sea and an azure sky           |
|-- await us in the by and bye.           |
|-- .ce off                               |
```

Placing a 'D' in the LIMA:

```
|E=>                                      |
|-- The horizon of man is unlimited       |
|-- .ce on                                |
|d- A calm sea and an azure sky           |
|-- await us in the by and bye.           |
|-- .ce off                               |
```

produces this result:

```
|E=>                                      |
|-- The horizon of man is unlimited       |
|-- .ce on                                |
|-- await us in the by and bye.           |
|-- .ce off                               |
|-- .sk                                   |
```

(Note the line is deleted and the text moved up)

using Repeat to create 2 additional copies:

```
|E=>                                      |
|r2- The horizon of man is unlimited      |
|-- .ce on                                |
|-- A calm sea and an azure sky           |
|-- await us in the by and bye.           |
|-- .ce off                               |
```


Franklin press , *Harper's*
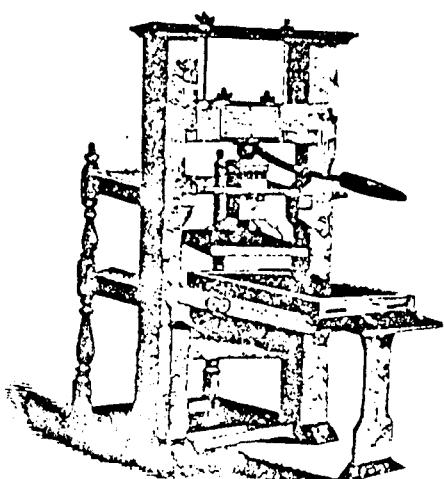
produces this:

```
 _____
|E=>                                    |
|— The horizon of man is unlimited      |
|— The horizon of man is unlimited      |
|— The horizon of man is unlimited      |
|— .ce on                               |
|— A calm sea and an azure sky          |
 _____
```

using the slash to select the new top screen line:

```
 _____
|E=>                                    |
|— The horizon of man is unlimited      |
|— .ce on                               |
|— A calm sea and an azure sky          |
|/— await us in the by and bye.         |
|— .ce off                              |
 _____
```

produces this:

```
 _____
|E=>                                    |
|— await us in the by and bye.          |
|— .ce off                              |
|— .pa                                  |
|— The noble experiment appears to      |
|— have succeeded far longer than       |
 _____
```

naming a point:

```
 _____
|E=>                                    |
|— The horizon of man is unlimited      |
|— .ce on                               |
|.b— A calm sea and an azure sky        |
|— await us in the by and bye.          |
|— .ce off                              |
 _____
```
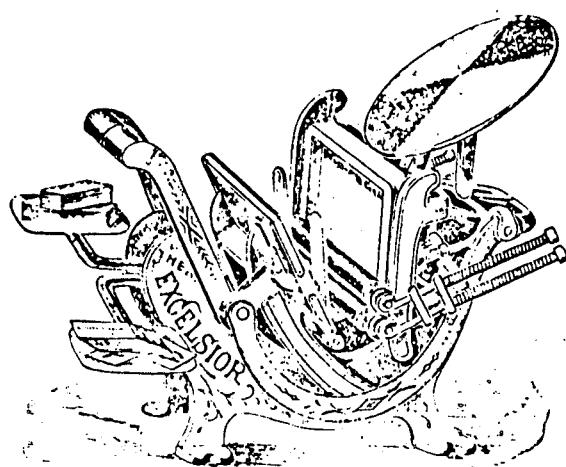

Printing press.

and referring to it subsequently:

```
|E=>    .B                              |
|—— Now, we are in some totally         |
|—— different place in the file,        |
|—— but wish to return to point .B      |
|—— And the command shown above will    |
|—— do just that                        |
```

will result in this:

```
|E=>                                    |
|—— A calm sea and an azure sky         |
|—— await us in the by and bye.         |
|—— .ce off                             |
|—— .pa                                 |
|—— The noble experiment seems to       |
```

Named points may also be used with "MOVE", "COPY", and "DELETE", and are explained within those commands later in this section.

This completes the discussion of the LIMA. The remainder of this section will present the commands recognized by EDIT. These commands must be placed on the primary command line, which is at the very top of the screen, next to the "E=>" prompt, and take effect only after the <ENTER> key is pressed. If necessary, review the "Sequence of Screen Processing" described earlier: commands are processed LAST (except for "WHOOPS").

Old fashioned horn bo

EDIT Commands by Category

The commands of the Editor fall into six categories: screen content, file handling, text changes, searching, status, and indirect. The purpose of each of the first five categories is probably clear. The "indirect" category contains several commands that actually cause commands in the other categories to be executed, and can reduce your typing time.

The commands in each of these categories are shown below. The CAPITALIZED portion of each of these words represents the shortest acceptable abbreviation recognized by EDIT.

SCREEN	FILES	CHANGES	SEARCHES	STATUS	INDIRECT
Backpage	AUtosave	ALter	Find	FRee	X
BOttom	DBlanks	BReak	Locate	Hardcopy	Y
Down	DIr	Change	Zone	LEngth	XY
FLow	ENd	COpy	/	NAme	=
Forward	Getfile	DElete	~		?
GRid	Kill	DString			&
Next	QUit	Insert			
Top	SAve	Join			
Up		Move			
View		Replace			
		String			
		Whoops			

In the remainder of this Section, each of these commands is described, along with an example. The commands are arranged alphabetically, without regard to category.

ALTER

```
ALter      /c1/c2/    <n   <G>>
           n1 n2       *    *
                       1    1
```

(If you're just learning NEWSCRIPT, I suggest that you skip over this command for the time being. It comes first alphabetically, but is hard to understand and rarely used.)

The ALter command is used to change any one character to any one other character. In this sense, it is similar to its more powerful cousin, "CHANGE". However, "ALter" allows you to specify ASCII (Hexadecimal, or whatever you wish to call them) values in the range 1-255, so you can place into a document almost any printer-specific control codes you need. NEWSCRIPT has already defined most of the common control codes in a more readily understood manner (see ".ES" in Section IV), but if you have special requirements that cannot be met in any other way, then "ALter" can be used.

There must be at least two values following the "ALter" command, and they must be bounded by a delimiter, such as a slash. The first value represents a character or ASCII value that already exists someplace in your text. If it's a normal keyboard character (a digit, letter, etc.), you can type it directly. If it's a control code or graphic, you must give its ASCII-decimal equivalent. For example, the largest TRS-80 graphics block is represented as a '191'.

The second value follows the same rules as the first, and represents the value that is to replace the existing value in the text. If the second value is omitted and the first value is found, then the occurrence(s) of that first value will be deleted from the text. If the first value is omitted and the second value is found, then that second value's single ASCII character representation will be placed at the extreme left-hand side of the line(s) within the range of the "ALter" command.

The default vertical range of "ALter" is the "current line", that is, the line at the top of the screen. the default horizontal range of "ALter" is the first occurrence of 'c1' or 'n1' within the currently-defined ZONE of columns. If you want to search and replace across several lines, the third value, 'n' must be specified as the number of lines to be searched. (This is NOT the number of occurrences of the first string.) If you want to search and replace all the occurrences on each line that is, within vertical range, the fourth value, 'G' must be specified as the letter 'G' or as an asterisk. In this case, the third value must also be specified, even if it is '1'. If you want the search/replace to apply to the entire file, position the viewing window to the top of the file ("TOP" command), and then use asterisks as the third and fourth values (see third example, below).

It is also possible to enter most special codes directly, using <SHIFT> <CLEAR>, which was covered just a few pages ago. However, some values will be taken as cursor controls, so "ALter" is needed for these characters. "ALter" is also useful for making multiple changes at one time.

Also see the "Change" comand and <SHIFT> <CLEAR>.

Examples

The first pair of examples show how to enclose a word within square brackets. This can also be done using <CLEAR> <up-arrow> and <SHIFT> <CLEAR> <left-arrow>, but it illustrates the technique.

    alter /?/91/
    alter /+/93/

    If the original text had been: Sing a peon ?sic+ of praise

    the first ALter will produce: Sing a peon [sic+ of praise

    and the second will produce: Sing a peon [sic] of praise

    alter /@/191/ * *

    would change every occurrence of the '@' sign to the large
    graphic block, beginning at the current line and continuing
    through the end of the file that is in memory.

For most purposes, you'll want to use "CHANGE", not "ALTER".

AUTOSAVE

        AUtosave <n>

AUTOSAVE is a feature that allows EDIT to periodically write your file out to disk, automatically. Frequent saving of a text file during an edit session is a way to protect yourself against power, equipment, and program failures. The "SAVE" command can be used to perform this function manually, but if you are in "INPUT" mode, you may forget to issue the "SAVE", which can only be issued when in "EDIT" mode.

If 'n' is specified, then the Editor will perform a "SAVE" for you after every 'n' lines have been added, deleted, replaced, or changed. If AUTOSAVE is issued without an operand ('n' omitted), then the current AUTOSAVE value and the number of changes since the last SAVE, will be displayed:

        AUTOSAVE = 32767 , CURRENT CHANGE COUNT = 17

AUTOSAVE is initially set to 32767, which is the same as being turned off. '32767' was chosen because it is the largest positive integer recognized by TRS-80 BASIC.

Examples

    au
    auto 50

The first example would return the current AUTOSAVE limit (32767 if you haven't set it yet). The second example would set AUTOSAVE so that a SAVE would be performed after every fifty changes, inserts, etc.

BACKPAGE

        Backpage   <n>
                   1

     This is used to move the viewing window back one full screen, that is, fifteen
lines. It is equivalent to "UP 15". However, if 'n' is specified, then the window is
moved up 'n' full screens, that is, 15*n lines. If the top of the file is encountered,
the current line will be set to "*TOF*".

     BACKPAGE, and its counter-part, FORWARDPAGE, are useful in scanning a text
file rapidly by eye.

Examples

        b
        back 3

     The first example moves the window 15 lines towards the top line of the file.
The second example moves the window 45 lines towards the top of the file, or to the
top of the file if it occurs before 45 lines.

BOTTOM

        BOttom

     This moves the viewing window down to the last line of the text file. That last
line becomes the current line. There are no operands with this command.

Example

        bo

BREAK

        BReak   /string/

     This command splits a text line in two. The split begins at "string", which must
be preceded by a delimiter.

     To join two lines together, see the "JOIN" command. Also see the "Text Split"
control key. Note that this command is completely different from the SCRIPT ".BR"
control word, and both differ from pressing the <BREAK> key.

Example

If the current line of text was:

        Fourscore and seven years ago, our fathers brought forth on

and this "BREAK" command was given:

        br /fa

(notice that just enough of "fathers" is needed to make "string" unique). The result of this command would be these two lines:

        Fourscore and seven years ago, our
        fathers brought forth on

CHANGE

            Change   /string1/string2/   <n <G>>
                                            x   x
                                            1   1

     This is probably the single most important command in any editor. It is used to change one string of text into another string. If the new string is the same as the old one, and the range of the command is specified as the entire file, then the Editor will display every occurrence of the specified string.

     If 'string1' is "null" (that is, there's nothing between the first two delimiters), then 'string2' will be placed at the left-hand side of each affected line, and the rest of each of those lines will be moved to the right.

     If 'string2' is "null", then each occurrence of 'string1' will be deleted from the file. If both strings are "null", an error message will be given.

     If any resulting line is longer than screen size (60 characters in length), then the line will be split into two or more lines that fit on the screen. This feature can be disabled through use of the "FLOW" command.

     The default range of "Change" is the first occurrence of "string1" on the current line. If 'n' is specified, then a search for "string1" is made across the next 'n' lines beginning with the current line. If 'G' is specified, then every occurrence of 'string1' within the range of the command will be changed to 'string2'. To specify the entire rest of the file, use the asterisk (*).

     The "Change" command performs its search only within the currently-defined zone (see the "ZONE" command). By default, the zone is all column positions of each line, and remains so unless you use "ZONE" to restrict the search.

     The maximum range of the "Change" command is from the current line through the last line of the file.

     The "SHIFT-CLEAR" control function, followed by almost any two-character hexadecimal value, may be used to convert to or from values not on the normal keyboard.

     If 'string1' and 'string2' are the same, and a global range is specified, "CHANGE" can be used to just display all occurrences of 'string1', without actually changing anything. However, as soon as the last change occurs, EDIT will refresh the screen to the current viewing window.

Also see "ALTER", and the control key functions for deleting, inserting, and overlaying text directly on the screen.

Examples

In each of the examples below, the text initially contains the following lines, the first of which is always the "current line":

        Fourscore and seven years ago, our fathers
        brought forth on this continent a new nation,
        conceived in liberty and dedicated to the
        proposition that all men are created equal.

    c /ago/ago (87 years)/

The first line of text would now read:

        Fourscore and seven years ago (87 years), our fathers

and the remaining lines would be unchanged.

    c.e.??.1g

the first line would then read:

        Fourscor?? and s??v??n y??ars ago, our fath??rs

change//:->/ *

The text file would then look like this:

        :->Fourscore and seven years ago, our fathers
        :->brought forth on this continent a new nation,
        :->conceived in liberty and dedicated to the
        :->proposition that all men are created equal.

And finally, if this were entered:

c/nation/country/

this error message would be displayed on the command line:

        * * ERROR 8 : STRING NOT FOUND * *

Although 'nation' does appear in the text, it is not on the current line, and global search ('n') was not specified.

COPY

```
COPY    .B
  '     .C
```

One or more lines of text may be copied (duplicated) from one point in a text file to another through use of this command. The source line(s) and the target line (the line after which the new lines are to be placed) must have been labelled in advance by Named Points that were placed in their LIMA's.

To copy (duplicate) a single line, give it the name: ".A" (that isn't an example; it must be ".A", not any other letter, although it may be lowercase). Then find the line below which the copy should appear, and name that line ".B". Finally, move to the command line and say:

<center>COPY .B</center>

To copy a block of lines in this way, name the first line ".A", the last line of the block ".B", and place ".C" on the line below which the block should be duplicated. Then, move to the command line and say:

<center>COPY .C</center>

Note that the Named points do not all have to be on the same screen, do not even have to be on the screen when the COPY (or MOVE, or DELETE) is issued, and may be named at any time before COPY is issued.

Also see "MOVE" and the LIMA commands.

Example

This will copy (duplicate) six lines to just below ".C":

```
|E=> CO .C                        |
|.A- Fourscore and seven years ago, |
|—  our fathers brought forth on  |
|—  this continent a new nation,  |
|—  conceived in liberty and      |
|—  dedicated to the proposition  |
|.B- that all men are created equal. |
|—  .sk 2;.ce                     |
|—  THE GETTYSBURG ADDRESS        |
|.C- .sk                          |
|—  text will be copied above here. |
```

DBLANKS

DBlanks   on | off

This tells EDIT whether or not to Delete Blank lines when saving the text file to disk. Since it's easy to create extra blank lines when editing, EDIT assumes that such lines should be deleted, not saved. If you set "DBlanks OFF", then blank lines are saved (as lines of zero length). However, the next time such a file is edited, EDIT has no way of knowing that you wanted these lines saved, and will delete them unless you set "DBlanks OFF" every time you edit the file. If you make the first character of a blank line a X'80' (SHIFT-CLEAR then 80), the line will be kept permanently, regardless of the DBlanks setting.

Example

    db off
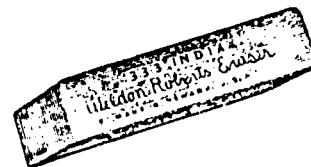
DELETE

DElete   <n>
           *
           1
          .B

This is used to remove one or more lines of text from a file. Deletion begins with the current line. If a range is given, that number of lines will be deleted, unless the end-of-file is encountered, in which case, all lines from the current line through the last line of the file will be deleted. The asterisk (*) specifies that all lines from the current one through EOF should be deleted. If no quantity is given, just the current line is deleted.

If the named point ".B" is given and points ".A" and ".B" are currently assigned to lines through the LIMA, and if ".B" occurs below ".A", then the entire range of lines from ".A" through ".B", including both of those lines, will be deleted.

Also see "DSTRING", "REPLACE", "COPY", <CLEAR-d>, and the LIMA.

Examples

    delete 5
    del
    de *

These would delete 5, 1, and all remaining lines, respectively. Lines above the current line would never be affected.

DIR

                 DIr   < <:>n >

     This displays the directory of drive 'n'. If 'n' is omitted, the system default
(usually 0) is taken. The colon may be included or omitted at your option.

     This command is **NOT** supported under Model I TRSDOS or NEWDOS. It <u>is</u>
supported on the Model I under NEWDOS/80, DOSPLUS, and LDOS. It is also
supported on the Model III. If used under DOSPLUS' "TBASIC", attempts to read the
Directory of a non-DOSPLUS diskette may be catastrophic: DOSPLUS (as of this
writing) may shift back to DOS level and the in-memory copy of the file you've been
editing will be lost.

     Once displayed, the directory remains on the screen until any key is depressed, at
which time the current text is re-displayed and control reverts to EDIT mode.

     Within the limitations stated above, any directory may also be displayed during
the initialization of EDIT <u>or</u> SCRIPT: whenever you're asked to enter a 'fileid', you
may reply with a question mark and the number of the drive whose directory you wish
to examine. After the directory is displayed, you will again be asked to enter a
'fileid' or to accept the default (if there is one).

<u>Examples</u>

        dir
        dir 1
        di :1
        (during EDIT or SCRIPT file I.D. input): ?1

DOWN

                 Down   <n>
                        <u>1</u>

     This moves the viewing window down (towards the bottom of the file) 'n' lines, or
one line if no quantity is given. The 'n'th line below the old current line becomes the
new current line. If 'n' is greater than the remaining number of lines in the file, the
"*EOF*" marker becomes the current line.

     Also see: NEXT, and LIMA "/"

<u>Examples</u>

        down 4
        d

DSTRING

> DString /string/

This is much like the "DELETE" command, but instead of specifying a number of lines to be deleted, a string, marking the end of the range of deletion, is used. The string may be anyplace in any line that follows the current line. If the string is not found, no deletions are made. If the string is found, then all lines, from the current line up to but not including the line containing 'string', are deleted.

The command is useful when a large number of lines must be deleted and counting the exact number would be a chore. It is normal to insert a dummy line, such as "xxxxx" following the last line to be deleted, then re-positioning to the first line to be deleted, before issuing this command. The dummy line should be deleted, via "DELETE", afterwards.

Example

If the lines beginning at the current line were:

> It is an Ancient Mariner,
> and he stoppeth one of three;
> By thy long grey beard and glistening eye,
> Now wherefore stoppeth thou me?

and the command used was:

> ds/grey

Then two lines would be deleted, and the current line would be:

> By thy long grey beard and glistening eye,

END

> ENd <fileid>

This usually is the last command entered during the editing of a document. It causes the current in-memory copy of the text to be written out to diskette, and then asks whether you want the file passed to SCRIPT.

If "fileid" is omitted, then the name you originally supplied when starting the Editor is used. If "fileid" is supplied, the file is saved under that name.

If any I/O error occur during the saving of the file (DISK FULL, WRITE PROTECTED, or PARITY ERROR), then an error message is given and instead of chaining to SCRIPT, control reverts to EDIT mode. After correcting the error, you should re-issue the "END" command. Since any disk error may occur, it is not feasible to describe recovery procedures here. The most common errors are likely to be "DISK IS FULL" (NEWDOS/80 may say "DIRECTORY WRITE ERROR") or "WRITE-PROTECTED DISKETTE". If you do get any of these errors, see "Avoiding Lost Files" in SECTION VI. The reference in the Index is under "errors".

One common cause of "DISK IS WRITE PROTECTED" is a write-protect tab on drive 0, when drive 0 contains a file of the same name as the one you've been editing. This easily can happen with "EDIT1/EX". The solution is to either remove the write-protect tab, change the name or drive number of what you're editing, or, best of all, to not place any text files on drive 0. Of course, that's not possible on a one-drive system. The point here is that the error can be corrected!

Also see "SAVE", "AUTOSAVE", and "QUIT". In particular, the distinctions among these similar commands are summarized under "SAVE".

Examples

The most normal form is simply: END

If you change your mind about what to call the file, or if you have been editing an existing file and do not want to update/replace it, you would supply a fileid:

            end newcopy/let

This would write the in-memory text to disk under the name "NEWCOPY/LET". For purposes of this example, it's assumed that the name chosen would create a new file, rather than replace an existing one.

FIND

            FInd character-mask

This performs a column-dependent search, beginning with the line after the current one. One blank is assumed after the command itself, and everything thereafter is treated as data. If a column in 'character-mask' is non-blank, it is compared against the corresponding column in the line being scanned. If a column in the mask is blank, no compare is made.

Several characters may be contained in the 'mask'. It is a useful way to search for scattered data, but its primary use is to perform a rapid scan along the left-hand side of the document. Used in this way, it is faster than "LOCATE", which scans for a string throughout each line. "FIND" is not restricted by the presence of "ZONE" values.

Kerosene lamp.
Sears Catalogue

Example

If the text starting at the current line was:

        We are met here on a great battlefield
        of that war. We have come to dedicate
        a portion of that field as a final
        resting-place for the brave men who

and this command was entered:

        fi a portion

then the third line would become the current line.

FLOW

        FLow   <ON> | <OFF>

When FLOW is ON, long lines modified by "CHANGE" are split into multiple lines, each of which can fit on one line of the screen. The command has no effect on lines that are typed directly into the data area, as they are flowed automatically. It only affects lines modified by the "CHANGE" command.

If you have constructed a wide table by moving the Viewing window back and forth, you should be sure to turn Flow Off if the Change command is used in the vicinity of that table. Otherwise, the table may be inadvertantly broken up into multiple lines.

Example

    flow off

FORWARDPAGE

        Forward   <n>

This moves the viewing window down in 15-line (one screen page) steps. Like "BACKPAGE", it is useful when rapidly reviewing your text. If 'n' is specified, then the window moves 15*n lines down towards the end of the file.

Also see "Down".

Example

    f 5

This would move the viewing window 95 lines (five full screens) towards the end of the file. If there were fewer than 95 lines left, it would move the window to the artificial "*EOF*" line that follows the last real line.

FREE

FRee   <n | 15>

This is used to determine how much space remains available for further editing. If 'n' is specified, it is used as a new "WARNING" threshold. If omitted, the default is "15", or the last specified value of 'n'.

When this command is issued, the following message is displayed on the command line:

LINES USED = n, LINES LEFT = n, CHARS LEFT = n, WARN = n

The first value is the number of text lines currently in your file. The second value indicates how many more lines you may add to the file, and the third value indicates how much string space is left for these additions. When "CHARS LEFT" is less than 300, no more additions may be made to the file. When "LINES LEFT" reaches zero, the next request for additional space results in dynamic allocation of fifty more buffers (if that much space is still available). If your document contains mostly short lines (15-20 characters maximum per line), EDIT may run out of pre-defined buffers even though there is space available for them. This would occur when the "MAX" number of buffers, displayed during initialization, is reached.

The "WARN" value indicates the maximum number of additional lines that can be made available before you run out of space. When you reach that small number, a warning message should be given:

* 15 LINES LEFT

This message gives you some advance notice of the imminent lack of additional space. When it occurs, you should stop editing, possibly place an ".AP filespec" as the very bottom line of the file, and then "END" the editing of this particular file. After the "SAVE" function of "END" completes, you can choose the "A" (append) option of EDIT to continue editing in another, appended file.

Examples

        free
        free 50

GETFILE

GEtfile fileid < /string1/<string2/>>

This is used to bring some or all of the contents of another file into the in-memory file being edited. The new text is inserted just below the current line.

'fileid' is required. If the file is not found, an error message is given, and you may issue another EDIT command (or try again, with the correct file name).

If only the 'fileid' is given, the entire file, or at least, as much of it as will fit into memory, is read in.

If 'string1' (surrounded by delimiters) is specified, then each line of 'fileid' is scanned for an occurrence of 'string1'. No lines from 'fileid' are inserted until 'string1' is found, and all lines starting with the first one containing 'string1', are then inserted into the current in-memory file.

If 'string2' is also specified, then insertion of lines ends after the first line in 'fileid', after the line containing 'string1', that is found to contain 'string2'. To put it another way, 'string1' and 'string2' are used to identify a range of lines to be copied from a disk file into memory.

'string1' and/or 'string2' may be "null". If 'string1' is null and 'string2' contains data, then copying begins with the first line of the file and continues through and including the line containing 'string2'.

If 'string1' is not found, nothing is copied. If 'string2' is not found, copying includes the entire file from 'string1' through end-of-file.

Examples

To read an entire file: get logo/let

To read part of a file: get standard/con @We will be@as stated@

Notes:

1. The slash (/) may be used as the delimiter even though it also occurs in the Fileid.

2. GETFILE is somewhat slower than reading a file during the Editor's initialization. Therefore, if you want to switch from editing one file to editing another one, do not do this:

```
        save file1
        top
        delete *
        get file2
```

That would work, but it would be faster to do this:

```
        end
        <BREAK>      -- hit the <BREAK> key
        run
        file2        --- reply to the first question asked by EDIT
```

GRID

        GRid   <ON> | <OFF>

   This controls the inclusion of a column-marking grid on the screen. It is useful when dealing with tabular data.

   When the grid is shown, it always appears at the bottom of the screen. The grid is numbered by tens from 10 to 250, and every fifth position is marked by a colon (:). The numbers appear just to the left of every other colon. If "VIEW" is used to move the window to the right, then the grid markings will show you which columns are currently in view (so will the "VIEW" command).

Example

      grid

will cause the following to appear on the bottom of the screen:

            ....:..10:....:..20:....:..30:....:..40:....:..50:....:..60:

HARDCOPY

        Hardcopy   <n>
                   <* <*>>
                   15

   This causes some or all of the file being edited to be printed. By default, fifteen lines, beginning with the current line, are printed. If 'n' is specified, then 'n' lines, beginning with the current line, are printed. If '*' is specified, then all lines from the current line to end-of-file are printed. If '**' is specified, then the entire file is printed.

   This is NOT a substitute for SCRIPT! No formatting occurs, and each line is printed just as-is. This differs from a screen- print in that the entire content of each line is included, not just the 60 characters currently being displayed.

Example

     h 20

      That will cause the next twenty lines to be printed, as-is.

INSERT

Insert   <string>

This is used to insert one line of text directly after the current line. 'string' is the line of text you wish to add. If 'string' is omitted, a screen-full of blank lines is generated to give you room for bulk entry of additional text. The maximum length of 'string' is 58 bytes, since the command line cannot be continued into the data area of the screen.

A convenient way to perform bulk entry is to place "&I" in the command line and press <ENTER>. Then, text can be typed as needed on the screen. When the bottom of the blank area is reached, a slash ("/") may be placed in the LIMA of the last line you've just typed, and the <ENTER> key pressed. This will copy the screen contents to permanent internal memory (not to disk), move that last line to the top of the screen, and generate another group of blank lines in which your typing may continue.

Example

     i This line is being added for clarity

JOIN

Join   <B>

This command combines the current line and the one that follows it. The result is a single line, with the contents of the original current line being first. If the optional "B" is specified, then one blank space is placed between the two portions of the resulting line. The rule is: "No 'B', no blank."

JOIN can create lines that are longer than the screen width of 60 characters. The longest resulting line can be 255, if necessary. Of course, only 60 of these will be visible in the window at any one time. If you want to see the rest, you may use "VIEW" to move the window left or right; or if you haven't been setting up a wide line on purpose, you can split it up into several shorter lines by changing a blank to a blank (c/ / /).

Also see "BREAK" and the LIMA "text split".

Example

If the current and next lines are:

          aaaaaaaaaaaaa
          bbbbbb

then this command: j

would produce: aaaaaaaaaaaaabbbbbb

But, this command: j b

would produce: aaaaaaaaaaaaa bbbbbb

KILL

        Kill   fileid

    This is equivalent to the DOS "KILL" command, and is used to delete a file from an on-line diskette. The quotation marks required by the "KILL" command of BASIC MUST NOT be supplied in this command. If the 'fileid' is omitted or the specified file is not found, an error message is given, and you remain in "EDIT" mode.

Example

    kill oldcopy/let

LENGTH

        LEngth

    This is used to find the character-length of the current line. The value is displayed on the command line, and represents the entire line length, which may include text that is outside the current viewing widow. No operands are used.

Example

    len

LOCATE

        Locate < /string/ >
        Locate < -string- >
        /<string/>
        -<string->

    "LOCATE" is used to search for a given string. The search begins at the line after the current line and continues through the end of the file, or until a line containing "string" is found. If the current line is the "*EOF*" marker, the search begins at the top of the file.

    The range of the scan is limited to the currently-defined ZONE, which defaults to all columns unless you have used the "ZONE" command.

    If the string is NOT found, the current line is not changed, and an error message is displayed.

    "LOCATE" has four forms, as shown in the command format. In the first form, the delimiter (shown as a slash or dash above) may be any character that is not within "string", and the search is performed as just described. Although a closing delimiter is shown in the command definition, it may be omitted.

Because the slash has been used in examples such as these since 1968, it has become something of a default standard, even though any other character is equally acceptable. In order to reduce the typing requirements, the slash itself is recognized as being equivalent to the full "LOCATE" command, and is the third form shown above. Again, the search is done as described.

If the delimiter is a dash (the minus sign), the command is interpreted as a "LOCATE-NOT" function. In this case, the first line beyond the current line that does NOT contain "string" within the current ZONE will satisfy the search. This is the second form shown above. The dash itself is recognized as being equivalent to a "LOCATE-NOT".

If 'string' is omitted, the string used in the most recent previous use of "LOCATE" is re-used. This can reduce typing time.

Also see the "FIND" and "ZONE" commands.

Examples

In each of the following examples, the text, beginning at the current line, is:

            Alone, alone, all all alone,
            Alone on a wide, wide sea,
            And never the Saints took pity on,
            My soul in agony.

```
loc/the/      <-- sets the third line as current
/aints/       <-- also sets the third line as current
1-Alone       <-- also sets the third line as current
-o            <-- fails to find a match ('o' is in all lines)
```

MOVE

            Move   .B
                    .C

This is used to move one or more lines of text to another place in the file being edited. The block of text is deleted from its original location and inserted at the new location.

Before issuing the "MOVE", the source line(s) and the line below which the text is to be moved must be identified as Named Points in the LIMA. The first (or only) source line must be ".A" (only ".A", not ".B", or any other name). If only one line is to be moved, the target line must be named ".B". If several lines are to be moved as a block, the last line in the block must be named ".B" and the target line must be named ".C".

Naming of points doesn't have to be done all at once, and the points don't have to be on the screen when the "MOVE" command is finally issued. However, the two or three points in question must have been named and still be valid by the time the "MOVE" command is issued.

The example shown under "COPY" also applies to "MOVE". The only difference is that "MOVE" deletes the original block of lines, whereas "COPY" does not.

Also see "COPY", "GETFILE", and the LIMA Named Points.

Examples

```
        move .b
               (move line marked ".A" to just after ".B")
        move .c
               (move lines ".A" through ".B" to just after ".C")
```

NAME

```
        NAme     <fileid>
```

This command is used to display or change the name (filespec or fileid) of the current file on the command line. If nothing follows the command, the current file name is displayed. Otherwise, the following string becomes the permanent file I.D. No error checking is done at this time to ensure a valid name. The name must not be in quotes.

Examples

```
    name

    name part4/fil
```

NEXT

```
        Next    <n>
                1
```

This is a synonym for the "DOWN" command, and moves the viewing window down '1' or 'n' lines (towards end-of-file).

Example

```
    n3
```

QUIT

QUit

This is used to terminate editing without saving the in-memory copy of the file being edited. If you want to start editing a different file, just hit <BREAK> and then type <RUN>, or else reply "N" when asked if you want to pass the file to SCRIPT.

If you don't want to save any changes (or haven't made any), but do want to pass the file to SCRIPT, you can use this "QUIT" command, then hit <ENTER> when asked about going to SCRIPT. This might occur when you were just reviewing the contents of a document to make sure it was ready for printing.

If you've been entering a new text file and have never issued a successful "SAVE" or "END", then issuing a "QUIT" will cause the text to be lost permanently (or until you type it all back in again), since it never had been written to disk. Remember, "QUIT" means just what it says: "stop what you're doing, computer, and don't save the in-memory copy of the file." However, it doesn't erase or kill anything that's already been written out to disk.

Please note that if you've made changes and issued a "SAVE" or "END", those changes have already been permanently stored on disk, so "QUIT" can't magically cancel them out. If you want to be able to back-out such changes, you will have to keep a backup copy of each file or disk.

To reduce the likelihood of losing any changes you may have made, the Editor will check to see if you did make any changes that were not saved. If not, it will allow you to transfer to SCRIPT. However, if it finds some changes, it will ask:

FILE CONTAINS n CHANGES. QUIT OR END?

Reply "END" to save the changed file, "QUIT" if you really mean it, or just hit <ENTER> to continue editing the current file.

This command does not expect or use any operands.

Example

qu

REPLACE

Replace   string

The line of data following the "REPLACE" command replaces the current line, whose contents is discarded. To change a portion of the line, use "CHANGE" or just overlay/insert material by moving the cursor to the appropriate position on the screen and typing on it. 'string' cannot exceed 58 characters, since the command line cannot overflow into the data area.

Example

r This line looks better when phrased this way.

SAVE

SAve   <fileid>

The current in-memory copy of the file being edited is written to disk, after which "EDIT" mode continues. If 'fileid' is omitted, then the original file I.D. used during Edit initialization is used. In this case, any pre-existing copy of the file will be replaced on disk. If 'fileid' is supplied, the file is saved under that name.

"SAVE" is a way of reducing your rework time in the event of a system failure. EDIT also has an automatic version of this command: see "AUTOSAVE". If you want to save and terminate editing, use "END". If you don't want to save the current image of the file, use "QUIT".

Summary of differences:

AUTOSAVE tells EDIT to save a copy periodically;
END tells EDIT to save a copy and switch to SCRIPT;
QUIT tells EDIT to switch to SCRIPT without saving;
SAVE tells EDIT to save a copy and not switch.

CAUTION:

Never press the <BREAK> key while a file is being saved.

Examples

    save
    sa newfile/doc

STRING

STring   n,c

This creates a string, 'n' characters long, of the character 'c'. It is functionally identical to BASIC's "STRING$" statement, but 'c' can only be a single character, not a multi-digit number. The resulting string is placed after the current line, and becomes the new current line. This command is useful when creating separator lines of asterisks, dashes, etc. Be sure to include the comma between 'n' and 'c'.

Example
    string 15,*

will produce this: ***************

TOP

>   Top

The viewing window is set to the first (top) line of the file. "Top" is the opposite of "BOttom".

Example

>   top

UP

>   Up   <n>
>   1

The viewing window is moved up (towards the beginning of the text file) 'n' lines, or '1' line, if no value is given. If this movement encounters the top of the file, the "*TOF*" marker becomes the current line. "UP" is the opposite of "DOWN".

Also see: DOWN, NEXT, FORWARD, BACK, cursor movement with up-arrow and down-arrow, and the "/" and "point" functions of LIMA.

Examples

>   u 5
>   up
>   u7
>   u
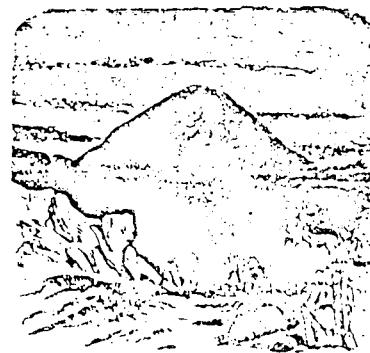
VIEW

>   View   <n1 <n2>>
>   +n
>   -n
>   *

This moves the viewing window to the left or the right. At most 60 characters of any input line may be seen at once, so longer input lines may be examined by shifting the window around with "VIEW".

If no operands are provided, "VIEW" displays the current starting and ending columns being displayed. These default to '1,60'. To reset to the default, use the asterisk: "V*".

If one number is given, then columns 1-n are displayed, but if 'n' is greater than 60, only the first 60 columns are displayed. If two numbers are given, then columns 'n1' through 'n2' are displayed, again with the provision that not more than 60 text columns will be shown from each line.

   The window can be shifted by a relative number of columns if a plus (right-shift) or minus (left-shift) sign precedes a single number. EDIT will not allow the window to go to the left of column 1, nor to the right of column 255.

NOTE: If portions of a line are not visible in the viewing window (lines longer than 60 characters, or "VIEW" set to start past column 1), any changes made by direct overlay/insert/delete will affect only the portion of text currently on the screen. For instance, if you use CLEAR-D (character deletion via the control key), text on the screen will shift to the left, but any additional text that is too far to the right to be visible will NOT be shifted onto the screen. Instead, additional blanks will be generated on the right side of the screen, and these will be placed back into the original long line when that line is updated by EDIT. This feature is intended to help you maintain column-alignment in tables.

   Also see: ZONE, LIMA, cursor movement, GRID.

Examples for VIEW

In these examples, only a "mini-window" is shown, for the purpose of showing what "VIEW" does. The "mini-window" is 10 characters wide, and the text in the file is:

```
....:...10:....:..20:
Fourscore and seven
years ago, our
fathers
brought forth
```

Initially, the window will show:

```
....:...10:
Fourscore
years ago,
fathers
brought fo
```

If this were issued: view 5,14
    (which asks for 10 columns)
then this will be displayed:

```
:..10:....
score and
s ago, our
ers
ght forth
```

If this were issued next: v-1

then the window would contain:

```
.:..10:...
rscore and
rs ago, ou
hers
ught forth
```

To get back to the original window: v*

    By the way, you can display fewer than 60 characters at a time. Also, when the starting column for "VIEW" is not column 1, it may be helpful to turn on the "GRID".

WHOOPS

Whoops

This does pretty much what you would hope it would do: it throws away any recent mistakes you've made and gives you a chance to try again.

If the "Whoops" command ("w" is sufficient) is given, then any changes still on the screen and whose lines are marked by the ">" change flag are discarded, and the screen is refreshed to its latest known condition. Anything that has scrolled off the screen, and anything changed prior to the previous pressing of <ENTER>, is unaffected by "Whoops" (such previous changes have been incorporated into the permanent in-memory copy of the document, although they will not have been written out to disk unless you had also issued "SAVE" or "END").

If you think you've really messed up the document, the best thing to do is "QUIT" (or <BREAK>), and start editing the file from scratch.

Example

        whoops

X

        X   <edit-command>
            <n>
            1

"X" and "Y" are indirect commands.   That is, you can assign another command, complete with operands, to "X" (and/or to "Y"). Thereafter, if you type "X", the command assigned to it will be executed. The value of this lies in the ability of "X" to be followed by a number, representing the number of times the command is to be executed; and in that you can use "X" much later on, as often as you need to, without having to type in the command and operands it is set to represent.

The values assigned to "X" (and/or "Y") can be changed whenever you wish, but cannot be set back to "null" until the Editor is re-started via the "RUN" command.

Also see "XY".

Examples
        x c/e/??
        x
        d7
        x2

In the above sequence, "X" was set to represent a "CHANGE" command. It was used, the viewing window was moved down seven lines, and "X" was used again. This time, however, the "CHANGE" was performed twice, since "X2" was specified.

"X" and "Y" are both used in exactly the same way. Having two indirect commands just gives you more flexibility, since both may be active at once, representing different commands or different variations on the same command.

"X", "Y", and "XY" are not simple, beginner-type commands, so a little experimentation with them will help clarify their uses.

Y

The "Y" command is identical in function to "X", which is described directly above. They are not synonyms for each other, however, since you can assign different command values to each and be using them both at any time.

XY

                    XY   <n>
                        1

"XY" is not quite the same as "X" or "Y". You cannot assign a value to "XY", but you can invoke it with a numeric repeat value. When used, "XY" causes the command assigned to "X" to be executed, and the command assigned to "Y" to be executed. Both indirect commands are executed in sequence ("X" is always executed first, and there is no "YX" command). If 'n' is specified, then the pair of commands represented by "X" and "Y" will be executed 'n' times.

If "X" or "Y" is not assigned a command, error messages are likely to result. Both must be set before "XY" is used.

Example

Suppose you have numbered a series of points in a letter, and now wish to move the numbers one space to the right (so that the single-digit numbers will line up with the two-digit numbers that follow them). Further suppose that the numbers are followed by a space, a dash, and another space, as in:

                    1 - Before turning power on, remove base plate.

If we assume that the three characters ' - ' only occur in this position, then "XY" can be used as follows:

                    x / - /
                    y c// /
                    top
                    fi 1 -
                    up
                    xy 9

These six commands perform the following actions:

1. "X" is set to represent a "LOCATE" command for the three characters in question.
2. "Y" is set to change the null string (that always precedes a line) to a single blank.
3. The viewing window is moved to the start of the file.
4. A "FIND" command is issued (for speed) to get to the first numbered paragraph. This step could have been omitted, but step 6 probably would have taken longer to complete.
5. Move back up one line, since "LOCATE" begins on the line following the current line.
6. "XY" is invoked with nine repetitions. The "X" portion searches for the dash, while the "Y" portion makes the needed change.

ZONE

               Zone   <n1 <n2>>
                      <*>

     This command is used to limit the horizontal search range of the "LOCATE", "CHANGE", and "ALTER" commands. By default, all columns are searched, but if you wish to confine a search, or a set of changes, within a range of columns, you can use ZONE to tell the Editor what that range is.

If no values are given, then the current ZONE (default=1,255) is displayed on the command line. If one value is given, the ZONE is assumed to be from column 1 through column 'n1'. If two values are given, the ZONE is column 'n1' through column 'n2', inclusive. To reset the ZONE to the default value, you can use the asterisk (*).

     "FIND", "MOVE", "COPY", "BREAK", and "GETFILE" do not use "ZONE". "LOCATE" and "CHANGE" do use it.

Examples

               z 1 25
               zone 15
               Z *
               z

SPECIAL EDIT SYMBOLS: = / — ? &

     These special symbols are recognized as commands by EDIT, and provide some of its nicest and most convenient features.

     The "equals sign" (=) tells the Editor to repeat the previous command.

     The slash (/) is shorthand for "LOCATE", and is described under that command. It is different from the LIMA "/".

The minus sign (-) is shorthand for "LOCATE-NOT", and is described under the "LOCATE" command.

The question mark (?) causes the last command to re-appear on the command line. You can look at it (to see what went wrong, usually), edit it via overlay / insert / delete, or blank it out or retype it if you don't want it any more. After you've finished fixing it up, you can hit <ENTER>, and the command in its current form will be executed. This is nice when you've typed in an incorrect "LOCATE" or "CHANGE": instead of having to re-type the whole thing correctly, you can just pull it back onto the screen and make the corrections selectively. If you realize that you issued the wrong Locate or global Change, you can press <ENTER> to interrupt them, and then issue "?" to recall and fix them up. However, any changes that occurred before then will remain in effect.

The ampersand ("&") can precede any other command. There may be NO blanks between the "&" and the command. The effect of this is to cause that command to be executed, and to then re-appear on the command line, along with the "&". Therefore, you could just hit <ENTER> again and have the command repeat ... and repeat ... and ... and ...

The ampersand remains in effect until blanked out. It is useful to use this feature in such things as scanning the file:

                              &F

would let you page forward 15 lines at a time by just hitting <ENTER> over and over.

THIS COMPLETES THE DESCRIPTION OF EDIT COMMANDS.
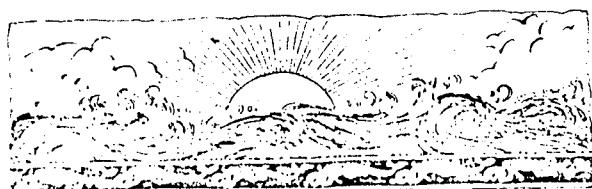
SECTION IV - SCRIPT CONTROL WORDS

SCRIPT Control Words by Category

The control words of the Text Formatter fall into seven categories: margins, spacing, formatting, page layout, files, titles, and miscellaneous:

MARGINS	SPACING	FORMATTING	LAYOUT	FILES	TITLES	MISC.
AO	BR	CE	BF	AP	BT	CM
BM	CF	CO	HI	IM	PN	DA
FM	DS	FO	IN	IX	FS	ES
FS	PA	JU	LL	RD	TT	TR
HM	SH		OF	ST		US
HS	SK		PL	TC		
LS	SP		PP			
TM	SS		SO			
			TB			

The remainder of this Section describes each of these control words, along with examples. The commands are arranged alphabetically, without regard to category.

Unlike EDIT commands, SCRIPT control words may neither be abbreviated nor entered as full words: they must always be two characters long and preceded by a period. Any line containing a control word must begin with a control word, and a control word cannot follow any other text, although it can follow certain other control words. If followed by alphameric operands (a mix of letters and numbers), at least one space must follow the control word. If followed by a numeric value, the space(s) may be omitted. Multiple numeric values must be separated by commas, and if several control words are placed on one line, they must be separated by semi-colons. Any mixture of upper- and lower-case is acceptable.

These are valid control words and combinations:

```
.sk                     - skip one line
.sk 2                   - skip two lines
.fo off                 - format off: no justification or text flow
.ce2                    - center the next two lines of text
.cp7                    - eject page if fewer than 7 lines left
.pa                     - start a new page
.pp                     - start a new paragraph
.tt /left/center/right/          - define a top title
.fo off;.sk3;.cp7;.ce            - several may be on one line
```

These are invalid ways of specifying control words:

```
unless you call. .pp        - cannot follow text
.fooff                      - requires a blank after 'fo'
.pa                         - didn't start in column 1
```

"Escape" Sequences

   The special features of many modern printers are supported by SCRIPT. Examples of this are underlining, super- and sub-scripts, the proportional font of the Line Printer IV, and the Emphasized modes of the Epson MX-80. Since these features often need to be turned on and off or used in combinations on the same line, a special set of two-character values has been defined to allow you to control: double-width characters, underlining, and half-spacing, which allows sub-scripts and super-scripts. These sequences are covered under the ".ES" control word later in this section.

Mini-Edit Mode

   SCRIPT allows you to make minor, _temporary_ changes to a document while it is being printed. These changes are _not_ saved to disk, since the feature is intended only for last-minute touch-up work. Mini-edit is NOT a substitute for EDIT.

   Mini-edit can be activated in any of three ways:

   1. By specifying "ED" as an option when SCRIPT asks you to enter options during initialization;

   2. By hitting the <ENTER> key during SCRIPT processing, and replying "E" when asked whether to continue processing.

   3. By replying "C" at End-of-Job, (EOJ) when asked whether or not to eject the page. This lets you add a "P.S." or something similar to the bottom of a letter. It is not intended for major additions to a book containing a table of contents, nor will it add text to form letters or multiple copies of a document being printed.

   Mini-edit can be deactivated by replying "T" to any "*EDIT*" prompt issued while in mini-edit.

Mini-Edit Commands

   When mini-edit is active, each line read from the current file will be displayed, and then this prompt will be given:

                              ?*EDIT*?

   There are _five_ valid replies to this prompt. Any other reply will ask you to re-issue the reply with a valid command (upper or lower case):

   1. <ENTER>. Just pressing the <ENTER> key causes the displayed line to be processed without change.

   2. D. This deletes the line (ignores it).

3. <u>I text</u>. The line of 'text' following "I" is inserted into the document <u>BEFORE</u> the line displayed on the screen, and then the displayed line is displayed again. You can insert as many lines as you wish in this way. Note that insertion <u>before</u> the displayed line is different from the methods of inserting text used in EDIT.

4. <u>R text</u>. The line of 'text' following "R" replaces the line displayed on the screen.

5. <u>T</u>. This <u>T</u>erminates mini-edit mode and causes processing to continue in non-interactive mode. You can hit <ENTER> later on to get the "DO YOU WANT TO CONTINUE PROCESSING" message, if you want to get back into mini-edit.

.AD

```
.AD   <n>
       5
```

"ADjust" controls the width of the left margin. 'n' is expressed in tenths of an inch and defaults to '5' (1/2 inch). The printer itself may add as much as 1/2 more due to its inability to print at the left edge of the paper, or due to how the tractors have positioned the paper relative to the left-most position of the print head.

If ADjust is used with Titles, it must precede the definitions of those titles.

Example

```
.AD 10
```

defines a one-inch left margin.

.AP

```
.AP   fileid
```

"APpend" is used to chain from one text file to another. The current file is closed, and the appended file becomes the current file. This allows you to create documents of unlimited length, since each small text file can end with ".AP fileid", effectively extending the size of the file to whatever length is needed.

'fileid' must NOT be enclosed in quotes, and must be a valid TRSDOS form. If the file is not found, an error message is given and end-of-job is raised.

If the file to be used next (appended) is not on a currently on-line diskette, you should use the ".STop" control word just before the ".APpend" (see ".ST"). Use of both of these permits you to create very long documents. For example, this manual pretty much fills five single-density 40-track diskettes. (And many of you, including my Treasurer, wish it was smaller but still explained everything better than it does now. If wishes were fishes we'd never want for food.)

If you want to use EDIT's automatic-chaining option ("A"), then ".AP" must be the first and only control word on a line. Also see ".IMbed", ."STop", "Table of Contents", and "Index".

Examples

```
.ap part2/doc

.st MOUNT DISKETTE # 15, THEN PRESS ENTER
.ap part99/doc
```

The first example chains immediately to another file. The second example allows the operator to change diskettes before the search for the next file is made.

.BF

.BF   737 | 10 | 12 | 16

"Begin Font" allows you to switch back and forth among the character sets supported by many printers. Actually, what changes is the pitch, or number of printed characters per inch. However, since the dot-matrix printers generally use different character shapes for the different pitches, the term "font" tends to be used in the printer manuals, and that terminology is continued here.

Proportional spacing is NEWSCRIPT's default on the Line Printer IV, Centronics 737 / 739, Atari 825, and Radio Shack Daisy Wheel II. It is specified as "737", regardless of which of these printers you are using. Note that other printers allow proportional spacing, but their method for doing so is entirely different from the family of printers covered by "737".

10 CPI (characters per inch) is the default for other printers. These other pitches are "monospace", that is, they are not proportional-spacing fonts. "10" specifies 10 character per inch (10 CPI), 12 specifies 12 CPI, and "16" specifies the range of 16-17 CPI. If you use "737", but are not using one of the printers in that family, SCRIPT will print in 10 CPI monospace.

Right justification is supported for all fonts, whether in single or double-width character style. Single and double width characters may be inter-mixed on a line without affecting right-justification in most cases.

Some printers allow inter-mixing of different fonts on the same printed line, while other printers do not. (The 737 does for the most part, the MX-80 does not, etc.) SCRIPT does not create a "control break" on a font change, so if your printer can handle it, you can print different character-spacings on the same line. However, SCRIPT will not right-justify such lines: justification will be based on the pitch (font) that is in effect at the end of the printed line. In other words, you probably should avoid inter-mixing fonts and pitches in this way. However, it is perfectly alright to intermix single and double-widths, and also italics on such printers as the MX-80 with GRAFTRAX.

Double-width characters are specified by the "escape sequences" defined within SCRIPT. There is no control word for double-width.

Most control words cause what is called a "break", that is, all text preceding the control word is formatted and printed before the control word is executed. ".BF" is one of the few exceptions to this (".US" is another), since inter-mixing of fonts is sometimes desired. If you want to start a new print line for the text following this control word, also use the ".BR" control word. NOTE: Since SCRIPT supports multiple control words on one line, ".BF" may follow most other control words. However, it cannot be followed by any control words on the same line.

Examples

```
.bf10
.br;.bf 737
.bf 16
```

However, this is invalid: .bf10;.in5

since nothing may follow a ".BF n".

.BM

            .BM   <n> | 6

This defines the size of the "Bottom Margin". The bottom margin is the space between the last line of the body of a document and the bottom edge of the page.

The bottom title(s), if any, will be printed within the bottom margin, and the number of blank lines (if any) specified by the Footing Margin will be placed between the last body line and the first bottom title. Therefore, the value 'n' defined for the Bottom Margin must be at least as large as the sum of the Footing Space and the Footing Margin:

$$BM >= FS + FM$$

The default '6' allows for a one-inch bottom margin.

Also see: .TM, .FS, .FM, .BT, .CP, and "Page Layout in Script".

Example

    .bm 3

.BR

            .BR

"BReak" is used to force subsequent text to begin on a new print line. A blank line is not created. This control word is useful when a list is being created and the ".FO" and ".CO" words are not being used. In practice, it is used when just two or three lines need to be separated, since ".FO OFF" would be easier to use if several lines in a row are to be kept separate.

Also see ".SP", ".SK", and ".FO".

Examples

        Please send these items:
        .sk
        one garden hose
        .br
        two hand spades
        .br
        one leaf rake

If ".BR" had not been specified, this would have been the result:

        Please send these items:

        one garden hose two hand spades one leaf rake

.BT

```
.BT   <n>   /left/center/right/
       1
```

This is used to specify a Bottom Title. Up to six such titles may be defined, and the one numbered '1' (or the unnumbered one, which defaults to '1') will be the first one printed. (If you are used to IBM SCRIPT, this sequence is the opposite of theirs.)

If 'n' is omitted, '1' is assumed. 'n' must be an integer between 1 and 6.

(The rest of this description applies to both Top Titles and Bottom Titles. Except for their positions on the page, they function identically.)

Three strings, properly delimited, must be specified. The first of these will be left-justified, the second will be centered, and the third will be right-justified. Centering is performed relative to the center of the current line length, so the number of blanks between 'left' and 'center' may be different from the number of blanks between 'center' and 'right' (if the lengths of 'left' and 'right' differ).

Any one of 'left', 'center', or 'right' may contain the "Page Symbol" (see ".PS"), along with any other text. If present, the Page Symbol will be replaced by the current page number. Use of ".PS" is optional, but if it appears in no Top or Bottom Title, and titles are being used, then pages will not be numbered. If no Top or Bottom Titles have been defined, then SCRIPT will number pages automatically, in the upper right-hand corner (page 1 will not be numbered unless a Top Title has been defined before any body text occurs).

Top Titles and Bottom Titles are entirely independent of each other in this implementation of SCRIPT. Either, both, or neither may be used at your discretion.

Any or all of the three strings may be "null", in which case blanks will be placed in the corresponding portion of the title. Leaving 'left' and 'right' null, but supplying a value for 'center', produces a centered title.

Bottom Titles are printed in the "Footing Space", and "Top Titles" are printed in the "Heading Space". If the space is too small (does not contain at least as many lines as there are titles for that space), then only as many titles as can fit will be printed. Since ".HS" and ".FS" both default to '1', multiple Top or Bottom titles can be produced only by changing the values of ".HS" and/or ".FS".

If you want titles only at the top of the page, use ".TT", and do not use ".BT". If you want titles only at the bottom, use ".BT" only. As you can see, this document uses both a top and a bottom title.

Examples

    Any of these will create one title at the top of the page only:

        .tt /CHAPTER 1/GREAT AMERICAN NOVEL/Page $/
        .tt //August 18, 1980//
        .tt /This is a left-justified top title///

This will create one title at the bottom of the page:

        .bt ..8/18/80..

    Notice that periods were used as delimiters, because slashes appeared in the text being delimited. This bottom title will be centered.

    This will create two top titles and one bottom title:

        .hs2
        .tt 1 /SUBSCRIPT/DOCUMENTATION/$/
        .tt 2 /and SUBEDIT/Version 1.3//
        .bt //by PROSOFT//

.CE

        .CE <1> | <n> | <ON> | <OFF>

    This allows you to "CEnter" one or more lines of text. The text to be centered occurs on the line(s) immediately following the "CEnter" control word. Formatting is suspended until centering ends. Centering occurs based on the current values of: "AD", "BF", "IN", "LL", and "OF".

If no parameters are supplied, then just the next line is centered. If a numeric value is supplied, then the next 'n' lines of text will be centered. To center a group of lines, it is best to use the "ON" parameter just before the group, and the "OFF" parameter to reset centering just after the group of lines.

    When centering multiple lines at a time, any valid control words may occur within the group. They are executed, taken into account as appropriate, and do not count towards the centering count (if "CE n" was used).

    If an input line to be centered contains too much text to fit in the line, centering is suppressed for that line.

    Centered text may be underlined or double-width. On some printers (but not all), it may be both.

Examples

    .ce
    This one line would be centered

    .ce 2
    This line would be centered

```
So would this one
This one would not be centered

.ce on
This line would be centered
.sk
This would be, too, with a blank line above it
Centering would continue indefinitely, until...
.ce off
And this would be the first non-centered line.
```

.CM

```
.CM    anything
```

"CoMment" is used to make private notes within a text·document. The control word, and the remainder of the line in which it occurs, is totally ignored by SCRIPT. This control word is one of the few that do NOT cause a control break. No other control word may follow ".CM", since the remainder of the line is ignored.

".CM" is useful in carrying file identification and update history within the text document itself.

Example

```
.cm Standard Thank You letter, revision 3, 6/78
```

.CO

```
.CO    <ON>  |  <OFF>
```

"COncatenation" is the transfer of excess text to the next print line, and the inclusion of text from the next input line on the current print line. It is used to ensure that as much text as possible will be printed on each line, without exceeding the currently-defined Line Length (".LL").

COncatenation is normally used with ".JUstify", and both are usually ON or both are OFF. The ".FO" control word changes both ".CO" and ".JU" at the same time. ".CO" is rarely used by itself, although ".JU" may be used to produce a right-ragged or right-justified margin. Leaving ".JU" on while turning ".CO" off can produce unnatural-looking lines and is not recommended.

Example

```
.co off
```

.CP

.CP   <n>

"Conditional Page" is used to ensure that sufficient lines remain on the current page to allow printing of material that should not be split across two pages. If there are fewer than 'n' lines remaining before the Bottom Margin, then an immediate "end-of-page" condition is raised: the paper is advanced, any Bottom Titles are printed, a page-eject is performed, any Top Titles are printed, and then the text block following ".CP" is processed.

Also see ".SKip", ".PParagrpah", and ".PAge".

Example

.cp 6

".CP" was used heavily when preparing this document, since I wanted to avoid placing the command definitions at the bottom of one page while the explanation appeared at the top of the next page.

.DA

.DA   <u>0</u> | 1 | 2 | 3

This controls the "DArkness" of printed output by forcing selected lines to be overstruck (printed twice without a paper movement). It is useful when printing in 10 cpi font, or when the ribbon is wearing out. NEWSCRIPT has no explicit facility for overstriking individual characters or words, although judicious use of the "backspace" escape sequence may achieve the desired result.

If '1' is specified, then all subsequent lines will be overstruck, until a ".DA 0" is encountered.

If the "DA" run-time option is selected and the file contains ".DA 0", lines after that control word will not be overstruck.

On most printers, only "0" or "1" may usefully be specified. On the Epson MX-80, this control word selects any of four special options as follows:

```
0 - Normal, single-pass printing
1 - Emphasized mode (this looks really nice)
2 - Overstrike mode (prints each line twice)
3 - Double-emphasized (great for bold headings)
```

Note: This won't work on all printers, as it requires either a backspace, reverse line-feed, or hardware overstrike capability.

Note: Since this may cause extra wear and tear on the printer, it should be used only when necessary. Also, as you can see from reading this document, the registration on some printers is not always perfect, so overstruck lines may not be as sharp as normal ones.

Example

```
This line is printed at 10 cpi without ".DA".
.da1
This line is printed at 10 cpi WITH ".DA".
.da0
And this is printed normally
```

.DS

```
.DS <ON> | <OFF>
```

This controls double-spacing. If "ON" or no value is specified, double-spacing is turned on, and remains on until cancelled by either ".SS" or by ".DS OFF". If "OFF" is specified, then double-spacing is turned off.

If the "DS" run-time option is also specified, double, not quadruple-spacing will occur. If you want your document single-spaced in its final printing, but double-spaced for proof-reading, use the "DS" run-time option, and not this ".DS" control word.

Example

```
.ds
```

.ES

```
.ES  c | <!>
```

This allows you to re-define the "EScape" character when you need to use an escape sequence as normal text. An escape sequence is two characters, beginning with the currently defined escape character. Escape sequences are used to control special hardware features of the printer. The effect of an escape sequence begins immediately, at the character following the sequence, and without causing a new line to begin. After re-defining the escape character, the old value becomes a normal character and remains that way unless it is subsequently defined again as the escape character.

If the escape character is not followed by one of the characters defined below, both it and that following character are treated as normal data, not as an escape sequence.

Escape Sequences

Nine escape sequences are supported by SUBSCRIPT:

> Begin double-width characters: !(
>   End double-width characters: !)
> Begin underlining: !$
>   End underlining: !%
> Begin italics: !/
>   End italics: !?
> Half-space down for subscript: !+
> Half-space up for superscript: !-
> Backspace one character: !<

    Not all printers support all of these features (see the chart in Section I). Depending on the printer, these will work as follows:

1. Double-width, underlining, and italics remain in effect until reset by the complementary escape sequence. Each of these may span multiple lines if necessary, and any combination of the three may be used if you wish. Remember that not all printers support all these features (at the time of this writing, only certain Epson's support italics, for example).

2. Half-spacing up or down gives a superscript or subscript. Some printers (Centronics 737, for example) perform a carriage return as part of the operation. In an attempt to maintain vertical registration of the data, NEWSCRIPT actually performs extra "reverse line feeds" and then comes forward again to the correct vertical position. Even so, you may have to experiment with the paper release lever (if there is one) to achieve the best result. Precise registration for subscripts, superscripts, and overstriking (the ".DA" control word) depends on the condition of your printer and the size of the holes in continuous-form paper.

3. Backspacing moves back one character position and allows overstriking (this can produce a slashed zero, or a darker letter, for instance). In proportional font, this may not work well. On the Centronics 737, it doesn't work well, since the printer always performs a carriage-return to the left margin and then tries to come back to where it believes it should continue printing. On standard MX-80's, backspacing isn't supported by the printer. However, on the MX-80 with GRAFTRAX, or on the MX-100, it will work. On the Diablo, Daisy Wheel II, and most other printers, it will also work.
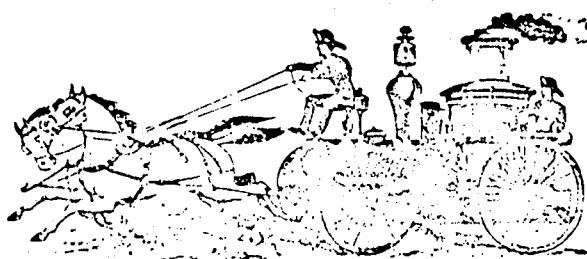
Examples

   Throughout this discussion, the default escape character (the exclamation mark) is used for illustration. In order to print these pages, my own document temporarily re-defined the escape character to be the '@' sign. This was done by:

   .es @

This line of input text:

                    The !$formula!% for !$!(water!)!% is: H!+2!-O

would be printed as:

                    The formula for water is: H$_2$O

(YUP! You really can do all that... on some printers.)

.FM

                    .FM   <n> | <1>

   The "Footing Margin" is the number of lines between the last body-line and the first Bottom Title. When changing its value, it may also be necessary to change ".BM" and/or ".FS" so as to ensure that the following remains true:

                         BM >= FM + FS

   This control word does not provide support for footnotes. At present, the only good way to do footnotes in SCRIPT is to place them at the end of a chapter. Placing them at the end of a page requires considerable manual effort.

Example

       .bm 4
       .fm 2
       .fs 2

In this example, the bottom margin has been re-defined to be 4 lines in height (2/3 of an inch); provision has been made for a two-line bottom title; and there will be a two-line gap between the last line of the body of the text and the first bottom title line. Actually, the result probably would be unacceptable, since the second title line would print on the very bottom of the page. If the paper had not been positioned correctly, that second bottom title line might print on the perforation.

.FO

        .FO <ON> | <OFF>

This allows you to turn "FOrmatting" on or off. Formatting controls both concatenation (flow of text from one line to another so as to fit, but not overflow, the current line length) and right-margin justification. While either can be set independently, they frequently are turned on or off at the same time, and this is a shortcut for that purpose.

When "Format" is "ON", as much text as will fit each line is used, and the result is a straight right-margin. When "Format"is "OFF", the text is printed as-is, regardless of its length. However, some printers perform automatic line feed/carriage returns upon reaching the right side of the carriage, and others just sit at the right side and print all remaining text in the last position. Therefore, this control word should be used carefully, and you should remember to turn formatting back on when passing from an unformatted text block to a block that you do want formatted.

Example

        .fo off
        .fo on

                This was done with FO turned on. The original text
                lines are identical to the ones below, except for the
                presence of the control word ".FO OFF" that preceded
                the next example.

                This was done with FO turned off. The original text lines
                are identical to the ones above, except for the presence of
                the control word ".FO ON" that preceded the above example.

.FS

        .FS <n> | <1>

The "Footing Space" lies within the Bottom Margin, below the Footing Margin, and may contain the Bottom Title(s). If you wish to use multiple Bottom Titles, then you must re-define the Footing Space, and possibly the Bottom Margin, so as to leave sufficient room for your titles.

Also see ".BM", ".FM", ".BT", and the "PAGE LAYOUT" diagram at the end of this Section.

Example

        .bm 7
        .fs 2
        .fm 2
        .bt 1 /Text Changes/Part 2//
        .bt 2 ///$/

.HM

.HM   <n> | <<u>1</u>>

The "Heading Margin" is the space between the last Top Title line and the first line of the body of the document. One or two blank lines in this position will separate the headings from the main part of the page, resulting in a familiar format.

Except for controlling what happens at the top of the page, this control word is the same as ".FM". Please refer to that control word for examples. Also see ".TM", ".HS", ".TT", and the "PAGE LAYOUT" diagram at the end of this section.

.HI

.HI   < < + | - > n > | < <u>0</u> >

"Hanging Indents" are delayed indents, also called "offsets". They are used to produce a series of bullets, in which a paragraph number or asterisk sticks out in the left margin, while the rest of the text of that paragraph is slightly indented to the right. When 'bullet' paragraphs are printed, the first line is not indented (unless '.IN' is also used), but all subsequent lines are indented 'n' tenths of an inch.

NEWSCRIPT supports hanging indents in two ways: if ".HI n" is specified, then the delayed indent value 'n' is reset each time a "control break" occurs (each time the next SCRIPT control word that causes a break is encountered). This automatic reset continues until ".HI 0" is issued to turn off the feature. The second way is to use ".OF n", in which case a delayed indent of 'n' tenths of an inch remains in effect until the next ".OF n" is encountered. This allows control breaks to occur within a bullet, without resetting the indent.

NOTE: A Hanging Indent remains in effect until another ".HI n" occurs. To turn the feature back off, use ".HI 0" (zero).

Because of the many possible characters that can be used between the paragraph number and the text itself, it is difficult to produce a perfectly flush left-margin when Hanging Indents are used with proportional font. However, SCRIPT is able to come within 0.02 inches of the ideal alignment in most cases. As a guide, when using asterisks as shown in the examples, ".HI 2" is correct; when numbering, ".HI 3" is best.

Also see ".OFfset", ".INdent", and ".ADjust".

<u>Examples</u>

    This input text:

        There are a number of advantages to using a Word
        Processing system, which is why they have become
        so popular:
        .sk
        .in 3
        .ll -3
        .hi 2
        * They reduce the time it takes to make corrections
        and revisions;
        .sk
        * They permit sophisticated, repeatable page layouts;
        .sk
        * They make it easy to produce customized Form Letters.
        .sk;.in0;.ll+3;.hi0

    will produce this formatted result:

        There are a number of advantages to using a
        Word Processing system, which is why they
        have become so popular:

           * They reduce the time it takes to
             make corrections and revisions;

           * They permit sophisticated,
             repeatable page layouts;

           * They make it easy to produce
             customized Form Letters.

.HS

        .HS   \<n\> | \<<u>1</u>\>

    The "Heading Space" lies within the Top Margin, between the top edge of the
paper and the first body line of text. The Top Title(s), if any, are placed in the
Heading Space. The default value of '1' allows for one Top Title. If more are needed,
then ".HS" must be re-defined.

    Except for controlling what happens at the top of the page, rather than at the
bottom, this control word is the same as ".FS". Please refer to that control word for
examples.

    Also see ".HM", ".TM", and ".TT".

<u>Example</u>

    .HS 2

.IM

.IM   fileid

This control word allows you to "IMbed" one file within another. The Imbedded file is used as though its contents occurred at the point of the ".IM" control word, and when the last line of the Imbedded file has been processed, the next line of the original file gains control.

IMBED differs from APPEND in that APPEND does not return control to the original file, whereas IMBED does return the flow of control to the original. Therefore, IMBED is useful in including stock phrases in an otherwise personalized letter, whereas APPEND is useful in chaining segments of a document together. For example, I often imbed my Logo, including the address and telephone number, this way.

RESTRICTION: ".IM" CANNOT BE USED WITHIN AN IMBEDDED FILE. That is, you can't "nest" Imbeds. This restriction exists because of the file allocation structure of TRS-80 BASIC and the requirement that file numbers be identified by constants rather than by variables. However, ".IM" can be used within an Appended file.

Imbed can also be used to manage the sequence of portions of a document: if the primary file contains only Imbeds, then by merely editing and rearranging the sequence of Imbeds in that primary, it is possible to cause the Imbedded files to occur in different order. The "no nest" restriction still applies.

To print mailing lists and form letters, use ".RD", and NOT ".IM". It's very easy to do form letters with ".RD", and next to impossible with ".IM".

Example of Imbed

If "FILEA" contained:

```
.ll 30
Steve,
.sk
Here is the inventory list I promised you.  Hope
it meets your needs.
.sk
.fo off
.im invenA
.sk
.im invenB
.sk
.in 10
Best Regards,
.sk 2
Chuck
```

and "invenA" contained:

```
Deluxe Widgets  50 @ .172
```

and "invenB" contained:

```
Collenders   22 @ 1.06
```

then the resulting letter would be printed as:

```
Steve,

Here is the inventory list I promised
you.  Hope it meets your needs.

Deluxe Widgets  50 @ .172
Collenders   22 @ 1.06

        Best Regards,


        Chuck
```

.IN

```
.IN  < < + | - > n >  |  <0>
```

This is used to "INdent" text, that is, to cause it to print to the right of the normal left margin. 'n' is expressed in tenths of an inch, regardless of the font being used. If a plus sign precedes 'n' then a relative shift of text is performed, further indenting the material. If a minus sign precedes 'n' then a relative text shift is also performed, but back out towards the left margin. If '-n' exceeds the current indentation value, then indentation reverts back to its original value of '0', at the left margin itself.

It is not possible to move past the left of the left margin through use of this control word. If you need such a format, you can use ".AD" to temporarily reduce the size of the left margin, and then ".AD" back to normal afterwards.

The absolute position of the right margin is not affected by Indentation. Instead, the temporary Line Length is changed to reflect the Indentation setting, and the right margin remains where it had been. Since it is often pleasing to the eye to indent both margins, reducing ".LL" by the same amount as the Indent will retain a symmetrical effect.

It is important to remember that an INdent remains in effect until changed by another INdent.

To indent the first line of a new paragraph, use ".PP", not ".IN". For "hanging indents", see ".OF" (offset).

Also see: .PP, .OF, .AD, and .LL.

<u>Example</u>

If the text file contained the following:

```
        .ll 30
        The bard's immortal words move us today,
        even as they did when first written over 300 years ago:
        .sk
        .in 5
        .ll -5
        Neither a borrower nor a lender be,
        But to thine own self be true;
        And it must follow as the night the day,
        That thou canst never be false to any man.
        .sk;.in0;.ll+5
        Of course, in many cases, the words move us only
        to negotiate some quick re-financing of our
        overdue bills.  Ah, progress!
```

Then the result would be:

```
        The bard's immortal words move us today,
        even as they did when first written over 300
        years ago:

                Neither a borrower nor a lender
                be, But to thine own self be
                true; And it must follow as the
                night the day, That thou canst
                never be false to any man.

        Of course, in many cases, the words move us
        only to negotiate some quick re-financing of
        our overdue bills,  Ah, progress!
```

.IX

```
        .IX string<;string<;string...>>
```

This identifies one or more words/phrases to be included in an index. Each 'string' may be a single word or a few words, as needed. SCRIPT collects each of these, along with the current page number, into a file. At end-of-job, another program, called "INDEX", can be run (automatically). That program will sort (that is, arrange in alphabetical sequence) and combine all page references, generate a new input file, and pass control back to SCRIPT, which can then print the index. If additional cleanup or changes are desired, this generated file can be edited (with EDIT) before allowing SCRIPT to print it.

Several independent index entries may be placed on a single line, as shown in one of the examples below. This saves space and typing time. However, SCRIPT does not support multiple indices, nor sub-terms within outer terms.

The ".IX" control word should be placed directly before or directly after the line in which the related terms occur in the normal text. It does not cause a control break, so if placed in the middle of a paragraph, that paragraph will still be printed as though the ".IX" wasn't even there.

NOTE: It is possible for an index reference to be "off" by one page, since a page break may occur in the middle of any line of text.

NOTE: Placing a term after ".IX" creates a reference <u>only to that one occurrence</u> of the term. If you want something to be indexed in several places, you <u>must</u> place a ".IX" reference in each needed place. This is what the "GENINDEX" program will do for you (see Section V for details on use of the Indexing facilities).

Index entries are named:

<div align="center">fileid/IDX</div>

where 'fileid' is the filename (but not the extension) of the current file. If file chaining through ".AP" or ".IM" is in effect, the fileid of the starting file, not the current one, is used throughout the process of adding to the index file.

Since the index is written to disk, the disk on which the index gets written must remain on-line for the duration of the print-run, must not be write protected, and must have sufficient room to hold all the entries. It may not be feasible to use this feature on one-drive systems.

The printed index will be 3 inches wide, but not two-up. It may be made two-up by pasting two pages together. (I apologize in advance for that, but it would take significant extra memory, not available on the TRS-80, to handle multiple column work. As you already realize, there's barely enough memory for NEWSCRIPT's components now.)

A complete discussion of the creation and use of indices may be found in SECTION V.

<u>Examples</u>

        .ix Greece
        .ix Rome;Carthage;elephants;roads

The first of these would generate a single index entry for the word "Greece". The second one would generate four independent entries.

.JU

.JU   <<u>ON</u>> | <OFF>

    This controls "JUstification" of the right margin. It normally is "ON", causing a straight and even margin (the form used through most of this document). However, if turned off, then a ragged right margin is printed, giving a result similar to what a typist would create.

    "JUstify" is usually associated with "COncatenate", and both can be controlled together through the ".FOrmat" control word. However, if ".CO" is "ON" and ".JU" is "OFF", then text will still be flowed between lines so as to maximize the number of words per printed line, without exceeding the currently-defined Line Length.

    Running with ".JU ON" and ".CO OFF" is not supported: SCRIPT normally will force concatenation in this case.

    Also see: FO and CO.

<u>Example</u>

    .ju off

.LL

.LL < < + | - > n > | <<u>65</u>>

    This sets the "Line Length", in tenths of an inch. The Line Length begins at the left margin (directly following the "ADjust" value). The right margin begins at "AD"+"LL".

    The Line Length may be expressed as an absolute value: "50", or as a relative value: "-3", "+7". When expressed as an absolute value, all subsequent body lines will be formatted within the defined length, and all titles defined <u>after</u> the new Line Length definition will be of that length. Previously defined titles retain their original length.

    When Line Length is expressed as a signed number, the old length is changed by the requested amount. The comments made in the preceding paragraphs regarding Title lengths still apply.

    If no value is given, the Line Length reverts to its last <u>absolute</u> value. This is useful when a series of relative line lengths (+/- n) have been used and it has become difficult to keep track of the current result. If you haven't set any absolute line lengths, then the original default of 65 (6.5 inches) will be used.

    The Line Length must always be in the range:

        10 <= LL <= 132 (1.0 inches to 13.2 inches)

although only in 16 CPI font can the longer lengths print properly on narrow printers. If the line length plus the left margin "ADjust" exceeds the physical length of your printer, the printer may type the rest of the line at the extreme

right-hand side (Diablo, Selectric), or may force an automatic line-feed / carriage-return, printing the excess on the next line. In either case, the result will be unacceptable, and in the second case, SCRIPT and the printer will be out of sync regarding where top-of-form is.

Most word processing languages expect to deal with 10 cpi (monospace) fonts, and therefore provide a means of expressing Line Length in terms of number of characters per line. Since the number of characters on a proportional printer line varies, and since it is so easy to change from 10 cpi to 12 or 16 cpi on most modern printers, expressing length in tenths of an inch offers a way of retaining compatibility with these other systems... ".LL 60" produces a six-inch line in SCRIPT as well as in other systems.

Examples

        .ll 20

will give a two-inch line length:

                        This little paragraph was
                        created with a line length
                        of 20.

but this: .ll +5

will modify that '20' to '25', producing:

                        This slightly less little
                        paragraph was created with a
                        line length of 25.

If you make the Line Length too short, SCRIPT may decide that it is a "short line", and therefore not right-justifiy it. Such short lines can fall in the middle of your text, not just as the last line of a paragraph. You can control this through ".SD".

.LS

                .LS n

The Logo Space occurs only at the top of the very first page of a letter. The control word is used to allow you to start printing below your logo, and yet not have to leave a large top margin or remember to do a ".SP 10" at the top of the first page.

The number of lines 'n' specified in this control word are added to the line position counter maintained by SCRIPT. You should position the paper so that printing will begin where you want it to: SCRIPT will not advance the paper for top margins or titles on this first page, but will assume that you've set the paper where you want the printing to start.

Example

.ls 15

tells SCRIPT you've positioned the paper 2-1/2 inches down from the top edge of the sheet (if running 6 lines/inch).

.OF

.OF < < + | - > n > | < 0 >

"OFset" is a delayed "INdent", sometimes called a "hanging indent". The line printed just after it will be left-justified according to the ".ADjust" and ".INdent" settings then in effect. Subsequent lines will be indented to the right by the OFFSET amount, as expressed in tenths of an inch. This numeric value may be absolute (unsigned), or relative (signed). If absolute, the offset is changed to that value; if relative, the value is added to the current offset. If the result is negative, the offset is set to zero (no offset).

NOTE: An OFFSET remains in effect until a new OFFSET is given. If you want to create a series of "bullets", you must precede each new bullet with another ".OF n".

The purpose of OFFSET is to create "bullets", that is, numbered paragraphs with smooth left margins, but with the paragraph numbers or asterisks standing out to the left. ".OFfset" is often used along with ".INdent" and ".LL" to produce eye-pleasing results.

When the last numbered (or lettered) offset paragraph is complete, you must always set ".OF" back to zero. SCRIPT has no way of knowing that you're done unless you tell it so.

Because of the many possible characters that can be used between the paragraph number and the text itself, it is difficult to produce a perfectly flush left-margin on OFfset text when using proportional font. However, SCRIPT is able to come within 0.02 inches of the ideal alignment in most cases. As a guide, when numbering as shown below, an ".OF3" is correct; when just using asterisks ("*"), an ".OF2" is best.

Also see ".HI" (Hanging Indents), ".INdent", and ".ADjust".
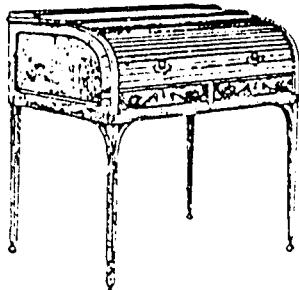
Tambour writing table.
Century Dictionary

Example

    There are a number of advantages to using a Word
Processing system, which is why they have become so
popular:
      1. They reduce the time it takes to make
        corrections and revisions;
      2. They permit sophisticated, repeatable
        page layouts;
      3. They can automate production of Tables of
        Contents.

The formatted bullets above were produced by this:

    There are a number of advantages to using a Word Processing
system, which is why they have become so popular:
.sh;.in+5;.ll-5;.of 3
1. They reduce the time it takes to make corrections
and revisions;
.of3
2. They permit sophisticated, repeatable page layouts;
.of3
3. They can automate production of Tables of Contents.
.sk;.of 0;.in0;.ll+15

.PA

        .PA   <n> | <Odd> | <Even>

    This causes an immediate "PAge" eject. Any Bottom Titles are printed on the
current page, the Page Number is advanced by '1', any Top Titles are printed on the
new page, and then the formatting and printing of the body text continues.

    If 'n' is specified, it is used as the Page Number of the new page, overriding the
default advancement of the page numbering facility. Subsequent pages will be
numbered from 'n+1'.

    If "ODD" or "EVEN" is specified (only the first letter of either word is needed),
then the page counter will advance by either 1 or 2, so as to reach the next odd or
even page number, as required.

    If ".PN OFFNO" is in effect and ".PA n" is specified, the internal page number
counter will be set to 'n' anyway.

    'n' must always be a simple integer number. Decimal points are ignored and
unsupported. Roman numerals are not supported. If 'n' is less than '1', it is ignored.

Examples

    .pa
    .pa 7
    .pa 1
    .pa odd

.PL

```
.PL   n
      <66>
```

This allows you to specify the "Page Length" in terms of the number of lines on a physical page of paper. The default is '66', based on 11-inch paper, six lines per inch (11 times 6 = 66).

'n' must be a positive integer, not greater then 32767.

Page Length should always be greater than the sum of the Top and Bottom Margins, or SCRIPT will do nothing except print titles and eject paper indefinitely. No check is made for this error condition.

Example

```
.tm 0
.bm 0
.hs 0
.fs 0
.hm 0
.fm 0
.pl 6
```

This series of page and vertical-margin specifications might be used in printing labels. Another, equally acceptable way to handle labels or other short pages might be this:

```
.hm 0;.fm 0;.hs 0;.fs 0;.tm 0;.bm 0;.pl 32767
```

.PN

```
.PN   <ON>   |   <OFF>   |   <OFFNO>
```

This controls automatic "Page Numbering". By default, SCRIPT will number each page, beginning with Page 2, in the upper-right hand corner, as follows:

<p align="center">Page   2</p>

'2' will be whatever the current page number happens to be. Page 1 normally isn't numbered, but if you do want it numbered, you can specify a Top Title (".TT") at the start of the document, and place the Page Symbol (normally a dollar sign) wherever you want the number to appear.

If you do not want page numbers printed, or not printed on certain pages, specify ".PN OFF". The internal page number counter will still be active, but will not print until ".PN ON" is specified.

If you do not want page numbers printed, and also do not want the internal counter to change with each page eject, specify ".PN OFFNO". To re-activate the counter, specify ".PN OFF"; to reactivate printing (and the counter), specify ".PN ON".

".PN OFF" is not effective when Titles are in use. The only way to suppress the page numbers in a title is to re-define any titles containing the Page-numbering Symbol so that the symbol no longer appears in the title definition.

<u>Example</u>

```
.pn off
.pn on
.pn offno
```

.PP

```
.PP    <cp <,sk <,in <,sh > > > >
        2,   2,   5,   0
```

This starts a new paragraph. It must be the last (or only) control word on the line. When this control word is encountered, a control break occurs, causing all text up to this point to be formatted and printed. Then, each of the four current values for paragraph are processed in turn. The defaults shown above are used unless you've changed them, in which case the current or most recent settings are re-used.

cp stands for "conditional page". There must be at least this number (default=2) of lines left on the page, after the "sk" value occurs, for the next paragraph to be placed on the current page. Otherwise, the paragraph is placed at the top of the next page.

sk stands for "skip". The new paragraph will be started "sk" lines (default=2) below the current printed line. If "sk" is set to '1', then the paragraph begins one line down, which is the very next line. That is to say, no blank lines will be left between the paragraph. The default ("2") causes one blank line between each paragraph.

in specifies the indentation, in tenths of an inch, for the first line of the paragraph (default=5, which is half-an-inch). This is not the same as the ".IN" control word, which indents all following text. This "indent" only indents the first line of the paragraph.

sh causes the "sk" value to be taken in half-lines instead of full lines. It may be set to "1" (meaning "space in half lines) or to "0" (the default, meaning "space in full lines). If your printer cannot easily perform half-spaces, this value should be left at "0". At present, the MX-80 is not supported in half-space mode by NEWSCRIPT. In general, if NEWSCRIPT supports sub-scripts and super-scripts for a given printer, it also supports this value.

<u>Examples</u>

```
this is the end of a paragraph.
.pp
This will start a new paragraph.
.pp 3,2,4
```

There will be one blank line above this paragraph,
there must be at least 3 lines left on the page, and
the first line will be indented 0.4 inches.

.PS

```
.PS    c
       <$>
```

This allows you to re-define the "Page-number Symbol". The "PS" is used in Top
and Bottom Titles to indicate where you want the page number to be placed.

SCRIPT performs page numbering automatically. If no titles are in effect, then
each new page is numbered sequentially in the upper right-hand corner, unless ".PN
OFF" or ".PN OFFNO" is in effect (see ".PN"). However, if any titles have been
defined, page numbers are printed only if at least one of the defined titles contains
the current Page-number Symbol. Therefore, if you change the default, do so before
using it in a title, and do not change it thereafter unless you also change the title.

".PS" is used in cases where the default symbol (the dollar sign) is needed as
normal text within a title.

If printing in proportional ('737') font, which is the default on the Centronics
737, the Page-number Symbol (which cannot occur elsewhere in any title you define)
should be a character that occupies twelve "dot-spaces". These characters meet
that requirement:

$$\$ \ * \ + \ = \ ?$$

In fact, many others do also, so you are not restricted to the five suggestions
shown here. If you are using a monospace font and/or printer, any character may be
used, so.long as it doesn't appear elsewhere in any title.

Example

```
.ps ?
.tt //MEALS UNDER $5.00/Page ?/
```

Since the desired title contained the default Page-number Symbol ($), that
symbol was re-defined before the title was defined. The symbol should not be
re-defined thereafter.

.RD

```
        .RD  n  <fileid>
             c  fileid
```

This instructs SCRIPT to "ReaD" one (*[handwritten: Type "RD 5 When inside address will be Script.]* ƅm
'fileid' and to print them as-is in the ne* *[handwritten: When letter gets]* to
create Form Letters, and can do so in any of *[handwritten: to ✓ address it]*

'n' must be a positive number, if supplied. *[handwritten: will stop Type]* be
   read at a time from the N&A (Name & Add *[handwritten: on ✓ pe]*

'c' stands for any non-numeric character, *[handwritten illegible]* le
   number of lines to be read at a time.

These are the ways in which NEWSCRIPT can   _ _ _ _ _ _ _ _ .

1. Text may be entered directly from the keyboard. This occurs if 'n' is
   specified but 'fileid' is omitted. 'n' lines are read from the keyboard and
   printed left-justified, as-is. Then, the rest of the letter is printed.

2. Text may be read from a file in the same manner as from the keyboard.
   This occurs if 'n' and 'fileid' are both specified. A fixed number of lines
   must be provided for each entry in this case, even if some lines are blank.

3. Text will be read from a file if 'c' and 'fileid' are both supplied. In this
   case, a lines of "selection codes" follows the character 'c' in the Name
   and Address file, and the Name line(s) and Address line(s) follow the
   "codes line". There may be as many such lines as you need, each entry may
   have a different number of lines, and a given Name and Address group
   ends with the next "codes line". Blank lines are not needed in this case.

4. The fourth kind of list is like the third, but with the addition of up to
   nine variable names and/or phrases that will be inserted within the body
   of the text based on symbols you place in the original form letter.

Each of these is further explained below, and examples are provided for each.
The "HOW TO..." section has further information on this subject. Note that Form
Letters is not the first thing you should try to do when learning NEWSCRIPT. The
feature has considerable capability, but is mastered only after study and practice.

When ".RD" is encountered, a control break occurs, causing all text up to the
control word to be formatted and printed. If the first kind of list (from the
keyboard) is used, SCRIPT will prompt:

                        > > > ENTER n LINES < < <

and for each specific line will prompt:

                                n: ?

In both prompts, 'n' will be a specific number, beginning with '1'. You should
type the exact line of data you want printed, and then press the <ENTER> key. If
several lines are to be read, you will be prompted repeatedly, and should enter one
text line each time. If you are entering a Name and Address, then you might set 'n'
to be '4' to '6', and when SCRIPT processes this portion of the text file, you would
enter the specified number of lines. If you need fewer lines than 'n', just hit the

```

<ENTER> key for the excess; this will produce a blank line each time.

The lines you enter are NOT concatenated (joined) to each other, nor to anything else. They are printed as-is. If proportional pitch (font) is active, the text will be dot-spaced according to the current ".SD" setting.

If 'fileid' is supplied, then SCRIPT will ensure that the file does exist. Then, if a quantity of lines was specified, it will read the first 'n' lines from that file and print them as-is, and then continue with the rest of the letter in the original file. This is the second kind of Form Letters. When the end of the letter is found, a page eject will occur, the letter will start over again, automatically and with no operator intervention. The next time the ".RD n fileid" is encountered, the next 'n' lines from the same file will be read and printed, etc. SCRIPT will keep track of its position in the Name and Address file, and repeat the form letter until it encounters the end of the N&A file. After the final letter has been printed, SCRIPT will perform its normal end-of-job processing.

If 'c' was specifed, a variable number of lines is read from the file. The rules are used for the third and fourth kinds of mailing lists are as follows:

1. Each entry in the N&A file must begin with a 'codes' line, and the first character of this 'codes' lines must be the character specified by 'c' (it normally won't be the letter "c", but something like a "@", "#", etc.).

2. The 'codes' line may optionally contain a series of selection codes of your choice. These could be one-letter codes, or words separated by blanks. The codes can be used to selectively print only certain names and addresses.

3. When the N&A file is accessed for the first time, you will be asked to enter the Selection Code for this particular print run. You may enter any character string you wish, or just hit the <ENTER> key to indicate that all entries in the file should be included into the letters that are about to be printed. This question is asked only once, and the code, if any, will be used for the rest of the run.

4. If the 'code' line contains the Selection Code, or if no Selection Code was given, then the next several lines of material from the N&A file will be read and printed as-is, left-justified, but subject to the current settings of ".IN" and ".AD".

5. If the 'code' line does not contain the Selection Code, the N&A file will be scanned until another 'code' line is encountered, and Step 4, above, will be repeated. This will continue until the end of the file is encountered.

6. Once a name and address entry has been selected for printing, all lines of text after the 'code' line, up to the next 'code' line (or end of file), will be printed. When the next 'code' line (or end of file) is found, control reverts to the next line of input text from the letter itself. A separate letter will be printed for each qualifying entry in the N&A file. When the end of the N&A file is reached, the final letter will be completed and SCRIPT will stop.

7. Variable information may be included in any or all N&A groups. This information is stored until needed, then printed within the body of the

letter. Up to nine variables may be defined within each group, and if none are defined for a given N&A, SCRIPT will print a letter anyway, but ignore the signal for printing the variables. This variable information is not printed along with the normal N&A lines. However, you can create a N&A file that contains only variable information and no fixed N&A's. Then, only the variables will be printed, and they will print only where you specify.

To define variable information, use the code character twice, followed by a number from 1 to 9, followed by one blank, followed by the information you wish to substitute. Exact positioning must be used (no blanks after the double code character, exactly one blank after the number). For example, if '@' is the code character:

<p style="text-align:center">@@1 Mr. Smith</p>

would define "Mr. Smith" as variable #1 for this copy of the letter being printed.

8. There is one "permanent" or "global" variable: "@@0" ("@@" would be whatever code character you've used, but zero is the identifier under discussion here). It's value comes from the keyboard, not from the N&A file. You will be prompted to enter it on the first letter only, and it's value will be re-used for all subsequent letters printed during the current run. The purpose of this variable is to provide a date in the letter, although any other similar use would be acceptable.


Only one N&A file may be used per letter. If you have more than one ".RD n/c fileid" in your form letter, all data will be taken, sequentially, from 'fileid'. This precludes your using the same data twice in the same letter, unless you've repeated it in the N&A file also.

The "CC" run-time option should not be used to produce customized form letters: if you use "CC" and ".RD n fileid", you will get a complete set of letters (one for each 'n' lines in the N&A file), and then another complete set, etc.

".RD", not ".IM", should be used to create form letters from a mailing list. ".IM" is intended for imbedding text from another file, and would be a very cumbersome way to do mailing lists.

## FORMATS OF A NAME AND ADDRESS FILE

SCRIPT can process either of two simple formats of mailing lists: a fixed number of lines for each N&A; or a variable number of lines, each preceded by a 'code' line. Each of these two formats is described below.


### Fixed Number of Lines

SCRIPT expects a very simply structured Name and Address file. It must be ASCII (the kind of file produced by your own BASIC programs, and also by EDIT), with each entry created by BASIC'S "PRINT #n" statement. This kind of data is read correctly by the "LINE INPUT #n" statement.

When the fixed number of lines approach ('n') is used, each Name and Address within the file must occupy the same number of lines as all the others, even if some of these lines are left blank.  This number must be the same as the value of 'n' specified in the ".RD n fileid" control word used in the form letter itself.

If SCRIPT encounters end-of-file unexpectedly in the N&A file, it will just print extra blank lines for that final form letter and continue normal processing.  No special error message is given.

When the last form letter has been printed, this message is displayed:

<p align="center">*** LAST LINE READ FROM N&A FILE ***</p>

If 'fileid' is not found on an on-line diskette, this message is displayed, and SCRIPT goes to end-of-job:

<p align="center">!!! FILE fileid NOT FOUND FOR '.RD' !!!</p>

Name and Address files may easily be created using EDIT, or most other programs, for that matter.  If you are using a Mailing List program that carries its data in a different, incompatible format, it will be necessary to write a simple BASIC program that creates a new N&A file that can be read by SCRIPT.

Examples With Fixed Number of Lines

First consider a Form Letter whose address lines will be entered from the keyboard. The letter might look something like this:

```
.in 30
.fo off
Box 839
No. Hollywood, Ca 91603
.cm get the date
.rd 1
.in 0
.sk 2
.cm get the Name and Address
.rd 6
.cm print the body of the form letter
.sk;.fo on
Thank you for your interest in our products... (etc.)
```

If you run this through SCRIPT, you will be prompted to enter one line (the date), and then to enter six lines (the Name, Address, and salutation, if that's what you wanted printed).  If you wanted to run several letters, you would have to take SCRIPT's end-of-job default (the <ENTER> key) to re-run the same file again, and would also have to ask SCRIPT to EJECT the last page of each copy of the letter.


For the second example of Form Letters with a fixed number of lines, consider the use of a Name and Address file.  Each N&A occupies four lines.  The SCRIPT input might be very similar to the above, except for the omission of the date (you could still read the date from the keyboard each time, but SCRIPT does not remember what you enter from one letter to the next, since you might be entering variable information such as amounts due):

```
.rd 4 GOODGUYS/NA
We appreciate your recent order, and hope that...
```

The 'GOODGUYS/NA' file should look something like this:

```
J. R. Jones
1234 Oak Street
Palm Springs, Ca

Ms. Jennifer Leeds
Acme Widgets, Inc.
1555 Main Street
Los Angeles, Ca.
Harry Hopkins
4777 Hollywood Blvd
Suite 4
Hollywood, Ca.
```

Notice that the very first entry (to J. R. Jones) only needed three lines, so a blank line was entered to fulfill the '4' line requirement. If some entries in the N&A file had to be more than 4 lines, then all entries would have to be changed to meet the maximum requirement.

Variable Lines Format

The example below supposes that we want to print letters to only a few selected people on our list. The letter itself might look like this, where the pound sign ("#") is the "codes signal":

```
.im logo
.rd # club/lst
.sk
Dear ##1,
.pp
The Valley Chapter of "Computer Addicts" will hold its
next meeting on Saturday, April 4th, at the Victory Inn.
The meeting should be particularly interesting because
we will have no one talking about Word Processing!
See you there!
```

This is an example of the fourth type of Form Letter, in which a variable is used to completely customize the letter. If "##1" were replaced by "Member" or some other non-variable, the example would apply to the third type of letter described earlier.

Suppose the mailing list looked like this (we will only show three entries):

```
     #A,VP,7/81
     Renee Desmond
     123 Oxnard St.
     No. Hollywood, Ca. 91604
     ##1 Renee
     #X,6/81
     Ron Rogers
     MagiPrint, Inc.
     Box 5544
     Van Nuys, Ca. 91607
     ##1 Mr. Rogers
     #A,12/81
     Mike Salven
     Open Roads Bikes, Inc.
     8811 Vanowen St.
     Suite 3B
     Sun Valley, Ca. 92005
     ##1 Mike
```

When the letter is processed by SCRIPT, and the ".RD" is encountered for the first copy, the pound sign (#) will be picked up as the 'code' line marker, the file will be opened, and you will be asked:

ENTER SELECTION CODE:

If you enter "A" (it would have to be upper-case, since the codes were done in upper-csae), then letters would be printed for the first and third entries in the N&A file (Renee Desmond and Mike Salven). The second entry doesn't have the letter "A" in its code line, and therefore wouldn't be printed.

The variable defined for each of these entries ("Renee" and "Mike") would replace the "#1" in the salutation of each resulting letter. However, if "#1" were omitted from the letter, the variables would be ignored. If "#1" occurred more than once in the letter, the substitution would occur as often as needed. If several variables were defined and also occurred in the letter, substitution would occur as needed.

If one of the codes happened to be an expiration date (5/81, for instance), it would be possible to send out "dues notices" from a list like the one shown, since the 'codes' lines contain such dates.

.SD

```
.SD   n1,n2,n3
       1, 5, 1
```

"Spacing Definitions" is used to override the default inter-character dot-spacing values. If you do not like the range of blank space left between the letters of words printed by SCRIPT, the Spacing Definitions can be used to specify a different range. When creating proportional output, all three values are used to represent dot spaces. When creating 10, 12, or 16 cpi output, only the second value is used for the purpose of limiting the number of spaces between words. However, all three values must always be specified.

Left to itself, the printer leaves one dot-space between letters when in proportional-font mode. The result may look too crowded, so SCRIPT generates a small amount of extra space between each and every character. Further, in order to achieve right-justification, this extra space is varied so as to evenly distribute the space needed to get the even right margin.

When in monospace font (10 cpi or 16.7 cpi), all extra space is left between the words, and none between the letters within a word (this is the way most Word Processing systems perform right-justification, since they expect to be dealing only with monospace printers). This means that ".SD" is normally not used in monospace printing.

Three values are controlled by the Spacing Definitions, and all three must be specified each time:

1. <u>Minimum</u> number of dot-spaces allowed. A dot-space is 1/60'th to 1/150'th of an inch, depending on the printer being used. This parameter limits the crowding of the letters, and defaults to '1'. Making it much larger than '3' may produce an excessively-spaced line.

2. <u>Maximum</u> number of dot-spaces allowed. This parameter limits the spreading of the letters, and defaults to '5'. Making it much larger than '6' can produce justified lines even when there is very little text. Making it smaller than the default will cause more lines to be considered 'short', and 'short lines' are not right-justified. This can be a problem in that SCRIPT will assume you want lines in the middle of a paragraph to be non-justified. If 'Maximum' is set too close to 'Minimum', then most of the blank spacing will fall between the words, not between the characters. This is not necessarily objectionable; the appearance of your documents should please <u>your</u> eye, and SCRIPT is set up to encourage that result.

3. <u>Absolute</u> number of dot-spaces to be used when <u>Maximum</u> is exceeded. This is used when SCRIPT determines that a 'short line' is being processed. "Absolute" can be set to be the same as <u>Minimum</u>, but its supplied default is "1".

If "Min" and "Abs" are both zero, but "Max" is non-zero, then all padding will be placed between the words, not between the letters. If all three values are zero, right justification will not occur.

Examples

    .sd 1,5,1

Most of this document, including the paragraph you are reading right now,
uses the default values of 1,5,1.

T h i s   p a r a g r a p h   u s e s   v a l u e s   o f   6 , 9 , 6 .   Y o u   c a n   s e e   h o w
e v e r y t h i n g   i s   s p r e a d   f u r t h e r   a p a r t   t h a n   n o r m a l .

This paragraph uses values of 0,5,0. This is way the printer would print if left to
itself. The reason for forcing it to leave some white space between the letters is
pretty obvious. Note that SCRIPT still right-justifies nicely.

By contrast, this paragraph is done at 0,0,0. If you've tried to use the proportional
font of your printer before, this is what it used to look like, n'est pas?

.SH

        .SH   < <-> n>

This causes the printer to "Space Half" 'n' lines at a time (on printers that
support such spacing, of course). 'n' may be any number from -32768 to 32767. If
omitted, '1' half-space is performed. If going to the screen, '1' full space is
performed regardless of the value of 'n'.

End-of-page is raised within one-half space of the integer value currently in
effect, and an extra half-space will be generated to regain correct alignment for the
Bottom Titles and next page. Therefore, you don't have to do anything special about
it.

Judicious use of ".SH -n", ".LL", and ".AD" can allow you to produce multiple
columns on a single page, even though multiple columns really are not supported by
SCRIPT. However, if a page-eject occurs in the middle of this, your calculations will
be thrown off. Please do not try to create multiple columns unless you are prepared
to experiment.

".SH" differs from the half-line escape sequences in that it causes a control
break and a carriage return.

Examples

    .sh 3

That would leave 1-1/2 blank lines.

This shows how to get three-up columns. Remember, it isn't easy to do, and if a page-overflow occurs in the middle, it won't work at all.

The original text is:

```
.ll 20
Fourscore and seven years ago, our fathers brought forth
upon this continent a new nation,
.sh -8
.ad 30
conceived in liberty and dedicated to the proposition that
all men are created equal.
.sh -8
.ad 55
Now, we are engaged in a great civil war, testing whether
that nation, or any nation...
```

If I've calculated the '8' of ".SH -8" correctly, then SCRIPT will back up four lines (8 half-spaces), adjust the left margin, and produce the extra columns. I've also used ".CP 10" to ensure that there is enough room on the page, and am aiming for an overall line length of 66:

Fourscore and seven years ago, our fathers brought forth on this continent a new nation,

conceived in liberty and dedicated to the proposition that all men are created equal.

Now, we are engaged in a great civil war, testing whether that nation, or any nation...

.SK

```
.SK   <n>
        1
```

This causes a "SKip" of 'n' lines. A control break occurs, any text preceding the "SKip" is printed, and up to 'n' blank lines will be skipped on the printer. If end-of-page is encountered during this SKip, the remainder of the skip is cancelled. If you want an absolute number of lines to be skipped, even if some of those would be placed at the top of the next page, then use ".SP" instead of ".SK". Normally, ".SK" should be used in preference to ".SP".

If 'n' is omitted, one blank line will be placed in the printed output.

Examples

```
.sk
.sk 10
```

.SP

```
.SP   <n>
       1
```

This causes "SPacing" of 'n' lines in the printed output. A control break occurs, all text preceding the SPace is printed, and then 'n' lines (default '1') are left blank. If end-of-page is encountered in the middle of this SPacing, Bottom Titles (if any) are printed, a page eject occurs, Top Titles or automatic page numbering printing occurs, and the remainder of the "SPaces" are printed.

".SP" and ".SK" are identical except for the treatment of the end-of-page condition. If you don't want blanks at the top of the next page, use ".SK" instead of ".SP". Normally, ".SK" is used in preference to ".SP".

Examples

```
.sp
.sp 3
```

.SS

```
.SS
```

This forces "Single Spacing" of printed output to resume. It normally is used to cancel the effect of a previous ".DS".

If the "DS" run-time option is in effect when ".SS" is encountered, the run-time option is cancelled.

All text printed after ".SS" is printed single-spaced, unless a subsequent ".DS" control word is encountered.

Example

```
.ss
```

.ST

              .ST   <message>

    "STop" causes SCRIPT to process any text up to the STop, display the optional "message", and then pause. When you press the <ENTER> key, SCRIPT will resume execution.

    STop is used when operator intervention is required. This might involve changing diskettes when a large, multi-segment document is being processed, or it might be needed for a change of special forms in the printer.

    The "message" may contain anything you like. It should give a reason for the pause, and instruct the operator as to what should be done before <ENTER> is pressed.

    This control word should NOT be used for the purpose of inserting single sheets of paper when running with cut forms. The "ST" run-time option should be used for that purpose. It's easier, it's far more accurate, and it allows SCRIPT to decide what goes on which page.

Examples

        .st Place Invoice Forms # 1044 in Printer

The next example shows how large, multi-segment, multi-diskette documents are chained together:

        .st Change to Diskette #2, Press <ENTER>
        .ap part17/doc

.TB

```
.TB   c   n1,n2,n3...n16
      ;   1,5,10,15...
```

This defines the "TaB" character and "TaB" stops for use in monospace, no-format mode only. It provides a means for producing tables and columnar data. This is not a way to produce two-up multi-column output of the sort used in newspapers. Wide tables can be created using EDIT: the Viewing window can be moved right or left for this purpose.

'c' is a character, such as a semi-colon or '@' sign, that will appear in the text only to signal that a tab to the next tab-column should be performed. Be sure to select a character that will not be used as normal data while tabbing is in effect.

From one to sixteen tab columns may be defined, separated by commas. It is not necessary to specify more tabs than you expect to use, but if any of your text lines contain more tab characters than there are defined tabs, the right-hand side of such lines will be lost.

Tab stops are column-dependent, and are not expressed in inches or 1/10's of an inch, but in characters.

Tabbing is supported only in single-width, ".FO OFF", monospace modes only, because the calculations needed to support tabbing in proportional mode were considered excessively space-consuming and were therefore omitted from SCRIPT.

When underlining a tabbed heading, column alignment will be lost unless the body of the table (the parts not underlined) is preceded by another tab definition. This second definition must define the tabs to be "2" less than the positions used for the underlined text (except for the starting position). The second example below shows this technique.

Text may exceed any given tab zone or zones without error, so long as the final tab column is not exceeded. If your tabs are five columns apart and some of the text is eight characters long, alignment will resume at the next available tab column. Text is not overlayed.

The first text character is always printed in the column designated by the first tab value (tabs do not occupy any space by themselves). Therefore, if the first tab value is not '1', no text will be placed in column 1.

If the conditions for tabbing are not met (monospace, ".FO OFF", single-width characters), then the tab character is treated as normal text and will be printed.

Since the semi-colon (";") is the default tab character for NEWSCRIPT, it cannot be used in monospace (10, 12, 16 cpi) mode when ".FO" is off, unless you re-define the tab character to something else. Similarly, if you've decided to use the "@" sign as the tab character, you can't use the "@" sign as part of your text. For example:

```
.tb @ 1,15,30,50
Widgets: 20 @ $0.30 = $6.00
```

wouldn't work, since the "@" sign would be taken as a tab, and not as it the simple text character it was intended to be.

Examples

This input text (difficult to read because the text is not tabbed out yet):

```
.fo off;.bf16
.tb @ 1,5,10,20
 #@ DATE@ITEM
.sk
37@08/10/80@100 Widgets
41@08/13/80@ 17 Diskettes
55@09/01/80@ 1 Radio
```

will produce this result:

```
#   DATE        ITEM

37  08/10/80    100 Widgets
41  08/13/80     17 Diskettes
55  09/01/80      1 Radio
```

The following example shows how to underline the heading of a table:

```
.fo off;.bf16
.tb @ 1,20,30,40
!$ PART@QTY@PRICE@AMOUNT!%
.TB @ 1,18,28,38
WIDGET@ 5@ 0.25@$ 1.25
DISK@ 2@ 2.67@ 5.34
```

The result will be:

PART	QTY	PRICE	AMOUNT
WIDGET	5	0.25	$ 1.25
DISK	2	2.67	5.34

.TC

>   .TC   string

This causes the 'string' to be added as the next line of the "Table of Contents".
If 'string' is omitted, then a blank line will be printed in the Table of Contents.
'string' may be from 0 to 50 characters in length, and may contain any combination of
characters, including blanks and leading blanks.

The Table of Contents is written to its own disk file, and when SCRIPT goes to
end-of-job, you will be asked whether you want the Table of Contents printed
(default is "YES"). If there were no ".TC" control words in the document, this
question is not asked.

The Table of Contents is printed in a fixed format, over which you have no
control except for the font (".BF") in effect at the time printing of the Table begins.
Automatic page ejection is handled as necessary. The number of the page on which
the ".TC" entry was found is printed, right-justified, to the right of the 'string'.

If you are working with a document whose segments span more than one
diskette, you must ensure that the Table of Contents file will be written to a
diskette that is never removed during diskette changes. In this situation, you must
have more than one disk drive. You should make sure that the Table of Contents
file does not initially exist on any diskette that will be removed during document
formatting, before starting the SCRIPT run. The Table of Contents file is always
named according to the filename (but not the extension) of the file first specified to
SCRIPT. For instance, if you were printing "MSTR1/DOC", the Table of Contents
would be named:

                        MSTR1/TCT

A good place for it is on SYSRES (drive zero) or drive 2 or 3. If you are
processing a document that scans more than one diskette, neither the Table of
Contents file nor the Index file should be allowed to be written to any of those
diskettes, since they may be taken off-line later on.

If you are using NEWDOS/80 with the "AO" option not set to zero, but want the
T/C on drive 0, then you should create a "dummy" "fileid/TCT" file on SYSRES
before starting SCRIPT. Regardless of which disk you use, make sure it will have
enough room for the Table of Contents (and the Index, if it will be created as well),
and make sure the diskette is not write-protected.

If you switch diskettes around, it is possible for SCRIPT to find the wrong
Table of Contents file when it begins to print that table. If you remove the
diskette containing the started Table of Contents file before SCRIPT closes it, you
may destroy the integrity of one or more diskette Directories.

In other words, if using Table of Contents in a multi-diskette document,

BE CAREFUL

Example

```
.tc INTRODUCTION
.tc
.tc   Understanding Word Processing
.tc   Components of SCRIPT and EDIT
.tc SCRIPT
.tc   Control Words
```

.TM

```
.TM   <n>   |   <6>
```

This defines the number of lines in the "Top Margin". The Top Margin is the space between the top edge of the paper and the first line of the body. Top Titles and/or automatic pagination, if any, are placed in the Heading Space, which lies within this Top Margin.

The Top Margin defaults to '6', which is equivalent to one inch. The Heading Space defaults to '1', which is sufficient for printing automatic pagination; and the Heading Margin defaults to '1', leaving one blank line between the Heading Space and the body of the text. Set this way, four blank lines (6-1-1) are left over, and they are placed above the Heading Space.

The Top Margin, Heading Space, and Heading Margin interact, and the Top Margin value must never be less than the sum of the Heading Space and the Heading Margin.

Also see ".BM", ".HM", ".HS", ".PN", ".TT", and ".LS".

Example

This would provide for a larger Top Margin, and would be useful if printing on letterhead:

```
.tm 12
```

That provides for a two-inch Top Margin (6 lines per inch).

.TR

          .TR   n1,n2,n3

     This lets you specify a range of ASCII values (n1 through n2) that should be
translated to a different set of ASCII values. The translation is performed by
adding 'n3' to every character in your text that falls in the range of 'n1' through
'n2'. The control word is intended mostly for use in printing "screen graphics" on
printers that support such low-resolution graphics. The Epsons are examples of
such printers. The Microlines can print TRS-80 screen graphics also, but don't
require translation to do so.

     This does not provide a general-purpose translation table, nor does it provide
support for high-resolution or pin-addressable graphics (such as the capabilities of
the Anadex DP9500 or the MX-100).

Example

     If you wanted to draw something in graphics blocks on the screen of the
     TRS-80, you could do so in EDIT, using either "SHIFT-CLEAR" and the HEX
     values 80-BF, or by using a convenient symbol such as the "*", and then
     performing a global "ALTER" afterwards to convert the "*" to the large
     graphic block whose ASCII value is 191 (X'BF'). To print the result on an
     Epson MX-80, you would do the following:

     .sk;fo off
     .tr 128,191,32
                    ***   ***   *******   *
                     *   *   *   *
                     *   *   *   *
                    *********   *   *
                     *   *   *   *
                     *   *   *   *
                    ***   ***   *******   *
     .tr 0,0,0

(I used asterisks instead of ASCII 191's because my 737 doesn't print TRS-80
graphics). Note that formatting was turned off before the graphics began, and
SCRIPT was told to add '32' to all characters whose ASCII values fell in the
range of '128' through '191'. That happens to be the range of graphics on the
TRS-80, but on the MX-80, the graphics begin '32' higher. This is the way to
make the MX-80 print screen graphics from SCRIPT.

.TT

             .TT \<n\> /string1/string2/string3/
                  1

   This is used to specify from one to six "Top Titles", or headings. If 'n' is omitted, SCRIPT assumes you are defining or re-defining the first top title.

   The three 'strings' are, respectively, left-justified, centered, and right-justified. Any of them may be omitted by placing two delimiters in succession. The delimiter may be any character that does not occur in any of the strings.

   The Page-number Symbol (default is '$') may appear anyplace in any Title, and if found, will be replaced by the current page number. If any titles are in use, page numbers appear only if the Page-number Symbol is found in at least one title. ".PN OFFNO" will not suppress the appearance of the page number in a title.

   If more than one Top Title is defined, then the Heading Space (.HS) must be re-defined so as to be large enough to accommodate the number of Top Titles. It may be necessary to increase the size of the Top Margin in this case, since the size of the Top Margin must not be less than the sum of the Heading Space and the Heading Margin (TM >= HS + HM).

   Titles cannot contain escape sequences, underlining, or double-width characters. Titles are always printed in the font (10, 12, 16, or proportional) that was in effect when the titles were first defined.

   To produce just a left-justified heading:

                    .tt /THIS IS MY HEADING///

   Also see ".BT", ".HM", ".HS", ".PS", and ".TM".

Examples

       .tt /COMMANDS/SECTION III/$/

       .tt 1 /HISTORY/POLITICAL CURRENTS/Page $/
       .tt 2 //IN EUROPE//

The second example-pair would produce:

            HISTORY   POLITICAL CURRENTS   Page 7
                                IN EUROPE

.US

> .US   string

This is used to "UnderScore" (underline) some text. The text is the 'string' that follows the control word. There may be more than one word in the string, and the string, when printed, may span more than one line. Blanks are not underlined, but all other characters are underlined.

This is one of the few control words that does not cause a "control break", so you can use it to underscore words in the middle of the line (as I just did). UnderScoring may be used in combination with other features, such as double-width characters, although on some printers, it will only partially underline double-width material.

To underline blanks as well as text, use the special escape sequence explained at the beginning of SECTION IV.

Example

    This shows how just a few words
    .us can be underlined
    in mid-sentence.

The result will be:

    This shows how just a few words can be underlined in mid-sentence.

        THIS COMPLETES THE DEFINITIONS OF SCRIPT CONTROL WORDS

Page Layout in SCRIPT

The diagram below illustrates the areas on a physical page, as viewed by SCRIPT. The asterisks represent the edges of the paper, the dots show the ranges of the control words, and the lines represent margins. Note that the right-margin is defined implicitly: it is what's left over after ".AD" and ".LL" have been defined. In addition, the space above the Top Titles and the space below the Bottom Titles are defined implicitly: they are what is left over after ".HS" and ".HM" have been subtracted from ".TM"; and after ".FS" and ".FM" have been subtracted from ".BM".

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X    .                                                .   X
X.AD...|.............................LL....................|  .   X
X    .                                                .   X
X    .    |-----------------------------------------|    .   X
X    .    |                                         |    .   X
X    .    |    HS      TOP TITLES GO HERE           |  TM X
X    .    |         .                               |    .   X
X    .    |-----------------------------------------|    .   X
X    .         .                                         .   X
X    .         HM                                        .   X
X    .         .                                         -   X
X    .    |-----------------------------------------|        X
X    .    |                                         |        X
X    .    |        BODY OF TEXT GOES HERE           |    .   X
X    .    |                                         |        X
X    .    |                                         |        X
X  PL |...IN                                        |        X
X    .    |     ...OF                               |        X
X    .    |                                         |        X
X    .    |                                         |        X
X    .    |                                         |        X
X    .    |-----------------------------------------|        X
X    .         .                               -   X
X    .         FM                               .   X
X    .         .                               .   X
X    .    |-----------------------------------------|    .   X
X    .    |         .                               |    .   X
X    .    |    FS      BOTTOM TITLES GO HERE        |  BM X
X    .    |         .                               |    .   X
X    .    |-----------------------------------------|    .   X
X    .                                                .   X
X    .                                                .   X
X    -                                                -   X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

SECTION V - INDEXING

These features allow you to create an index for the back of a book. The procedures to follow are fairly simple and will result in a passable result. Production of a really good index will take some work on your part: selection of the terms that should be included in the index, weeding out page references that shouldn't be there, etc. On the other hand, if you're writing a book that would benefit from an index, you'll be glad to participate in such an effort, as long as the computer is able to do most of the work. And that's where NEWSCRIPT can help.

There are five steps to producing an index:

1. Select the words and phrases to be indexed;

2. Place markers for these words throughout a document;

3. Run SCRIPT, which will produce a list associating each marker with the page assigned to it;

4. Run INDEX, which will sort (alphabetize) and merge these markers into coherent form;

5. Run SCRIPT again to format and print this index.

Step 1 must be done by a person. Step 2 can either be done manually (that is, use EDIT, scan the text, and insert markers where they are needed) or automatically through use of the GENINDEX component of NEWSCRIPT. Steps 3-5 are done automatically, although you can stop at any point you wish.

The remainder of this chapter discusses each of these five steps as they pertain to using NEWSCRIPT.

Selection of Words and Phrases

In terms of planning and mental effort, this is the most difficult part of Index creation. In many cases, you'll want to pick terms that don't appear in the document quite the way they should appear in the index. There will be singulars and plurals, tenses, capitals, etc. NEWSCRIPT helps a little: the GENINDEX and INDEX programs are "case-blind" ("A" and "a" are the same to them).

NEWSCRIPT allows you to use single words and/or multi-word terms as appropriate, so you can concentrate on making the index useful to your readers, and let the mechanics take care of themselves.

Placing Markers In The Document

A "marker" is just a way of telling NEWSCRIPT that a particular word/phrase should be indexed. The marker is a control word:

.IX

followed by the term to be indexed:

.ix transistors

The marker and its word/phrase should be placed in the document directly after <u>each</u> <u>line</u> containing the information or word that is to be referenced in the index:

        The twentieth century has seen enormous strides
        forward in many areas. One of the most visible
        of these is in solid-state electronics, starting
        with the transistor. It would have been
        .ix transistor
        impossible to predict how this small device...

Most "control words" in NEWSCRIPT cause a "control break": text on the lines after such control words are formatted starting on a new line. A few control words do <u>not</u> cause a control break: after performing the requested function, the next line of the input document is treated as a continuation of the line before the control word. This distinction is relevant here because ".IX" does <u>not</u> cause a control break. So, be sure to place the "marker" exactly where you want it.

To save a little space and typing, you can place several index references on a single line. If you do this, you must separate them by semi-colons:

        .ix transistors;electronics;technological progress

You should only do this if the text line contained multiple references, of course.

If you believe that automation can save you time, then you may wish to try the "GENINDEX" component of NEWSCRIPT. This program takes a word/phrase list (created by you using EDIT), reads a specified file, and inserts markers wherever it finds a match. To use it:

    1. Create the list using EDIT
    2. RUN "GENINDEX"
    3. Specify the Word List file
    4. Specify the input document I.D.
    5. Specify the output document I.D. to be used.

The output file will contain the entire input file, plus the inserted markers. The output file I.D. must not be the same as the input I.D.

The advantage to this approach, of course, is the speed and accuracy of the computer. The drawback is that it'll pick up <u>every</u> reference, including irrelevant ones; and it'll miss slight variations that a human would have caught. You've probably heard the computerese term, "GIGO" (garbage in, garbage out). Use of GENINDEX without careful planning is one of the more extravagent ways of demonstrating this phenomenon. However, you can use GENINDEX as a first-pass, and then go through the result with EDIT and eventually wind up with a really classy result.

Running SCRIPT

A normal run of SCRIPT, with <u>five</u> files specified for BASIC, will create a formatted printout and up to two new files. The first of these contains any Table of Contents references, which will be processed after printing is complete. The second new file contains the raw index references, and must be processed further. This file is created automatically if any ".IX" control words are found.

Running INDEX

"INDEX" is a combination BASIC/Machine Language program. It reads the raw references, sorts (alphabetizes) them (fairly fast, using machine language) without regard to upper / lower case, then merges all page references in ascending order for each word / phrase. If a word was referenced more than once on a given page, the page number will be retained only once. The output of "INDEX" is itself a SCRIPT file, and can be processed and printed immediately by SCRIPT.

If SCRIPT creates an Index file, it will ask you whether you want to produce the index (default is "NO" and the index should be bypassed until you're satisfied with the body of the document). If you reply "Y", it'll run "INDEX" for you, passing along the correct name for the index file. The "INDEX" program will ask you to verify the input file I.D., perform its sort/merge function, and then pass control back to SCRIPT.

Re-running SCRIPT

The second run of SCRIPT prints the index. All you have to do is hit the <ENTER> key each time a choice is presented, and the correct defaults will be taken. NOTE: if you EDIT the original document immediately afterwards, the default will be the name of the Index file, not the name of the original document.

The index is printed half-width, but not 2-up. If there are too many page references to fit on one line, the extras will be slightly indented (offset). If you want your camera-ready document to have a 2-up index, it'll be necessary to cut and paste.

SUMMARY

Indices are normally thought of as being appropriate for large books and manuals. However, since NEWSCRIPT makes it so easy to produce passable-to-good indices, it may well be worth your while to make them for small reports as well.

Punch

SECTION VI - HOW TO...

This section contains hints, references, and examples for a number of things that you may want to do with a Word Processor. As time goes by, it'll probably be revised (that's a euphemism for "made more relevant") based on the feedback we get from our customers. The topics covered here are:

* Letters
* Form Letters
* Mailing Lists
* Tabbing
* Titles
* Table of Contents
* Indexing
* Pre-defined standard setups
* Long documents (big documents)
* Large EDIT files
* Scanning and searching
* Underlining
* Centering
* Italics
* Double-width letters and headings
* Double-spaced output
* Bullets (Hanging Indents)
* Keyboard Debounce and Repeat Speed
* Avoiding lost files
* Seeing wide lines off the end of the screen
* Graphics

LETTERS

This deals with personal or business correspondence. The next topic deals with Form, or Repetitive Letters.

If you just want to write a simple letter, with your return address, the date, the name and address of the recipient, the body of the letter, and the signature: take a look at "EDIT1/EX", which is listed in SECTION I of this manual and is also on the distribution diskette.

The control words you'll probably want to use in letters are:

```
.IN 35    - indent to type your address on the right
.FO OFF - prevent the address lines from running together
.SK       - to skip a line (leave a blank line)
.PP       - to start a new paragraph
```

Now, if you want to type your logo at the top of the letter, there are two ways to do it: you could enter it at the beginning of every letter you write (borrrrinnnggggg), or you could place your logo in a file of its own, then imbed that file each time you need it. Being addicted to laziness, I'll just show you the "imbed" approach.

Using EDIT, create a tiny file called "LOGO" (I like to be creative in my choice of names). It would contain something like this:

```
.ce on
SUPERIOR WIDGETS, INC.
4411 Franklin Ave.
New York, N.Y. 10023
(519) 483-2500
.ce off;.sk 2
```

The logo file could be saved on the Word Processing diskette itself, since it will be used frequently and takes very little room. Notice that I put a ".sk 2" at the end of it to avoid having to type the spacing into the main document every time.

To use this in a letter takes only one line:

```
.im logo
.in 35
November 5, 1980
.in0;.sk 2;.fo off
G. Willekers
123 Hope Street
Compton, Ca. 92302
.sk 2
Dear George,
.fo on;.pp
Thank you for....
```

FORM LETTERS AND MAILING LISTS

When you want to send the same letter to a number of different people, you're dealing with form letters. If you want to merge such letters with name and addresses that are in another computer file, then the form letters will be sent to the people in that mailing list. Of course, you may just want to enter the name and address by hand, but still have a "canned" letter when you do this.

If you want to do mailing lists and form letters, these are the steps to follow:

1. Create the mailing list using EDIT or some other program;
2. Create the form letter using EDIT;
3. Print the series of letters using SCRIPT.

The formats of mailing lists are described under the ".RD" control word in Section IV. The ".RD" control word is what you put into the form letter where the recipient's name and address is to be printed.

If you want to type the names in as needed, you still use ".RD", but don't specify a file. Just specify the number of lines to be entered from the keyboard. Allow for the largest number of lines you expect to need, and when entering fewer lines, just hit the <ENTER> key to add blank lines. When using this approach, I leave space for two extra lines: one to be left blank, the other for the "Dear ..." line.

Names and addresses can be placed into Form Letters from the keyboard or from a mailing list file. When a file is used, it can be in either of two formats, as described in SECTION IV under the ".RD" control word. When the "variable" format is used, a "code line" must begin each entry in the file. This acts as a kind of key to indicate where a new entry may be found, and also to allow extraction of certain Names and Addresses (N&A's) on a selective basis.

If you find yourself using these codes for several purposes, you'll want to make sure that identical-looking codes don't accidentally occur. One way to minimize such duplication is to place spaces or periods between the codes. For instance, if the code line signal in a given file was the pound sign (#):

        #.A.VP.03.

When supplying the selection key, as described in SECTION IV, the periods would be included to ensure uniqueness:

                        .VP.

Form Letters are discussed in some detail, with several examples, under ".RD" in Section IV.

TABBING, TABLES, AND COLUMNS

Tabbing takes place at print time, so EDIT won't show you the final positions of your tabbed lines of material. However, you can turn ".FO OFF", and then use EDIT to lay your tables out exactly the way you want them to be printed. Section IV contains a complete description of tabbing under the ".TB" control word.

There are several things to remember when setting up tables:

1. Turn format off before the table starts: .FO OFF
2. Use a "monocase" font, not a proportional one.
3. When creating the table, move the window back and forth.
4. If you use EDIT's "CHANGE" command, turn FLOW OFF.
5. Underlining of table headings is explained under ".TB".

TITLES, HEADINGS, AND FOOTINGS

"Titles" are repeated at the top (and/or bottom) of every page, until they are replaced or blanked out. There is enough room in the default setups for a one-line title at the top of the page ("top title", or ".TT") and a one-line title at the bottom of the page ("bottom title", or ".BT"). If this isn't enough, you can define up to six lines each of top and bottom titles. However, you must also tell SCRIPT to make room for these extra lines. You do that through the "Heading Space" (.HS) or "Footing Space" (.FS) control word.

If you plan to use titles, you should probably study the "PAGE LAYOUT" diagram at the end of SECTION IV, since it illustrates the effect of each control word involved.

There should also be one or two blank lines between the title and the body of the document. This is called the "Heading Margin" (.HM) or the "Footing Margin" (.FM).

Finally, you have to leave enough room at the top and/or bottom of the page itself. These areas are the "Top Margin" (.TM) and the "Bottom Margin" (.BM). The Top Margin includes the space above the title, the title itself, and the space between the title and the body of the document. You can define the Heading Space, the Heading Margin, and the Top Margin. What's left over, if anything, is the space above the title, and that is not defined except by arithmetic. (All of this applies in reverse to the bottom of the page.)

Here are some things you should remember about titles:

1. They remain in effect until cancelled;
2. They are printed in the format used when defined, not the format in use on any particular page, and with the line length in effect when they were defined;
3. Their contents cannot be underlined, italicized, or double-width.
4. Top titles take effect on the <u>next</u> page (unless you're at the top of the first page of a document), and bottom titles take effect on the <u>current</u> page. So, if you want to change a top title, do so before you force a new page via ".PA".

Headings

Headings are different from titles. Headings change frequently, and often are centered, underlined, double-width, or some combination of these. NEWSCRIPT supports all of this!

THIS IS A HEADING

That was done as follows:

```
.sk;.ce
!$!(THIS IS A HEADING!)!%
```

A control word was used to center the text, and two pairs.of "escape sequences" were used to both underline and double-width the material itself:

    !$!%   starts and stops underlining
    !(!)   starts and stops double-width

(These don't.work on all printers. For instance, double-width won't work on a daisy wheel printer.)

You can use either or both of these escape sequences; they do not have to be used together.

If you want a left-justified, underlined heading, you could still use the escape sequence (but omit the ".CE"), or you could use the ".US" control word. It differs from the escape sequence in that it does not underline blanks.

TABLE OF CONTENTS

NEWSCRIPT's Table of Contents feature lets you supply text that is to be assigned a page number, collected, and then printed coherently. To save time, the T/C is printed at the end of the document, so you must move its page(s) to the front.

The T/C is printed in the font currently in effect, not in the font(s) that were in effect as each individual entry was encountered. Therefore, you can select the font by making it one of the last lines of the document (".BF").

The material that follows the ".TC" control word is saved, along with the current page number, and then printed along with all other ".TC" entries after the document ends. You can place blank lines in the T/C by not providing any text after ".TC" (SCRIPT will not print the page number in this case; it knows what you mean). You can also leave some leading blanks if you want to indent the material, but no control words are used during the actual printing of the T/C, so you can't use ".IN".

The material for the T/C is written out to a disk file, and this disk file must remain on-line throughout the processing of the document. This means you can't use ".TC" with a multi-diskette document if you have only one disk drive. If you have two drives, make sure the T/C is written to drive 0 so that it will never be taken off-line. To do this, remove the write-protect tab on drive 0, make sure there are some free granules on drive 0, and make sure that there is no Table of Contents file (extension: /TCT) on any other drive (1-3). By the way, these same procedures apply to creation of an Index.

CREATING AN INDEX

This is covered in Section V.

STANDARD SETUPS

You may find yourself using the same sequences of control words over and over throughout a document. If these are short, the extra typing may be acceptable. However, if the sequences become long, or are complex and difficult to remember, you may wish to create them once, carefully, and then store them on disk as a reference library.

These pre-defined setups may be brought into your document in either of two ways: you can imbed them (".IM fileid"), or you can GET them ("GETFILE fileid"). The first approach saves space, requires additional time when SCRIPT runs, and cannot be used within another imbedded file. The second approach makes the setup a permanent part of the primary file at EDIT time. I don't really have a recommendation to make on this one, since I usually just type the sequences several to a line (and that's a third way to do it).

Typical uses of ".IMbed" are:

           * logo at the top of a letter
           * signature at the end of a letter
           * legal clauses to be placed within a contract

However, "Form Letters" should <u>not</u> be done with ".IMbed". They should be done with ".RD" (Read from file).

LONG DOCUMENTS

This could be taken in a couple of ways. If you want to print on legal-size paper (long paper), just set the page length (".PL.") to 84:

.PL 84

However, what I really wanted to discuss here is large documents such as this manual (well, maybe not that large, but bigger than 3 to 6 single-spaced pages).

SCRIPT imposes no upper limit on the size of a document (well, not totally true: when titles are being used, it can only page number up to 999), but EDIT can handle only as much text as it can fit into memory. On a 48K machine, that's comes to about 2,000 words of text. If all of NEWSCRIPT needed zero room, and all you had was the ROM and the Operating System, you could fit perhaps 6,000 words in memory, which would be nicer, but would only postpone, not solve the problem.

The solution, of course, is to edit several small files, and let the "Append" (.AP) control word chain them together. (Both EDIT and SCRIPT understand this.) "Append" is the last control word of a file, as far as SCRIPT is concerned, since it immediately switches to the identified file and never comes back to the original one.

This scheme works fine until you get to medium-big manuscripts (8,000 - 16,000 words on a Model I, 20,000 - 30,000 words on a Model III). Documents larger than that will not fit on a single diskette. However, SCRIPT also has a control word that lets you switch disks as needed. That is the "STOP" (.ST) control word, not to be confused with the "STOP" run-time option (which is used for cut-sheet paper). You can place whatever message you wish after this control word. When it is encounterd, the message is displayed and the computer stops until you press any key. This gives you whatever time you need to switch disks, change paper, or whatever.

You can chain files and disks together indefinitely, so the number of disks you can afford becomes the real limiting factor in the size of a manuscript. (A smaller limit might be how much you need to say, of course.)

WHAT TO DO WHEN A FILE BECOMES TOO LARGE FOR EDIT

When creating a document, it's good practice to leave some room for future growth. So, if you expect to revise your text considerably, it's wise to not go much over 150-200 lines. Of course, even if you adhere to this guideline, the additions later on may push a file to the limits of memory, and at some future time when you attempt to edit that file, you may get this message:

FILE IS TOO LARGE TO BE EDITED ALL AT ONCE

DO NOT PANIC!!!' NEWSCRIPT includes a utility program called "FITLINE", and it will divide a large text file into several smaller ones. It's described in Section XI. Just run your file through it, allowing perhaps 150 or so output lines per new file. FITLINE will always divide the text at a SCRIPT control word, so you'll have something of a logical breaking point in the result. FITLINE will also generate the necessary ".APpend" control word, so EDIT and SCRIPT will be able to chain from one file to the next.

After running FITLINE, just run EDIT again, and continue from where you had left off.

FITLINE may also be used to "flow" text from one line to another just for the purpose of combining short lines. This is strictly cosmetic for editing purposes, since SCRIPT pays no attention to how the text appears during EDIT: SCRIPT formats based solely on control words.

SCANNING AND SEARCHING

One of the marvelous, hidden uses of an editor is its ability to quickly (and accurately) search through general material and find things that match other things. In the case of EDIT, there are four ways to do this:

1. "LOCATE" will search a file from the second line on the screen through the end of the file, staying within the currently defined ZONE, if any, until it finds a match to the argument you've given it, or reaches the end of the file, or until you press any key to interrupt the search. If a match to the argument is found, the line containing it becomes the current line (the top one on the screen).

2. "LOCATE-NOT" is like "LOCATE", but searches for the first line that does not contain the argument. If every remaining line in the file does contain that argument, the "STRING NOT FOUND" error message is displayed. Of course, in this case, it really means "STRING NOT NOT FOUND", but that would be even more confusing.

3. "FIND" scans down the left-hand side, looking at the beginning of each line for a match. When it does this, it ignores column positions that are blank in the search argument.

4. "CHANGE" can be used to display all occurrences of a given string: just make the 'old' and 'new' strings identical, and specify global range:

                   change /widgets/widgets/ *

Bear in mind that although CHANGE will display what it finds, EDIT will replace what's on the screen with the current data lines as soon as CHANGE ends.

UNDERLINING

There are two ways to do underlining: the ".US" control word, and the "!$!%" escape sequence pair.

The ".US" control word underscores only the text that follows it on the same line (this is an exception to the general rule that text does not occur on the same line as a control word), and it does not underline blanks between the words. So, the control word approach should be used when only the words, and not the spaces between them, are to be underlined. If the resulting text prints on more than one line, it will, of course, be underlined on all necessary lines; but the input text all has to be on one line. If you want to underline words, but not blanks, across several text lines on the edit screen, then precede each line with the ".US" control word.

The escape sequence is more flexible and easier to use: just place the starting sequence ahead of the text that is to be underlined, and the ending sequence directly following that block of text. The block may be of any length, and underlining will span several printed lines if necessary. Within the block, everything will be underlined, including the blanks between the words.

The escape sequences normally are placed right inside the text, not on separate lines of their own. For instance:

We want to !$underline just!% those two words.

will produce:

We want to underline just those two words.

(I reset the escape character using "\ES" to show you what I was doing.)

A complete list of the escape sequences is given in Section IV, under the ".ES" control word.

Underlining can be done in conjunction with centering, and if done through the escape sequence, even a single letter can be underlined.

CENTERING

Centering is very easy with NEWSCRIPT. All you have to do is precede the text with the ".CE" control word. You can center one line or many lines this way. If you know how many lines you want centered, you can specify the quantity; otherwise, you can turn centering on or off as needed.

Both single and double-width characters in any font may be centered. Within the limitations of the printer (one character position plus or minus in some cases, one dot-space in others), centering will be effective even when single and double-width characters are inter-mixed. However, intermixing of fonts will usually result in non-centered text.

ITALICS

Italics are possible if your printer has an italics character set. As this is written, the only printers having this feature and supported by NEWSCRIPT are the Epson MX-80 with the GRAFTRAX option, and the Epson MX-100.

To do italics, just surround the word or phrase with the italics escape sequence pair. For example:

Italics are !/very easy!? to produce with NEWSCRIPT.

If this had been printed on a suitable printer, then "very easy" would have been italicized.

Like underlining and double-width, any word, phrase, or block of text can be in italics, even if several lines are printed as a result. In fact, all <u>three</u> can be in effect simultaneously, and the result will be double-width, underlined, italics. On the Epson, Emphasized, over-strike, or Double-Emphasized modes could be used in combination with any or all of those high-lighting features, and the result would really stand out on a page.

DOUBLE-WIDTH <u>LETTERS</u> <u>AND</u> <u>HEADINGS</u>

"Letters" in this context means characters such as "A", "b", "+", and so forth. Many dot-matrix printers have a double-width mode for each of the normal fonts they can print. SCRIPT can turn this feature on and off based on the escape sequence:

!(!)

Everything between this pair of sequences will be double-width, if the printer can do it and if SCRIPT knows how to tell the printer what to do. The result can print on several lines, if necessary, even if the printer normally turns off double-width at the end of each line.

DOUBLE-SPACED <u>OUTPUT</u>

There are two ways to double-space a printout: the easier of the two is the "DS" run-time option of SCRIPT. If you select this option when asked, the entire document will be printed double-spaced. The other approach is to use the ".DS" control word. This gives you more flexibility, since you can turn double-spacing on and off as needed. However, it's harder to remove from a document than the run-time option, which isn't there in the first place.

BULLETS (HANGING INDENTS, OFFSETS)

The explanation below is presented as a series of six bullets, with the numbers "hanging out" relative to the rest of the text.

1. Whenever you want separate thoughts or points to be highlighted, you can use hanging, or delayed indents.

2. SCRIPT supports two kinds of hanging indents. The first kind is called a "hanging indent" (".HI") and the second kind is called an "offset" (".OF"). The two are very similar. The difference is that ".HI" resets itself for the next bullet each time any control break occurs, while ".OF" must be reset by another ".OF". If your text won't have any blank lines or other control breaks within a bulleted paragraph, ".HI" is more convenient to use.

3. Each bullet must be preceded by a control word line containing ".HI" or ".OF" and the number of tenths of an inch by which the subsequent lines should be indented.

4. The indentation value depends on the nature of the bullet. Numbers or letters followed by periods do best with ".HI 4", while single characters such as asterisks and dashes do best with ".HI 3" (or ".OF 2").

5. If you use ".OF", then each bullet must be preceded by another ".OF n" (where "n" is a number), since SCRIPT has no magical way of knowing where one bullet ends and another begins. If you use ".HI", then SCRIPT will assume that each bullet ends at the next control break.

6. When you are finished with the last bullet, you must remember to set ".HI 0" or ".OF 0" (turn it off).

7. Examples of how this is done may be found in Section IV, under the ".HI" and ".OF" control words.

KEYBOARD DEBOUNCE AND REPEAT SPEED

It is possible to adjust the degree of keyboard debounce and the key-repeat speed of NEWSCRIPT. The menu program called "NSINIT" contains lines of code for this purpose. If you wish to make these adjustments, load "NSINIT" while in BASIC, scan through it for the comments regarding the adjustments you wish to make, change the numbers accordingly, and then SAVE "NSINIT" back to diskette.

AVOIDING LOST FILES

Files can be lost for a variety of reasons. You can protect yourself against most, but not all of these. The list below is necessarily incomplete, but attempts to identify the more common situations.

Disk Full or Write-Protected. When this happens during editing, remove the write-protect tab, if there is one, and try again to save the file. Don't give up and restart EDIT, or go back to DOS. If the disk really is full, find another formatted disk and try to save your file on it. You can always move it to its proper home later on, but only if you've saved it in the first place.

It's also possible that a file of same name as the one you're now using already exists on a write-protected disk. NEWSCRIPT lets the DOS perform all file handling, and DOS will try to replace a file if it finds it. Since it can't write to a protected disk, it'll generate this error. When that happens, you can do any of the following:

1. Remove the write-protect tab and try again to "SAVE" or "END";
2. Remove the write-protect tab and KILL the spurious file, then replace the tab;
3. Specify an explicit drive number as part of the filespec:

                    name test/fil:1
                    save test/fil:1
                    end test/fil:1

Using the wrong file name. The trivial version of this is that you haven't lost the file at all, but merely given it a name you don't recognize. The serious version of this occurs when you use the name of a different file, thereby destroying that other file and possibly winding up with two almost identical copies of the current file. This error is less likely to happen with EDIT than with BASIC, since you don't have to supply a file name when saving a file.

EDIT Error. Perish the thought! But, if it does happen, or if there's a hardware glitch (likely story), all is not necessarily lost. If you get a BASIC error and remain in BASIC, you can try to correct the problem and get EDIT to continue (use BASIC's "CONT" command or, even better, its "GOTO" command). Don't use "RUN" until you've given up, since that will cause loss of the in-memory copy of the file. To use the "GOTO" approach, you must find the line to which BASIC should go. The line number shown below can change from release to release, but the content is stable and also unique, so you can look for such a line and GOTO it. If you don't know anything about BASIC programming, you may not wish to attempt this kind of recovery, but the steps following the program / hardware / disk error are:

1. Hit <BREAK>
2. Type: LIST 1358. It may or may not be the right line.
3. The line you need to find says: OPEN "O",1,F$
4. LIST the entire program if necessary to find this.
5. Type: GOTO 1358 <ENTER> (or whatever the line # is)

If you keep getting errors, try changing the value assigned to "F$":

    1. Hit <BREAK>
    2. TYPE: F$="TEMP/NAM"
    3. Hit the <ENTER> key.
    4. Type: GOTO 1358 <ENTER>   (or whatever the line # is)

    If you aren't a programmer, this may have been unclear or even meaningless (if it was unclear to programmers, I apologize for my poor explanation). However, if you have an hour or two of text tucked away in memory and can't get it out to a disk, come back to this explanation and try it again. Better still, when you have some spare time (whatever _that_ is), just experiment with this by editing a file and hitting <BREAK>. Of course, don't do it with a file that means anything to you!

Losing the Screen. If you're running EDIT and hit <BREAK>, EDIT will save the screen image before honoring the <BREAK>. However, if you're running, say, NEWDOS/80, and hit "DFG", then the screen image becomes whatever NEWDOS/80 puts up there. If this isn't what you want in your current text file (and it won't be), then when you type "MDRET", you should get to EDIT's command line (SHIFT up-arrow), type "Whoops", and hit <ENTER>. The "Whoops" command will restore the screen for you. If you just hit <ENTER>, or issue some other command, the NEWDOS/80-produced material will replace whatever used to be in that part of the file.

WIDE LINES

    EDIT can handle lines of up to 255 characters, but since the screen can display only 60 characters per line, EDIT normally breaks your text up into 60 character (or less) segments. SCRIPT re-combines them for printing, so an optimum balance is maintained for you: all your material is visible, but it can be printed differently than it appears on the screen.

    However, there are situations where the text may be wider than 60 characters: a table created that way by using EDIT; or a file created elsewhere and being edited now, or even a file created by SUBEDIT (predecessor to NEWSCRIPT) and not yet processed by the "FITLINE" utility.

    Wide lines can be seen in segments of 60 characters at a time. The video screen may be thought of as a window, and the document as a long, wide piece of paper. "UP" and "DOWN", "FORWARD" and "BACK" will move that window vertically. "VIEW", followed by a numeric value, will move it horizontally. By using "VIEW", you can move the window back and forth across wide lines. When you do this, you may wish to turn on the GRID to help you keep track of where you are.

GRAPHICS

Graphics has to do with pictures and designs, rather than with words and text. Some printers can handle only letters, others can also print the same kinds of graphics characters as the TRS-80 can display (this does not include the clever little characters on the Model III, such as card suits), and still others can print small dots in any pattern at all.

The last of these, sometimes called "pin addressable graphics" or "graphics mode", is not directly supported by NEWSCRIPT. It's possible to create the necessary control sequences through the use of <SHIFT><CLEAR>, but that would take a _very_ long time, and be impossible to verify or change should you make a mistake. It's also possible to create the necessary graphics streams outside of NEWSCRIPT, perhaps through use of a BASIC program, and then to ".IMbed" the resulting file for printing. That would be a lot easier, but still not a solution totally within NEWSCRIPT. If you have a good graphics package, you can use it to produce what you need, and then combine those results with NEWSCRIPT output by doing a paste-up.

There _is_ a form of graphics supported by NEWSCRIPT, however. It works easily on with the Microline series: just use <SHIFT><CLEAR> or "ALTER" to create screen graphics blocks on the TRS-80. These graphics can be part of a normal text file. Then, save the file and run SCRIPT. The printer will recognize and print the result.

If you're using an Epson, you can do the same sort of thing, but the Epson's notion of TRS-80 graphics is slightly different than the TRS-80's notion. In fact, they differ by a value of '32' for each graphics character. NEWSCRIPT lets you get around this easily. Just draw your picture using <SHIFT><CLEAR> and the necessary "hexadecimal" values defined for TRS-80 screen graphics. A much easier way would be to use asterisks, then convert them to the final screen graphics characters by using the "ALTER" command of EDIT. In either case, once the graphics have been produced on the EDIT screen, you should surround the resulting block of material with the ".TR" (translate) control word. To enable the "shifting" of values, specify the range and the degree of shift. To disable this, specify all zeros. The specific values to use for the Epson printers are:

```
                    .TR 128,191,32
                        ... TRS-80 graphics blocks go here ...
                             for as many lines as you need
                    .TR 0,0,0
```

Remember, this doesn't work on all printers, only on the ones that support TRS-80·screen graphics.

THIS CONCLUDES THE "HOW TO" SECTION

SUMMARY OF EDIT COMMANDS

COMMAND	OPTIONS	PURPOSE
ALter	/s1/s2/ <n <g>>	like Change, but for chars, not on keyboard
AUtosave	<n>	set or display autosave limit
Backpage	<n>	scroll 'n' pages towards top of file
BOttom		set current line pointer at last line of file
BReak	/s1/ <B>	split current line into 2 lines at s1
Change	/s1/s2/ <n <g>>	change string1 to string2
COpy	n dir m|target	copies 'n' lines UP or DOWN 'm' lines, or TO target
DBlanks	<ON>|<OFF>	delete or save blank lines into file
DElete	<n>	deletes 'n' lines starting at current line
DIr	<:n>	display DIRECTORY (NEWDOS/80, DOSPLUS, LDOS, MODEL III)
DString	/s1/	deletes from current line to just before first line containing 's1'
Down	<n>	moves current line pointer down 'n' lines
ENd	<fileid>	saves current file, terminates editing
FInd	string	performs column-dependent search
FLow	<ON>	<OFF>	splits long input lines to screen size
Forward	<n>	scroll forward (towards eof) 'n' pages
FRee		shows lines in use, left, available string space
Getfile	fileid </s1</s2>>	gets all or part of 'fileid'
GRid	<ON>|<OFF>	turns a column grid-guide on or off
Hardcopy	<n> <x<x>>	prints part or all of the file
Insert	ssss	inserts 'ssss' after current line
Join	<B>	concatenates current and next lines.
Kill	fileid	kills a disk file
LEngth		returns length of current line
Locate	/string/	scans within current zone for a string. If delimiter is '-', it locates first line NOT containing 'string' (locate-NOT)
Move	n dir m|target	moves 'n' lines UP or DOWN 'm' lines, or TO target
NAme	<fileid>	displays current File I.D. or changes it
Next	<n>	moves current line pointer down 'n' lines
QUit		terminates editing without saving file
Replace	ssss	replaces current line by 'ssss'
SAve	<fileid>	saves file to disk, remains in Edit
STring	n s	creates a string, 'n' chars long, of 's'
Top		sets current line pointer to first line of file
Up	<n>	moves current line pointer 'n' lines towards top
View	<<s>n1,<<s>n2>>	sets or displays columns to be displayed on screen 's' can be '+' or '-' to make a relative shift right or left
Whoops		ignores changes on screen and refreshes it
X	<cmd>	<n>	sets or uses an indirect command 'n' times
Y	<cmd>	<n>	like X. allows a second value to be used
XY	<n>	alternates execution of X and Y settings
Zone	<n1,<n2>>	sets or displays columns to be scanned by Change or Locate.
/		shorthand for 'locate', acts as the delimiter
-		shorthand for 'locate-NOT', acts as the delimiter
=		repeats the last-entered command
?		brings last command back to command line for re-use
&		allows current command to remain on screen for re-use

LIMA commands: D<n> I<n> R<n> / .A .B .C
CONTROL KEY: <CLEAR> FUNCTIONS: I, D, blank, right-arrow

SUMMARY OF SCRIPT CONTROL WORDS

WORD	OPERANDS	DESCRIPTION
.AD	n	left margin size, in 1/10 inches
.AP	fileid	continuation file, append to current file
.BF	<10>|<12>|<16>|<737>	begin font, proportional or c.p.i.
.BM	<n>	# lines for bottom margin
.BR		break: temporarily suspend concatenation
.BT	<n>/s1/s2/s3/	bottom title
.CE	<n>	<on>	<off>	center next line(s)
.CM	<anything>	comment: totally ignored
.CO	<on>	<off>	concatenate input lines
.CP	n	conditional page eject if fewer than 'n' lines left
.DA	<0,1,2,3>	number of overstrikes for darkness
.DS		double space
.ES	c	re-define the within-text escape character
.FM	<n>	number of lines for footing margin
.FO	<on>	<off>	format (justify and concatenate)
.FS	<n>	number of lines for footing space
.HI	<n>	hanging indent at each control break
.HM	<n>	number of lines for heading margin
.HS	<n>	number of lines for heading space
.IM	fileid	imbed contents of 'fileid' here, then continue
.IN	<+	->n	indentation in tenths of inch, abs. or rel.
.IX	string	defines an entry for the Index
.JU	<on>	<off>	right margin justification
.LL	<+	->n	line length in tenths of inch, abs. or rel.
.LS	n	number of lines manually bypassed on first page
.OF	<n>	offset (indent) lines AFTER this one
.PA	<n>|<Odd>|<Even>	page eject <next page number>
.PL	<n>	number of lines in physical page size
.PN	<on>	<off>	<offno>	automatic page numbering
.PP	<cp<,sk<,in<,sh>>>>	start new paragraph
.PS	<$>	<c>	page-numbering symbol, used in title definitions
.RO	n|c <fileid>	imbed 'n' as-is lines from keyboard or 'fileid'
.SD	min,max,abs	proportional inter-character dot-spacing
.SH	< <->n>	space 1 or more half-lines up or down
.SK	<n>	space 'n' lines or to bottom of page
.SP	<n>	space 'n' lines, possibly across page-end
.SS		single-space
.ST	<message>	stop for operator action, then continue
.TB	c n1,n2...	tab character and column settings
.TC	string	table of contents entry
.TM	<n>	number of lines in top margin
.TR	n1,n2,n3	add 'n3' to all chars in range 'n1' - 'n2'
.TT	<n>/s1/s2/s3/	top title
.US	string	underscore 'string'

SUMMARY OF WITHIN-TEXT 'ESCAPE' COMMANDS

!(	begin double-width characters	!)	end double-width
!$	begin underlining	!%	end underlining
!+	half-space down for subscript	!-	half-space up for superscript
!/	begin italics	!?	end italics
!<	backspace one character		

SUMMARY OF EDIT RUN-TIME OPTIONS

After a "QUIT" or "END" command completes, this question is asked:

PASS 'fileid' TO SCRIPT (Y/N/C/A, DEFAULT = Y) ?

Y	YES, chain to and run SCRIPT
N	NO, just return to BASIC
C	CONTINUE editing the current file
A	EDIT the file Appended to this one

SUMMARY OF SCRIPT RUN-TIME OPTIONS

V	output to video screen instead of to printer
CC n	number of copies to be printed
DA	overstrike for darker printing
DS,TS	double-space or triple-space printed output
ED	allow interactive edit of input text
ID	print input File I.D.'s in left margin
PG n1 <,n2>	starting <and ending> pages to be printed
ST	stop at bottom-of-page (for next cut sheet)
NU	number each line of each page in left margin
TC	print only the Table of Contents for current document

In addition, using a semicolon as the name of the input file causes all input to be taken from the keyboard instead of from any disk file.

SUMMARY OF SCRIPT'S MINI-EDIT COMMANDS

<ENTER>	process displayed line
D	delete displayed line (ignore it)
I string	insert 'string' ahead of displayed line
R string	replace displayed line with 'string'
T	terminate mini-edit and continue normally

SUMMARY OF OPTIONS WHEN SCRIPT IS INTERRUPTED

Y	continue processing (ignore the interrupt)
N	stop processing and go to end-of-job
E	activate mini-edit mode and continue processing

EDIT Error Messages

1. **No more room!** This should have been preceded by a series of warning messages saying "nn LINES LEFT". All available string space has been used up, or the number of lines EDIT can handle has been reached.

   SAVE the file immediately, then use FITLINE to break the file into two smaller ones. Then, just continue by editing whichever piece you were working on.

2. **Unknown command.** Probably a typo. Try again. If it persists and is a valid EDIT command, you may have had a memory fault. Try to SAVE the file, then re-boot and continue. It is possible in this case for part of the file to have been lost or garbled.

3. **Invalid or missing operands.** Similar to #2 above, but the parameters you used were not correct for the command. Same recovery procedure as #2.

4. **Invalid bounds for Verify or Zone.** The left-most column must be at least '1', and cannot be greater than the right-most column. The right-most column must be less than 256.

5. **File not found.** The 'fileid' specified in "GET" is not on any of the on-line diskettes. Insert the correct diskette, or change the spelling of 'fileid'.

6. **At TOF or EOF.** You are trying to do something outside the bounds of the file. The markers can't be changed.

7. **At EOF.** You are trying to do something with the EOF marker.

8. **String or point not found.** Very common. The argument of a Locate, Change, Find, Copy, Move, or Dstring wasn't found. This could be because it isn't in the remainder of the file, because you typed it incorrectly, because it's not within the Zones you've set, or because it's outside the range of the command. If a Point, it may have been deleted.

9. **String larger than currently-defined zone.** If you've set a narrow column zone and your argument to Change or Locate is bigger than that zone, the argument would never be found. If the argument is correct, widen the zone.

10. **String would exceed 255 characters.** An attempt to Change or Join a string failed because the resulting line would be too long. Consider using EDIT's "BREAK" command first.

11. **Disk Error!   (error message).** This message is given when any disk I/O error occurs, including parity or hardware problems. If possible, correct the problem (switch diskettes) and re-issue the "SAVE" or "END" command, possibly specifying a different disk drive. DO NOT disturb EDIT while attempting to correct the problem; you can hit <BREAK> if necessary, and when ready to re-try, type the BASIC command: "CONT". NOTE: Some Operating Systems force EDIT back into BASIC after DIRECTORY WRITE ERRORs. After making the diskette writeable, just type "CONT" to BASIC, and re-issue SAVE/END if EDIT doesn't re-try automatically. If a write-protected disk contains a file with the name you've assigned to the current file, specify a drive number in the name, or use a different name.

12. <u>Directory Unreadable.</u> This message may occur within EDIT or SCRIPT when you use either the "DIR" command or the "?" response to the prompt for a filename. The message means that NEWSCRIPT can't display the requested directory. There may be two reasons for this: 1) you're running under Model I TRSDOS or NEWDOS, and NEWSCRIPT doesn't support the "DIR" command with these; 2) you're trying to read a directory from an unsupported format of diskette. For example, under "TDOS", which is used to distribute NEWSCRIPT, a Model III Directory cannot be read (neither can such a diskette, except with the "CONVERT" program).

This message does <u>not</u> mean there's anything wrong with the diskette whose Directory you tried to display. It just means that NEWSCRIPT can'te read or display it. .

SCRIPT Error Messages

1. Invalid address for input string. SCRIPT error. Re-boot and re-run. If error persists, please report it with sample data file and explanation of how/where it happened.

2. Invalid address for output string. Same as #1.

3. Requested line length not between 10 and 132. Your ".LL n" value is out of bounds. Correct and retry. If error persists, its same as #1.

4. Input string truncated to fit output length. This should rarely, if ever, occur. It may be due to some strange combination of ".SD" and ".LL".

5. Input line too long to be spaced as requested. Same as #4, but can also occur if Concatenate or Format is OFF and the text line exceeds the Line Length.

6. Generated output cannot fit into output buffer. Same as #1.

7. Line cannot be spaced within requested limits. Same as #3.

8. Invalid SCRIPT control word. Check and correct the mnemonic. If error persists, see #1.

9. Top or bottom margin too small. The values of "TM, HM, HS", or the values of "BM, FM, FS" are in conflict. Either increase "TM" or "BM", or reduce some of the others.

10. File to be imbedded not found. Put the correct diskette on-line and retry, or correct its name.

11. Nested imbedded files are not allowed. There is an ".IM" in an imbedded file. Remove it, or make the higher level file an ".AP" at the end of the file that used to ".IM" it.

END OF ERROR MESSAGES

SECTION VIII - PRODUCT DIFFERENCES

This section is intended only for people who have used SUBEDIT/SUBSCRIPT or IBM's CMS Editor, EDGAR, DCF, or SCRIPT.

Differences Between NEWSCRIPT and SUBSCRIPT

1. Full-Screen Editor - all new.
2. These Edit commands deleted: Overlay, Input, Replace, Cline,Dup.
3. These Edit commands added or changed: Name, Whoops, "?", "&", FREE, DIR.
4. This EDIT command's name is changed: "VERIFY" is "VIEW".
5. These Script control words added or changed: BF, DA, IX, LS, PP, RD, TR.
6. This SCRIPT control word no longer causes a control break: ST.
7. These Script escape sequences added: !< (backspace), !/ (start italics), and !? (end italics).

The Full-Screen editor, the "Whoops" command, and the ".PP" and ".RD" control words offer significant new function over SUBSCRIPT, and should be learned by anyone who wants to use their functions.

Differences Between EDIT and the CMS EDIT Command

If you're reading this section, I'll assume you already know how to use SCRIPT and either EDIT or EDGAR under CMS. Therefore, I'll only identify where incompatibilities exist, and leave it to you to look up the commands in SECTIONS III and IV. The Full-Screen editor is almost identical to EDGAR, although its LIMA (the Type III area) is on the left side. The editor does not support SOS commands or split-screens, but does support dynamic windowing, scrolling, and text splitting. The <CLEAR> key acts much like the <ALT> key on the 3278, and while it is pressed, <blank> is like "erase EOF", "I" sets/resets insert mode, "D" is the delete key, and the right-arrow performs a text-split at the cursor. Shift-up-arrow is like the ALT-back-tab key, taking you to the command line.

If you like what you've just read, and find it to be true, please spread the word among other CMS users: EDGAR and SCRIPT are alive and well and running on the TRS-80!

Different Syntax

          ALTER BACKPAGE FORWARD GETFILE LOCATE VIEW =

New Commands

          BREAK COPY DUP FLOW FREE GRID HARDCOPY JOIN WHOOPS
               KILL MOVE NAME STRING XY - ?

Commands that Have Additional or Different Options

          GETFILE LOCATE OVERLAY VIEW

CMS Editor Commands that are not Implemented

    CASE CMS FILE FMODE FNAME FORMAT IMAGE INPUT LINEMODE LONG
      PRESERVE PROMPT RECFM RENUM REPEAT RETYPE RESTORE RETURN
      REUSE SCROLL SERIAL SHORT SOS STACK TABSET TRUNC TYPE NNNNN

    Some of these are implemented under other names. In particular, "END" is used
in place of "FILE". Most of the non-implemented commands are either not needed or
make no sense on a TRS-80.

    The current line pointer is handled differently by EDIT than by CMS EDIT: when
a search error occurs, the current line pointer does not move in EDIT, so you can try
again easily. "Change" does not move the line pointer, either.

Differences Between SCRIPT and CMS SCRIPT

    If you're wondering how I'm going to handle this one, its by mostly ducking the
question. There are at least six Script processors available on CMS as of mid-1980,
and both IBM and Waterloo seem to be enhancing their versions quarterly. I don't
expect to keep up with either of them, so I'll just identify those SCRIPT control
words that are not functionally identical to their better known parents. Some
SCRIPT functions do not exist in the bigger versions (as far as I know), but I'm not
about to stick my neck out by making any claims about them.

SCRIPT Syntatical Differences

                                  DA TB TC

If you're not sure about SCRIPT's implementation, please refer to SECTION IV.

SECTION IX - INSTALLING and TAILORING NEWSCRIPT

The normal installation procedures were covered in Section I. The material in this section covers special conditions:

    1. Changing "EDIT" defaults   4. Other Operating Systems
    2. Connection of Parallel Printers 5. One-drive operation
    3. Serial Printers   6. How to get assistance from PROSOFT

PROSOFT software is protected from unauthorized distribution by Copyright and the good will of our customers. We do <u>not</u> attempt to protect our software or the diskette by making them hard to copy, so the <u>first</u> thing we want you to do is:

PLEASE MAKE A BACKUP COPY

and then store the original PROSOFT diskette away in a safe place. The original is your ultimate backup, and your proof of purchase should you ever need updates or upgrades to the programs. Original diskettes shouldn't be used for everyday work, and shouldn't be altered in any way.

PARALLEL PRINTER CONSIDERATIONS

EPSON Printers (MX-80 and MX-100)

The EPSON MX-80 should be in "MX-80" mode, not in "TRS-80" mode, and a standard Centronics/Radio Shack cable should be used. If you use an EPSON cable, set the printer switches to force a line feed with each carriage return. The setup should be the same as the one you would use to list a BASIC program via "LLIST". On some MX-80's, "16.7" characters-per-inch actually print at about 17.1 cpi. NEWSCRIPT does not compensate for this discrepancy. If you have the GRAFTRAX-80 option on the MX-80, then <u>italics</u> and selective boldface can be used. Most MX-100's <u>cannot</u> print italics, since "GRAFTRAX" and "GRAFTRAX-80" are different. To use cut-sheets on an MX-80, some form of friction/tractor option must be installed on the printer.

SWITCH SETTINGS ON EPSON's

```
                          (Y=ON, N=OFF)
                      SWITCH 1           SWITCH 2
                    1 2 3 4 5 6 7 8      1 2 3 4
       No GRAFTRAX: Y N N Y Y N N Y      Y Y Y N
          GRAFTRAX: N N N N N N N Y      N N Y N
```

MICROLINE Printers

The automatic line feed switch should be "on" (that's the normal default). The printer does not permit underlining or overstriking. Double-width, 10 cpi, and 16 cpi are supported by NEWSCRIPT.

DAISY WHEEL PRINTER II

NEWSCRIPT performs pitch selection under software control, so position of the "10 / 12 / PROP" lever on the front doesn't matter. However, after printing with NEWSCRIPT, the printer has to be turned off and back on to give control back to that switch.

The DW2 does not underline blanks. To circumvent this, you can place underscore characters _directly_ into the text where such blanks occur. To do this, position the cursor at a blank, hold down <SHIFT>, press <CLEAR>, release both, and then press the <minus> key.

C. ITOH 8510, NEC 8023A

The switches on these printers should be set as follows:

```
        (Y=ON or CLOSED, N=OFF or OPEN)
          SWITCH 1            SWITCH 2
        1 2 3 4 5 6 7 8     1 2 3 4 5 6 7 8
        N N N N N Y N Y     N N N N N N Y N
```

SERIAL PRINTER CONSIDERATIONS

The first time you run NEWSCRIPT, you will be asked to identify your printer, and whether it is connected in parallel or serially. NEWSCRIPT includes support for parallel printers, but relies on facilities that you must provide for serial printers.

Some operating systems contain support routines for serial printers. For instance, using the supplied TDOS, or a full version of DOSPLUS, the commands are "FORMS" and "RS232". If you need this support, A file called "SERIAL/MIN" is provided with NEWSCRIPT (it's on side two if you received the Model I version). To use it, just "KILL STARTUP/MIN", then "RENAME SERIAL/MIN TO STARTUP/MIN". (On the Model I, COPY it from Side 2.) See Section X for more information on the "FORMS" and "RS232" commands.

If you're using TRSDOS on the Model III, see the TRSDOS "ROUTE" command. On the Model I, depending on your operating system, you probably have been using a "serial printer driver." Please note that such programs _must_ run lower in memory than NS/CMD (60000), and may have to be modified to do so. If you don't know how to do this, it may be necessary to call the original supplier of that printer software. In all cases, be sure to add the name of the driver to "STARTUP/MIN" to ensure it gets used.

You may place other commands in "STARTUP/MIN" as long as the total number of characters in the entire file, including the <ENTER>'s, does not exceed _128_. NS/CMD _cannot_ be used with DO, CHAIN, or other command list programs. "STARTUP/MIN" is _not_ a normal "BLD" file, and should not be used with "DO".

TRSDOS, NEWDOS (all versions), LDOS, DOSPLUS

NEWSCRIPT runs very nicely under these Operating Systems, but the initialization sequence for BASIC must be replaced. This sequence is stored in "STARTUP/MIN", and is used by the chaining facility of NEWSCRIPT when running under TDOS and DOSPLUS. "STARTUP/MIN" must be replaced if you want to run with a different Operating System, and files for this purpose have been provided. They are:

<div align="center">TRSDOS/MIN   NEWDOS/MIN   LDOS/MIN</div>

We recommend you use NEWSCRIPT with the supplied DOSPLUS, but it can be migrated to any of these other systems if necessary. Please note that operation under TRSDOS is slower than under NEWDOS, LDOS, or DOSPLUS.

If you have a Model I, just copy all the "Side 1" NEWSCRIPT programs to your own system disk, then copy the appropriate "MIN" file from side 2, as follows:

<div align="center">COPY opsys/MIN:1 STARTUP/MIN:0</div>

(where 'opsys' is one of: LDOS, NEWDOS, TRSDOS)

If you are using LDOS double-density, then you should also:

<div align="center">COPY NSLDOSDD/CMD:1 NS/CMD:0</div>

To complete installation, see step 9, below.

If you have a Model III and want to move NEWSCRIPT to another operating system (except for DOSPLUS, of course):

1. Place NEWSCRIPT in drive 0 and a blank disk in drive 1.
2. FORMAT drive 1 as single-density, 35-tracks.
3. Issue the NEWSCRIPT command: DO TRSDOS
   This will copy all the needed parts of NEWSCRIPT onto this single-density diskette. It takes 5-10 minutes.
4. KILL STARTUP/MIN:1
5. RENAME opsys/MIN:1 STARTUP/MIN
   (where 'opsys' is one of: LDOS, NEWDOS, TRSDOS)
6. Remove NEWSCRIPT from drive 0 and insert a copy of your preferred operating system its place. This disk must have enough room to hold NEWSCRIPT (about 1/3 of a data disk of space).
7. Boot your operating system.
8. The disk in drive 1 looks like a Model I disk. Follow your operating system's instructions for converting it. This would be the "CONVERT" command of TRSDOS, the "PDRIVE" command of NEWDOS, etc.
9. When conversion is complete, remove the single-density disk and type "NS" (no quotes). Once you answer the configuration questions, NEWSCRIPT will be ready for use. We suggest you make at least one backup at this point.

ONE-DRIVE OPERATION

Always begin operation with the primary NEWSCRIPT production diskette (not the one we sent you, but a copy). Issue the "NS" command, allow the primary menu to appear, select the NEWSCRIPT facility you need (EDIT, SCRIPT, INDEX, etc.) and allow the selected program to load and initialize.

Next, remove the NEWSCRIPT diskette and insert a disk that will hold your documents. (This disk must contain the same operating system that is on the NEWSCRIPT diskette, but instead of the NEWSCRIPT programs, should have space for your text files. It's a data storage diskette.)

You can work on this "data" disk until you need another NEWSCRIPT program. Then, when asked if you want the file passed to SCRIPT (or EDIT), remove the "data" disk and insert the NEWSCRIPT disk again. After the next program initializes, just switch back to the "data" disk.

ASSISTANCE

All PROSOFT products include limited on-going support. If you have problems installing or using our software, we'd like to try to help resolve the problems and answer your questions. We can't promise that we will be able to answer every question or fix every problem, but can promise that we will try. If the distribution diskette isn't readable, just send it back, carefully wrapped and with a note of explanation. We'll replace it at no charge within the first 30 days.

The best way to report a problem is by mail: send in a description of the problem and a sample printout. Please be sure to indicate your printer, CPU model, the release and date of NEWSCRIPT, and your registration number. This approach gives us the best chance of determining what's wrong and how to fix it.

Telephone assistance is available during our normal business hours: Monday - Friday, 9:00 A.M. - 7:00 P.M. Limited assistance is available on Saturday by prior arrangement. We do not accept collect calls, and when no one answers the phone, an answering machine picks up after four rings. The Toll-Free numbers ring only at an outside order-taking service that does not relay messages. Since many problems are caused by faulty hardware, please try to determine that suspected problems are re-creatable. Please be prepared to describe exactly what sequence of steps led to the problem you're having. It may be necessary for you to send in print samples and a disk containing a file that will re-create the error if we follow your detailed directions (all disks will be returned.)

Our address and telephone number are shown below. Please note the Box number differs from the one on the cover of the manual.

PROSOFT
Box 560
North Hollywood, Calif. 91603
(213) 764-3131

We think you're going to find that writing can be delightful with NEWSCRIPT, hope you enjoy using it as much as we do, and

T H A N K   Y O U

for choosing

P R O S O F T

SECTION X -- DOSPLUS[1]

OVERVIEW

NEWSCRIPT is distributed on a ready-to-run diskette. It uses a tiny version of the excellent "DOSPLUS" Operating System, hereafter called "TDOS" and "TBASIC". TDOS is very similar to TRSDOS, but runs faster and is somewhat more reliable. All features needed for Word Processing are in this system, but the many additional capabilities of DOSPLUS are not supplied. If you find that you like TDOS so well that you want to obtain the full DOSPLUS, it may be purchased from many software dealers, including PROSOFT. If you wish to run NEWSCRIPT under a different Operating System (TRSDOS, NEWDOS, NEWDOS/80, or LDOS), you can copy all the non-system files from the distribution diskette to a TRSDOS-formatted data diskette. Then, you may use that diskette with your own operating system or may copy its contents to any disk you wish.

SUMMARY OF TDOS COMMANDS

These are the TDOS commands most likey to be needed with NEWSCRIPT:

   AUTO BACKUP CAT CONVERT COPY DIR DO FORMAT KILL TBASIC

If you have a serial printer, you will also need these commands: FORMS RS232

If you have only one disk drive, you will also need this command: COPY1

All these commands are described in the remainder of this section. However, it is assumed that you also have a TRSDOS manual from Radio Shack, should you need more detail.

These commands must be used from "DOSPLUS" level, and cannot be used while in TBASIC. Also, there is no "BASIC" on the diskette: "TBASIC" replaces it.

Briefly, the commands covered in this section do the following:

```
    AUTO          - sets a command to be used automatically on startup
    BACKUP        - copies an entire diskette to another diskette
    CONVERT       - converts TRSDOS diskettes/files to TDOS format
    COPY          - copies a file from one diskette to another using 2 drives
    COPY1         - copies a file from one diskette to another using 1 drive
    DIR or CAT    - displays the directory of any on-line diskette
    DO            - executes a series of commands in a "BLD" file
    FORMAT        - formats (initializes and erases) a diskette for data use
    FORMS         - sets paper control for a serial printer with NEWSCRIPT
    KILL          - deletes a file from an on-line, unprotected diskette
    RS232         - sets speed and protocol for a serial printer
    TBASIC        - invokes Tiny BASIC for use with NEWSCRIPT
```

---

[1] DOSPLUS is a trademark of Micro-Systems Software, Inc.

AUTO

This identifies a command that is to be executed automatically whenever you power up or reset the computer. The command (and any needed parameters) follows the word "Auto", separated by a blank:

<p align="center">AUTO NS</p>

is used to invoke NEWSCRIPT automatically.

To suppress the "AUTO" function, hold down <ENTER> when you power up or reset the computer. To disable it entirely, issue the "AUTO" command with no operands. To set a different "AUTO", issue "AUTO" with the name of the command to be used.

BACKUP

Use this to make copies of your Word Processing diskettes. We urge you to make multiple backups of program and text diskettes, since the cost of diskettes is far less than the value of your time.

When you issue the "BACKUP" command, you will be prompted as follows:

        SOURCE DRIVE NUMBER ?
        DESTINATION DRIVE NUMBER ?
        BACKUP DATE (MM/DD/YY) ?

Answer each question appropriately. A normal backup is from drive 0 to drive 1, using today's date for future reference. If the destination diskette already has something on it, you will be asked whether it should be used anyway. Reply "Y" (no quotes) to proceed, or "N" if you want to use a diferent diskette.

If both the source and destination drives are "0", a one-drive backup will be performed. In this case, you will be prompted repeatedly to switch diskettes. Do this carefully! The safest thing to do, in fact, is to put a write-protect tab on the source diskette before you start.

When BACKUP completes, the message "INSERT SYSTEM DISK AND PRESS <ENTER>" will appear. Just press <ENTER> to continue.

CONVERT

This command applies only to Model III systems. It is used to copy files from TRSDOS-formatted diskettes to TDOS-formatted diskettes. The command has two distinct uses and several distinct forms:

MODEL I TRSDOS OR NEWDOS to MODEL III TDOS: CONVERT :1

This makes a Model I diskette readable to TDOS on the Model III. It is not for use on the Model I, since Model I TDOS can read and write normal Model I diskettes without conversion. This form of CONVERT changes some of the control information on the Model I diskette, and the resulting diskette cannot be read by TRSDOS or NEWDOS+ thereafter (it can be used by LDOS and NEWDOS/80). This form of the command is mostly for one-time conversions from Model I to Model III. If you plan to use the source diskette on the Model I, make a Model I backup of it first and use the backup disk. Remember, if you use this form without making a backup, or run both the original and the backup through "CONVERT", you won't be able to use the diskette on the Model I afterwards.

MODEL III TRSDOS to TDOS: CONVERT :1 :0 (OLD or V13)

This copies all non-system files from a TRSDOS diskette to a TDOS diskette. The word "OLD" or "V13", in parentheses, is needed depending on the release of TRSDOS being converted (see below). The TRSDOS diskette is not altered in any way. If you have more than two drives, you may specify any pair of drives for the conversion. Files are read from the first-specified drive ('1' in this case) and written to the second-specified drive ('0' in this case). This command will not move files from TDOS to TRSDOS. A way to do that is explained in the Installation Instructions.

There are several versions of TRSDOS and two versions of DOSPLUS to consider in determining whether to use "OLD", "V13", or neither. If your TDOS or DOSPLUS recognizes the "CAT" command to display the directory, use "V13" to convert TRSDOS 1.3 files, and omit the parameters for other version of Model III TRSDOS. If "CAT" is not recognized, use "OLD" to convert TRSDOS 1.1 or 1.2 files, and omit the parameters for 1.3.

COPY

This copies a file from one disk drive to another. For a one-drive copy, use "COPY1", not "COPY". The destination diskette must already be formatted and not write-protected. The destination diskette must be in either TDOS or Model I single-density TRSDOS format. It cannot be in LDOS format or in any double-density format other than that used by Model III TDOS. However, once you've copied something onto a Model I TRSDOS-compatible diskette, you can copy the result to any other operating system.

The format of the command is:

                    COPY filespec:d TO filesec:d
                                or
                      COPY filespec:d :d

'filespec' must conform to normal TRSDOS conventions, e.g.,

              TEST/FIL:1 or DOC1 or GREATDOC:0 etc.

If the source and destination names are the same, it is sufficient to just give the destination drive number as the second filespec.

COPY1

This performs a one-drive copy. It copies a single file from one diskette to another, but you must switch diskettes back and forth within one disk drive. The format is:

COPY1 filespec

If 'filespec' does not contain a drive number, drive zero is used. However, you can copy from drive 1 to drive 1 if you wish to do so. When using this command, carefully follow the prompts for switching diskettes back and forth. For safety, put a write-protect tab on the source diskette before starting. The destination diskette must already be formatted in TDOS form or in single-density Model I TRSDOS form.

DIR or CAT

These display the directory on any available drive. "CAT" just shows an abbreviated directory, whereas "DIR" shows considerable information. Only one of these is provided on our TDOS, and we are phasing out "DIR" in favor of "CAT". If your copy uses "DIR", the display stops automatically each time the screen fills. To see the next 'page', press <ENTER>. To see the next line, press <SPACE>. The command format is:

DIR :d (S,I,D,P)

If ':d' (drive number) is omitted, drive zero is assumed. If any options are used, they must be enclosed in matched parentheses (you need the closing parenthesis). The meanings of the options are:

    S -- show System Files as well as user files
    I -- show Invisible Files also
    D -- show all files marked for deletion also
    P -- print the Directory on the printer

Any combination of options may be used at once with DIR; "CAT" has no parameters or options.

DO

This executes a series of commands previously entered in a 'BUILD' file. The "DO TRSDOS" facility, which transfers NEWSCRIPT to a single-density Model I diskette, is an example of such a series of commands. The format is:

DO filespec

where 'filespec' is the name of the file containing the commands. 'BUILD' is not needed when using NEWSCRIPT, and is not explained here. However, it's format and use are the same as those used by Model III TRSDOS.

FORMAT

This formats data diskettes. After issuing the command, you will be prompted as follows (depending on your computer and whether the new diskette contains data, not all of these questions may be asked):

        WHICH DRIVE IS TO BE USED ?
        DISKETTE NAME ?
        FORMAT DATE ?
        MASTER PASSWORD ?
        NUMBER OF TRACKS (35-80) ?
        SINGLE OR DOUBLE DENSITY ?
        DISKETTE CONTAINS DATA, USE OR NOT ?

Answer each question in turn. The PROSOFT diskette password is 'PASSWORD'. The 'name' may be anything you like, up to eight alphanumeric characters, of which the first is a letter. The number of tracks defaults to '35' on the Model I and to '40' on the Model III. Density is always 'single' on the Model I, and defaults to 'double' on the Model III. If you are creating a TRSDOS-compatible diskette, be sure to specify 'single' on the Model III.

Remember that FORMAT completely erases the diskette, so don't use it on a diskette whose contents you may need afterwards.

When FORMAT completes, it displays this message: "INSERT SYSTEM DISK, PRESS <ENTER>". If TDOS is still in drive 0, just press <ENTER>.

FORMS

This should be used only if you are running NEWSCRIPT with a serial printer connected to the RS-232 interface of your computer. In this case, you'll need both the FORMS command and the RS232 command. In the case of the FORMS command, you must specify the following:

                        FORMS (L=66,S)

It also may be necessary to place either or both of these values inside the parentheses, suitably separated by commas. This depends on your printer:

            LF   (auto linefeed)
            NUL   (line feed on all carriage returns)

KILL

This is used to delete a file from a given diskette. The space used by the file is reclaimed, and the comand format is the one used by TRSDOS:

KILL filespec

RS232

This is used in conjunction with the FORMS command when running NEWSCRIPT with a serial printer that is connected to the RS-232 interface of your computer. If you're using the "TRS-232" serial printer interface on a Model I, these commands do not apply. However, PROSOFT offers an interface for the TRS-232, and it can drive a Diablo or other Daisy Wheel printer at about 45 characters/second.

The format of the RS232 command is as follows:

RS232 (parm1,parm2,parm3, etc.)

where the parameters are:

```
BAUD=300 or 1200    (or any standard baud rate)
WORD=7 or 8         (bits per character, usually 7)
STOPS=1 or 2        (usually 1)
PE                  parity enable (also use ODD or EVEN)
PI                  parity disable (ODD/EVEN are ineffective)
EVEN                even parity (used only with "PE")
ODD                 odd parity (used only with "PE")
RTS                 Request To Send, if your printer needs it.
DTR                 Data Terminal Ready. If your printer needs it.
NOWAIT              tells RS232 to not wait for next byte.
```

If you type "RS232" (no quotes) and no parameters, the current settings will be shown. If you're running a daisy wheel type printer at 30-55 cps, a setting that usually works is:

RS232 (BAUD=300,WORD=7,STOPS=1,PI)

Remember that "FORMS (S,L=66)" must be issued in conjunction with RS232 for anything useful to happen.

TBASIC

This invokes Tiny BASIC. Portions of NEWSCRIPT run under control of BASIC, and TBASIC must be in control when using NEWSCRIPT. Note that 'NS/CMD' and a printer driver such as 'NSPRT' must have been activated from DOS before entering TBASIC.

The format of the TBASIC command differs from that of most other BASIC commands (it's format has been picked up by Radio Shack as an option in TRSDOS 1.3):

TBASIC filespec-F:n-M:mmmmm

For example: TBASIC SCRIPT-F:5-M:60000

'filespec' is identifes a BASIC program that is to be run as soon as TBASIC is initialized. If omitted, you will be placed in TBASIC with a "READY" message.

'-F:n' specifies the number of files to be used while in TBASIC. Unlike TRSDOS, the default here is <u>zero</u>. NEWSCRIPT requires 4 files and TBASIC will not prompt you for 'FILES' or 'MEMORY SIZE', so both must be specified here if needed.

'-M:mmmmm' specifies the protected memory address. It is not needed with NEWSCRIPT unless you add a printer driver of your own that doesn't set the "HIMEM" address.

TBASIC is similar to Model I Disk BASIC, and not to Model III Disk BASIC or NEWDOS BASIC. That is, it doesn't honor abbreviations, the period, comma, or other short-cuts. It also doesn't display detailed error messages. In return, it takes much less memory than full BASIC, and legally can be distributed to support NEWSCRIPT. (We pay a small license fee to Micro-Systems Software for every copy we distribute. It makes life far more pleasant for us, and we hope, for you also.)

When using TBASIC, the "SAVE", "LOAD", "KILL", "LIST", "LPRINT", and "LLIST" commands, as well as all of Disk BASIC, may be used. The command "CMD" takes you back to DOSPLUS, and the "S" used with TRSDOS must be omitted. "TBASIC *" may or may not allow you to return to TBASIC level from DOSPLUS for the purpose of trying to salvage a text file under NEWSCRIPT's EDIT. Depending on what has happened, it may or may not be possible to complete the salvage. To attempt recovery, <RESET>, type "TBASIC *", <ENTER>, "GOTO 9999" <ENTER> (no quotes or brackets anyplace.)

SUMMARY

This completes the summary of TDOS and TBASIC as used with NEWSCRIPT. A full explanation of DOSPLUS may be found in the documentation that accompanies that Operating System.

SECTION XI - FITLINE UTILITY

This utility program is used for three purposes:

1. To split large files into smaller ones. This becomes necessary when revisions to a file have made it too large for EDIT.

2. To "flow" short lines of text together, while honoring the SCRIPT control words. This is for cosmetic appearance while editing only! SCRIPT treats all input as a continuous stream of characters, regardless of line length on the screen.

3. To convert large, wide files from other systems into dimensions that fit the NEWSCRIPT Editor. However, the format controls of these other systems are not converted.

FITLINE is written in BASIC. It reads an existing file that contains SCRIPT control words, and writes one or more new files back out. It can shorten or combine lines to fit the 60 character-wide screen used by NEWSCRIPT (hence its name), and can limit the number of records in such a new file by creating several small output files. As it performs this conversion, it takes account of the meanings of the SCRIPT control words it finds, so it knows when to stop flowing text back and forth and when to start a new file (only at a control word).

OPERATION

1. Activate BASIC with at least two files. Memory size doesn't matter. If NEWSCRIPT is already active, this step is omitted.

2. RUN "FITLINE"

3. The program will ask you to enter the I.D. of the input file.

4. The program will ask you to enter the I.D. of the (first) output file. This I.D. must be different from the input file I.D., and there must be enough room left on an on-line diskette to hold the resulting file.

5. The program will ask for the maximum number of characters per output line (default=60). Just hit <ENTER> to accept this default.

6. The program will ask for the maximum number of records per output file (default=9999). Enter a value of 125-175 to ensure that you will be able to edit the new files with room for additional text.

7. FITLINE will now run until it has processed the input file. If the output file record limit is reached, FITLINE will write additional records until it encounters a SCRIPT control word. Then, it will ask you for the I.D. of the next output file. If that file already exists, you'll be given a chance to supply a different name. Then, FITLINE will generate a SCRIPT control word, ".AP filespec" as the last line of the previous output file, close that old file, open the new one, and continue reformatting and writing the input file. The generation of the ".AP" allows EDIT and SCRIPT to chain from one segment of a document to the next.

8. This process will continue until the input file has been completely processed.

Then, you can run FITLINE again for another file, or run "EDIT" again to resume your editing.

Writing desk.

SECTION XII - MAILING LABELS

MAILING LABELS IS AN OPTIONAL, EXTRA COST FEATURE OF NEWSCRIPT. THIS DOCUMENTATION IS INCLUDED WHETHER OR NOT YOU'VE PURCHASED THE PROGRAM.

INTRODUCTION

"LABELS" is used to print mailing labels from the same kinds of lists that are used for Form Letters in NEWSCRIPT. It lets you specify the dimensions of the labels and the number of labels across and down. It lets you specify which names and addresses should be extracted from the mailing list. Finally, it lets you join several lists together, either by entering each list-name at the keyboard, or by placing all the list-names in a Super-list.

The mailing lists used by LABELS must be in the same format, and use the same selection codes, as the ones defined by the ".RD" control word of NEWSCRIPT. LABELS does not offer any sorting capabilities and does not insert selected names into the middle of a letter.

If printing 10, 12, or 16 cpi, labels may be 1-up, 2-up, 3-up, etc. However, if printing in proportional font, only 1-up labels are supported. Labels may be on sheets, such as Avery 5375 / 5380, or on continuous rolls.

INSTALLATION

The LABELS program is written in BASIC and distributed on diskette. It should be copied from the distribution diskette to a disk of your own, and the original disk should be saved for backup and possible updates from PROSOFT. If you received it along with NEWSCRIPT, it's on the Model III distribution diskette or the second Model I diskette.

TAILORING

LABELS contains a complete set of defaults that describe the dimensions of a set of labels. Those defaults may be used as-is, permanently changed in the program, or over-ridden each time you run LABELS. As distributed, the defaults are as follows (they may be found near the beginning of the program if you need to change them):

Gummed labels.

VARIABLE	DEFAULT	DESCRIPTION
CF$	S	C=continuous form: S=single sheets
NC	3	# labels across a . (1-up, 2-up, etc.)
NR	10	# labels down ? p. (single sheets only)
NL	6	# printable li..s p.r label
VS	0	# lines between ea(h label (vertical gap)
SP	1	start position of first label (for left margin)
NP	25	# print positions/label line
NG	3	# characters betwee. labels (horizontal gap)
CS$	#	code signal for cod. records

FORMAT OF A MAILING LIST

LABELS is compatible with the rest of NEWSCRIPT, and expects lists to be in the "Variable Lines Format" described under ".RD" in the NEWSCRIPT manual, Section IV. Please refer to that documentation for a complete description and examples. Also note that the method of selective printing used by NEWSCRIPT is also used by LABELS. However, LABELS does not use variable words.

OPERATING INSTRUCTIONS

1. Prepare a mailing list in the format described by NEWSCRIPT. The EDIT command of NEWSCRIPT may be used for this purpose, or you may wish to write a simple BASIC program to convert pre-existing lists to the required format. In either case, it's a good idea to place several unique identifier codes in the "codes" line that begins each N&A entry. This will give you good selection flexibility in the future without having to re-edit the lists. To make sure the identifiers are unique, separate them by commas or some other character that doesn't occur in any of the identifiers themselves. For example:

       @05/15/81,widgets,whsl,CALIF,

2. If you have a large list, you may want or need to segment it into several smaller ones. Alternately, you already may have several lists that you want treated as a single large list. If this is the case, see "SUPER-LISTS" later on. The current instructions describe processing of a simple mailing list only.

3. Setup any special printer drivers you may require (e.g., SETCOM and ROUTE on a Model III to a serial printer), and make sure your printer is hooked up.

4. Activate BASIC with at least one file, then RUN "LABELS"

5. You will be asked to enter the I.D. of the mailing list or the I.D. of the Super-list. Make sure the file to be used is in a disk drive, and reply to this question with the I.D. of the file to be used.

6. If the file is not found, you will be given a chance to re-enter its name correctly. Note that an invalid file I.D. (such as "MASTERLIST", which is too long) is treated as a "file not found" condition.

7. The current setup defaults will be displayed. If you can use them as-is, just press the <ENTER> key. If you need to change even one of them, press the <CLEAR> key instead.

8. If you press <CLEAR>, you will be asked a series of questions. In each case, the default setting will be shown. To accept that setting, just press <ENTER>. To change it, type the value you need,then press <ENTER>. The questions are asked in the sequence used to display the defaults, and the list on the first page of this document is also in that sequence. Note that there is no short-cut: if you want to change even one value, you must answer all the questions, even though <ENTER> is a sufficient answer when the default is correct.

   If you have a standard setup for labels, and it doesn't match the distributed defaults, you should change the permanent defaults in the LABELS program, and then save the program back to disk. Your defaults will be used thereafter. As distributed, LABELS assumes the single sheets of labels offered by Avery (#5375 or #5380).

9. After bypassing or over-riding the defaults, you will be asked to enter the Selection Criterion for this run. If you just hit <ENTER>, all N&A's in the file will be printed as labels. If you give a selection code, then each "code line" preceding a N&A will be examined to determine whether it contains that code. Only N&A's whose code lines contain that code will be selected and printed.

   This selectivity feature allows you to print just a few labels for special purposes. The selection criterion may be changed each time you run labels. However, only one criterion may be specified at a time. You can "fool" the program by giving two or three criteria as a single criterion, provided the codes in question were placed adjacent to each other in the codes lines of the N&A file. For example, the first line below gives a single criterion, but the second one actually give two criteria:

                              widgets
                              widgets,whsl

10. LABELS is now ready to print. It will ask you to press <ENTER> when the printer is ready and the labels are correctly positioned in the printer.

    Given the cost of paper versus the cost of labels, it's a good idea, maybe even a great idea, to make a trial run on plain paper before doing the real run on labels. Once you've settled on a standard setup that you use regularly, this may be unnecessary. However, when first learning to use LABELS, it'll save time, money, and grief.

    If you've just entered a new batch of Names and Addresses into a file, it's a good idea to run LABELS on plain paper and then proof-read the results. You may find mis-spellings, typos, and botched "code" lines this way. These proof-sheets also make useful permanent records for bookkeeping purposes.

11. Once you press the <ENTER> key to allow printing to begin, the program runs without further attention until it reaches the end of a single sheet or the end of the input file. If you are using continuous forms, only end-of-file requires attention. If you are using single sheets, the program will ask you to change sheets and press <ENTER> from time to time.

If you are using single sheets, your printer may or may not be able to print on the last row or two of labels, so you may have to specify fewer labels-down than you really have. For example, the Avery labels sheets contain 11 rows of labels, but the default of LABELS is only '10', to ensure that the printer still can advance the page as it nears the bottom.

12. When the end of the input file is reached, you will be asked to enter the I.D. of the next file, or to just press <ENTER> if there are no more files to go. This is one way in which you can combine several files together without wasting any labels at all. If there <u>are</u> more files, then enter the I.D. of the next one at this time, and make sure it's on a mounted diskette. Otherwise, just hit <ENTER>, and the last row of labels will be printed.

13. Finally, you will be asked whether the page should be ejected from the printer. The default is "YES", and the alternative answer is "N" (or "n").

This completes the Operating Instructions.

SUPER-LISTS

Large mailing lists are awkward to maintain, and take lots of time to process. It would be very nice to be able to segment a large list into several smaller ones. For example, it may make sense to start a new list each month, or keep a list for each state, each product, etc. Of course, when printing these little lists, it would be nice to be able to treat them as a single list. That's where the "Super-list" comes in. (It isn't supported by NEWSCRIPT at present, only by LABELS).

A Super-list doesn't contain Names and Addresses. It contains the file I.D.'s of mailing lists. Each of these lists is a file unto itself, and contains code lines, names, and addresses. Each such list is maintained separately and can be processed independently of any others when not using a Super-list. There may be up to 100 such I.D.'s in a Super-list, and if that isn't enough, the LABELS program can be changed easily to accommodate 1,000 or more (just change the maximum value defined for "MF" from 'MF=100' to some larger value).

This is an example of what a Super-list might contain:

        JAN81/ML
        FEB81/ML
        MAR81/ML

The Super-list must be created before you run LABELS. NEWSCRIPT's editor is an easy way to accomplish this. Of course, the files identified in the Super-list must also be created before you run LABELS.

When LABELS asks for the I.D. of the file or Super-list, reply with the I.D. of the Super-list, but place an '@' sign just ahead of that I.D. (no blanks). For instance, if the list shown above was called "ORDERS", the reply would be:

                    @ORDERS

The '@' is the way you tell LABELS that you're using a Super-list instead of a

simple mailing list.

LABELS will make sure the Super-list exists, read all of it into memory, make sure the first file it identifies exists, and then show you the setup defaults. The rest of the operating procedures remain the way they were described earlier, except for the file-to-file transitions. LABELS will run non-stop from one file to another, without asking you to enter the next I.D. (since it gets the next I.D. from the Super-list). When the last file has been processed, LABELS will print the final row of labels and ask whether to eject the page.

The Super-list doesn't have to stay in a disk drive after its been read during initialization. If you have so many normal lists that a diskette change is required, then LABELS will stop when it doesn't find the "next" list, ask you to insert the required disk, and then it will try again. If it doesn't find the "next" list on the second try, it will skip to the list after that one. If the missing list is an unacceptable error, hit <BREAK> to stop the program, find the missing list (or correct the spelling within the Super-list), and start the run over again.

This completes the instructions for use of "LABELS".

THANK YOU
for choosing
PROSOFT

Punch

INDEX

Release 7.1 is a minor revision to NEWSCRIPT. It includes accumulated changes, corrections, and enhancements, most of which will not be noticeable to you, and some new features that have been requested by our customers:

1. Total exit from NEWSCRIPT without re-booting;
2. Optional use of Operating System's keyboard driver;
3. Support for more printers;
4. Additional formatting capabilities in SCRIPT;
5. Documented compatibility with high memory "drivers".
6. Support for the LOBO MAX-80 and TRS-80 Model IV;
7. Multiple column printing on some printers;

EXITING FROM NEWSCRIPT

Option #8 (EXIT TO DOS) of the Primary Menu will ask you whether you want to terminate NEWSCRIPT. If you make this selection, NEWSCRIPT will completely disconnect itself from the Operating System and exit to DOS. You can restart NEWSCRIPT without re-booting after this, but it will not remember the name of the file you had been using.

If you just want to exit temporarily, NEWSCRIPT will still exit to DOS, but remain in control. To return to NEWSCRIPT afterwards, just issue the "BASIC" command the way it appears whenever NEWSCRIPT first begins. This is operating system dependent, of course:

TDOS/DOSPLUS:	TBASIC NSINIT-F:4
LDOS:	LBASIC (E=N,F=4) RUN"NSINIT"
NEWDOS/MULTIDOS:	BASIC 4,RUN"NSINIT"

OPTIONAL USE OF LDOS OR DOSPLUS KEYBOARD DRIVER

When you run the Installation Procedure, you'll be asked whether to use the keyboard driver in NEWSCRIPT or the one in your Operating System. Normally, you should use ours (it's very fast, fully featured, and matches the documentation in our book), but if you want to use certain features in your Operating System, it may be necessary to also use their keyboard driver (KI/DVR).

Two or three important differences should be noted, however: our "SHIFT-CLEAR" won't work at all; the "CLEAR" (control) functions in the EDITOR may have to be accessed through "SHIFT-CLEAR" (LDOS KI/DVR), since "CLEAR" is used by KI/DVR itself; and keyboard repeat becomes a function of the operating system, not of NEWSCRIPT. (Also see "High Memory Drivers" on the next page.)

ADDITIONAL PRINTERS

The printer selection menu (in INSTALLATION) continues on a second screen, so if your printer isn't in the range 1-23, just press <ENTER>, and a second screen will appear. If your printer isn't there, either, press <ENTER> again to get back to the first screen, and pick either #9 (general typewriter) or #18 (Teletypewriter, which provides basic printer support).

Many printers are "lookalikes" of the supported ones, so if your printer isn't listed, and the information in the NEWSCRIPT manual doesn't give you any clues, it's still possible that something on those menus will give excellent support to your printer. In particular, many printers are DIABLO-compatible (#11), and except for proportional printing and boldface (which are included in the "DWP" option), NEWSCRIPT supports such printers nicely.

Release 7.1 is a minor revision to NEWSCRIPT. It includes accumulated changes, corrections, and enhancements, most of which will not be noticeable to you, and some new features that have been requested by our customers:

1. Total exit from NEWSCRIPT without re-booting;
2. Optional use of Operating System's keyboard driver;
3. Support for more printers;
4. Additional formatting capabilities in SCRIPT;
5. Documented compatibility with high memory "drivers".
6. Support for the LOBO MAX-80 and TRS-80 Model IV;
7. Multiple column printing on some printers;

EXITING FROM NEWSCRIPT

Option #8 (EXIT TO DOS) of the Primary Menu will ask you whether you want to terminate NEWSCRIPT. If you make this selection, NEWSCRIPT will completely disconnect itself from the Operating System and exit to DOS. You can restart NEWSCRIPT without re-booting after this, but it will not remember the name of the file you had been using.

If you just want to exit temporarily, NEWSCRIPT will still exit to DOS, but remain in control. To return to NEWSCRIPT afterwards, just issue the "BASIC" command the way it appears whenever NEWSCRIPT first begins. This is operating system dependent, of course:

TDOS/DOSPLUS:	TBASIC NSINIT-F:4
LDOS:	LBASIC (E=N,F=4) RUN"NSINIT"
NEWDOS/MULTIDOS:	BASIC 4,RUN"NSINIT"

OPTIONAL USE OF LDOS OR DOSPLUS KEYBOARD DRIVER

When you run the Installation Procedure, you'll be asked whether to use the keyboard driver in NEWSCRIPT or the one in your Operating System. Normally, you should use ours (it's very fast, fully featured, and matches the documentation in our book), but if you want to use certain features in your Operating System, it may be necessary to also use their keyboard driver (KI/DVR).

Two or three important differences should be noted, however: our "SHIFT-CLEAR" won't work at all; the "CLEAR" (control) functions in the EDITOR may have to be accessed through "SHIFT-CLEAR" (LDOS KI/DVR), since "CLEAR" is used by KI/DVR itself; and keyboard repeat becomes a function of the operating system, not of NEWSCRIPT. (Also see "High Memory Drivers" on the next page.)

ADDITIONAL PRINTERS

The printer selection menu (in INSTALLATION) continues on a second screen, so if your printer isn't in the range 1-23, just press <ENTER>, and a second screen will appear. If your printer isn't there, either, press <ENTER> again to get back to the first screen, and pick either #9 (general typewriter) or #18 (Teletypewriter, which provides basic printer support).

Many printers are "lookalikes" of the supported ones, so if your printer isn't listed, and the information in the NEWSCRIPT manual doesn't give you any clues, it's still possible that something on those menus will give excellent support to your printer. In particular, many printers are DIABLO-compatible (#11), and except for proportional printing and boldface (which are included in the "DWP" option), NEWSCRIPT supports such printers nicely.