PROSOFT™

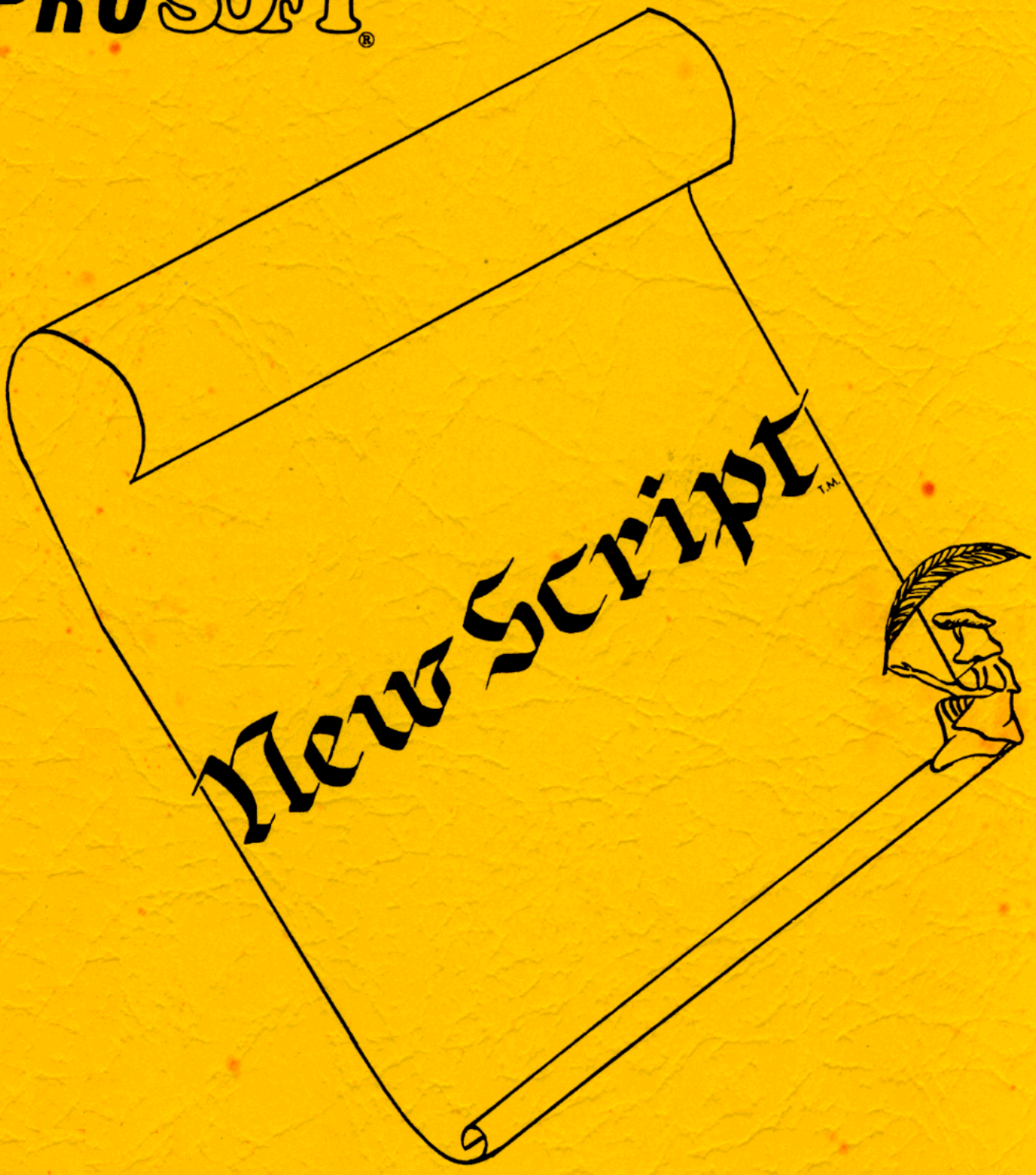
Box 839 · North Hollywood, California 91603 · (213) 764-3131

MODEL III

WITH *MAILING LABELS* OPTION

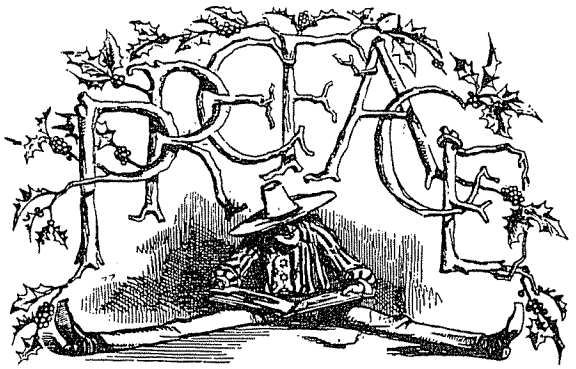
PROSOFT™

PROSOFT®



(714) 755-3043
DBA NEW DIRECTIONS
RT. 1, BOX 72, DEL MAR, CA 92014

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



HOW TO USE THIS BOOK

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 Section I and practice the examples. This section will show you how to install and operate NEWSCRIPT, then step you through a hands-on tutorial for a simple letter.

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. Since this entire book was written and printed with NEWSCRIPT, you'll be seeing many of the output formats that you, too, can produce.

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 major revision of the NEWSCRIPT documentation applies to PROSOFT's NEWSCRIPT Release 7.0 and above. The text in this book was written and printed using NEWSCRIPT. Additional Supplement sheets may be issued from time to time along with updates to the programs.

Revision Date: July 1, 1982

Printing: 10 9 8 7 6 5 4 3 2 1

Published by:
PROSOFT, a division of TTSC
Box 560
North Hollywood, Ca. 91603

N O T I C E

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

PROSOFT, TTSC, and/or the authors shall have no responsibility or liability to the purchaser or any other person, persons, or entity with respect to 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.

WARRANTIES

PROSOFT warrants that any software supplied on diskette is free from mechanical or recording defects, and that we will replace any diskette with recording or mechanical defects within 30 days of the date of purchase. No other warranties are expressed or implied as to the operation, use, or suitability of NEWSCRIPT.

REGISTERING YOUR PURCHASE

A Registration Card was included with your NEWSCRIPT diskette. Its serial number matches the one on that disk. Please fill out and return that card to us immediately. We won't know you exist until we receive that card, and until we know about you, we can't give you any support.

REPORTING PROBLEMS AND OBTAINING UPDATES

Although all PROSOFT computer programs are sold "as-is", we welcome all properly-documented reports of programming or documentation errors, and suggestions for enhancements.

Suspected errors in NEWSCRIPT, or difficulties in using it, should be described to us as clearly as possible. Please Make sure you give us enough information so that we can 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 Primary 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 instructions you provide. All requests for updates or corrections to programs must be accompanied by an original PROSOFT diskette and a $10.00 update service fee. All diskettes will be returned to you, hopefully with corrections and relevant documentation updates.

CORRECTIONS WILL BE DISTRIBUTED ONLY TO REGISTERED LICENSEES.

11978

Please copy the Registration Number from the Registration Card: _____

When you run NEWSCRIPT, the "PRIMARY OPTIONS MENU" displays a release number (such as 7.0) and a date. Please record them below and refer to them if you need assistance or wish to report a problem:

VER 7.0

Release Number, Date: 08/16/82

THANK YOU

for choosing

P R O S O F T

CREDITS AND ACKNOWLEDGMENTS

The NEWSCRIPT Word Processor was written by Chuck Tesler of VM-CMS Consulting Services, Inc., and Glenn Tesler of PROSOFT.

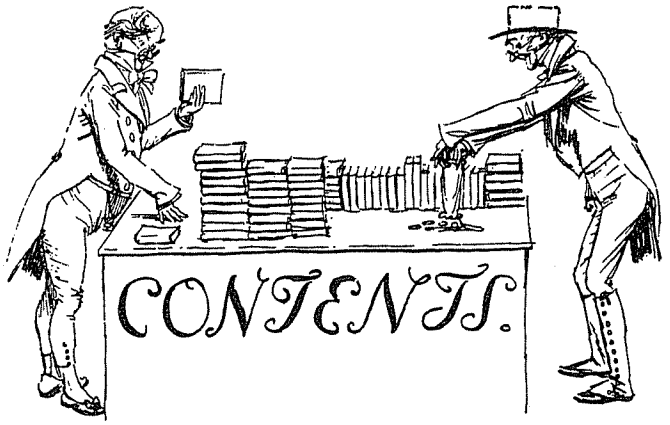
This book was written by Chuck Tesler and Bruce Powel Douglass. Mr. Douglass is a scientist and professional writer well-known and respected in the micro-computer industry. He specializes in writing tutorial and technical manuals for computer software and hardware products, and may be contacted at the following address:

Bruce Powel Douglass
A - Priori Software
1005 West Main
Vermillion, South Dakota 57069

We at PROSOFT wish to express our thanks to Mr. Douglass for planning and writing most of the Tutorial Section of this book. His talent for explaining technical material to new users of computers is reflected both here and in the magazine columns he writes each month.

We also offer our thanks to the many customers who offered suggestions for improving both NEWSCRIPT and this book. To the extent we've succeeded, they deserve a large share of the credit.

The manuscript was developed entirely with NEWSCRIPT, and spanned six Model III diskettes. The spelling was checked with ELECTRIC WEBSTER, one of the excellent dictionary programs available for the TRS-80. All the printing was done by NEWSCRIPT, and a variety of printers was used to demonstrate the capabilities of each. We confess that the drawings were not done by NEWSCRIPT: they were taken from catalogs of 19th century illustrations.



SECTION I

INTRODUCTION, INSTALLATION, AND TUTORIAL

OVERVIEW

"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 "EDIT". 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".

"EDIT" is based on the text-editing methods used by IBM on its "mainframe" computers. A number of features have been added to make it friendlier and more suitable for use on a personal computer. "SCRIPT" is also based on IBM's mainframe text processing systems. It traces its origins to "PROJECT MAC", which was begun at MIT in the early 1960's. This approach to formatting text has been refined by a number of large computer manufacturers and universities for over fifteen years, and its use of English-language mnemonics has been shown to be easy to learn and excellent for making revisions. The PROSOFT version of SCRIPT was begun in 1980 and today reflects many of the features suggested by magazine reviewers and our customers.

Learning NEWSCRIPT

Between EDIT and SCRIPT, NEWSCRIPT probably contains over two hundred separate features. The trick in mastering them is to do so a little at a time. If you try to learn everything at once, you'll wind up thinking NEWSCRIPT is very hard, but if you just set out to master the fundamentals, it'll take surprisingly little time and seem very logical, even simple.

To write with NEWSCRIPT, you'll use something called "EDIT". When doing so, you'll give it "commands", and a dozen of them are more than enough to write and revise a letter.

As you type your letter (or any other text, for that matter), you'll want to indicate where paragraphs start, where dates should go, which lines should be centered, and so forth. In other words, you'll want to control the formatting of the printed pages. The portion of NEWSCRIPT that does the formatting and printing is called "SCRIPT". When supplying formatting instructions to tell SCRIPT how your pages should look, you'll use EDIT to put "control words" right inside your text.

Control words actually are commands to SCRIPT, even though they are typed in along with normal text. The difference between "commands" and "control words" is that commands are obeyed immediately by EDIT, whereas control words are used much later on by SCRIPT, when it formats and prints your material. For the most part, EDIT doesn't understand control words, and treats them as normal data.

About this Book

We are writing this book for many kinds of people. Your computer background and Word Processing requirements probably are different from those of other readers, so some of what we cover may be important to you immediately, some may be old hat, and some may not make any sense until much later on.

This book is divided into about a dozen major sections. This first section is a tutorial, intended to help you learn about your computer and about NEWSCRIPT. It contains several short chapters, each of which will teach you a little bit about using NEWSCRIPT. If you're just starting to use a microcomputer, you'll find the first chapters of this section very useful: they cover the fundamentals of using your computer and the handling of diskettes. If you already know about these matters (or don't care), just skip ahead until you find chapters dealing with NEWSCRIPT itself. If you've used NEWSCRIPT (or its mainframe big brothers) before, you may want to follow the Installation instructions and then just skim the sections detailing EDIT and SCRIPT.

If you've been using another Word Processor up until now, we ask you to try to learn this one from scratch. Our methods are different from theirs (whoever "they" are), but most of us tend to stick with our "first love", comparing everything else to it. There is a chapter later on that does some of these comparisons, and if it covers your present Word Processor, you may want to review its contents.

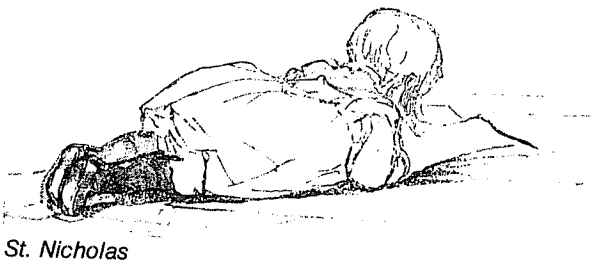
Using This Book

NEWSCRIPT has many capabilities... too many to fit within the TRS-80's memory all at once. So, we've separated these capabilities into several "programs", which are groups of instructions that the computer will follow if all goes well. We've also linked these programs together to make the transitions between them simple or invisible, so that you can concentrate on writing and not have to worry about the mechanics of the word processor.

This book explains how and when to use each program and each feature. The topics are covered in this sequence:

Section I - Introductory Material

1. using the computer; handling of diskettes
2. installing and backing up NEWSCRIPT
3. overview of fundamental facilities
4. activating NEWSCRIPT
5. introduction to EDIT - the screen, moving the cursor
6. EDIT control keys - the quick, easy way to move around
7. inserting and deleting text
8. obtaining status information
9. introduction to the SCRIPT language of control words



St. Nicholas

Section II - Advanced Tutorial

1. boldface, italics, sub-scripts, super-scripts
2. hanging indents
3. moving, copying, and deleting blocks of text within a file
4. moving blocks of text between files

Reference Sections

* EDIT - alphabetical description of all commands
* SCRIPT - alphabetical description of all control words
* Indexing facilities
* HOW TO... - tricks and techniques
* How to write with a computerized Word Processor
* Supplemental Information about Printers and Operating Systems
* Summary of Tiny DOSPLUS Operating System Commands
* Additional and Optional Features

As you read this document, please note that it was written entirely in EDIT, formatted by SCRIPT, and printed on a good dot matrix printer with proportional capabilities. 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 d o u b l e - w i d t h letters, for example.

Some pages were done on different printers to show you the capabilities of those printers when used with NEWSCRIPT. The art work wasn't done with NEWSCRIPT: most of it was taken from catalogs of 19th century illustrations and included to provide a light note here and there.

REQUIRED EQUIPMENT

To use NEWSCRIPT, you need these pieces of equipment:

* TRS-80 Model I or III or equivalent (LNW, etc.)
* 48K of memory
* one disk drive (2 are recommended on Model I)
* lower-case hardware (cannot type or print lower-case without it)
* a printer

Chapter 1 - Getting Started

USING THE TRS-80

The first part of this chapter covers the fundamentals of using the TRS-80 and diskettes. If you're familiar with these subjects, just skip ahead until you hit a topic that's new to you. If you've just started using your new micro-computer, some of what we say here may make the machine more understandable.

The Model I is turned on by pressing a plunger-switch along the back of the keyboard, near the right side. Generally, it's good practice to turn on the disk drives, display, and Expansion Interface before turning on the keyboard. The power switch on the Model III is underneath the right edge of the keyboard area. When turning either computer on or off, diskettes should be removed from the disk drives, or the doors to the disk drives should be open. Otherwise, momentary voltage surges can erase small patches of information from the diskettes. Since there are more than 50,000 bits of information per square inch on these diskettes, it takes very little erasure to destroy a whole lot of information.

When we refer to certain keys and buttons, we usually will show them like this:

<ENTER>

That refers to the white key marked "ENTER", not to seven keystrokes you should type. Most of the time, it will be necessary to press <ENTER> to tell the computer you've finished typing in a command and want the computer to obey that command. Similarly, it sometimes will be necessary to press <CLEAR>, <BREAK>, or one of the other special keys on the keyboard. Finally, it sometimes will be necessary to press the <RESET> key to start everything over from scratch. On the Model I, <RESET> is a round plunger-type button in the back of the keyboard, just to the left of the cable connecting the keyboard to the Expansion Interface. On the Model III, <RESET> is the orange button at the extreme right side of the keyboard.

The Parts of A Computer

For our purposes, the TRS-80 consists of a keyboard, a TV-like display (sometimes called the "video" or "monitor"), one or more disk drives, a printer, internal "memory" (sometimes called "RAM" for Random Access Memory), and a very fancy calculator called a "micro-processor" (sometimes called "CPU" for Central Processing Unit). The TRS-80 uses a micro-processor called the "Z-80". This remarkable device is housed in a gum-stick sized piece of plastic, but the device itself is only about 1/4 of an inch square, and considerably thinner than a dime. It has the ability to follow instructions (which are contained in computer "programs"), do arithmetic, compare things and select alternative sets of instructions based on the results of these compares, and to electronically move information around.

When you turn the computer on, it begins to follow instructions permanently stored in a portion of memory called "ROM" (Read Only Memory). This ROM was built for Radio Shack and retains its instructions even when the computer is turned off. The ROM cannot be changed except by damaging it, which is why it's called "Read Only". By contrast, RAM can be changed, and is used to store things temporarily.

The main functions of a computer are divided into five categories: arithmetic, logic ("is 'A' less than 'B'?), storage (in internal "memory"), receiving information from the outside world (called "input" or "reading"), and sending information from its memory to outside machines (called "output" or "writing"). The keyboard is an "input" device, the video display is an "output" device, and disk drives are used for both input and output. Sometimes, we will refer to input and output together by using the term, "I/O".

The instructions in the ROM attempt to "read" information from a diskette in drive 0. If this succeeds, those instructions are placed in RAM (an area that can be changed very easily) and the CPU begins to follow those instructions. The instructions are part of something called an "Operating System", which is discussed elsewhere and is often abbreviated "O/S" (this has nothing to do with Dorothy). Since Operating Systems usually are stored on disks, they may also be referred to as "Disk Operating Systems", or "DOS". NEWSCRIPT is supplied with an O/S called TDOS, which is a tiny version of the well-known DOSPLUS. However, it can be used with several other O/S's if you prefer. (If you've wondered why computer people use so many abbreviations, its because they come up with so many big words, and then need shortcuts to avoid tripping over their tongues.)

Once the DOS takes control of the computer, you can begin to issue "commands" to it. Commands usually are the names of computer programs that were written by such people as the authors of DOSPLUS, the authors of NEWSCRIPT, or yourself. However, you don't need to write programs to use a computer... you can buy them, just as you've bought NEWSCRIPT.

Because the RAM (memory) of computers is fairly small, and because turning off the computer causes the contents of RAM to be lost, it's necessary to store programs and "data" (data may be a letter, information about your payroll, a mailing list, etc.) on something that is fairly permanent. Otherwise, you'd have to type everything back in every time you wanted to use the computer. (When computers were first developed in the early 1940's, there was no permanent storage, and it took days or weeks to enter each program. This probably had some influence on the fact that there were only a dozen or so machines built in the entire decade.) The two most common forms of storage on micro-computers are magnetic tape (music cassettes) and flexible magnetic diskettes. Since information and programs can be read from and written to diskettes much faster and more reliably than from/to cassettes, we'll just talk about diskettes from now on.

USING THE KEYBOARD

Throughout this tutorial, when we want to indicate the use of a particular key on the keyboard, we'll show it like this:

<SPACE> or <ENTER> or <N> or <CLEAR><T>

When a key is shown that way, it means it should be pressed to produce the specified result. If two keys are shown together, as in the last example above, it means the first and then the second should be pressed in sequence. Unless otherwise stated, the first key should be held down until the second one is pressed, and then both can be released.

Most of the TRS-80 keyboard works just like a typewriter keyboard: when you press a letter or a number, it will appear on the screen and the cursor (a small line or square) will advance to the next position. When you hold down the <SHIFT> key and press another key, either the capitalized form of the letter, or a special character, will appear.

Several keys work differently on the TRS-80 than they do on a typewriter:

<LEFT ARROW> moves the cursor back over the character to its immediate left. Most of the time, it also erases that character from the screen, but when using the full screen editor, it just super-imposes the cursor over the previous character without erasing it. <LEFT ARROW> is the computer's "backspace" key.

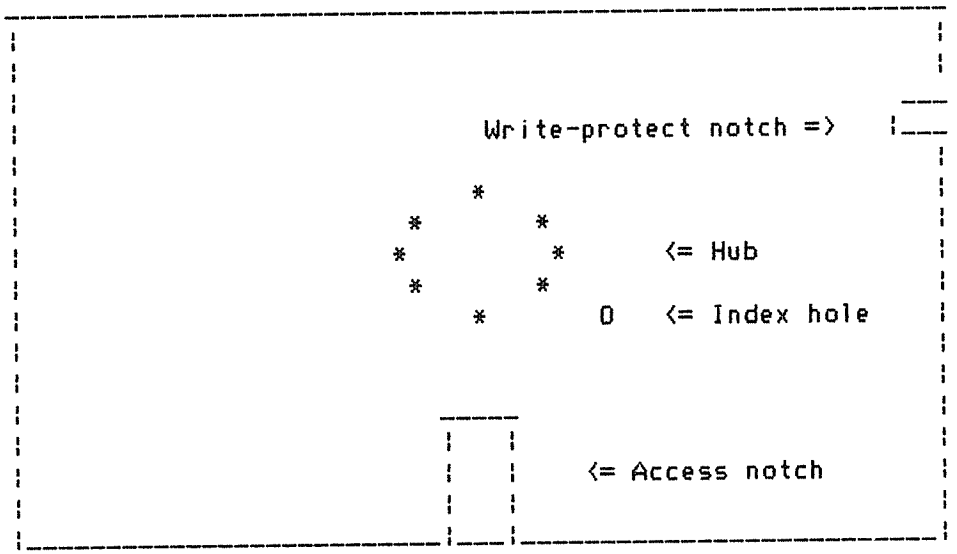
<ENTER> does two things: it indicates that what's been typed should now be examined and processed; and moves the cursor to the start of the next line on the screen. When used within the full-screen editor, it functions a little differently, and the differences will be explained later on.

<SHIFT><@> tells the TRS-80 to pause in whatever it's doing. To tell it to resume, just press <ENTER> or almost any other key. This "pause" function is honored by most programs, but not by all: it depends on what's being done.

<BREAK> is a key that should be used cautiously. Usually, it tells the TRS-80 to quit whatever it's doing. If you aren't sure how to get things to continue afterwards, this key should be avoided unless our instructions give a safe, special use for it.

THE CARE AND FEEDING OF DISKETTES

The diagram below identifies the main parts of a diskette. As this is written in mid-1982, the most popular form of diskette for the TRS-80 is the 5-1/4 inch "mini-floppy", a thin, circular piece of plastic coated on both sides with iron oxide and enclosed in a lined paper jacket.



We need to notice several things about the diskette:

First, the "U" shaped "ACCESS NOTCH" is where the magnetic head of your disk drive sits when it is reading from, or writing to, the diskette. YOU SHOULD BE CAREFUL NEVER TO TOUCH THIS PORTION OF THE DISKETTE, as it is very sensitive to dust, dirt, and oil. The oil deposited by a fingerprint will cause dust to stick, and that will cause scratches on the soft plastic surface. A scratch too small to see can erase dozens of "bits" of information. By the way, if you have "single-sided" drives (the kind that record information on only one of the disk's two surfaces), the recording surface is the underside of the disk, not the side with the label. When inserting a diskette in a disk drive, the access notch should enter the drive first, and the label should face the movable side of the door (usually face up or right).

The little hole marked "INDEX HOLE" is used by the computer to synchronize the drive with the locations on the disk it needs to access. If you have a Model I, the disk you received from us probably was a "flippy" disk, which means it is recorded on both sides. Your computer only can read one side at a time (even if you have two-sided drives), so you'll have to turn the disk over to read the second side. We bring this up now because these special disks of ours have two index holes and two write-protect notches. Side 1 is accessed as a normal disk, with the label "up" (or to the right), while Side 2 is accessed by flipping the disk over.

The "WRITE-PROTECT NOTCH" is used to tell your Disk Operating System (or DOS) whether or not writing to the disk is to be allowed. When you buy a box of diskettes, the box usually contains several small, opaque, adhesive tabs that can be used to cover the write-protect notch. They are called "write-protect tabs" and should not be placed anywhere else on the diskette, particularly at the access notch. When using 5-1/4 inch disks on the TRS-80, covering up the write-protect notch with a write-protect tab tells the computer that it must not modify the contents of the diskette (must not write to it). When the notch is uncovered, it is O.K. to write to the disk.

Your original NEWSCRIPT diskette should have silver or black tabs covering the write-protect notch(es). If not, please put one on before using the disk. The tab should never be removed: that will help protect you against accidentally erasing or writing over NEWSCRIPT. In a moment or two, we'll explain how to make copies, or "backups", of NEWSCRIPT. You should make changes ONLY to backups, never to original disks. The originals serve as a starting-point in case you destroy everything else by mistake, so if you also destroy the original, you'll have to send it back to us for a refresh. This will take a week to ten days, and we believe you'll agree it's cheaper to use an extra backup disk to save the time. Since returning the original disk to us may be the only way you can receive an update to NEWSCRIPT, please make sure you don't lose it, remove our labels, or otherwise change it in any way.

You may wish to place labels on your diskettes so that you can tell which of your diskettes is which. The best procedure for this is to write your label before sticking it on the diskette, or else use a felt-tipped pen. When you place a label on your diskette, make sure it is well away from the write-protect notch, index hole, the drive hole, and especially the access notch.

Whenever you have a diskette that is particularly important to you, be sure to always keep at least one current backup copy (the procedure for this is described below), store it separately from the working copy, and put a write-protect tab on it (unless you need to write to that particular diskette, in which case you may want to keep 2 backup copies). Recovering data from damaged disks is a difficult, painstaking process, even if you know the procedure. While we have recovered diskettes by this process, we find it much easier to simply make another backup copy from the safely-stored master. As you read through this book, you'll find a continuing (some would say, "obsessive") emphasis on the need to make and keep backups of all programs and data files. There's no such thing as too much backup: a

diskette costs less than $3.00, and recovery of its lost contents could take anywheres from a few minutes if you're lucky, to days or never if you're not. One of Murphy's Laws states that "...the probability of lost data is directly proportional to the difficulty of recreating that data." You probably can think of several other grim remarks, all of which are funny until they come to life at the worst possible time.

A few more words about handling diskettes...

A computer diskette is truly a wonderous invention. You can store 100,000 characters of data (or more) on a single side of a piece of mylar plastic 5 1/4" in diameter. You can read or write data from or to it as it spins in your disk drive at 300 RPM and presses against a magnetic read/write head. But with this fantastic power comes some responsibility - YOU MUST TAKE CARE OF YOUR DISKETTES! Here are some DOs and DONTs of diskette handling:

* DO make backup copies of your important diskettes, and keep "off-site" copies to guard against damage or theft

* DO frequently save your text files as you enter them to prevent loss of data

* DO write-protect your important diskettes

* DO keep your diskettes away from ALL magnetic fields, including:

    a. motors, including your disk drive (never lay diskettes on top of your drives!)

    b. your CRT (especially the left side of the CRT)

    c. magnets, magnetic screwdrivers, etc.

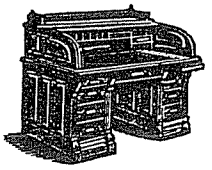
* DO keep your diskettes in their paper covers and in dust-proof diskette boxes when not in use!

* DON'T EVER touch the plastic surface of the diskette itself -- handle the diskette by the jacket and center hub only

* DON'T write on your diskettes with anything harder than a Q-TIP! Use Felt-tip pens.

* DON'T EVER power-up or power-down your computer with diskettes in the drives! This can erase spots on them, causing permanent loss of data

Treat your diskettes as you would an old cherished original Bach manuscript, and they'll literally give you years of service.



OPERATING SYSTEMS

Computers are machines that obey electronically encoded instructions. A group of instructions intended to perform a given set of functions is called a "program". A group of programs that work together to perform a variety of related functions is called a "system" (just as several sets of accounting procedures might be called a bookkeeping system).

If these functions were simply calculations and "yes-no" kinds of decisions, computers would be fairly simple things to use. However, most of us want our computers to store information for later use (usually on diskettes in the case of the TRS-80), retrieve this information when needed, display the results of calculations, and accept new information from the keyboard. This complicates things tremendously, since storing information on diskettes (called "writing") and retrieving it later on (called "reading") involves precise timings, careful placement (we don't want to write over something else, or be unable to find things later on), and frequent error recovery (sometimes information is not written correctly the first time, or is hard to read back for a variety of reasons).

A specialized, sophisticated group of programs is needed to cope with these requirements. They are called "Operating Systems" because they perform many of the functions needed to keep the computer operating productively. There are several very good Operating Systems available for the TRS-80, and NEWSCRIPT can be used with most of them. We've included "TDOS", which contains a portion of the O/S called "DOSPLUS", to make it easier for you to start using NEWSCRIPT immediately. TDOS is very easy to use, and the instructions given below for making disk backups (copies) assume you're using TDOS (or DOSPLUS). If you prefer to use some other O/S, by all means do so, and follow the equivalent procedures supplied with it. If you're using a Model III and want to move NEWSCRIPT to another Operating System, please see Section IX, near the end of this manual.

Operating Systems That Work with NEWSCRIPT

NEWSCRIPT has been tested successfully with all of the operating systems listed below. We cannot commit to maintain compatibility with future, as-yet unreleased versions of any of these, since we obviously cannot predict what their authors may do to their products. Also, we do not claim that NEWSCRIPT will function correctly with modified versions of these operating systems, or with other operating systems.

TRS-80 Model I	TRS-80 Model III
DOSPLUS 3.3, 3.4, 4.0	DOSPLUS 3.3, 3.4, 4.0
LDOS 5.1.1, 5.1.2	LDOS 5.1.1, 5.1.2
MULTIDOS 1.4	MULTIDOS 1.3
NEWDOS 2.1	NEWDOS/80 Version 2
NEWDOS40	TDOS 3.3, 3.4
NEWDOS/80 Versions 1, 2	TRSDOS 1.2, 1.3
TDOS 3.3, 3.4	
TRSDOS 2.3, 2.7DD	

Chapter 3 - Installing NEWSCRIPT

BACKING UP YOUR NEWSCRIPT DISKETTE

The Model I version of NEWSCRIPT is in 35-track, single density, standard format. It is distributed either as a two-sided diskette or as two one-sided diskettes. When you follow the procedure below, copy both diskettes or sides, winding up with two diskettes of your own. Whether we refer to "side 1" "diskette #1" hereafter, it'll mean the same thing.

The Model III version of NEWSCRIPT is in 40-track, double density, DOSPLUS format. It cannot be read by TRSDOS, but its contents can be transferred to other Operating Systems if necessary. NEWSCRIPT fits on one side of a normal, 40-track double-density disk, so only that one side should be backed-up.

<*> C A U T I O N <*>

NEVER REMOVE THE WRITE-PROTECT TAB FROM THE ORIGINAL DISTRIBUTION DISK. NEVER WRITE TO OR MODIFY THE CONTENTS OF THE ORIGINAL DISTRIBUTION DISK. BEGIN BY MAKING A BACKUP COPY OF IT AS DESCRIBED BELOW. MAKE COPIES OF THE INSTALLED VERSION, AND MAKE REGULAR COPIES OF ALL DISKS CONTAINING ANY TEXT YOU WRITE. THE COST OF A DISKETTE IS FAR LESS THAN THE VALUE OF THE TIME YOU'LL LOSE IF DATA IS LOST!

Backup Procedure with TDOS

To use NEWSCRIPT with any other Operating System, refer to SECTION IX, near the end of this book. The Backup, Format, and Installation procedures described here will only work with the supplied TDOS.

1. If your computer is off, turn it on. Make sure no diskettes are in the drives when power is first applied.
2. After the disk drive motors stop spinning, insert Side 1 of the NEWSCRIPT diskette in drive 0 (the first disk drive) and press the <RESET> key. On the Model I, this is located at the rear of the keyboard unit on the left side, next to the cable connection for the expansion interface. Model III users will find it at the upper right corner of the keyboard (the orange button). A message called "WELCOME TO NEWSCRIPT" should be displayed, after which you should see the prompting word, "DOSPLUS". Whenever this prompt is on the screen and nothing else is happening, it means you can enter a "command" to tell the computer what to do next. If you hit the wrong key as you type, you can erase the keystroke by pressing <LEFT ARROW> (unless you've hit <ENTER>).

3. To initiate backup, just type this:

   BACKUP (and press <ENTER>)

4. You will be asked the following questions:

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

5. The source drive is '0'. The answer to the second question depends on the number of disk drives in your system. If you have 2 or more drives, then enter '1' (the second drive number); if you only have one drive, then enter '0'. Remember that when you enter each answer, you must terminate it by pressing the <ENTER> key on your keyboard. Supplying the date lets you keep track of your diskettes, and should be today's date. If the day is "July 4, 1982", it must be entered as "07/04/82", not as "7/4/82".

   If the destination diskette had been used previously, you may get a message asking you whether to use it anyway. Reply 'Y' if it's O.K. to write over what used to be there. Otherwise, reply 'N' and find another disk. If you do reply 'N', you'll be told to press <ENTER> when a "SYSTEM" disk is in drive 0. The NEWSCRIPT disk is a SYSTEM disk, so you can just press <ENTER> immediately. Once you've found another disk, type in the "BACKUP" command again and start over.

6. If you have entered '0' to both the first two questions (meaning you have only one drive), then you'll have to swap the source disk (NEWSCRIPT master disk) and the destination disk (the diskette to which you are copying NEWSCRIPT) back and forth several times. The DOS will tell you when and which diskette to insert.

7. If the backup fails for any reason, the DOS will inform you of that fact. If this should occur, restart the backup procedure using another blank diskette. If the problem appears to be with the SOURCE diskette (the original NEWSCRIPT), carefully wrap it in cardboard and mail it back to us, with a brief explanation of the problem. We will replace the disk at no charge. Please don't use staples, don't overseal the package, and don't send back the documentation...just the bad disk. It's not uncommon for disks created on one drive to be unreadable on another, or for disks to be damaged or erased in transit. We generally will have a replacement in the mail to you the day after we receive the old disk, and will send it to you via First Class mail. (If you're sending it back because you suspect a "bug" [that's a problem with a program], it may take longer, since we'll have to look at the problem.)

8. When BACKUP finishes, it will prompt you to insert a System disk and press <ENTER>. A "System" disk is one with the Operating System on it, and at this point, both the original NEWSCRIPT disk and the copy you've made qualify. So, remove the original disk, move the new one from drive 1 to drive 0, and press <ENTER> as requested.

9. The disk you've just created will become your everyday working copy of NEWSCRIPT, and this is a very good time to place a label on it. Before actually using NEWSCRIPT, you'll also have to follow the Installation procedures that will customize it to your printer and typing preferences. At this point, it's a good idea to tell the Operating System you'll want NEWSCRIPT started automatically each time you press <RESET>. This setting can be made whenever the "DOSPLUS" prompt is on the screen and a square cursor is below it. Type:

AUTO NS

By the way, if you ever want to prevent NEWSCRIPT from starting automatically, just hold down the <ENTER> key when you press <RESET>, and keep holding it until "DOSPLUS" flashes on the screen. Then release it, and you can issue any of the commands supported by this tiny version of DOSPLUS, which we will call "TDOS". If the word, "what?" appears when you do this, just ignore it.

PREPARING DATA DISKS (USING THE "FORMAT" COMMAND)

Once you have a copy of the NEWSCRIPT program disk, you'll want to prepare a few "data" disks. These will hold the letters and documents you create with NEWSCRIPT. If you have more than one drive, use the "FORMAT" command described below. If you have only one drive, then you must make another backup of NEWSCRIPT and use the DIR command to list the files on the diskette. Using the KILL command, KILL all the files displayed by the directory. Then use this diskette to store your data (text) files. It will contain the DOSPLUS Operating System as well as your data, and by switching back and forth between these data disks and the NEWSCRIPT disk, you'll be able to create and print your documents. Here's how you might begin clearing space on a backup of the backup:

```
DIR
KILL TRSDOS/MSG
KILL TRSDOS/BLD
KILL EDIT
      (...and so on.)
```

FORMAT COMMAND

If you have more than one drive, you'll want to prepare data disks through use of the "FORMAT" command, which is similar to "BACKUP" except that it doesn't copy anything from one disk to another. Its main purpose is to format a diskette for use in storing data, since a diskette cannot be used until it has been formatted (the BACKUP command performs a FORMAT operation before copying anything from the source disk to the destination).

From the DOSPLUS prompt, enter FORMAT. You will be asked several questions:

```
WHICH DRIVE IS TO BE USED?
DISKETTE NAME?
FORMAT DATE?
MASTER PASSWORD?
NUMBER OF TRACKS?
SINGLE OR DOUBLE DENSITY?
DISKETTE CONTAINS DATA, USE OR NOT?
```

The last of those will be asked ONLY if there already is some data on the diskette.

Answer each question in turn. Enter the drive number to be used (0-3), and the name that you wish to call your data diskette, such as DATA01, or something equally original. The format date should be today's date. The master password should be something you will not easily forget. NEWSCRIPT just leaves the password blank, and you may with to do the same.

The number of tracks to be used depends on the capabilities of your disk drives. Some old Model I drives are limited to 35 tracks, but now there are 40 and 80 track drives. Most drives can access 40 tracks. If in doubt, "35" always will work (but may waste disk space). If you've only used MODEL I TRSDOS until now, you may find to your surprise and glee that your drives can handle 40 tracks. Try specifying '40' tracks; if your drive can't handle it, the FORMAT program will issue lots of error messages, after which you can specify '35' when you retry the command. If you don't get error messages, then you've just increased your diskette capacity by 14%. Not bad, not bad.

On the Model I, TDOS only supports single-density drives. Other Operating Systems, including full DOSPLUS, can handle double-density, two-sides, etc. On the Model III, both single- and double-density are supported, but of course you'll normally want to use double-density. TDOS only supports single-sided drives, but can handle up to 96 tracks if you have such drives.

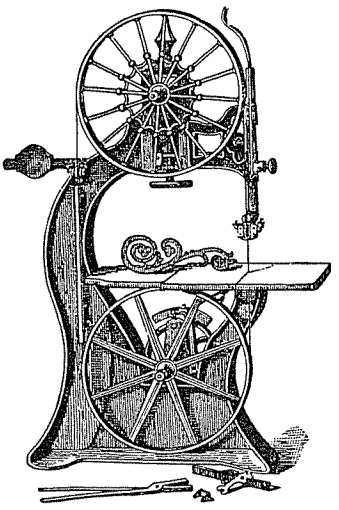
If you've used a diskette previously, it will contain data. To protect you from accidentally erasing valuable information on such a disk, DOS will ask you if you are sure you want to proceed. If not, just answer 'N'. A response of 'Y' will cause formatting to occur.

FORMAT PERMANENTLY ERASES THE DATA ON A DISKETTE -
DON'T USE IT ON DATA YOU WISH TO KEEP!!

Note that on the Model I, the DOSPLUS disks may be read by TRSDOS, but on the Model III, they cannot. If you're using a Model III and want to transfer programs or data from DOSPLUS to TRSDOS, then FORMAT a single-density, 35-track disk with DOSPLUS (yes, it works just fine), copy what you need onto it, and then switch to TRSDOS. The "CONVERT" command of TRSDOS will transfer your files and make them readable on a native TRSDOS diskette.

If you want to move NEWSCRIPT to NEWDOS/80 on the Model III, create the single-density diskette with TDOS, and then use "PDRIVE" in NEWDOS to describe the diskette. The directory will be on track 17 and there are two granules per lump.

If you want to use NEWSCRIPT with LDOS, the single-density diskette can be read directly by LDOS. It will try to read the disk a few times, accompanied by much groaning and arm movement in the disk drive, and then switch to single-density and proceed successfully.



INSTALLING AND CUSTOMIZING NEWSCRIPT

If you don't have the "DAISYWHEEL PROPORTIONAL OPTION" ("DWP"), please skip down to the next topic, "Installation Procedure." If you have a Radio Shack Daisy Wheel Printer II, you don't need, and can't use, this option, so please don't start worrying that something is missing: the option is only for Spinwriter and Diablo-type printers.

Installing The Daisy Wheel Proportional Option

DWP must be installed before the regular Installation Procedure is performed, or NEWSCRIPT will not work. If you add DWP later on, you must run the installation procedure again. The Option is supplied on a single-density, 35-track disk. If you have a Model III, then you'll have to make this diskette readable, and copy one file from it to your own System disk before going any further. Here's how:

TDOS or DOSPLUS

1. If there's a write-protect tab on the DWP disk, remove it.
2. Insert DWP in drive 1, and the backup of NEWSCRIPT in 0.
3. Type: CONVERT :1
4. Type: COPY PROP/CIM:1 :0

TRSDOS

1. Insert DWP in drive 1, and a TRSDOS disk containing the rest of NEWSCRIPT in drive 0.
2. Type: CONVERT :1 :0

NEWDOS/80

1. Insert DWP in drive 1, and a NEWDOS/80 disk containing the rest of NEWSCRIPT in drive 0.
2. Type: PDRIVE 0,1,TI=A,TD=A,TC=35,SPT=10,GPL=2,DDSL=17,DDGA=2
3. Type: COPY PROP/CIM:1 :0

LDOS

1. Insert DWP in drive 1, and an LDOS disk containing the rest of NEWSCRIPT in drive 0.
2. Type: COPY PROP/CIM:1 :0

If you have a Model I, just insert the DWP disk in drive 1, the NEWSCRIPT backup of Side 1 in drive 0, and type:

COPY PROP/CIM:1 PROP/CIM:0

Eventually, you may want to copy some of the other files and programs from DWP. If you have a NEC 5500 series printer and plan to use one of the proportional thimbles, then "SPIN/TAB" should also be copied at this time.

This ends the extra steps needed if you have the DWP option, so you can put its diskette away.

Regular Installation Procedure

Now, you are ready to customize NEWSCRIPT for your system. To install NEWSCRIPT on an operating system other than the supplied TDOS, please follow the instructions in Section IX, near the end of this book.

The installation procedure displays its own instructions, which are not repeated here since installation may change slightly as time goes by. However, NEWSCRIPT won't function until you've gone through this procedure at least once, and if you move NEWSCRIPT to a different model of TRS-80, or decide to use a different Operating System, it will be necessary to re-run the installation procedure first.

The installation procedure covers three main areas: your printer, your Operating System, and your typing preferences, including the degree of "debounce" needed on your keyboard, how fast you want the repeat speed to be, and whether the <BREAK> key should be partially disabled. If you have a Model I, the procedure is on Side 2 of the distribution diskette, which corresponds to the second backup diskette you (hopefully) made a few minutes ago. On the Model III, it's on Side 1, since there is no side 2.

To start the installation procedure with TDOS, insert the correct backup disk in drive 0 and type:

DO NSINSTAL <ENTER>

If you're using any other Operating System:

1. Activate BASIC with at least one file;
2. RUN "NSINSTAL"

If you're using a Model I, once the Installation Procedure begins, you'll have to remove the Side 2 disk and insert a copy of the Side 1 diskette in its place. Make sure this Side 1 is NOT write-protected (and make sure it's a backup, of course), since your answers will be written onto it.

The first few screens of the Installation Procedure will explain what should be done. Once you are presented with choices, you can usually just hit the <ENTER> key to indicate that a default value should be used. A "default" is something we selected, in our infinite wisdom, in case you don't have any strong opinions to the contrary. At first, most of the defaults should be used, but as you become accustomed to NEWSCRIPT, or find that something just doesn't work with your printer, you may wish (or need) to go back through installation and pick different answers.

The first and last questions of installation are the only ones that don't have defaults. The first question asks you to identify your printer. The last question asks you to confirm that your answers should be stored on disk. Everything in between can be answered just with <ENTER>, unless, of course, you need to supply an explicit choice. The defaults are marked with an asterisk in each of these menus.

Once the Installation Procedure completes successfully, set the "AUTO" command to "NS" as shown below. If you don't do this, you'll keep getting the "WELCOME" message every time you press <RESET>. In fact, since you have no further use for that one-time greeting, this would be an excellent time to "Kill" it (if NSINSTAL didn't kill it for you). Type:

```
KILL WELCOME      (press <ENTER>)
AUTO NS           (press <ENTER>)
```

Then, place a write-protect tab on this working copy of NEWSCRIPT, and make a backup of it. Thereafter, to use NEWSCRIPT, just insert the working copy, press <RESET>, and if you haven't set the "AUTO" command, type: NS (and press <ENTER>).

PRINTER SELECTION

NEWSCRIPT contains tables describing about two dozen different printers. Once you've told NEWSCRIPT which printer you're using, you won't have to know how your printer does something (such as underlining or proportional printing), but only whether it can do it. Since some printers can use the tables provided for other printers, and since there's limited room on the printer menu, your printer may be fully supported, but not explicitly listed. If that's the case, this information may be helpful. If it isn't, please call or write for help. Choosing the printer correctly, and setting the internal switches in many printers, is absolutely essential to satisfactory operation.

Automatic Line Feed

A "line feed" is a special control character sent out by the computer to cause a printer to advance the paper one print line. All Radio Shack printers do this automatically whenever they receive just a "carriage return" without a "line feed." Most other printers have a user-changeable switch to control whether "carriage return" should also force a "line feed", even in the absence of an explicit "line feed" character. If your printer has such a switch, it normally should be set to force a line feed whenever a carriage return is received. If there is no switch, or if the special instructions for specific printers in Section IX indicate that NEWSCRIPT should supply line feeds, then select this option when you run the installation procedure, and set the printer switch accordingly.

Other Printers

Here are some printers supported by NEWSCRIPT that are not on the printer menu:

Atari 825 - use Centronics 737
Centronics - use Centronics 737 and read up on ".BF 10"
C. Itoh Starwriter or F-10 - use Diablo
Olivetti 121 - use Modified Selectric
Qume - use Diablo (except for Radio Shack Daisy Wheel I)
SCM TP-1 - use Modified Selectric
Radio Shack Line Printer III, V, VI - use standard R/S
"typewriter-like printers" - use modified Selectric

If your printer is neither on the menu nor in the above list, the best choices are "None of the Above" or "Modified Selectric". If nothing seems to work, check the printer switches carefully, turn the printer off, then on again (most printers don't check the switch settings except when first turned on), and try again. If things still don't work, please call or write to us for assistance. If you've recently made a change to your printer (for example, added a version of "GRAFTRAX" to an Epson), be sure to re-run the Installation procedure before trying to use the printer with NEWSCRIPT, and specify the new feature you've added.

This completes the general Installation of NEWSCRIPT. You may also need to refer to SECTION IX, near the end of this book, for information on switch settings, other Operating Systems, and use of your own "printer driver." (If you don't know what that means, please use the driver built right into NEWSCRIPT; you don't need to know about it, or even that it's there.)

Chapter 3 - NEWSCRIPT Components

In this chapter, we'll identify the major components of NEWSCRIPT, give you a short list of the most fundamental features you'll need to use, and try to explain some of the zillion computerese terms we will be forced to use from time to time. However, we'd like to reminisce with you for a moment or two first...

WHAT'S A WORD PROCESSOR, ANYWAY?

Some of you older types may be familiar with an archaic, mechanical instrument known as the "typewriter." Back in the days of bear skins and stone knives (ca 1850-1950), this instrument was used in an attempt to put people's thoughts onto paper. It had a keyboard somewhat similar to the computer keyboard that accompanies your TRS-80, but resorted to using keys, bails, and levers to mechanically place words on paper. While this was an improvement over the use of a pen or pencil, it was still a very inefficient way to produce documents. Why, someone even invented white paint to be used to cover-up the errors that one made during the typing process! Towards the end of this pre-computer era, IBM invented a version that used a "golf ball" typing element to let this process go more quickly, although no more accurately.

In the late 1950's, Friden came along with a device that could type letters repetitively from information stored in punched paper tape. Once you got the tape right, you could print hundreds of form letters quickly and accurately. This was the start of word processing, and was followed by a variety of devices that would let you record text on magnetic surfaces (tape, card, or disk) and subsequently modify that text without having to retype all of it from scratch.

Needless to say, the initial thrill of being able to change text without retyping everything was soon replaced by demands for easier ways to do full-scale text manipulation and formatting. Computers were becoming widely used in the early 1960's, and one of the most ingenious developments was the ability to "edit" text. This, of course, led to full-scale word processing, which has recently become one of the most popular uses of micro-computers.

NEWSCRIPT is one of these word processors. It's based on the methods used on large computers, but has been adapted for use on the TRS-80 and includes features that take advantage of the unique aspects of personal computers. NEWSCRIPT's facilities are divided into several programs, most of which work together. You'll only need to be aware of a couple of them, but for the sake of completeness, the main programs are:

EDIT	-	allows full-screen text entry, revision, and update
SCRIPT	-	prints previously edited files; uses 'PROP/CIM'
INDEX	-	sorts and collates index references
FITLINE	-	splits large text files into several small ones
NSINSTAL	-	used to install and customize NEWSCRIPT
NSINIT	-	contains menus for selecting major functions
NS/CMD	-	central control for keyboard, printer, and other functions
FEDIT/CIM	-	performs full-screen editing for 'EDIT'
PROP/CIM	-	formats text; used by 'SCRIPT'
STARTUP/MIN	-	Operating System-dependent entry to BASIC
LABELS	-	optional address-label printing program
LIST	-	lists document files at variable speed
APPEND	-	joins several lists or files together

Several other programs and files are provided in case you want to run with other Operating Systems, but we'll ignore them for now and just talk about the four most visible functions of NEWSCRIPT.

NS/CMD

The only correct way to activate NEWSCRIPT is by issuing this as a DOS command:

NS

You can't start NEWSCRIPT by just going into BASIC and trying to run one of the programs, since most of the functions of NEWSCRIPT aren't written in BASIC. For those of you who care, "NS" is a Z-80 machine language program (one of three obvious ones in NEWSCRIPT, the others being "FEDIT/CIM" and "PROP/CIM"). It takes control of the computer, replacing parts of whatever Operating System you are using, and looks for instructions for what do do next. These instructions normally are in a small file called "STARTUP/MIN", and tell NEWSCRIPT to activate BASIC and then run a program called "NSINIT."

Note: The "NS" command can be issued only once for each time the computer is <RESET>. If you try to issue it a second time without pressing <RESET> first, the system will fail and the only way out will be to press <RESET> anyway.

NSINIT

Most of the major capabilities of NEWSCRIPT are selected from the menu or menus shown by "NSINIT". For example, if you want to see what files are on a diskette, you can usually display the "Directory" for that diskette by choosing option #6 from the "PRIMARY OPTIONS MENU". When you finish with any major function, you often will return to this menu to select the next thing to be done. The two most important parts of NEWSCRIPT are shown first on this menu: EDIT and SCRIPT.

EDIT

EDIT is a full-screen text editor. This means that it allows you to enter and change text anywhere in the document simply by moving the flashing cursor around, and making your changes. We will discuss how this is done in detail later. This kind of program is called an editor to associate it with the people at a newspaper or printing house who have to clean up the manuscripts submitted by writers.

SCRIPT

SCRIPT is a text formatter. SCRIPT reads the text that you have entered into your computer with EDIT and, obeying the "control word" instructions imbedded in your document, formats the text into the finished document and then prints it out on your printer.

While NEWSCRIPT is composed of several programs, the selection and operation of most of these programs is done by the computer, so that you, the user, needn't worry about which program should be loaded and executed. It is a good idea to learn about how NEWSCRIPT operates, but it is not necessary for the production of good documents.

NEWSCRIPT knows what it should be doing, and it will ask you for answers to questions as needed. All the questions (with the exception of the I.D. of the file) have default values that will be used if you just press the <ENTER> key on the keyboard of your computer.

EZEDIT AND EZSCRIPT

Only a few EDIT commands and SCRIPT control words are needed to produce excellent-looking letters and short reports. The fundamental ones are listed below just to give you an idea of what's involved. Each of these will be covered in the tutorial that begins in the next chapter, and covered again in the Reference sections. As your skills and needs increase, you'll want to learn more about NEWSCRIPT, but these make a good "starter set:"

EDIT

<CLEAR> KEYS		PRIMARY COMMANDS		CURSOR
<F> forward	<B> back	Locate	Change	four arrows
<T> topfile	<E> endfile	SAve	ENd	
<D> delete	<I> insert	Whoops	QUit	
<W> word delete	<H> help			

SCRIPT

BLANK LINES	PAGE LAYOUT	BIG FILES	ESCAPE CODES
.SK skip	.FO format on/off	.AP append	!$ start underline
.PP paragraph	.CE center	.IM imbed	!% end underline
.PA new page	.BF begin font		!(start double width
	.IN indent		!) end double width

Looks simple, doesn't it? Since NEWSCRIPT uses mnemonics, you already know most of these commands. Now, it's just a matter of finding out exactly what they do and when to use them.

DISPLAYING THE DIRECTORY

Once you begin to create disk files, you'll find it difficult to remember all their names or where they are stored. Keeping track of them on paper, on labels, or with a cataloging program is a good way of overcoming this problem. With NEWSCRIPT, there are four ways to examine the directory (this depends on which Operating System you use):

1. From DOS, this command may be issued: DIR
2. From the PRIMARY OPTIONS MENU, option '6' may be used. It won't work with Model I TRSDOS or NEWDOS 2.1, nor will the other methods below. Only #1 works with those two;
3. When EDIT or SCRIPT asks for a fileid, you can reply with a question mark, optionally followed by the drive number;
4. When in EDIT, you can enter this command: DIR n

(where 'n' is the drive number.)

The TDOS directory display is very brief: it lists only the I.D.'s of the files on the disk in question. This is also true when the directory is displayed from EDIT or SCRIPT under TDOS. When it is displayed from the PRIMARY MENU, the amount of "free space" remaining on the disk is also shown, and the files are listed alphabetically.

When the directory is displayed with other Operating Systems, the level of detail shown depends on that O/S.

ABBREVIATIONS, SYMBOLS, AND BUZZ-WORDS

Before going further, it might be a good idea to define some of the terms we will be using. We've been doing this all along and will continue to do so as necessary.

Notation

1. The CAPITALIZED portions of commands must be typed in exactly as shown. The lower-case portions may be omitted.
2. Most commands accept "operands" (see definition below). When operands are shown in square [brackets], they are optional; when shown in angular <brackets>, a choice must be made of one of the values shown. Even though brackets are shown with many commands, they never should be typed in.
3. the lower-case letter 'n', possibly followed by a number, 'n1', 'n2', etc., indicates that a number must be used.

ASCII - a way of representing letters, numbers, special symbols, and control values in a computer. All characters, regardless of whether they are letters, numerals, or symbols, can be identified by a number in the range 0-255. The term, "ASCII FILE" is sometimes used to describe files that can be displayed or printed as-is, without having to convert special codes to printable values. All NEWSCRIPT text files are in "ASCII FILE" format. The only ASCII value that you must never enter is '13', and the only value NEWSCRIPT won't accept is '0'. However, attempts to use other values below '32' may or may not work, depending on how and why they are entered.

BASIC - a high-level computer language used in NEWSCRIPT. Because it resembles English, BASIC is easy to learn. It has become very popular in the last few years, and if you decide to learn just a computer programming language, this is the one to start with.

command - a word that tells the Operating System or NEWSCRIPT what to do. 'DIR' and 'NS' are commands to the O/S; 'EDIT' is a command to BASIC (but you'll almost never have to use it since it's also on the NEWSCRIPT menus); and 'CHANGE' is a command to EDIT.

control word - a formatting command to the SCRIPT text formatter. All control words consist of a period and a two-letter mnemonic. For example, ".CE" causes the next line of text to be centered, ".PP" starts a new paragraph, and ".PA" starts a new page. Control words are entered into text files when you use EDIT. They must either be right at the start of a line, or must follow other control words and be separated by semi-colons (example: .PA;.CE). NEWSCRIPT understands over 50 control words, but you only need to learn four or five of them to write most letters.

cursor - a character that is used to indicate your current position on the video display. In NEWSCRIPT, the cursor usually is a blinking square that may be thought of as the letter 'oh'. When you're using EDIT, the cursor may have any of three shapes ('o', 'i', or 'c', for 'overlay', 'insert', or 'control' modes), depending on what you're doing at the moment.

data - information that can be read or written by a computer.

default - this is what will be assumed and used when you give DOS or NEWSCRIPT an ambiguous instruction. For example, if you don't indicate the disk drive onto which a file should be written, the "default" will be drive zero, unless drive zero is "write-protected", or unless the file in question already exists on another drive. When you don't want a default to be used, you'll have to supply an explicit value to be used instead.

DOS - short for Disk Operating System, which is a set of programs that oversees the operation of the computer and the disk drives. Examples of this for the TRS-80 are: TRSDOS, NEWDOS, LDOS, TDOS, and DOSPLUS. (There are several others and no recommendation or criticism is intended by listing or omitting any O/S here.)

file - a file refers to a block of space on disk that contains either a computer program or text. We normally will use this term only in reference to text and data, not in reference to programs.

fileid - the name or designator of a file. On the TRS-80, a fileid (or 'filespec'... we may use either of these interchangeably) consists of at least one part, and may have up to four parts, which, if used, must be in this order:

filename/extension.password:drive#

filename is always required. It may be from one to eight characters in length. The first character must be a letter, and the other characters may be letters or numerals. Special symbols are not allowed.

extension is optional. If used, it must be preceded with a slash (the diagonal line shown above), begin with a letter, and be not more than 3 characters in length. Extensions usually are used to identify the nature or classification of the file. For instance, "/BAS" usually means a BASIC program and "/CMD" means a machine language command program.

password is optional. If used, it must be preceded by a period. It may be up to eight characters in length. NEWSCRIPT doesn't use passwords, but the Operating Systems do.

drive number is optional. If used, it must be preceded by a colon. It is a single number, usually between 0 and 3 (sometimes higher). It may be necessary to include the drive number in a fileid to make sure the file is written to, or read from, the correct disk drive.

font - also known as 'pitch', this is the spacing between printed characters. In proportional font, the spaces between characters are adjusted according to the width of each character. The other fonts are called monospace fonts and all characters are given the same amount of print space. If your printer has the capability, NEWSCRIPT lets you print at any of these pitches: 10, 12, 16, or proportional; and any of those may be in double-width if the printer allows it. By the way, when we refer to '16' pitch, we really mean "approximately 16 characters per inch (cpi)", since different printers may use 16.0, 16.5, 16.7, or even 17.1 cpi.

hexadecimal - the base 16 representation of a number. The hexadecimal representation is used to enter special codes with the <SHIFT><CLEAR> commands. You don't need to learn about this unless you want to enter certain special graphics or foreign print symbols.

machine language - the binary (or hexadecimal) codes that form instructions for a computer. The TRS-80 (and therefore NEWSCRIPT) use "Z-80 Machine Language". The "Z-80" is the "brains" of the computer. Three of the main components of NEWSCRIPT are programs written in "machine language", and other portions of NEWSCRIPT contain small amounts of "machine language" code to obtain extra speed or special capabilities..

mnemonic - an abbreviation whose letters remind you of the full word(s) for which it stands. "IOU" is an example of this, as is "C.O.D." You'll be using mnemonics constantly when you work with NEWSCRIPT. We've based all commands and "control words" (see below) on the English language, so learning NEWSCRIPT is largely a matter of finding out which synonyms we've chosen.

monitor, screen, or video - the monitor is the display, or TV, that is used by your computer to display words, text, and graphics.

operand - a word or number that follows a command. Most commands require or can accept one or more operands. In this book, operands are shown either in square brackets: [ ] or angled brackets: < >. The brackets themselves never should be typed in along with the values; we use them only to indicate where you should choose one of the values shown or substitute a constant for a variable.

record - a unit of related information. In NEWSCRIPT, each line of text entered into the editor is counted as one record.

special keys in the tutorial are indicated by being contained within "< >" brackets. Thus <ENTER>, <CLEAR>, <BREAK>, <SHIFT>, <SPACE>, <UP ARROW>, <DOWN ARROW>, <LEFT ARROW>, and <RIGHT ARROW> are the main special keys on the TRS-80 computer. These keys have special meaning in NEWSCRIPT, as we shall see later, in the section on using EDIT. When an ordinary key, such as "D", is used for a special purpose, rather than as a letter, it'll be shown the same way: <D>. We'll also use this kind of bracket to show where you must choose something from a list.

text - the material you are writing and want printed. We also may call it "data"

text file - a file that contains text or words, as in a written document. This, of course, is what you'll be creating and printing with NEWSCRIPT

variable - something you replace with something else. We show variables as general symbols, since we can't know exactly what you will need to type into the computer.

Chapter 4 - Tutorial: Learning the Editor

The next two chapters will show you how to write a simple business letter with NEWSCRIPT. Actually, you'll learn much more than this, since you'll be learning how to use the editor (in this chapter), and how to indicate the formatting to be used when printing occurs (in the next chapter).

You should use an installed backup copy of NEWSCRIPT from now on, never the original. On the disk is a letter called "**EDIT1/EX**". We'll use it for most of this tutorial, both to save you the typing time and also to make sure we're all talking about the same thing. To break up the monotony, some of the examples won't use this particular letter, but the principles we show will always apply.

You can use this example as a starting-point for your own letters: just replace the names, date, and text with what you need, and go from there. Some NEWSCRIPT users keep an outline version of this letter for just that purpose.

GENERAL APPROACH TO USING NEWSCRIPT

1. get NEWSCRIPT running;
2. select the EDIT function;
3. give the name of the file you want to create or change;
4. type in your text, intermixed with format control words;
5. save the text to disk;
6. select the SCRIPT function;
7. select any special options needed;
8. allow printing (or video checkout) to occur;
9. if changes are needed, re-select the EDIT function;
10. repeat the cycle until the printed result is satisfactory, or until you want to stop for a while;
11. exit from NEWSCRIPT or start working on another document.

STARTING NEWSCRIPT

Let's start at the very beginning...

1. Power on the computer, if it's off
2. Insert the installed backup of NEWSCRIPT in drive 0
3. Press <RESET>
4. If you didn't set "AUTO NS", type: NS <ENTER>
5. A copyright notice should appear and BASIC should start
6. Next, the "Primary Options Menu" should appear
7. Press "1" for EDIT (no quotes, no <ENTER>).

The Primary Menu is shown below; it will be explained later:

```
        (*) (*)  N E W S C R I P T   7.0  (c) 1982, PROSOFT (*)(*)
            Box 839, No. Hollywood, Ca. 91603   (213) 764-3131

 _|    (printer type)      | 06/01/82 |  SERIAL #:    12345    |_
_|_____________________________________________________________|_
|        PRIMARY OPTIONS MENU, PLEASE CHOOSE BY NUMBER? _       |
|______________________________________________________________|
|  1. EDIT (CREATE OR CHANGE DOCUMENT)   6. DISPLAY DIRECTORY  |
|  2. SCRIPT (PRINT A DOCUMENT)          7. (unassigned)       |
|  3. MICROPROOF / EW (CHECK SPELLING)   8. EXIT TO DOS        |
|  4. GEAP (GRAPHICS EDITOR)             9. CUSTOMIZE/INSTALL  |
|  5. PRINT MAILING LABELS               0. EXIT TO BASIC      |
|______________________________________________________________|
                CURRENT DOCUMENT IS: (unassigned)
```

Starting The Editor

EDIT is used to enter and revise text, so most of your time with NEWSCRIPT will be spent using the editor. The steps we've covered so far will bring the computer to the point where you can start editing whatever it is you want to write or change. It will take a few seconds to bring EDIT into computer memory. Once it takes control, it will display its identification. You may also notice another brief use of the disk drive, then...

1. This prompting message should be displayed:

   ENTER I.D. OF FILE TO BE EDITED?

2. If you had been editing or scripting before, it will also say:

   (DEFAULT IS 'fileid'):

3. Answer as follows: edit1/ex <ENTER>
   (Note: With most operating systems, fileid's may be entered in lower or upper case. With TRSDOS 2.7DD, upper-case is not the same as lower-case, so you'll have to decide which case to use, and then stick with it. "EDIT1/EX" should be typed in UPPER-CASE, since that's how we created it.)
4. A counter will show that the file is being read.
5. Once the file has been read, the screen will look like this (the numbers after "E=>" may vary):

```
E=> * LINES USED= 35 LINES LEFT=365 CHARS LEFT=12345 WARN= 15
___ *TOF:..10:....:..20:....:..30:....:..40:....:..50:....:..60:
___ .ce
___ P R O S O F T
___ .sk 2
___ .in 40
___ June 1, 1982
___ .in 0
___ .sk 2
___ .fo off
___ Mrs. Joanne Riley
___ 55 South Hope Street
___ Los Angeles, Ca 90001
___ .sk
___ Dear Mrs. Riley:
___ .fo on
```

The EDIT screen (or "window") is divided into three parts:

1. The top line, which starts with "E=>", is called the "command line". When you need to give "primary commands" to EDIT, you must enter them here. When EDIT needs to display a message for you, it normally displays it on the command line, preceded by a star ("*"). Messages disappear as soon as you begin to type, but should be read first.

2. Down the left-hand side are rows of dashes (Model I) or a line 1/2 inch wide (Model III). This is a secondary command area called the "Line Manipulation Area", abbreviated "LIMA". We won't say much more about it for a while.

3. The rest of the screen is the "data" or "text" area. This is where you'll be doing most of your typing. Each line in this area can show up to 60 characters at a time. By moving the window to the right, up to 240 characters can be placed on each line. However, since NEWSCRIPT handles text formatting for you, you usually won't need to move the window sideways: consecutive lines of text will be treated as a continuous stream of information, and as much as possible will be fitted to each printed line, even if you had only one word per screen line. (If you wanted your material to print "as-is", there's a way to do that, too.)

The first line of text in the data area is called "the current line." Whatever line of text happens to be in that position will be used as a starting-point for many EDIT commands.

When the displayed text is taken from the very beginning of the file, an extra line is displayed. This is the line beginning "*TOF*..10:....", and is called a "GRID" or "column-marker." Its function is to assist you in developing tables, and it will probably disappear soon. There's a way to keep it on the bottom of the screen if you need it, but it isn't needed for this letter, so we'll make no special effort to keep or get rid of it. "TOF" is an abbreviation for "Top Of File". There's also a grid at the end of the file, and its abbreviation is "EOF" (End Of File).

The data area contains two distinct kinds of material, as you have undoubtedly noticed. Some lines contain normal words and sentences (these will be printed as part of the letter), while others begin with periods and contain codes. Those codes are the "control words" we mentioned at the start of the book. They tell NEWSCRIPT about the special formats you want used. For example, that first one, ".CE", stands for "center", and will cause the next line of regular text ("P R O S O F T"), to be centered when printed.

While we're on the subject... most control words are either the first two letters of an English word, or else "mnemonics" for a pair of English words. So, ".SK" means "skip a line", while ".PP" means "start new paragraph", referring to the convention most people use to mark a new paragraph. Once you begin to catch onto the way NEWSCRIPT uses English words for EDIT commands and SCRIPT control words, you'll see why this approach to word processing is so easy to remember: you already know the words and just have to find out how NEWSCRIPT uses them.

MOVING THE CURSOR

The cursor is that small blinking square at the beginning of the command line, right after the "E=>:". It shows you where you'll be typing. Right now, its blinking on top of the star that marks a message. We'll ignore the message and concentrate on the cursor for now.

The cursor can be moved by typing something or by pressing any of the four arrows. If the cursor is in certain extreme places on the screen or in the file, then it won't move further. Please experiment with it now. Also notice that if an arrow (or any other key) is held down, it begins to repeat automatically. If you hold the right arrow down, the cursor will move quickly across the command line. When it reaches the end, it'll just wrap around to the start of the next line. Conversely, holding down the left arrow eventually wraps the cursor back up to the end of the previous line. Moving the cursor through use of the arrows doesn't change any text; only the character keys and the space bar do that.

There are some limits to cursor movement: normally, it won't go into the LIMA, nor will it go leftward over the "E=>", so once the cursor is at the start of the command line, the left arrow stops working. On the other hand, if you hold the right arrow long enough, it'll reach lower right corner of the screen and wrap onto the next line, which previously was not on the screen. When this happens, everything else shifts up a line, and the topmost line of text disappears.

Try pressing <DOWN ARROW>, and see how the cursor moves to the bottom of the screen. If you continue to press it, the text will begin to move up the screen. This is called "scrolling". If you switch to the <UP ARROW>, the cursor will move to the top of the screen, and then the text will start moving down.

If you press an arrow while <SHIFT> is also pressed, something different happens. And if you press <CLEAR> and then press an arrow, yet another set of controls is engaged. The list below summarizes all the uses of the arrows, and we invite you to experiment with each of them as you read:

No <SHIFT>, no <CLEAR>

<LEFT ARROW> moves cursor to left, one character at a time. Wraps to right end of previous line when the left edge of data area is reached. Will not go back past start of command line.

<RIGHT ARROW> moves cursor to right, one character at a time. Wraps to left end of next line when right edge of data area is reached. Will cause scrolling when it reaches lower right corner of screen.

<UP ARROW> moves cursor straight up one line. Causes scrolling when cursor is on command line, unless screen is at top of file.

<DOWN ARROW> moves cursor straight down one line. Causes scrolling when cursor is on bottom line of screen. If cursor is at end of file, a new, blank line is created beneath it.

<SHIFT>, no <CLEAR>

<LEFT ARROW> moves cursor to extreme left side of data area if not already in that position. If cursor is at extreme left, cursor is moved into the LIMA. (This is the only way to move cursor into LIMA.) If cursor is at extreme left of LIMA, it is moved to extreme left of previous data line.

<RIGHT ARROW> moves cursor just past last data character on the line. If already there, moves cursor to start of next line. If line is 60 characters wide, cursor cannot be positioned to the right of text, and moves to next line. If cursor is in LIMA, it is moved to left side of data area.

<UP ARROW> moves cursor to start of command line, regardless of where it was before. This usually is the quickest way to get there.

<DOWN ARROW> if cursor is in data area, it moves to left side of next line. If in the LIMA, cursor moves into the data area. If on command line, it has no effect.

<CLEAR>, no <SHIFT>

<LEFT ARROW> moves screen horizontally to the left, (or text to the right), usually 20 positions at a time. Will not move to the left of position one.

<RIGHT ARROW> moves screen horizontally to the right, (or text to the left), usually 20 positions at a time. Will not move to the right of position 240.

<UP ARROW> places a left bracket into the text. On the Model I, this appears as an up arrow on the screen. This is the easiest way to get this character on the screen. Incidentally, to get a right bracket, press <SHIFT<CLEAR>, then press the left arrow by itself. On the Model I, it appears on the screen as a left arrow.

<DOWN ARROW> unassigned, has no effect.

ENTERING AND CORRECTING TEXT

Text is entered in the data area. You cannot type on a GRID line, but can type anyplace else within this area. Even though there's already some text on the screen, why not move the cursor someplace into that text and type a few characters? As you do so, what you type will appear, and if the cursor was positioned on previously existing text, what you type will replace (overlays) what used to be there. This is one of the ways of correcting minor errors: just place the cursor over the error and retype it. Much better than an eraser or white paint, isn't it?

As you type, you'll see a "greater than" symbol (">") appear between the LIMA and the data area. This is called the "changed line signal", and indicates that a line has been changed but not yet stored. Text is not stored into memory until one of the following things happens:

1. The changed line scrolls up or down off the screen;
2. The screen is scrolled left or right;
3. You press <SHIFT><ENTER> in the data area;
4. The cursor is on the command line or in the LIMA, and you press <ENTER> or <SHIFT><ENTER>;
5. Saving of text to disk occurs.

Upper and Lower Case

NEWSCRIPT begins in lower-case. There are two ways to enter text in upper-case:

* hold down <SHIFT> when typing, just as on a typewriter
* hold down <SHIFT> and press "Zero" to "lock" upper-case

If you "lock" upper-case, you can "unlock" it by pressing <SHIFT><ZERO> again. This method is standard for most TRS-80 Operating Systems. If you're using TRSDOS 2.7DD, be aware that, unlike any other TRSDOS, it doesn't convert lower-case fileid's to upper-case. If you need to "RUN" a program, you'll have to type it's name in UPPER-CASE (unless it has a lower-case name).

Our First Command

Since we don't want to keep any of the changes you've just made while experimenting, let's introduce our first Primary Command (in the future, we'll just call these "commands"). These go on the Command Line and must be followed by <ENTER>. All commands can be abbreviated to either one or two letters, and we will show the minimum abbreviation in capital letters as we introduce each of them. It's ok to type the full word, but only the capitalized part is needed. If you type only one letter when two is the minimum, then either some other command will be used, or an error message will appear.

Whoops

"Whoops" is the command, not an indication we just said something wrong in this narrative. It's used when you want to throw away all the changes still on the screen, and restore the screen to the way it was before. Only lines still marked with the "change signal" can be restored; once they've been stored as explained above, it's too late. Of course, if you haven't saved the text onto disk yet, there's a way to terminate editing and start all over. Or, you can retype things correctly. However, WHOOPS often is sufficient. It

can be abbreviated to its first letter, "W".

Primary commands like "WHOOPS" must be entered on the command line at the top of the screen. Move the cursor there now, type in "W", and press <ENTER>. An exact sequence for doing this is:

1. press <SHIFT><UP ARROW>;
2. press the letter, "W" (lower-case ok, no quotes);
3. press the <ENTER> key.

When you do this, the screen will be restored instantly.

DELETING AND INSERTING CHARACTERS AND WORDS

You've already seen how very minor errors can be corrected by positioning the cursor on the error and overstriking.. Now, we'll show you how to delete (remove) characters and words, or insert them. The keys to be pressed are shown in brackets like this: <D>.

1. Use the arrows to position the cursor on the first (or only) character/word to be deleted, or on the character ahead of which text is to be inserted.

2. Press <CLEAR>. The shape of the cursor will change to a large, square "C". Once it does so, which is instantly, you can release <CLEAR> or keep pressing it; it makes no difference. (by the way, if you have an LNW or "Pencil" control key, you can use it instead of <CLEAR>.)

3. To delete the character at the cursor, press <D>. If you want to delete two or three characters, continue holding down <CLEAR> and press <D> repeatedly. If you want to delete several characters at a time, hold both of them down until auto-repeat kicks in, and text will be gobbled up at about 25 characters per second. Since that's very fast, you'll want to keep your eye on what's happening to avoid deleting too much. This kind of deletion won't affect the next line on the screen, nor will it affect any text that is off the screen to the right of the window you can see. As soon as you release <D>, deletion stops. If the cursor is still a big "C", just hit <CLEAR> again to turn off control mode.

4. To delete the word, or remainder of the word, at the cursor, press <W> (while, or after, pressing <CLEAR>). This is a kind of high-powered <D>, and the rest of the explanation for <D> applies.

5. To insert one or more characters, press <I> (in conjunction with <CLEAR>). The cursor shape will become tall and thin (like a capital "I"), indicating you are in "insert" mode.

   <I> is different from <D> or <W>, because they stay in effect only as long as you keep pleasing those control functions. "Insert" mode stays in effect until you turn it off, or until a command is issued.

   While in "insert" mode, what you type appears at the cursor, and everything on the line to the right of the cursor moves further to the right. If the text being pushed ahead of the cursor reaches the right edge of the screen, it'll be moved to a newly-created next line. If the cursor reaches the right edge, it'll also move to a newly-created next line. As a result, everything will stay in order on the screen, but you may wind up with a couple of short lines. This makes no difference whatsoever at print time, provided that formatting is turned on. (Formatting is something we will talk about shortly.)

   When you want to get out of "insert" mode, just press <CLEAR><I> again, and the cursor will revert to its small, square "oh" shape (for "overlay" mode).

If you want to practice a bit of this with the sample letter, this is a good time to do so. The examples below use the first few lines of "EDIT1/EX". The black square or rectangle represents the cursor, and since we can't show it blinking, please remember that the character at that position will be seen on the monitor along with the cursor.

Example: Delete the letter "S" in line 2

```
___ .ce      ■
___ P R O S O F T
___ .sk 2
```

Action: move cursor onto the "S": down once, then left 3 times
Result:

```
___ .ce
___ P R O ■ O F T
___ .sk 2
```

Action: press <CLEAR><D>
Result:

```
___ .ce
___ P R O ■O F T
___ .sk 2
```

Example: Insert the "S" back in place

Before:

```
___ .ce
___ P R O ■O F T
___ .sk 2
```

Action: press <CLEAR><I>
Result:

```
___ .ce
___ P R O ▮O F T
___ .sk 2
```

Action: type the additional text: S
Result:

```
___ .ce
___ P R O S▮O F T
___ .sk 2
```

Action: exit "insert" mode: <CLEAR><I>
Result:

```
___ .ce
___ P R O S■O F T
___ .sk 2
```


BLANK LINES

Depending on what you want to do, blank lines may be necessary, convenient, or a nuisance. The following information is likely to be useful in dealing with them:

1. Blank lines are created automatically at the end of the text file as the screen scrolls along while you type or hold the down arrow, right arrow, etc., when at the end of the file.
2. Blank lines are created automatically to take up the overflow when in "insert mode".
3. Blank lines are discarded automatically when text is saved to disk. Normally, this is what you'll want NEWSCRIPT to do, since you or it may create more blank lines than you really need. However, if you have placed blank lines in a file on purpose, type this on the command line before saving the file (it will stay in effect only while you remain in EDIT; if you edit the file again later on, you will have to re-issue the command at that time):

   DBLANKS OFF <ENTER>

4. If you want one blank line to appear someplace on the screen, just place the cursor at the start of the line below that point, and press <SHIFT><ENTER>.
5. If you want a whole bunch of blank lines, there are several ways to do it. One way is to move the cursor into the LIMA by pressing <SHIFT><LEFT ARROW>, typing "I" (for "insert") and the number of lines you want, then pressing <SHIFT><ENTER>.
6. If extra blank lines are on the screen, you can ignore them, since they won't be saved, as explained earlier. If you just can't stand them until then, move the cursor onto those lines, then into the LIMA, type "D" (for "delete"), the number of lines to be deleted, and press <SHIFT><ENTER>.
7. If you want blank lines between printed paragraphs, you'll need to use some of SCRIPT's control words, and we'll cover them later on. One such control word is ".SK", which stands for "skip." Another is ".PP" (paragraph).

MOVING THE SCREEN WITH CONTROL KEYS AND COMMANDS

Once you start writing text that takes more than 15 lines on the screen, you'll want to have some easy ways of moving quickly from one place to another. To do that, you'll have to make EDIT move the text, or move the viewing window, and that's the subject of the next several paragraphs.

Let's begin by getting our sense of direction in order:

TOP	- the first line of the file
END or BOTTOM	- the last line of the file
UP	- move towards the top, a line at a time
NEXT or DOWN	- move towards the bottom, a line at a time
FORWARD	- move towards the bottom, a screen at a time
BACK	- move towards the top, a screen at a time
RIGHT	- move to the right (text moves left)
LEFT	- move to the left (text moves right)

By using the appropriate commands, you can move around in small or large steps through your text. Each of the eight "directions" listed above has two kinds of commands associated with it:

<CLEAR> key commands are issued by pressing <CLEAR> and then one other key. You've already learned several of these for character deletion and insertion. All <CLEAR> commands are obeyed immediately, so <ENTER> shouldn't be pressed. The keys to use are shown in brackets below, but you should not type in the brackets, only the keys:

<T> - top of file
<E> - end of file
<U> - move screen up one line (text moves down)
<N> - move screen down to next line (text moves up)
(we couldn't use <D>, since it stands for "delete")
<F> - moves screen forward 15 lines (one screen down)
<B> - moves screen backward 15 lines (one screen up)
<RIGHT ARROW> - moves screen 20 columns to the right
<LEFT ARROW> - moves screen 20 columns to the left

Primary, or Global commands are issued by typing them on the command line and then pressing <ENTER>. This takes longer, but these commands can do more than the <CLEAR> commands. For the most part, they use the same letters as the <CLEAR> commands, but we don't show them in brackets since they aren't control keys.

When the first <u>two</u> letters are capitalized in this list, it means you have to type at least those first two letters for the command to be recognized. If only the first letter is capitalized, it is sufficient by itself.

When the letter 'n' appears in brackets, it means that, if you put a number after the command, the screen will move that many units. The 'n' stands for a number, and the letter 'n' itself is <u>not</u> typed. If you omit the number, the screen will move one unit. If you use a number, don't type the brackets; we use them here only to mark what you must type.

Top - move screen to top of file
BOttom - move screen to bottom of file
Up [n] - move screen 'n' lines towards top of file
DOwn [n] - move screen 'n' lines towards bottom of file
Next [n] - synonym for "DOwn"
Forward [n] - move screen 'n' times 15 lines towards bottom
Back [n] - move screen 'n' times 15 lines towards top

```
**************************************************************************
*                                                                        *
*     Do you see how we've chosen English-language words for the         *
*     commands? Any of several words could have been used to stand       *
*     for these functions, so all you have to do is learn which ones     *
*     we've chosen for our little command language. By remembering       *
*     that NEWSCRIPT uses mnemonics, you'll find that you already        *
*     know our commands, since they're part of our ordinary language.    *
*     And, since the <CLEAR> keys work the same way, you don't need      *
*     to stick labels on the keys: they're already labelled right on     *
*     top!                                                               *
*                                                                        *
**************************************************************************
```


Please try each of those control keys in turn to see what happens. Pretty soon, they'll become second-nature, and you won't have to think about how to move around, but will just be pressing the right keys as easily as you move your fingers.

DISK FILE COMMANDS - SAVING TEXT TO DISK

EDIT offers several commands for moving text between disk and memory. We'll only cover four of them now; the others are very useful, but aren't appropriate for an introductory tutorial. Three of the four are Primary Commands, issued on the top line and followed by <ENTER>. The fourth is a <CLEAR> key command, added as a convenience. The primary commands in question are:

SAve [fileid] - writes the in-memory text to disk. By default, the name used when you started EDIT is used. If you wish, you can specify a different name. Doing so will assign that name to the material being written, but won't affect a pre-existing copy under the original name, and won't change the name EDIT remembers from when it started. When "SAVE" is finished, you'll regain control and can continue editing the same text. 'fileid' is shown in brackets to indicate that it's optional. If you just type "SA", the text will be written using the name you gave when you started EDIT. Remember: DON'T TYPE THE BRACKETS.

ENd [fileid] - just like "SAVE", except that, when it finishes, control is passed to an exit menu that lets you select what you want to do next. Therefore, you'll want to use "SAVE" when you just want to make a copy for safety and then continue editing; and you'll use "END" when you're finished.

QUit - this tells the editor to exit without saving anything to disk. It's useful when you've messed things up and just want to start over again, or when you've just been reading something and didn't have any changes or additions worth saving. If you did make any changes, EDIT will tell you so, and ask you to confirm that you really want to quit. Reply "Q" if so, or just press <ENTER> to resume editing. If you take the exit, you'll get the same menu as you would after "END".

The control key function doesn't have to be used only from the command line:

<CLEAR><S> - works just like "SAVE" without a "fileid", but is quicker and easier to use. When the "SAVE" finishes, the cursor will reappear where it was just before you hit the control key. I use it when I've just captured a particularly lucid thought and don't want to risk losing it to a power failure or system problem. A good rule of thumb is to issue "SAVE" when you wouldn't want to have to retype all your changes should something go wrong.

These "file-oriented" commands must be used whenever you want to keep what you've written. If you don't "save" your text to disk, but just press <RESET>, turn the computer off, or anything else along those lines, then what you've written will be permanently lost. To recreate it, you'll have to type it all back in, because NEWSCRIPT doesn't print from internal memory, only from files stored on disk. Of course, if you don't want to save something, then "QUIT" is appropriate.

SEARCHING AND REPLACING TEXT - LOCATE AND CHANGE

One of the advantages of working with a word processor instead of a typewriter is that if you don't like a word or phrase, you can easily change it by replacing it with a different phrase, inserting new text, or deleting the offending characters. Changes to text may be done within a single line by typing over the characters, as we have seen before, and small scale deletion and insertion is easily done with the <CLEAR> control keys. However, these methods are useful mostly for single occurrences of text. What if you want to make changes all the way through your text, every time the phrase is encountered? Or, what if you don't even know where the text lies in a document? EDIT allows you to perform this feat with incredible ease through use of some of its Primary Commands. The ones we'll cover now are "LOCATE" and "CHANGE".

Locate /text/

"Search", "scan", and "locate" are three words with very similar meanings, and various text editors have used all of them as commands for the same function. NEWSCRIPT uses "Locate", which can be abbreviated to its first letter, "L". The purpose of the command is to scan the text until a specified phrase is found.

Actually, you can search for any sequence of characters, not just words or phrases, and the term used for any of these arbitrary sequences is "string". Since a string may contain leading, trailing, or imbedded blanks, it's necessary to mark the exact content that is to be found. We do this by placing a "delimiter" before and after the string. A delimiter is any character (except a blank) that does not occur within the string. A delimiter must precede the string, and may be included or omitted after the end of the string. The same delimiter must occur at the end as at the beginning, if a trailing delimiter is used. For example:

```
L OCATE   hell o there      - valid (delimiters are not needed, if you don't need blank spaces
                                     at the beinging or end of string)
          /hello there/     - valid
          ?X ?              - valid (different delimiter)
          /wherever         - valid (trailing delimiter optional)
          ?wherever/        - may be wrong if '/' isn't part of text
          /X=A/B/           - invalid: delimiter occurs within text
```

The fourth example isn't invalid unless the slash was supposed to be a closing delimiter. Since "?" was used as the starting delimiter, it would have to be used at the end also. As the third example shows, the trailing delimiter can be omitted. It was needed in the second example since there was a trailing blank, which would not have been included in the string without that second delimiter.

Now, this kind of string searching has been used widely for over 15 years, and the books describing the editors usually showed their examples like this:

```
locate /string/
```

Needless to say, most people who learned from these books used the slash "/" as a delimiter. Since they didn't always understand what the book was trying to say, they copied exactly what they saw. (I did this when I started, and I'll bet you're doing it too!) Well, as time went by, and authors of editors looked for better features, they realized that the command word "locate", even if abbreviated to "L", could be omitted if the slash was present, since everybody was typing "L/string/" anyway. So, a convention was adopted to allow this shorthand, and in keeping with tradition and laziness, we recognize both "L" and "/" as meaning "locate". Consequently, all of these commands mean exactly the same thing:


```
locate /apple/
L/apple
/apple
```

Regardless of how you type it, you will be telling EDIT to "locate the next occurrence of /apple/". EDIT obediently will start at the second text line in the data area, and search for the string of text contained within the slash marks. It will keep searching until any of three things happen:

1. a line containing the string is found;
2. the end of the file is reached without finding a match;
3. you press any key to interrupt the search.

The slash (/) is the only character that can be used without the "L" command, so if the "/" is in the string, you'll have to pick a different delimiter and include the command word (or "L").

NOTE: That this is NOT the same as the slash used in the LIMA commands, since it's entered on the command line, not in the LIMA area. (That will be covered later.) Also, the minus sign has a special meaning when used as the delimiter: it stands for "locate not", and tells the editor to search for the first line that does not contain the string.

"Locate" is one of the two most powerful features in most good editors (the other being "change"), so we'll spend some more time discussing and practicing it.

Locate is used to search for the next occurrence of a string of characters. The search begins one line after the current line (the "current line" is the first data line on the screen at the moment, and is either the line below the "E=>" or the line below the "*TOF*" grid if you're at the top of the file), and scans all the text in each line unless you have specified a ZONE (see ZONE, below) command to restrict the search to just a few of the columns on each line. The search continues until the first occurrence of the string is found, or until the end of the file is encountered. If no match is found, this error message will appear on the command line:

```
* * ERROR 8: STRING OR POINT NOT FOUND * *
```

Nothing has been changed, and you can do something else immediately. As soon as you touch any character key, the error message will disappear. If you realize you've made a mistake before the search completes, just press <ENTER> and the search will terminate immediately, giving the above error message.

There are several ways to use "LOCATE":

```
1. L /string/      - search to end of file for "string"
2. L -string-      - search to end of file for "not string"
3. LU /string/     - search up to top of file for "string"
4. LU -string-     - search up to top of file for "not string"
5. L               - search using latest Locate "string"
6. LU              - search upwards using latest "string"
7. /               - search using latest Locate "string"
```


```
E=>
___ This is an example
___ of simple and nonsensical
___ text without another thing
___ in it.


L /an/   <-- sets the cursor at the second line
L?an?    <-- also sets the cursor at the second line
/an      <-- also puts the cursor at the second line
/xx/     <-- fails to find a match (xx is never found)
L-an     <-- sets the cursor at the fourth line
-an-     <-- also sets the cursor at the fourth line
-i-      <-- fails to find a match (i is used in all lines)
```

Any character may serve as a delimiter if "L" or "LOCATE" is specified, but only the slash and dash are recognized as shorthand for "L" or "LU".

As you can see, this command is quite powerful and very useful in editing documents. We will use it later when we go through the process of producing a real document.

Change /string1/string2/ [n [*]] [#]

This probably is the most useful and powerful command in any text editor, and is a lot easier to use than that mess above might indicate. It lets you change something to something else, once or many times, and specify the range over which these changes will be made. Since the editor performs the chore of searching for the string, it is much faster and more accurate than doing it by hand.

In NEWSCRIPT, "CHANGE" is quite versatile, so there are several ways of using it. Consequently, the notation we've had to use in defining it may be a little confusing: the things in square brackets are optional and therefore may be omitted if you don't need them. The brackets themselves never are typed. Here are a few examples:

```
c /Sam/Samuel/                  - change current line only
c.08/14/81.August 14, 1981      - couldn't use '/' as delimiter
c/expres/express/ * *           - change every remaining occurrence
c/ alright/ all right/5         - change all occurrences in next 5 lines
c/0/Zero/**#                    - global change, suppress FLOW
c//    / 10                     - insert 4 blanks at start of 10 lines
```

The command can be abbreviated to its first letter, "c", and must be followed by two strings contained in a delimiter. In the prototype format shown above, the slash, "/" is used as the delimiter, but you can select any character that does not appear in either string.

Either string may be "null", that is to say, omitted; in this case, two delimiters would appear right next to each other. If the first string is null, then the second string will appear at the extreme left of the line(s). If the second string is null, the first string will be deleted from the line(s) in which it occurs.

'n' is a number, and if used will tell EDIT to look for (and replace if found) 'string1' across the next 'n' lines of text, beginning with the current line. The '*' is also optional and if present means that every occurrence within each line should be replaced. If it's missing, then only the first occurrence of the string will be replaced. Of course, if the string doesn't appear in a line, that line won't be changed. Also, if the string is in a line that is outside the range you specify for the scan, it won't be changed. The range is both 'n' lines and the columns defined by the current "ZONE". (We haven't talked about "ZONE" yet; it's the horizontal columns within which LOCATE and CHANGE should occur. If you don't re-define it, everything will be within the active zone.)

Note that the range of "*" is different than the 'n' range: 'n' is vertical, while "*" is horizontal. Also note that the vertical range can be any number of lines, but the horizontal range is either the first occurrence or all occurrences.

The default values of "n" and "*" are both 1, so that only the current line is searched if neither is specified. You may, however, use an asterisk (*) for 'n' to denote the entire remainder of the file (from the top of the screen to the end of the file) as the search range, and a second "*" to specify changing all occurrences of string1 to string2 throughout the file. Like "Locate", "Change" can be stopped in the middle by pressing <ENTER>. However, any completed changes will remain changed, and "Whoops" won't change them back.

The pound sign, "#", is optional. It's used to tell EDIT to suppress the automatic line-splitting that occurs when "Change" creates a line longer than 60 characters. To see what it does, try making something longer in an already wide line, first without "#", and then with it.

Since "Change" is so powerful and important, let's do some examples of its operation. Let's work with the screen shown below:

```
E=>
___ A box contains 3 red hats and 2 black hats. Three
___ men each reach into the box and place the hat on
___ their head without looking at it. The men are arranged
___ in a row so that the first man can see the two men in
___ front of him, the second man can see only the man in
___ front of him, and the last man cannot see anyone.
___ Since each man cannot see his own hat, but only the
___ hat(s) of the people in front of him, they are asked
___ if they know
___ what color hat they have on. The first man says, "I
___ don't know". The second man says, "Neither do I."
___ However, the last man says, "yes, I know!". What color
___ hat did he have on?
```

Aside from being a fun little logic problem, this paragraph can serve as a useful example. Suppose we wanted to change every occurrence of 'last' to 'latter'. The command we would enter on the command line would be:

C/last/latter/ * * <ENTER>

This would tell EDIT to locate every occurrence of the word 'last' and change it to 'latter'. (Remember, you press the <ENTER> key; you don't type in the word.) What would we do if we wanted to change only the first occurrence on each line?

That's right, we would enter the command 'C/last/latter/*'. What would 'C/last/latter/' do? It would look only at the first line, since that is the default value for 'n'. Since the word 'last' does not appear in the first line, the command would cause no changes to be made, and the message "string not found" would be displayed on the command line.

Now, you moved the window down four lines, 'last' would be the top line on the screen, and 'C/last/letter/' would work. Remember: when using "CHANGE", the search starts with the top line on the screen (the "current line"), but when using "LOCATE", the search starts with the second line on the screen. If you stop to think about it, this makes sense, since you frequently will want to issue "LOCATE" several times before making a "CHANGE".

You do have to be a little careful when using this command: suppose that in the first line you spelled 'three' as 'tiree'. Could you use the change command to correct the spelling? Well, yes, but you should NOT enter the command 'C/i/h/' since that will change only the first occurrence of the letter 'i' in the line and no other. This is wrong since you do not want to change the first letter 'i' to 'h' and you did not change the letter desired. The point is that the change command will not know which text characters you mean, so it will change all the ones specified in the command parameters. If you used the global option for the line 'C/i/h/ 1 *' then all the 'i's in the line would be changed to 'h's, also an undesirable state of affairs. If you used the complete global option 'C/i/h/ * *' then ALL 'i's in the entire text file would be changed to 'h's. How can we change only the one we want with the change command?

It is very simple. You must make sure that the change command references a set of characters that are unique to the words you wish to change, and at the same time is present in all the words you wish to correct. In this case, a simple way to do this would be to enter the command, 'C/tiree/three/', but 'C/tir/thr/' or 'C/iree/hree/' would also have worked. You only have to give enough characters in "string1" to make it unique; you don't have to finish words just to be neat.

TABLES AND WIDE LINES

NOTE: This topic can be skipped temporarily.

When using the editor, the video screen acts like an imaginary "window" that can be moved over your text to display various parts of the text. We discussed several ways of moving the screen earlier, but for special circumstances, there are two other commands that can be very useful: VIEW and GRID.

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

The VIEW command moves the text window to the right or left without moving it up or down. This is particularly useful when you have lines longer than 60 characters. The <CLEAR><arrow> keys can shift the screen also, but VIEW can do so in large steps, or can show just a few columns at a time. This makes it more flexible than the control key approach. Please note, however, that even if you define a narrow VIEW window, you still can type in the "undefined" area; narrow views are useful for specialized purposes, but just as in politics, they can lead to problems.

If no parameters are specified with VIEW, then the current starting and ending columns are displayed. The default for V is 1,60, which means that your default viewing screen is limited to columns 1 through 60 inclusive. This can be changed to view a different number of columns as well as view the end of long lines, but can never exceed 60 consecutive columns at a time.

If only 'n1' is given, then the viewing window includes columns 1 through n1, but if n1 is greater than 60, then only the first 60 columns are displayed, and n2 is set to n1+59. If two numbers are given on the command line, then columns n1 through n2 are displayed, although if n2 - n1 is greater than 60, then only 60 columns will be displayed.

You may also shift a relative number of columns. For example, if you are currently viewing columns 1-60 and would like to look at 11-70, all you need to do is enter 'V +10'. To move back, you could enter 'V -10', but of course if you're already at column 1, the command will be ignored. A quick way of returning to default viewing is to enter the '*' parameter, ie. 'V *', and this will return the viewing screen to columns 1-60.

It is important to remember that all changes entered within a certain viewing window affect only that portion of the line on the screen. Inserting or deleting characters will not move the portion of the line not shown on the screen. This approach makes it very easy for you to maintain column alignment when constructing tables.

For example:

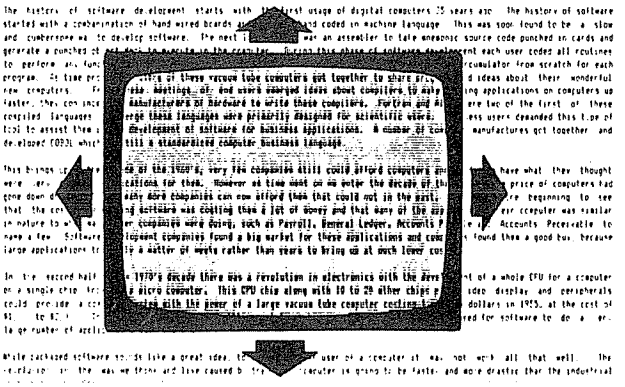
```
....:..10:....:..20:....:..30
This is an example of text
used to illustrate the
utility of the VIEW command
```

If you enter the command on the command line 'V 20' you would get:

```
....:..10:....:..20:
This is an example o
used to illustrate t
utility of the VIEW
```

If you then entered 'V +3', you would see:

```
.:..10:....:..20:...
s is an example of t
d to illustrate the
lity of the VIEW com
```



The above example illustrates that the viewing window retains its size (in number of columns) when moved by the relative commands 'V +n' and 'V -n', and is merely moved to the right or left. In the above example, the width of the window remained the same, but 'V +3' moved that window to the right by three columns. Enter the command on the command line 'V *', and the window returns to normal.

HOrizontal n

This command is a companion to VIEW, but instead of moving the text window, it defines how large the steps should be when <CLEAR><LEFT ARROW> or <CLEAR><RIGHT ARROW> is used. 'n' is initially '20', but you can make it anything from '1' to '240'.

GRid <<ON> <OFF>>

This command displays a column-marking grid on the screen. Its main utility lies in creating tables of data. There are only two states for this command; either it is ON or it is OFF. If you enter the command 'GR', then it is assumed that you want it turned on. If you enter 'GR OFF', then it will be turned off. In order to demonstrate an example of a data table, we must invoke a SCRIPT formatting command, even though we have not yet discussed SCRIPT at all. This command is the FOrmat OFF command. These will be dealt with in much greater detail in a later chapter. All you need to know now is that the '.fo off' command tells SCRIPT not to format the text until it sees a '.fo on' command. This enables us to create column-dependent tables, without having SCRIPT mess up the works by formatting the lines.

> (When you actually start to setup tables, it may also be necessary to tell SCRIPT to not use proportional printing, since columns can't be aligned when the characters are of different widths. The control word to use is: '.bf', and will be covered in the next chapter.)

This example will work fine on the screen, but it won't turn out right on the printer unless you turn formatting off. At any rate, suppose you have laboratory data for a medical experiment.You might want to set up the data table with the use of GRID:

```
E=> ____________________________________________________
___ .fo off;.bf12
___   group no.      10 ug     20 ug     30 ug
___   ------------------------------------------
___      I             15        45        60
___      II            30        37        41
___      III            2         2         1
___   ------------------------------------------
___ .fo on
___ ....:..10:....:..20:....:..30:....:..40:....:..50:
```

The GRID command, as demonstrated above, is merely a convenience for creating column-dependent tabular data. This is all there is to creating data tables! You don't have to use the GRID, but it certainly makes it easier to line things up.

As you've probably noticed already, there usually is more than one way to do something in NEWSCRIPT. That applies to tables, too: later on, you'll see that "tab" positions can be defined to help print tables.

STATUS COMMANDS

The last group of commands dealt with in this chapter don't change text; they just give you information about what's going on with whatever you're editing. These commands give the file's name, length, amount of free space, and the length of any given line. We have included the HArdcopy and HElp commands in this section as well. The "DIRectory" command belongs here too, but we covered it earlier.

Some of these commands can take optional parameters, as shown in angular brackets. If a parameter is given, it changes the setting; if omitted, the current setting is shown. The brackets never are entered.

NAme < string >

The NAME command is used to display the file's current name or to change it to another name. If no parameter is given with the command, then the file's current name will be displayed on the command line. If 'string' is given, then the name of the file will be changed to 'string'. For example, 'NA' will return the file's name, such as 'EDIT1/EX', whereas 'NA EXAM1/NS' will update the name to be 'EXAM1/NS'. If you change the name, it doesn't affect anything on disk. However, the next time you save the file, it'll be saved under the new name. This means the old one, if any, will still be on disk in its original form.

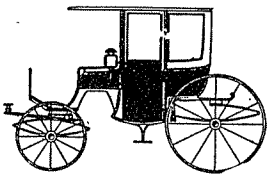
FRee < n >

This command returns the number of lines used, the (estimated) number of lines left, the number of actual characters left, and the number of lines below which EDIT will start warning you that you're running out of room. The second and third values give you an estimate of the maximum number of lines or data characters you may enter into the current file. The actual limit is either 400 lines or the number of characters available in memory, whichever comes first. If you're using TDOS, this limit is about 13,000 to 13,500 characters.

The last value is the number of lines left when a warning message will be displayed on the command line. This is what the parameter on the command line will affect. The default is 15 lines, which means that when there is room for only 15 more lines, or 15 times 40 = 600 characters (a guestimate made by EDIT), then you will be warned. This limit should, in practice, rarely be reached. We highly recommend keeping your data files relatively short, 150-300 lines, for a variety of reasons, as discussed earlier. If you change the warn limit, the warning message will appear if and when the new limit is reached.

When you reach the warning limit, a prominent message appears on the command line. We urge you to stop at that point, move the cursor to the command line, erase the message by pressing <CLEAR><SPACE>, and then saving your file. Then, you can decide whether to split it in two, or start another file that will be appended (chained) to it.

If you ignore the warning, then EDIT eventually will run out of room. When it does so, the message will tell you that there is no more room, and that you should save the text immediately. Any additional text you attempt to enter when there is no more room will be discarded, so you might as well save what you've got, and then split the file into two smaller ones. This can be done either with EDIT's "PUT" command or NEWSCRIPT's "FITLINE" utility program. Both of these are explained elsewhere in this book.



AUto <n>

If "AUTO" is typed with no parameter, then the current AUTOSAVE limit (the number of changes after which EDIT automatically writes your text to disk for safety) is displayed, along with the number of changes you've made since the last "SAVE" was issued, or since you started editing this time.

If "AUTO" is followed by a number "n", then the AUTOSAVE limit will be changed to that new value. "n" must be greater than zero and less than 32768.

LEngth

This command has no parameters. It displays the total number of characters in the current line (the first data line on the screen).

HElp

HELP has no parameters. It clears the screen and displays a list of EDIT and SCRIPT commands. To return to your document, just press <ENTER>. Another way to obtain the HELP screen is by pressing <CLEAR><H>.

HArdcopy <n> or < * < * > >

This command lists some or all of your document onto the printer. By default, fifteen lines (the current screen, beginning with the current line) are printed. If 'n' is specified, then 'n' lines will be printed, beginning with the current line. If a single '*' is specified, then all lines from the current line through the end of the file will be printed. If two '*'s are specified, then your entire document, from the top through the end of the file, will be printed. Note that HA is <u>not</u> a substitute for SCRIPT, since SCRIPT does a whole lot more than merely print out your document - it also formats, centers, determines proportional spacing, etc. HA only prints out the document as you see it on your screen, with the exception that the entire line, rather than just the current viewing window, is displayed, even if the line is longer than the viewing window's width.

Examples are:

```
HA       <=  prints out 15 lines beginning with the
             current line
HA 10    <=  prints out 10 lines beginning with the
             current line
HA *     <=  prints all the lines from the current
             line to the end of the file
HA * *   <=  prints all the lines in the file
```

<u>Summary</u>

This concludes the introductory tutorial for the editor (whew!. With a little practice, you'll become proficient very quickly. However, in order to control the way your documents will look when printed, you also need to learn a little about SCRIPT, the text formatter. So, the next part of the tutorial will teach you enough to create and print some simple, yet splendid-looking results.

Chapter 5 - Document Formatting: SCRIPT Tutorial

We use EDIT to get information into the computer, but it's SCRIPT that gets it out and onto paper. This chapter will teach you enough about text formatting so that you can begin immediate production of simple, but attractive and professional-looking, documents. The remainder of formatting commands will be dealt with later.

INTRODUCTION TO TEXT FORMATTING

Basically, text formatting is the procedure by which a bunch of characters is transformed into a beautifully designed document with page numbering, margins, spacings, and emphasis. Coming up with the words and typing them in is only half the job!

NEWSCRIPT takes an approach to text formatting that is flexible and straightforward, both to learn and to use. This is done through text formatting control codes that you include in your document when you EDIT it. SCRIPT, the text formatter, then reads in your document, and distinguishes between the codes that control formatting and the body of the document itself. It uses these codes to control the formatting of the text.

An obvious question to be asked is, "If the control codes are imbedded in the document, how can SCRIPT tell if they are control codes or text to be printed?". This is done in two ways: "control words" and "escape sequences."

In general, "control words" tell NEWSCRIPT where to put things on each printed page, while "escape sequences" tell the printer to invoke various special features such as double-width printing. There are a few exceptions to this, whereby certain control words cause SCRIPT to send printer escape sequences (for 10, 12, 16, or proportional pitch), and certain escape sequences are simulated by SCRIPT (underlining on a typewriter). In this chapter, we'll be concerned primarily with control words, and will mention only two escape sequences. When you're ready for the rest, see SECTION IV of this book.

All NEWSCRIPT control words begin with a period, and are two letters (characters) in length. They are placed in a document on lines of their own, with the periods at the extreme left side of the line. In English and other European languages, periods don't begin sentences, so it's pretty safe to assume that a line beginning with a period isn't normal text. (If you want a line full of periods for some reason, SCRIPT lets you do it without conflict.)

When you specify more than one control word at a time, you can either place each one on its own line, or can place several of them one after another on a shared line. If you do this, separate each of them by a semi-colon:

```
.sk 2;.ce
```

Control words and their operands may be entered in any combination of UPPER and lower-case.

Control Breaks

Normally, NEWSCRIPT treats all your text as a continuous stream of characters, and fits as much as possible on each print line before going on to the next one. As you will learn shortly, there are ways of preventing this when you want to start new paragraphs and pages, or when you want to create a table. Interrupting the flow of text is called a "control break", and most control words cause such a break to occur.

When a control word is encountered, all the text up to that point is processed, formatted, and printed. Then, the actions dictated by the control word are performed, and processing of additional text resumes, taking into account the new instructions contained in the control word. There may be several control words in a row, but that makes no difference.

Exceptions

A few control words don't cause breaks, due to the special nature of their functions. For instance, a comment ".CM" can be placed on any control word line, as can an index entry, ".IX", without interrupting the smooth flow of text.

A few control words must be the last ones on a control word line, again due to their special functions. Comments, index entries, and titles are examples of these, and if you think about it for a moment, you can see why they must be last: they can contain any information at all, including what might look like another control word! We'll note these as we go.

Use of Mnemonics

We've selected the mnemonics of these control words so that they stand for their English-language functions. ".PP", for example, means "begin a new paragraph", and ".FO OFF" means "turn the formatting of text off." The text of your document is placed between the lines of control words, or in a few special cases, right after the control words, on the same lines, depending on the control words used. These control words allow a great deal of flexibility in formatting text, yet, because they are abbreviations of English, they are easily learned and easy to use. For example, the sample letter, "EDIT1/EX", looks like this after being typed in with EDIT:

UNFORMATTED SAMPLE LETTER, AS ENTERED INTO EDIT

```
.ce
P R O S O F T
.sk 2
.in 40
June 1, 1982
.in 0
.sk 2
.fo off
Mrs. Joanne Riley
55 South Hope Street
Los Angeles, Ca 90001
.sk
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 $1.65 each.
Please use UPS and the above address as the destination.
.pp
I'd like to take this opportunity to express my
appreciation for the special help your !$Mr. Smith!% gave
us last week: we needed quick delivery of diskettes and
ribbons, and he took the trouble to drop them off here on
his way home. Now, !$that's!% !(service!!)
.sk 2
.fo off
.in 40
Sincerely,
.sk 4
Chuck Tesler
.in 0;.fo on;.sk 10
!$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 $2.75 a diskette, just like you are.  The
numbers were just to get your attention.
```


SAMPLE LETTER AS FORMATTED BY SCRIPT

When we let SCRIPT loose on this text, the printed result will look something like this (on a proportional printer; on other printers, it'll be a bit different, but not much):

P R O S O F T

June 1, 1982

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.

I'd like to take this opportunity to express my appreciation for the special help your Mr. Smith gave us last week: we needed quick delivery of diskettes and ribbons, and he took the trouble to drop them off here on his way home. Now, that's s e r v i c e !

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 $2.75 a diskette, just like you are. The numbers were just to get your attention.

EXPLANATION OF SAMPLE LETTER

I'll bet you'd like some explanation of what those control words mean, wouldn't you? Actually, they're pretty obvious, once explained:

```
.ce       - center the next line of text
.sk 2     - skip down two lines (leave two blank lines)
.in 40    - indent 4.0 inches (we measure in tenths of an inch)
.in 0     - indent 0 inches (move back to permanent left margin)
.sk 2     - skip down two more lines
.fo off   - turn formatting off (print the salutation as-is)
.sk       - skip down one line (one is the default)
.fo on    - turn formatting back on (to print the body of the letter)
.pp       - start a new paragraph with appropriate indentation, etc.
```

You may have enough information at this point to start writing letters and straightforward reports on your own. But, just in case we're still holding your attention, we'll briefly cover several of the most commonly-needed control words, and give you a better idea of how to design your pages using NEWSCRIPT.

Because formatting requirements vary so much, NEWSCRIPT has over 50 control words and a dozen escape sequences. Most of these are intended for special purposes, and you will rarely, if ever, use them all. In this introductory chapter, we won't even try to learn too many of the SCRIPT commands. We will concern ourselves with learning just enough to produce a document that is not too out-of-the-ordinary. A list of these simple commands follows, and since they all have defaults, you probably won't have to use more than half-a-dozen of them in most letters:

MOST COMMON SCRIPT CONTROL WORDS

Margins

```
.TM - top margin
.BM - bottom margin
.LM - left margin
.LL - line length
.IN - indentation
```

Formatting

```
.CE - center line(s)
.FO - full formatting
.JU - justification
.BF - begin font (pitch)
```

Spacing

```
.BR - new line
.SK - skip line(s)
.PP - new paragraph
.PA - new page
.DS - double space
.SS - single space
```

Large Documents

```
.AP - append (chain)
.IM - imbed text here
```

Miscellaneous

```
.CM - any comments you want
```

Most of this list is very easy to remember, because it basically just contains the things we do unconsciously when we write something out by hand. The 18 codes fit neatly into the five categories shown, and in fact if you check the Reference Summary Card (the "cheat sheet"), you'll see only six categories altogether, not counting those escape

sequences we keep saying we will talk about. Since the mnemonics are obvious (for the most part), you already know most of them, and would probably pick the same ones yourself if you were masochistic enough to write a word processor. (No, actually I enjoy doing this; it's only my keeper who tells me I'm nuts.)

MARGINS

NEWSCRIPT formats a page in terms of printed lines going down the page, and tenths of an inch across the page. The vertical units (lines) makes sense because that's how a typewriter works, and it's probably how we visualize what will be printed. But, on the face of it, it would seem to make more sense to have chosen "characters across" than "tenths of an inch". After all, when typing, we usually get 60 or 65 characters on a line, don't we?

Well, it depends. If your typewriter is set for "PICA", which is 10 characters per inch (cpi), this is true; but if your typewriter is set for "ELITE", which is 12 cpi, then you probably get 70-80 characters on a line, and moreover, in the same amount of horizontal space! So, if we made you tell NEWSCRIPT how many characters to print on each line, then you'd have to change your specifications if you ever printed on a different printer, or even at a different pitch.

It would be even worse to specify "characters" when using a proportional pitch, since the number of characters per line varies! In fact, it wouldn't be a workable arrangement, so there's no point even trying to figure it out. Instead, we'll just work in terms of inches, and to avoid having to put decimal points in, we'll use tenths of an inch.

This unit of measurement relates well to PICA, which is 10 cpi, and the kind of typing and computer printing we are most familiar with. So, if you were printing on standard size paper (8-1/2 by 11 inches), and wanted a one-inch margin on all four sides, you would tell SCRIPT:

.TM 6 - six lines at top of page (top margin)
.BM 6 - six lines at bottom of page (bottom margin)
.LM 10 - one inch for the left margin
.LL 65 - allow 6.5 inches across for printing

We didn't define "right margin", but instead chose to use ".LL" for "line length". So, the right margin starts where the sum of the left margin and the line length end. When you want to set up very narrow or very wide columns, you probably will find yourself thinking in terms of "inches wide", not in terms of the size of the right margin, and this approach saves you the trouble of doing a bunch of additions and subtractions..

Just this once, we'll show that this really does produce a one-inch right margin:

8.5 inches = 85 tenths; 85 - 10 - 65 = 10 tenths = 1 inch.

Having gone through this in excruciating detail, it is my pleasure to inform you that, if you want your margins to be one inch on each side, you don't have to specify anything at all: these values are the defaults already built into NEWSCRIPT!

I just told you a small lie. The left margin is actually defined as "5", not "10". This is because most printers are setup to leave about a half-inch left margin as a minimum, so all we have to do is move the margin the other half-inch to make it come out properly on the paper. If your printer doesn't work this way, or if you've moved the tractor/pin feed to

some other position, then you'll need to specify the left margin in each of your documents. Of course, another solution might be to move the tractor back to the half-inch mark, and then forget the whole thing.

The top and bottom margins can contain titles and page numbers. We won't cover this in the tutorial, but will mention that NEWSCRIPT generates page numbers automatically, beginning with "Page 2" (it isn't considered too smooth to put "Page 1" on a letter). This invaluable information is printed on the fourth physical line of each page, over on the right side. Since there are six lines in the top margin, this arrangement means that there will be three blank lines above the page number, and one blank line below the page number but above the start of the body of the text. If you don't like this arrangement, you can easily change it by using titles and the more advanced fine-tuning margin controls, which are described in Section IV:

.HS .HM .FS .FM .PS .TT .BT

The left margin sometimes needs to be indented in various ways. NEWSCRIPT has several control words for this purpose, but we'll only cover one of them here:

Indentation

.IN n or .IN +n or .IN -n

".IN" stands for "indent" (another original selection), and 'n' is a number in tenths of an inch. If the plus or minus sign is given, then the temporary left margin is moved in or out (indented) relative to where it was before. If an absolute number is given (no sign), the text is indented by that amount. If 'n' is omitted entirely, or specified as zero, then indentation is moved back to the permanent left margin. An indent of '10' causes printing to start 1 inch to the right of the left margin, which is column #11 at 10 cpi.

For further information on indentation, refer to these control words in Section IV:

.IN .OF .HI .JU .PP

SPACING BETWEEN LINES, PARAGRAPHS, AND PAGES

We won't attempt a full treatment here of the myriad ways provided in NEWSCRIPT for leaving space between printed lines. We will cover the most common situations:

1. When left to itself, NEWSCRIPT prints as much text as it can on each line, taking extra words from the next line that had been on the screen, or saving excess words for use with the first words of the next line from the screen. This is the normal ".FO ON" mode, and will be described further in a moment. If you want the next line of text to start on a new line on paper, you must indicate a "break" in the flow of text. Most control words provide such a break, but if you don't need to use a control word, but just want to separate a couple of lines, you can use:

.BR

Remember, as a control word, this goes on a line of its own, starting at the left edge of the data area on the screen. "BREAK", which is its full name, has no parameters. Most control words create breaks anyway, so this is needed only when you want to start a new line, but aren't using any other control words at that point.

2. Of a bit more interest is the ability to leave one or more blank lines in the text. One of the ways to do this is:

.SK [n]

'n' is the number of lines to skip. If omitted, one line is skipped. If 'n' is larger than the remaining number of lines on the page, a new page will be started, and the extra spacing lines will be discarded.

A variant of ".SK" is ".SP". They both work alike except for what happens at the top of a page. ".SP" is always effective, and generates the number of blank lines you specify. If a new page is started in the middle of the spacing, this can lead to unwanted blank lines at the top of that next page. ".SK" is conditionally effective, and doesn't generate blank lines at the top of a page. If it starts near the bottom of a page, and a new page is needed, the remaining count is just discarded. Better yet, if ".SK" is used at the top of a page, it is still ignored, and unexpected large gaps are thereby avoided. Of course, if you want spacing at the top of a page, just use ".SP".

Examples: .SK .SK 3 .SK2 .SK25

3. Of more interest yet is the ability to specify where new paragraphs are to begin. The control word to use is:

.PP

This control word can be followed by up to four parameters, none of which we will attempt to cover here. We do suggest you read up on ".PP" in Section IV at your first opportunity. In the meantime, we'll tell you what the defaults are when these mysterious parameters are omitted.

A. a blank line is created above the new paragraph;
B. if fewer than 3 lines are left, a new page is started;
C. the first line of the paragraph is indented 1/2 inch.

4. To start a new page, just use:

.PA <n> <+n> <odd> <even>

If you just specify ".PA" by itself, the current page is ended (and any bottom titles are printed if you specified them) and a new page is started (with appropriate paper movements and printing of top titles if you specified them). The optional parameters give you additional control:

n - number the next page 'n' (any number from 1 to 32767)
+n - number the next page 'n' higher than the current page
odd - make sure the next page bears an odd number
even - make sure the next page is evenly numbered

If 'odd' or 'even' is used, a page number will be skipped if necessary. If '+n' is used, you can hand-insert other pages from a separate printing afterwards.

Examples: .PA .PA 27 .PA ODD .PA +2

5. Finally, you can tell NEWSCRIPT to double-space (or go back to single-spacing) a document by selecting the appropriate one of these two control words:

.DS .SS

Does most of this seem straightforward so far? Using mnemonics as control words sure does help.

FORMATTING: RIGHT JUSTIFICATION AND PITCH

Left to itself, NEWSCRIPT formats all text so as to produce smooth left and right margins. This usually is called "right-justification", but more accurately should be called "left and right justification." Also, if you've selected a printer that NEWSCRIPT supports for proportional printing (except for Diablo and Spinwriter-compatible printers), then the proportional font of that printer is used by default.

If you don't put any control words in your text, your document will be printed as one long paragraph. Since this probably won't be what you want, the "spacing" control words we covered a moment ago can be used to indicate where new paragraphs, pages, etc., should begin. When you need additional kinds of formatting, other control words will have to be used.

The four most important of these are the ones we listed under "FORMATTING":

".CE", ".FO", ".JU", and ".BF".

.CE <n ON OFF>

As you may suspect, this tells NEWSCRIPT to center one or more lines of text. The text doesn't go on the same line as ".CE", but begins on the next non-control word line. The center point is always assumed to be the middle of the current "line length" (".LL"), and is calculated from the left margin. Any indentations currently in effect are ignored for this purpose.

The control word has four forms; the one to use depends on how many lines of text you want centered:

.CE	-	center the next one line of text
.CE n	-	center the next 'n' lines of text
.CE ON	-	center all lines until told to stop by "OFF"
.CE OFF	-	stop centering

In our sample letter, the word "P R O S O F T" is centered by using the simplest form of this control word.

.FO < OFF ON >

This tells SCRIPT whether to treat your text as separate lines to be printed as-is, or as a continuous stream of characters to be fitted as fully as possible onto each print line (this is called "concatenation", which is the connecting of two strings to form a single larger one). The default is "ON" and causes full fitting and right justification (smooth left and right margins) to occur. If formatting is turned "OFF", then text is printed as it appeared on the screen.

If formatting is off and an original line is too long to fit within the currently defined line length, an error message will be given. The solution is to either increase the line length, or else to shorten the original line.

In the sample letter, we turned format off in order to print the name and address on three separate lines, and then turned it back on to let NEWSCRIPT print the body of the letter. We didn't have to turn it off for the date or signature, since other control words caused natural breaks in the flow of text.

Examples: .fo off .fo on

.JU < ON OFF RIGHT ALL >

".FO" controls both justification and determination of whether to use words from several original lines or just from one line at a time (concatenation). Usually, you'll either want your text to be both concatenated and justified, and ".FO" is the appropriate control word to use in those cases. However, if you want the text to be treated as a continuous stream (concatenated) but not right-justified, then you'll have to use one of the forms of ".JU" (which, if course, stands for "justify):

ON	- left and right margins smooth (the default)
OFF	- left margin smooth, right margin ragged (like typing)
RIGHT	- left margin ragged, right margin smooth (special effect)
ALL	- left and right margins smooth even if last line of paragraph

Other than "ON", the most common use of JU is "OFF". If your printer can't, or isn't printing in proportional mode, then right justification is accomplished by generating extra spaces between the words. These wide gaps may be objectionable in appearance, and shout "COMPUTER PRINTOUT" for all the world to see (hear? mixed metaphor there, I guess; sorry about that). The way to avoid that appearance is to use ".JU OFF", rather than ".FO OFF". The result looks very good, especially when you consider the alternative.

Examples: .ju on .ju off .ju right .ju all

.BF < 10 12 16 737 >

When we discussed margins and line length, we explained how changing the pitch affects the number of characters that can be printed per line. The ".BF" control word is used to select the pitch you need, and any one of the four numbers shown above can be used.

Strictly speaking, we are talking about "pitch", or width; but most printers also use different shapes for the different character widths, and their manuals refer to these as "fonts." It was a toss-up as to whether we should have made this control word ".BF" or ".BP", and ".BF" won. Just remember we're talking about the same thing, regardless of which term is used.

The first proportional-spacing printer NEWSCRIPT supported was the Centronics 737. Since its number (737) was way too large to be confused with a monospace pitch (10, 12, 16, etc.), we used the printer number to designate proportional printing. The convention persists today, even though NEWSCRIPT now supports almost a dozen proportional-spacing printers. So, if you have a Diablo 1620, you must specify "737", not "1620", to select proportional printing (you must have NEWSCRIPT'S "Daisy Wheel Proportional Option" for this to work, of course).

You can change the font/pitch from one line to the next, but not within the same line. If your printer can print "double-width" characters (like this), you can switch between single and double width within a print line, which is a form of pitch change, but a very simple, specialized one. However, NEWSCRIPT won't let you use, say, 10 and 16 intermixed on the same line, because we found the calculations too complex and the need infrequent.

If the input text looked like this:

```
.bf 10
This line is printed at 10 cpi !(this is double!)
.bf 12
This line is printed at 12 cpi !(this is double!)
.bf 737
This line is printed in proportional !(this is double!)
.bf 16
This line is printed at 16.5 cpi !(this is double!)
```

the printed text would look like this:

```
This line is printed at 10 cpi this is double
This line is printed at 12 cpi this is double
This line is printed in proportional this is double
This line is printed at 16.5 cpi this is double
```


RESTRICTIONS

1. If your printer doesn't have a particular pitch, NEWSCRIPT can't use what isn't there. This material was printed on a printer that can do all four pitches. An EPSON M-80 doesn't have 12-pitch capability, so ".BF 12" won't work on it. it.
2. Daisy wheel printers can't print double-width; this is a dot-matrix exclusive.
3. If your printer lacks proportional capability, or if it's a Daisywheel or Spinwriter and you don't have our Daisywheel Proportional Option, then "737" won't work. Exception: the Radio Shack Daisy Wheel Printer II is supported in proportional by standard NEWSCRIPT, because it works almost identically to the 737 dot matrix printer.
4. "16" is a catch-all for 16.5, 16.7, 17.1, etc. Only use "16", never the exact numbers with decimal points. If your printer has a 15 cpi capability, specify "16", not "15", since "15" is not a recognized value.

If you want to print 132-character lines on an 8-1/2 inch dot matrix printer, such as the MX-80, 739, or 8510, use condensed pitch and set the line length to 80:

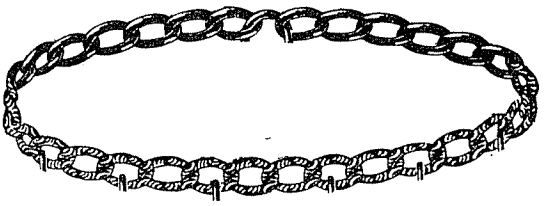
.BF16;.LL80

DO NOT set ".LL 132", since that would tell NEWSCRIPT your printer can print lines as wide as 13.2 inches, which it can't. At 16.5 cpi, you can print 132 characters in 8 inches.

Multiple Columns

Most of this book was printed with ".BF 737". The tables are either printed at 12 cpi, which approximates proportional, or by printing a column at a time and then indenting and reverse feeding the paper. Reverse paper feeding is the only way NEWSCRIPT handles multiple columns, and requires a printer with that mechanical capability. If yours has it, please be aware that the paper "slips" a little when fed in reverse. This can cause temporary loss of horizontal alignment, the severity of which varies from printer to printer. NEWSCRIPT attempts to compensate for this mechanical deficiency, but you may have to engage the "friction feed" to ensure satisfactory results.

Unless you use reverse paper feeds, vertically-aligned columns can be produced only in "monospace" pitches, never in proportional.



CHAINING FILES TOGETHER (LARGE DOCUMENTS)

Two control words fit into this category:

.IM - imbed another file "here" at print time
.AP - append (chain) to the next file in the sequence

These differ in effect. "Imbed" temporarily halts processing of the current file, and then processes a second file. When the processing of that second file is completed, SCRIPT resumes processing of the first file. "Append" is placed at the end of a file, and causes SCRIPT to stop processing the current file and begin processing the second one. Unlike the '.IM' word, ".AP" will not revert back to the original file after the second file is processed, so any text after the appearance of ".AP" will be ignored.

The syntax for these two control words is similar:

```
.im filename/ext:drive
.ap filename/ext:drive
```

The imbed must have a filename specified with it. You must specify the extension of the filename, if one exists. The drive number is optional unless you wish to ensure that the file is loaded from a specific drive and the file occurs on more than one drive. The syntax for "Append" is identical. Some examples are shown below:

```
.im EXAMPLE/TXT
.im letter/ns:2
.ap FILE2/AUT:0
.AP stats
```

These examples are independent of one another. However, if they all occurred in a single actual document, the last one (".AP stats") would never be used, as we explained above.

The Imbed control word is used when you wish to load some document into the middle of another. This enables you to keep the size of the documents small, since many documents may be imbedded into a single one. However, imbeds and appends are not permitted within an imbedded file.

The ".ap" is used to link several document files together at their ends. This "appending" is also known as "chaining". Control does not revert back to the original file, and so is suitable only for chaining, but not for imbedding. On the other hand, you may append as many files as you like. This allows documents of truly unlimited size, since you may even append files that are on separate diskettes!

When a document is so large that its components occupy more than one diskette, it may be necessary for you to switch diskettes from time to time. For that purpose, NEWSCRIPT also has a control word called ".ST" (STOP). Since you probably will not be creating any humongous documents immediately, we will forgo the problem of files scanning off-line diskettes for now, and just ask to you remember that such a facility exists.

Maximum Document Size

A document may consist of one or more files. The files may reside on one or more diskettes. Each file can contain up to about 13,000 characters, which is about 2,000 words. To create a larger document, it's necessary to chain them together with ".AP" or ".IM".

Because files can be chained and diskettes can be switched, there is no maximum document size with NEWSCRIPT. The largest page number is "32767" (about 30 times the size of War and Peace). A Model I diskette holds 85-100,000 characters, and a Model III diskette holds about 178,000 characters. (When you stop to think about, a diskette actually holds less, since the last "granule" of each file will only be half-full on the average). A printed page with no blank lines between paragraphs might hold 3,500 characters (54 lines of 65 characters each), so a normal page probably has about 2,500 characters. (If you're using proportional or 12 cpi pitch, you'll get 20% more per page.)

Based on these very general capacity estimates, you'll probably be able to store about 30 single-spaced pages (75000/2500) on a Model I, or 60 pages (150000/2500) on a Model III. (If you're using proportional pitch, these numbers will be about 25 and 50, respectively.) In practice, the actual ranges probably are 20-35 pages on a Model I, and 35-70 pages on a Model III. If you double-space your manuscripts, you can almost double these estimates.

It's a good idea to not completely fill a diskette, since you'll want to leave some room for revisions and expansion later on. The ".ST" and ".AP" control words come in handy at such times for switching diskettes.

SUGGESTION FOR WORKING WITH LARGE DOCUMENTS

If you're writing a one-to-four page letter, you won't be using ".AP", although you may find ".IM" useful for imbedding your logo, a stock paragraph of terms and conditions, etc. If you're writing a letter or report that is 5-20 pages in length, a series of "appends" is probably the easiest way to put everything together. However, if you wind up using NEWSCRIPT to write something large, such as this book, we recommend you use a combination of "appends" and "imbeds."

A book, large report, or term paper typically consists of a number of chapters or sections. Within each are a number of related topics. If you put each small topic in its own file, and then use a series of "imbeds" within another file to control the sequence of printing of those topics, it'll be very easy for you to move things around on a grand scale should you decide that a particular subject belongs elsewhere. This was done many times in the preparation of the NEWSCRIPT documentation, and saved days of effort.

The files containing the "imbeds" may be though of as master files. Each of them covers a large chapter or section, and ends with an "append" to the next master file. In this way, you can move an imbedded topic around within a section just by moving the one line that identifies it. You can move the topic to another section by deleting the line in one master file and inserting it in another. This is clearly quicker and easier than having to move a block of text from one file to another, and turns the "chore" of re-organizing material into an almost pleasurable activity.

Example

```
    Intro              Tutored            Tutorsc
.cm Section I      .cm EDIT Tutorial  .cm SCRIPT tutorial
.im overview       .im startns        .im textfmt
.im backgrnd       .im cursor         .im sampletr
.im install        .im txtentry       .im ctlwords
.im operate        .im delisrt        .im margins
.ap tutored        .ap tutorsc        .ap section2
```

Three "master files" are shown here. They are portions of the ones we used in constructing this book. Topics were tried in different sequences, and some topics were written several times in an attempt to find the clearest explanation. (How well we succeeded is a judgement you, not we, will have to make.) Rather than discard each attempt, we just stored the old ones and wrote new ones, often drawing on previously-written material to do so. The best combinations wound up here, sometimes in sequences different from the ones originally planned. By keeping each topic in a separate file, it was possible to do this quite easily.

This recommended approach should sound familiar: it mirrors the development of an outline, that crucial first step in the development of any large written document or plan. In fact, because the method kind of forces you to organize and plan your thoughts, it makes the actual development effort easier than it otherwise would be.

There's another section later in this book called "How to write with a Word Processor", and you may want to look at it after becoming more familiar with NEWSCRIPT.

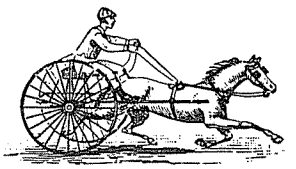
MISCELLANEOUS CONTROL WORDS

It would be tempting to cover several other control words at this point, but you've already learned more than enough to produce some really excellent documents. So, we'll just give you one more for now:

.CM any comments you want to have

This control word is ignored by NEWSCRIPT. It doesn't print, it doesn't cause a control break, it doesn't interrupt the formatting or processing of text. It's only there for you, to let you make notes to yourself for future reference. The "comments" immediately follow the ".CM", and can be anything you want to say. If you need more than one line, you can use more than one ".CM".

Since everything else on the line is treated as a comment, you can't put more control words after this one: they must go on the next line. The only other control word you've learned so far that shares this restriction is ".PP".



ESCAPE SEQUENCES

No, we're not talking about the chase scene in a movie. We're talking about those funny things in your printer manual that tell the printer to stop acting like a dumb typewriter and start doing all the fancy things that made you decide to buy it. Things like double-width, boldface, italics, underlining, sub-scripts, super-scripts, and so forth.

Now, NEWSCRIPT can't make your printer do something it doesn't know how to do: we can't do italics on any dot matrix printers except some Epson's (as of June, 1982, at any rate). But, if your printer has a special feature, chances are that NEWSCRIPT can let you use that feature easily. You don't have to know the codes the printer needs, or how it works; you only need to know that your printer has the feature, and NEWSCRIPT will do the rest. (NOTE: Not all features of all printers are supported.)

What makes this even nicer is that the method is printer-independent: if you specify underlining, the document can be printed on several different printers without touching the document again. Just tell NEWSCRIPT which printer you're using, and the appropriate escape sequences will be generated as required.

Escape sequences are very different from control words in appearance and usage. Control words always begin at the left edge of dedicated lines, and contain English mnemonics. Escape sequences can be placed anyplace at all within normal text, and are comprised of two special symbols each. Each symbol was chosen on the basis that its appearance in some way related to the appearance of the effect it would produce. Most

escape sequences are matched pairs: one starts a special effect, and the other ends it. All the text between the matched pair is affected.

The first character of an escape sequence is always the same: by default, it is the exclamation mark ("!"). The second character indicates what special function is needed. In our sample letter, we used two matched pairs of these:

!$ - start underlining (the line through an "S")
!% - stop underlining (strike out the underline)
!(- start double width (the characters bulge out)
!) - stop double width (matches the open parenthesis)

So, if you look back at the sample letter, you'll see that we have marked "!$Mr. Smith!%" for underlining, and something else marked for both underlining and double-width. Other escape sequences are covered in Section IV, where the full SCRIPT language is described.

Sooner or later, it should occur to you to wonder how we were able to print the escape sequences here, and yet use them for special effects in the letter. The answer involves the use of a control word we won't cover here: ".ES". This lets us (or you) re-define the escape character from "!" to something else (we used "@"). Once that's done, the "!" becomes a normal data character at all times, unless it is redefined again as the escape character.

It may also occur to you that the "!" may not be usable as a normal character for innocent uses such as emphasizing a sentence. Not so! Only when followed by one of the special function symbols is it interpreted as an "escape"; otherwise, it's just another data character.

There is a special circumstance that can arise, in which you want an exclamation point at the end of a parenthesized expression, such as:

(Remember to lock the door!)

If you try to print that without special precautions, the "!)" won't print, since NEWSCRIPT will think you've just ended double-width. There are two ways around it:

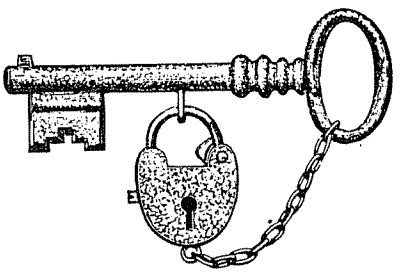
1. Re-define the escape character temporarily. We did that here, but we did so because these examples use "!)" repeatedly.
2. Trick NEWSCRIPT by writing it this way:

(Remember to lock the door!!))

The first "!" isn't followed by a special function, so it gets printed. The middle "!)" is used to turn off double-width printing, but if you were in single-width to begin with, it has no effect and does no harm. The final ")" is all by itself, so it gets printed, too. The result will be:

(Remember to lock the door!)

which is exactly what we wanted. Sneaky, yes?



Chapter 6 - Menus and Run-Time Options

Now that we've covered the use of the editor and some of the SCRIPT control words, all we need to do is finish learning about the mechanics of operating NEWSCRIPT. If you can remember back to Chapter 3, we started NEWSCRIPT by the DOS command, "NS" (or "AUTO NS"), were shown the "PRIMARY OPTIONS MENU", and immediately selected #1 (EDIT). Since then, we've learned how to create and change text, and how to tell SCRIPT about formatting it. The next step, of course, is getting something printed, and we'll step through that now.

EXITING FROM EDIT

When you're done editing some text and want to save it to disk and go on to something else, move the cursor to the command line and type:

END <ENTER>

A message will appear on the command line to indicate that saving of text is occurring: a counter will show the records being written to disk, and with some luck, one of the disk drives will be selected (the little red light) and recording your immortal prose.

If something goes wrong during this process, you'll be shown this message:

ERROR 11: DISK FULL, WRITE PROTECTED, OR I/O ERROR

Those are only the most likely causes of the problem, and there could be others. The thing to do, of course, is to try to correct the problem, most likely by trying another (already formatted) disk.

Once the text is written successfully, or you've used "QUIT" to just exit without saving anything, another menu will appear:

```
What next? Please choose by NUMBER ?_
1. SCRIPT this file (no SCRIPT options needed)
2. SCRIPT this file (allow options)
3. EDIT another file
4. EDIT APPENDED file
5. continue editing this file
6. return to PRIMARY MENU
```

To make your selection, just press a number from 1 to 6. Don't bother pressing <ENTER>, as it isn't needed. If you want to format and print the text you've been editing, and the printer is ready and waiting, and none of the options we'll cover in a minute are needed, then just press "1" and head for the refrigerator. SCRIPT will take over and perform the formatting.

If you want to select any SCRIPT options, or print a file other than the current one, pick #2; to edit some other file at random, pick #3; to let EDIT chain you automatically to the ".AP" (appended) file, pick #4; if you've just realized there's something more you want to do with the current file, pick #5 for instant return; and if you want to do anything else, pick #6 to get back to the PRIMARY MENU.

If you've chosen to just "QUIT", but have made changes to the file, EDIT won't let you get away so easily. It'll ask you whether you really want to quit. If you meant it, just reply "Q", press <ENTER>, and the exit menu shown above will be given.

OPERATING SCRIPT

SCRIPT is the program that formats and prints your text. There are several ways to get to it:

1. From the EDIT Exit menu (as you're doing now);
2. From the PRIMARY OPTIONS MENU, selection #2;
3. After processing an index (see Section V);

When SCRIPT starts, it'll display one of these prompts:

```
ENTER I.D. OF FILE TO BE SCRIPTED
                    (or)
ENTER I.D. OF FILE TO BE SCRIPTED (DEFAULT = somefileid )
```

If you've been using the editor, the second message will appear, and the default file will be the one you were just editing. Similarly, if you've been running SCRIPT and are running it again, there will be a default. However, if you activated NEWSCRIPT and went straight to SCRIPT, then the first message will be shown, since there would be no currently active file known to the system.

If there is a default and it's the file you want printed, just press <ENTER>; otherwise, type in the I.D. of the file to be used, and press <ENTER>. SCRIPT will check to see that the file is available if it isn't, you'll get an error message and an opportunity to try again.

If you're not sure what files are available, you can reply with a question mark, optionally followed by a drive number. Once you press <ENTER>, the directory in question will be displayed (this feature doesn't work with Model I TRSDOS 2.1, 2.2, 2.3, or NEWDOS 2.1 or NEWDOS40). After the display completes, you'll again be asked for the I.D.

If you decide you don't really want to print at this time, but want to use EDIT, reply with a minus sign and <ENTER>, and the SCRIPT Exit Menu will be shown. We'll cover that menu soon. By the way, the question mark and minus sign are also recognized by EDIT when it asks for a file I.D.

Once SCRIPT accepts the file I.D., it'll ask you to select any run-time options for this printing. If none of them are needed, just press <ENTER>. Otherwise, enter all the ones you need, separated by commas, and press <ENTER>. The options are fully explained in Section IV of this manual, but you won't need any of them at first.

Finally, you'll be asked to press <ENTER> when the printer is ready. (This question is not asked if you've asked SCRIPT to format to the video monitor instead of the printer.) If the printer isn't ready when you reply, you'll be given an error message and asked to try again. When the printer is ready and you press <ENTER> this third time, printing commences.

Positioning Paper in the Printer

Paper should be positioned to allow printing to start just below the top edge of each page. NEWSCRIPT will space down from that point to allow room for the top margin.

How to Pause or Cancel Printing

SCRIPT "buffers" the printing in memory as it runs. This means that it usually is several lines ahead of the printer (depending on the speed of the printer and how many control words are in your text... control words take longer to process than text does). The screen will show what SCRIPT is processing, and when the buffer fills up, the system will just wait for the printer to catch up a little.

If you want SCRIPT to stop pre-maturely, usually because you've seen a really bad error, or had a paper jam, just press the <ENTER> key for about 1/2 a second, and you'll be given this message:

DO YOU WANT TO CONTINUE PROCESSING (Y/N/E) ?_

The <ENTER> key must be pressed after entering one of those three letters. If you enter anything except "N" or "E", SCRIPT assumes you meant "YES", and will resume where it left off.

If you reply "N", you'll be given the SCRIPT exit menus. If you reply "E", you'll activate something called "Mini-Edit." That is a very nice feature built into SCRIPT, and it will be covered in Section II.

EXITING FROM SCRIPT

There are three ways to get to SCRIPT's exit menu:

1. let SCRIPT process the document(s) to completion;
2. press <ENTER> during processing, and reply "N" when asked if you want to continue processing;
3. reply with a minus sign at the very beginning, when asked for the file I.D. to be SCRIPTED.

At exit time, you'll be asked a few questions:

```
PRINT TABLE OF CONTENTS (Y/N, DEFAULT=Y)
EJECT PAGE (Y/N, DEFAULT=Y)
PROCESS THE INDEX (Y/N, DEFAULT=N)
PRESS <ENTER> TO PRINT AGAIN, <CLEAR> FOR EDIT, OR <M> FOR PRIMARY MENU
```

Of course, if your document doesn't have any table of contents or index references, those questions won't be asked; and if you had been using the VIDEO option, the page eject question won't be asked.

The really important question is the last one shown above. If you just press <ENTER>, SCRIPT will be re-initialized immediately. You can re-print the same document, or enter the name of another file to be printed.

If you press <CLEAR>, you'll be taken right back to EDIT. Again, you can re-edit the same file, or enter the name of a different one. This is the most common exit from SCRIPT when you're getting a document "just right."

Finally, if you press the letter <M>, you'll be taken back to the PRIMARY OPTIONS MENU. This is useful if you want to run some function other than SCRIPT or EDIT.

EXITING FROM NEWSCRIPT

There are several ways to get part-way out of NEWSCRIPT:

1. from the PRIMARY menu, use option zero. You'll be left in BASIC, and can run other programs. However, NEWSCRIPT is still active in the computer, and the only way to get rid of it is by resetting the machine.
2. from the PRIMARY menu, use option "8" to exit to DOS. You can issue any DOS commands you need. Again, NEWSCRIPT is still active. To get back into it fully, issue the command that takes you back into "BASIC" with four files, and run "NSINIT". If you're using TDOS, this is done by:

   TBASIC NSINIT-F:4

3. in an emergency, from almost any program, by pressing <BREAK> once or twice. This will leave you in BASIC. However, this approach is normally not a good one, since anything you were editing could be lost in memory, or what you were printing won't be completed.

Cancelling Printing

To speed up printing, NEWSCRIPT uses an internal "print buffer". Usually, NEWSCRIPT is a few lines ahead of your printer, so if you decide to stop SCRIPT while it was in the middle of processing a document, the printer will keep running for a few seconds. You can stop it more quickly by pressing:

<SHIFT><CLEAR> and then the letter <Q> (no shift)

NEWSCRIPT cannot be deactivated entirely except by pressing <RESET> or turning the computer off. We know this is objectionable to some people, and wish to explain that it was done to conserve memory. It would take more memory than we feel is justified to put everything back the way it was, and the percentage of NEWSCRIPT users who want this capability seems to be very low. We apologize to those of you who are inconvenienced by this, but ask you to consider which features you (and others) would be willing to have deleted to make room for a perfect exit.)

PRIMARY OPTIONS MENU

When we explained how to start NEWSCRIPT, we briefly showed you an illustration of this menu, but really didn't explain what it contains. Since it's the focal point of NEWSCRIPT, we'll cover it now.

The Primary Menu is shown below:

```
        <*> <*>  N E W S C R I P T   7.0  (c) 1982, PROSOFT <*><*>
            Box 839, No. Hollywood, Ca. 91603   (213) 764-3131
  ____________________________________________________________
 _|    (printer type)      | 06/01/82 |  SERIAL #:    12345   |_
_|_____________________________________________________________|_
|        PRIMARY OPTIONS MENU, PLEASE CHOOSE BY NUMBER? _       |
|______________________________________________________________|
|  1. EDIT (CREATE OR CHANGE DOCUMENT)   6. DISPLAY DIRECTORY  |
|  2. SCRIPT (PRINT A DOCUMENT)          7. (unassigned)       |
|  3. MICROPROOF / EW (CHECK SPELLING)   8. EXIT TO DOS        |
|  4. GEAP (GRAPHICS EDITOR)             9. CUSTOMIZE/INSTALL  |
|  5. PRINT MAILING LABELS               0. EXIT TO BASIC      |
|______________________________________________________________|
                CURRENT DOCUMENT IS: (unassigned)
```

First of all, there is a line of information that tells you which printer you selected during installation, the Update Level for your copy of NEWSCRIPT (that's the last time we monkeyed with it), and your unique serial number. If you have problems with NEWSCRIPT and suspect that it isn't working properly, make sure you've specified the correct printer, for openers. If you have more than one, this is a very common error. If you do contact us, be sure to tell us the update level,the serial number, the name of your printer, the Operating System you're using, and whether you're using a Model I or III. That'll give us a better idea of what corrections may apply to your problem, or whether your problem is even likely to be due to errors in NEWSCRIPT.

At the very bottom of the screen is the I.D. if the currently active file (if any). NEWSCRIPT remembers this I.D. until another file is selected, or until you press <RESET>.

The bulk of the screen lists ten options, each of which may be selected by number (don't press <ENTER>). At the time of this writing, option #7 was unassigned, which means we may use it in the future.

We've already covered options 1 and 2 (EDIT and SCRIPT).

GEAP is an optional product that may be purchased separately. It provides block (screen) graphics in a number of versatile ways. Printed results can be obtained on certain Epson and Microline printers. If you're interested, contact us or check our catalog before purchasing this option, as it doesn't work it all printers.

Microproof and Electric Webster are spelling checker programs. They are fully integrated with NEWSCRIPT, which means they can be selected from this menu, perform their functions, and then return to the menu. Several other spelling checkers also work with NEWSCRIPT, but as of the date of this writing, none of the others are fully integrated. If you wish to use some other spelling checker with NEWSCRIPT, first check with the manufacturer to ensure that it will work with NEWSCRIPT.

The LABELS option of NEWSCRIPT is used to print Address Labels from the same lists that are used for Form Letters. This option is fully documented in Section XII at the end of this book. It is not a standard part of NEWSCRIPT, but may be purchased as an extra-cost item.

Option #6 displays the Directory (except on Model I TRSDOS 2.1, 2.2, 2.3, NEWDOS 2.1 and NEWDOS40). If used with LDOS or TDOS, a sorted, numbered directory is displayed, along with a summary of how many "granules" of free space remain on the disk. If used with NEWDOS/80 or Model III TRSDOS, the Operating System directory display function is used.

Option #8 exits to DOS. As explained earlier, you can issue normal DOS commands after this, and can return to the menu without pressing <RESET> by activating BASIC with 4 files, and running "NSINIT."

Option #9 invokes the Installation procedure, "NSINSTAL". You may re-install NEWSCRIPT whenever you wish to change a setting, but after doing so, must press <RESET> and start NEWSCRIPT up again to have the changes take effect.

Option #0 just exits to BASIC. NEWSCRIPT is still active, including its "string compression elimination" feature, so you can run other BASIC programs under NEWSCRIPT and probably get a speed improvement as a bonus.

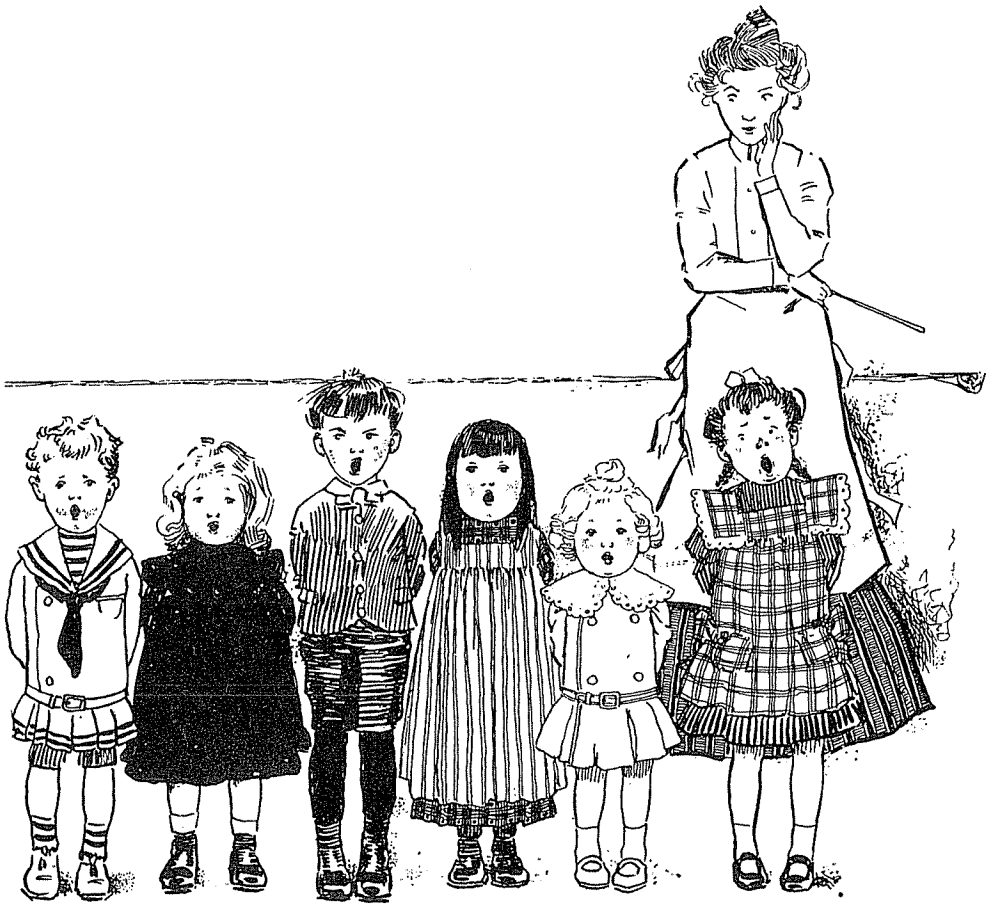
END OF TUTORIAL

This ends the beginner's tutorial. Of course, if you've been practicing as you go along, you're not a beginner anymore.

So, what do you do next? We suggest you just continue to use NEWSCRIPT, getting more practice with the commands and control words you've learned already. When you're ready for more information, you can start on Section II, which presents some advanced features, also in tutorial style. And, whenever you need specific information, you can start reading Sections III and IV on an as-needed basis.

Remember that there's a large index, as well as a good-sized table of contents, in this book, so it shouldn't be too hard to look things up as you need them. Section IX contains additional information about specific printers and Operating Systems, so if you're having trouble making NEWSCRIPT work with your equipment, take a look there. And, when you get weird error messages, look in Section VII, where we've listed and explained them.

Until we meet again in Section II... good luck!



SECTION II

ADVANCED TUTORIAL

The topics covered in this section aren't necessarily any harder than the ones in Section I; they're just things you normally won't need when you first begin to use a Word Processor. Once you've become comfortable with the facilities we've covered already, and want to start producing larger documents or fancier results, this material will be very helpful.

We're not going to cover the rest of NEWSCRIPT here, just the things that many of our existing customers have said is most important to them. And the first of these is how to take fuller advantage of the fancy features on you printer.

SPECIAL PRINTER FEATURES - ESCAPE SEQUENCES

In the first tutorial, we showed how underlining and double width can be specified by placing what we called "escape sequences right in the middle of your text. NEWSCRIPT actually supports several other printer features in a similar way. Remember, these will work only if your printer has a corresponding ability:

!$	- start underlining (most printers)
!%	- stop underlining (most printers)
!(	- start double-width (most dot matrix printers)
!)	- stop double-width (most dot matrix printers)
!-	- reverse half-line feed for superscript (some printers)
!+	- forward half-line feed for subscript (some printers)
!=	- cancel sub/super-scripts (GRAFTRAX-PLUS)
!/	- start italics (GRAFTRAX-80, GRAFTRAX-PLUS)
!?	- stop italics (same)
!*	- start boldface (requires printer feature or DWP)
!:	- stop boldface (same)
!<	- backspace a character or dot space (some printers)
!>	- pause in mid-line to change print element (not dot matrix)

Most of these occur in matched pairs. When a feature is turned on in this way, NEWSCRIPT keeps it on until you turn it back off. Even if the printer cancels the feature at the end of a line, NEWSCRIPT will keep turning it back on until it encounters the "stop" sequence.

Several features can be used at once:

The formula for water is: H_2O

was produced by:

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

Restrictions

If you printer cannot perform a reverse line feed, then sub-scripts and super-scripts won't work. There is one notable exception to this: the EPSON MX-80 and MX-100, if equipped with "GRAFTRAX-PLUS", will print half-sized characters when the NEWSCRIPT escape sequences are used. The result looks very good. However, even those printers cannot do this:

$$P_i=X^{Y^3}$$

NEWSCRIPT supports underlining only on printers that have at least one of these features:

1. have hardware underlining capability
2. recognize the backspace character
3. can perform a reverse line feed
4. can perform a very small line feed (1/72'nd of an inch)
5. can suppress the line feed and print a second time

The Radio Shack DaisyWheel Printer II is unique in that it has a hardware underline feature, but does not underline blanks. It's hardware underlining also tends to leave gaps between the lines in proportional pitch, since the underscore character on the wheels is too narrow. You can force underlining of blanks by placing an underline character right on the EDIT screen in place of those blanks. We'll explain more of this later, but for now, the method is to place the cursor on the blank to be underlined, and then:

1. Hold down <SHIFT> and press <CLEAR>
2. Release both, then press the minus sign.

NEWSCRIPT can print italics and boldface only on dot matrix printers having those character sets built right into them. We don't use high-resolution graphics to simulate such features. With the "Daisywheel Proportional Option" (DWP), NEWSCRIPT can do selective boldface on any letter(s) you select. It does so by striking the character(s) three times without otherwise moving the carriage. This produces a sharp, dark result, rather than the blurry one that often results from attempting to move forward slightly between overstrikes.

If you have a printer that falls in printer categories 9-12 (Selectric, DaisyWheel II, Diablo-compatible, Spinwriter), then the "pause" code can be used to stop the printer in mid-line for a wheel change. Pressing <ENTER> gets it going again. This feature doesn't work on dot matrix printers, since you can't change the print head in mid-line.

SPECIAL SYMBOLS AND FEATURES

In the first tutorial, we explained that there simply aren't enough keys on the keyboard to provide us with all the functions we may need to use. The <SHIFT> key is the typewriter's way of adding more characters without doubling the number of keys, and <CLEAR> and <SHIFT><CLEAR> are NEWSCRIPT's way of going even further.

We covered all the <CLEAR> (or <CONTROL>) functions back in Section I, and touched on a couple of <SHIFT><CLEAR> capabilities. Now, we'll cover the rest of them, and also introduce several ways of getting the computer and the printer to cooperate in printing special symbols that normally are hard or impossible to get at.

The <SHIFT><CLEAR> commands are used primarily to insert special codes within a document. There are several reasons why you might want to do this. You may, for example, want to output special printer codes, or output graphics characters to your printer (provided, of course, that your printer can take them). The other reason to use <SHIFT><CLEAR> is to control certain NEWSCRIPT functions. The following table lists all the options:

command	meaning
R	toggle auto-repeat on/off
P	copy current screen to the printer
V	toggle dual routing of video-to-printer on and off
J	toggle Japanese character set (Model 3 only, only on the screen)
S	toggle special characters and space compression codes (Model 3 only, only on the screen)
right arrow	<RIGHT ARROW> character
left arrow	<LEFT ARROW> character
down arrow	<DOWN ARROW> character
-	underscore character
=	unsplittable blank character
shift up arrow	left brace
shift left arrow	right brace
shift down arrow	vertical line
shift right arrow	tilda
00 - FF	insert byte with value in hexadecimal

When you press the <SHIFT><CLEAR> keys together, a double question mark appears in the lower right-hand corner of the video screen. This is to indicate that a special code is expected. When the question marks appear, the two keys should be released, and then one of the keys above pressed. If you get into this mode by accident, you can get back out by hitting <SHIFT><CLEAR> a second time.

If you press <R>, then the auto-repeat function will be turned on or off, depending on whether it was on before you pressed it or not. When NEWSCRIPT starts, auto-repeat is ON.

Pressing <P> sends the contents of the 16 x 64 video screen to the printer. Text outside the viewing window will not be sent to the printer.

Pressing <V> will cause everything sent by normal methods to the video screen to be sent to the printer as well. Pressing it again turns the feature off. Text that is placed directly on the screen (through BASIC 'POKE' or the techniques used by EDIT) cannot be printed this way, although <P> still will work.

Several other common print symbols can be accessed as listed. The left bracket is not on the list because the "up arrow" stands for it, but the up arrow is handled strangely by the circuits of the TRS-80. To circumvent this, you can get a left bracket by pressing <CLEAR><UP ARROW> when in full-screen edit mode. "J" and "S" don't affect what gets printed; they only control what appears on the screen of the Model III

Hexadecimal Characters

Please don't get scared off by the funny computer word! We only want to show you some more ways of fully utilizing your computer with printers that can produce characters that do not occur on the TRS-80 keyboard. These may be graphics, Greek letters, drawing symbols, brackets, etc. NEWSCRIPT offers you three separate ways of getting such characters to the printer:

1. <SHIFT><CLEAR> then a pair of hexadecimal digits
2. EDIT's "ALTER" command
3. SCRIPT's TRANSLATE (".TR") control word

Some special symbols can be created by any one of these, but others are either restricted or most convenient when done in one particular way. We'll cover some of this below, and the rest in Sections III and IV.

<SHIFT><CLEAR> can be used to enter codes that are not normally available in NEWSCRIPT. By pressing <SHIFT><CLEAR>, releasing them, and then entering any two hexadecimal digits (0-9 and A-F), the appropriate character is created and placed within your text at the current cursor location.

Certain codes cannot be created in this manner, since NEWSCRIPT or the TRS-80 will interpret them either as control symbols or arrow-key commands. In these cases, use either the EDIT command, "ALTER", or the SCRIPT control word, ".TR".

The ASCII 13 character (carriage return) should never be created except through ".TR" (translate), and the ASCII 0 character (null) should never be created at all, since the computer will ignore it. Other characters in the range ASCII 1-31 can always be created through ".TR", sometimes through "ALTER", and almost never through direct hexadecimal entry.

ALter /c1/c2/ <n <*>> <#>

The ALTER command is a cousin of the CHANGE command. The latter changes one string to another within your text. The former does the same, but with a single character at a time. The syntax is much the same as with the CHANGE command. The two required parameters, c1 and c2, must be surrounded by delimiters, as shown in the command line above. The optional parameters "n", "*", and "#", are the same as with CHANGE.

c1 and c2 may be entered in either of two ways, and these may be mixed. They may be a character, such as a 'P' or an 'O', or they may be the ASCII equivalent. Of course, you have to know the ASCII (numeric) equivalents of the characters you're using, and to some extent, this is printer-dependent. All of these produce the same result:

```
C/N/P/
AL/N/P/
AL/78/80/
AL/N/80
AL/78/P
```

We deliberately started with CHANGE instead of ALTER, since in this example, either could be used. We checked a table to find that the letter 'N' is an ASCII '78'. Of course, if we only needed to change one letter to another, we wouldn't bother with ALTER. However, to print Greek symbols (if your printer can do it) or certain other graphics, you might find this feature very useful. Example:

```
AL/*/217/**
```

.TR n1,n2,n3 (or) filespec

"Translation" means the same thing on a computer as it does with a spoken language: it is the substitution of one symbol for another. NEWSCRIPT allows you to specify translations on a character-by-character basis through use of this control word. The translation is performed at printing time, and doesn't affect anything else.

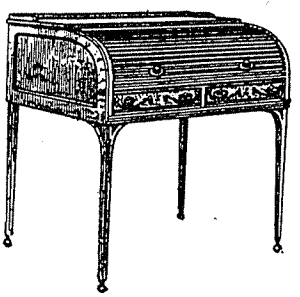
The control word takes either three numbers or one filespec. If the numbers are used, 'n1' and 'n2' specify the range of characters to be translated, and 'n3' is the quantity to be added or subtracted to each character in that range. Subtraction occurs if you put a minus sign ahead of 'n3'. 'n3' must be no smaller than '-128', and no larger than '127'.

The "filespec" identifies a disk-resident table. These tables, and the means of creating additional ones, are supplied with the Daisywheel Proportional Option, and are not part of standard NEWSCRIPT.

There are three primary uses for ".TR":

1. To print certain special letters or symbols;
2. To print block graphics with EPSON printers;
3. To identify transposition tables for special daisywheels.

Further information about this may be found in Section IV, where ".TR" is described fully.



Tambour writing table.
Century Dictionary

HANGING INDENTS

The hanging indent is best illustrated with an example:

> ** This is sometimes called a "bullet." As you can see, the first line hangs out further to the left than the subsequent lines do. In effect, all lines except the first are indented. They will continue to be indented until the next control word is encountered.
>
> ** This is the second bullet. A ".sK" command separates the two bullets. As you can see, the indent is delayed until the second line. Then it remains in effect until a control break, such as a space, occurs within the text. Then it starts over again, delaying the indent until the second line. Hanging indents remain in effect until set to zero, and forgetting to do so produces unwanted results.

The paragraphs shown above were specified as follows:

```
___ .sk2;.in10;.ll-10;.hi3
___ ** (text follows
___ and continues)
___ .sK
___ ** (text for second bullet)
___ .sk2;.in0;.ll;.hi0
___ .cm that last '.hi0' turns off hanging indent
___ .cm omitting 'n' resets '.ll' to its last absolute value
___ (text continues)
```

NEWSCRIPT has three facilities for producing "bullets":

.HI n	- automatic delayed indent after any control word
.OF n	- delayed indent after each ".OF" control word only
.PT n	- semi-automatic numbering of paragraphs

('n' is an optional number)

".HI" stands for "Hanging Indent"; ".OF" stands for "Offset"; and ".PT" stands for "Point." ".HI" and ".OF" are very similar in function: they both cause a delayed indent, in which the first line of text following the control word is <u>not</u> indented, but subsequent lines <u>are</u> indented. They differ in that ".HI" is reset automatically when any control word is encountered, whereas ".OF" is reset only when another ".OF" is encountered. By "reset", we mean that the next line of text will again "hang out" to the left, but lines after it will again be indented, etc., until the feature is turned off.

".PT" is generally used in conjunction with the other two. It merely generates a number, a period, and a space, then tacks these onto the left of the first line of the text following the '.pt' command. If 'n' (a number) isn't specified, '1' will be added to the value of the previous point. If a number <u>is</u> specified, it'll be used as the starting point. The first time '.PT' is used in a document, it will default to '1'.

Superficially, '.hi' and '.of' seem the same, but actually, there is an important difference. The indent is delayed for the first line in both cases, but with '.hi', when the next control word is encountered, such as '.sk', the line after it will not be indented, since '.hi' performs an automatic reset. This is not true with '.of': the indent caused by an '.of' command continues until another '.of' command is encountered, even if several paragraphs are created within the single major bullet.

For example, if you wanted to create a simple list of bullets, you would need only one '.hi' to turn it on, and one to turn the whole thing off when you want to return to normal formatting. You would separate the bullets with a '.sk', '.sp' or some other control word.

On the other hand, if you want the indent to continue to occur for several paragraphs, with only the first line of the first paragraph hanging out, then you should use '.of' just ahead of that first line. Also, if you have any other control words within a series of paragraphs, but don't want a reset on the hanging indent, then you should use '.of'.

Use of '.hi' can lead to unexpected, incorrect results when you accidentally place other control words within the scope of an active hanging indent. For example:

```
___ .hi3;.in15;.ll-15;.sk
___ * this is one bullet
___ .sk
___ * well this is too, but I want to
___ .sk;.ce;.us emphasize
___ .sk
___ what happens with this when you create
___ bullets with the hanging indent command and
___ control breaks within them
___ .hi;.in;.ll
___ .cm when numbers are omitted, .hi, .in, and .ll
___ .cm revert to their permanent settings
```

Formatting this text gives us:

* this is one bullet

* well this is too, but I want to

emphasize

what happens with this when you create bullets with the hanging indent command and control breaks within them

The word 'what' hung out incorrectly. Now, see the difference when '.OF' is used:

```
___ .of3;.in15;.ll-15;.sk
___ * this is one bullet
___ .sk;.of3
___ * well this is too, but I want to
___ .sk;.ce;.us emphasize
___ .sk
___ what happens with this when you create
___ bullets with the hanging indent command and
___ control breaks within them
```


This produces the following, which is what we wanted:

* this is one bullet

* well this is too, but I want to

emphasize

what happens with this when you create bullets with the hanging indent command and control breaks within them

Notice that the hanging indent command required only one command at the start to set the hanging indents and then a control break (such as a blank line created by '.sk') to restart the hanging indent. The '.of' command, however, must be invoked each time you want it to perform another offset.

The parameters for the hanging indents and offsets are in tenths of inches. '.hi 3' offsets 3/10 inches, regardless of the pitch or font being used. You also may use relative indents, such as ".of +4" or ".of -2".

Hanging indents and offsets remain in effect until set to zero:

```
.OF 0
.HI0
.OF
.hi
```

All of these turn the feature off, since omission of a number implies 'zero' for ".OF" and ".HI". Please note that other control words may take different defaults, depending on what makes the most sense for them.

Numbered Points

We used asterisks as bullets in the above examples. If you wanted to use numbers instead, you certainly could do so. However, if you had several things to number, and knew that you might rearrange them later on, or even add to the list, then you could use '.pt' to let NEWSCRIPT take care of the sequencing for you. Here's an example of it:

```
___ Today's meeting will address several subjects:
___ .sk;.in10;.ll-10;.hi3;.pt1
___ Impact of low-cost computers on educational methods;
___ .pt
___ Financial considerations for purchasing computers
___ .pt
___ Academic and hands-on training for faculty in advance
___ of making the computers available to students
___ .hi;.in;.ll
```


When printed, this would appear as follows:

Today's meeting will address several subjects:

1. Impact of low-cost computers on educational methods;
2. Financial considerations for purchasing computers
3. Academic and hands-on training for faculty in advance of making the computers available to students

MOVING, COPYING, AND DELETING BLOCKS OF LINES

In NEWSCRIPT, a "block" is a group of lines on the screen. A block may contain anywhere from one to 400 lines (400 being the maximum number of lines EDIT allows in one file). However, a block cannot start or end in the middle of a line.

Blocks are less important in NEWSCRIPT than in some other word processors because NEWSCRIPT "sees" text either as lines or as a continuous stream of text. However, when you want to manipulate several lines at a time, say to move a paragraph from one place to another, then blocks become very important.

The present discussion only covers the manipulation of blocks within a single EDIT file. The advanced tutorial in Section II explains how "GET" and "PUT" can be used to move blocks from one file to another.

Suppose that all of a sudden you decided that paragraphs 4 and 5 really ought to be placed after paragraph 9 rather than after 3, as they are now. What can you do? Retyping would be cruel but usual punishment, but with NEWSCRIPT it is a simple matter to move the entire block of two paragraphs to its new location. Let's see how this can be done.

The first thing you must learn to do is to MARK the lines to be manipulated. This is done by using two or three of the LIMA commands, ".A", ".B", and ".C" as the markers. These are not the same as SCRIPT "control words", which will be covered later on. They begin with a period to associate them with the word "point", since, after all, you're marking points in your text, which will serve as labels for the EDIT commands that manipulate them.

To place a marker or command in the LIMA:

1. move the cursor up/down to the line in question
2. If the cursor is at the left edge of the data area, press <CLEAR><LEFT ARROW> once; if it's in the middle of the data area, press that combination twice.
3. Type the label or command.
4. Press <ENTER> if you want it processed immediately. Otherwise, if you want to mark another line, move the cursor down to it and type in the second marker.
5. IMPORTANT: If a LIMA marker or command scrolls off the screen, it is lost, not processed. So, if that other line you want to mark is far away, press <ENTER> to tell EDIT to record the first mark. Then, you can find and mark the next line.
6. To get out of the LIMA, either press <ENTER> or hit the right arrow until the cursor enters the data area.

7. NOTE: Once a LIMA marker has been processed, it disappears from the screen. However, EDIT remembers where it was until the line is deleted, the marker is re-used, or you exit from from EDIT.

There are only three markers. We may refer to them as "named points", "labels", or "markers", but will always mean the same thing. These markers are ".A", ".B", and ".C". They are unique and must be used in the prescribed order, which is alphabetical.. No other labels are recognized, so if you try to use ".D", let alone ".Z", it will simply be ignored, and not even an error message will mark its passing.

Labeling a line to be moved is simple. First, move the cursor into the LIMA area of the line to be manipulated, and enter the appropriate label. Then press <ENTER> while the cursor is still within the LIMA area, and the line is labeled. This label will remain attached to that line until a manipulation of that line occurs or the label is invoked to mark another line.

Since the LIMA can be used for several purposes, the labels will not be shown after you press <ENTER>. However, if you don't remember where a label is, you can put its name, e.g., ".A" (no quotes) on the Primary Command line, press <ENTER>, and that line will appear at the top of the data area immediately.

If you're going to manipulate a single line of text, then label it ".A". If you are going to manipulate a block of lines, then use ".A" to label the start of the block, and ".B" to label the end of the block. The block of lines must be contiguous; that is, the line manipulated must be contained within a single block with no unwanted lines between them; and ".B" must come after ".A".

When you are going to copy or move these lines, then you must mark the line that just precedes where you wish the lines to be moved or copied. If you are moving or copying a single line, then you must use ".B" as the 'put it right after here' marker, while if you are moving or copying a block of lines, you must use the ".C" marker. (This is how EDIT distinguishes between moving a single line and several lines.)

VERY IMPORTANT:

".C" must never, never, NEVER lie between ".A" and ".B". It can be above ".A" or below ".B". If you place ".C" between the others and do a "MOVE" or "COPY" (described below), all the text involved will seem to disappear. It's still possible to get to it and recover it, but to avoid the problem, please be careful.

You may use these markers to indicate lines to be moved, copied, inserted, or deleted. Let's now examine the commands that make all this happen. Be aware that these commands (below) assume that the proper lines have been marked prior to invocation of the command. Also note that if the lines to be marked occur on the same page, then you may mark more than one line at a time before pressing the <ENTER> key by moving the cursor to the appropriate lines. This is true, in general, of all LIMA commands.

When manipulating blocks, we begin to use facilities from all EDIT categories: we move the cursor to the desired lines by using arrows or <CLEAR> commands; we mark things in the LIMA, and then we refer to these markers through Primary Commands. The discussion below introduces some of those Primary Commands.

DElete <n * .B>

The delete command causes a single line, some number of lines ("n"), or a marked block of lines (".B") to vanish from the document, and the text below it to be be moved up. If <n> is specified, then 'n' lines are deleted. The default, if only "DE" is given, is for the line at the top of the data area (the current line) to be deleted. If * is used as the parameter, then all lines from the current line to the end of the file will be deleted. The text that precedes the current line will not be affected.

You may also use the '.B' parameter instead, if you prefer. The command will work only if both .A and .B are currently marking lines, and .B occurs AFTER .A. The effect will be to delete all the lines from .A through .B, inclusive. A variant of DELETE will be discussed under the command DSTRING.

An example of the DELETE command is shown below after these commands are all introduced.

MOve <.B .C>
COpy <.B .C>

These two commands have identical syntax and similar functions. The "MOVE" command moves a block of lines from one location in a file to another location in the same file, while COPY command duplicates the block of lines where specified, but leaves the original alone. If you use the 'MO .B' then only one line will be moved; it will be the line marked with '.A', and it will be moved and inserted immediately after the line marked with '.B'. If you use a block of lines, then the start of the block must be marked with '.A', the end of the block must be marked with '.B', and the line just preceding where you want this block moved to must be marked with '.C'. After this marking is completed, then the command 'MO .C' will do the rest! Of course, if you used "CO .B" or "CO .C" instead of "Move", then the line(s) would be duplicated.

DString /string/

The DS command is a special form of DELETE. The command mnemonic stands for "Delete up to String." It looks for the first occurrence of "string" in the text and deletes all the lines starting with current line and ending with the line before the one containing "string". The "string" MUST be specified. This allows you to have special strings of characters to mark lines that you wish to delete. An example of this command is shown below.

Now let's devote some time to looking at examples. We will use a micro-screen patterned after the screen that you will see on your video, and will examine the effects of these global line-manipulation commands. Given the screen below, let's use the COPY command to copy some of the lines to another place on the screen.


```
E=>_____________________________________________________
.a_ The theory of problem solving is an interesting
.b_ one. Basically, it involves classifying a problem
___ into a major type and applying strategies that
.c_ are usually applicable to that category of problem.
___
___
```

Now that you have all the commands entered, and while you are still in the LIMA area, press the <ENTER> key to mark the lines. The cursor will be placed at the top of the screen. Now, you can enter the COPY command "CO .c", and press <ENTER> to get the following result:

```
E=>_____________________________________________
___ The theory of problem solving is an interesting
___ one. Basically, it involves classifying a problem
___ into a major type and applying strategies that
___ are usually applicable to that category of problem.
___ The theory of problem solving is an interesting
___ one. Basically, it involves classifying a problem
___
___
```

If we wanted to copy only a single line, we would have marked it with '.A' and used the command 'CO .B'. Since we really don't need these extra two lines here, let's delete them with the DELETE command:

```
E=>_____________________________________________
___ The theory of problem solving is an interesting
___ one. Basically, it involves classifying a problem
___ into a major type and applying strategies that
___ are usually applicable to that category of problem.
.a_ The theory of problem solving is an interesting
.b_ one. Basically, it involves classifying a problem
___
___
```

Then we invoke the Primary Command "DE .B", which gives:

```
E=>_____________________________________________
___ The theory of problem solving is an interesting
___ one. Basically, it involves classifying a problem
___ into a major type and applying strategies that
___ are usually applicable to that category of problem.
___
___
```


Next, we will use the MOVE command to interchange the last two lines of text:

```
E=>_____________________________________________
___ The theory of problem solving is an interesting
___ one. Basically, it involves classifying a problem
.a_ into a major type and applying strategies that
.b_ are usually applicable to that category of problem.
---
---
```

And by entering the command 'MO .B' on the command line, which places the line marked '.A' AFTER the line marked '.B', we get:

```
E=>_____________________________________________
___ The theory of problem solving is an interesting
___ one. Basically, it involves classifying a problem
___ are usually applicable to that category of problem.
___ into a major type and applying strategies that
---
---
```

MOVING BLOCKS BETWEEN FILES

The "MOVE" command of EDIT is fine for shifting a paragraph from one place to another within a single file, but doesn't solve the problem of moving a paragraph from one file to another. Also, there are times when you will want to include a "stock" paragraph and then tailor it to your specific needs; in such cases, ".IMbed" won't work. To cover these situations, EDIT has two special commands: "PUT" and "GET".

PUtfile filespec

This copies a block of text from the file being edited to a disk file. The block is not deleted from the original file, nor from the in-memory copy being edited.

The block begins with the line marked ".A" (in the LIMA), and ends with the line marked ".B". These points must both have been set ahead of time, but don't have to be on the screen when "PUT" is issued. ".B" must follow ".A" in the file. A minimum of two lines of text are written to disk by this command.

If "filespec" already exists, it can be replaced or appended. If the two points are not defined or valid, an error message will be given.

Once a block has been written successfully with "PUT", you can delete it from the original file by:

```
DE .B
```

This may take a few seconds to complete if the block is large. Once it's finished, you can save the file back to disk. If you want to use the extracted block in another file afterwards, just edit that second file and use "GET" as described below to pull in the extracted block.

GEtfile filename < /string1/<string2/>>

The GE command "gets" a file and places it after the current line (remember, this is the top data line on the screen). The filename MUST be specified (how else will it know what file to load?), but the other parameters are optional. If neither of these is specified, then the file, or as much of it as can be read into memory, will be loaded. If the file is not found, or a bad filename was entered, then you will receive an error message.

If 'string1' is specified, then it must be surrounded by delimiters. A delimiter is a character used to show the beginning and ending of a small block of text. Most often a slash (/) is used, but if a slash appears in the string that you are entering then another character, such as '@', may be used.

The two parameters string1 and string2 are used to define the range of text from the file to be loaded in. It is possible that you may not want the entire file loaded in, so by specifying special strings, you may limit this range.

String1 specifies that the text being loaded in will begin with the line containing 'string1'. String2, when specified, means that EDIT should stop loading the file after 'string2' is found. Some examples appear below:

```
GET file1  - get the entire file called 'file1' and load its contents immediately
             after the current line.

get file1 /This is/  - get all of file1 starting with the first occurrence of "This is"
                       within "file1."

ge file1 /This is/ending/  - get the portion of file1 that falls between the first
                             occurrence of 'This is' and the first occurrence
                             thereafter of 'ending'.

GE file1 //ending/  - get the beginning of file1, starting from the first line, and
                      ending with the first occurrence of 'ending'.
```

EDIT is smart enough to recognize whether the slash is within the filename or not, and so you may use the slash as the delimiter even though it appears in the file name. For example, "GE file1/txt /begin/end/" will look for 'file1/txt' and load the text between 'begin' and 'end'.

Note that if string1 is specified, but not found, then no text will be loaded. On the other hand, if string2 is specified but not found, then the entire file after string1 is loaded.

GET is very useful for loading short pieces of text into your current document. Suppose, for example, that you keep a standard contract condition called 'CANCEL/TRM' on disk, want it included in a contract you're writing, but need to make a small change to it. The need to modify it means you can't just say ".IM CANCEL/TRM", so instead:

1. scroll the screen until the top line of text is the one after which the external paragraph should go;
2. move the cursor to the command line;
3. type: get cancel/trm <ENTER>
4. if "cancel/trm" is on an available disk, it'll be read in, and then appear at the top of the screen. Now, you can make any changes you need, since this private copy will be saved as part of the document you're creating.

Now, if you have lots of little paragraphs to modify in this way, it may be very wasteful of disk space to make a separate file for each of them. After all, on a single-density diskette, each "granule" holds about 200 words; and on a double-density diskette, about 250 words. To conserve space, it might make sense to put several stock paragraphs in one big file, and give a unique starting and ending name to each. Then, you could "GET" just the part you need. It would take longer, but would save space. Here's an example of such a file, and a use of it:

```
.cm *penalty
.pp
It is agreed that the maximum liability to each party
shall not exceed _____?_____...
.cm *end
.cm *cancel
.pp
In the event that either party should wish to
withdraw from this agreement, and the other...
.cm *end
.cm *distributions
.pp
Distributions of profits shall be made at the end of
each ______?______...
.cm *end
```

Suppose this file was called "TERMS", and we wanted to include that cancellation clause. The screen might look like this just before we pressed <ENTER>, since the command to use is shown on the screen:

```
E=> get terms /*cancel/*end
___ shall be binding for 18 months from the starting date.
___ .pp
___ This contract shall be governed by the laws of the
___ State of California...
___
___
```


After the command completes, the screen will look like this:

```
E=> *GETTING: terms                    5    *RECORDS READ
___ shall be binding for 18 months from the starting date.
___ .cm *cancel
___ .pp
___ In the event that either party should wish to
___ withdraw from this agreement, and the other...
___ .cm *end
___ .pp
___ This contract shall be governed by the laws of the
___ State of California...
___
___
```

MINI-EDIT (WITHIN SCRIPT)

"Mini-edit" is a simple editor built right into SCRIPT. It only has five commands, and it can't save anything to disk. It's used for two purposes:

Mode 1: make minor or temporary changes during printing;

Mode 2: experiment with SCRIPT or write simple letters.

Mode 1 is used in conjunction with a file previously created with EDIT. The mode is activated in either of two ways:

1. specify the SCRIPT run-time option, "ED";
2. press <ENTER> while SCRIPT is formatting the file. When asked:

DO YOU WANT TO CONTINUE PROCESSING (Y/N/E)?

reply "E" <ENTER> (no quotes, upper or lower case).

Once Mini-edit is activated in either of these ways, it will let you examine and modify each line that is read in from disk. This applies to control word lines as well as to text lines. The line will be displayed, and then you'll be prompted:

* EDIT *?

There are five acceptable replies to this prompt:

<ENTER>	- accept line as-is
D	- delete line
I text	- insert line of 'text' ahead of this line
R text	- replace line by new line of 'text'
T	- terminate Mini-edit mode, resume automatic processing

Each of the last four replies must end with <ENTER>.

This mode of Mini-edit is useful in making minor corrections to a document without taking the time to re-run it through EDIT. It's also useful when printing a file from the middle of a large document, since you can insert margin controls that were set only in the first file of the chain.

Mode 2 of Mini-edit is even simpler than Mode 1: all you do is type text and control words directly into SCRIPT for immediate processing. The lines you type can be at most 3-1/2 screen lines in length (240 characters), and each must end with <ENTER>. You can use the left-arrow to correct mistakes before <ENTER> is pressed.

There are two ways to activate Mode 2:

1. when asked for a File I.D. to be SCRIPT'ed, reply with a semi-colon: ";" If you do this, nothing will be read from disk, and all input will be taken from the keyboard.
2. when SCRIPT finishes printing a disk file, it will ask:

   EJECT PAGE (Yes/No/Continue, DEFAULT="Y")

   If you reply "C" or "c" <ENTER>, Mini-edit Mode 2 will be activated.

Mode 2 prompts for a new line to be typed in as follows:

* ADD *?

You can type in control words or text, then press <ENTER>. The line will be processed just as though it came from disk, and then the prompt will re-appear. About the only control words you can't use are ".IM", ".AP", and ".RD", since they would use disk files.

To exit from Mode 2, just hit <ENTER> without typing any text. When you do so, any remaining text will be formatted and printed, and you'll be given one last chance to type some more:

PRESS <ENTER> TO CONFIRM EOJ, OR TYPE MORE DATA

"EOJ" is computerese for "end of job". It's just a way of indicating that the computer is finished with whatever it was doing. In this case, if you really are done, just press <ENTER>. Otherwise, continue typing.

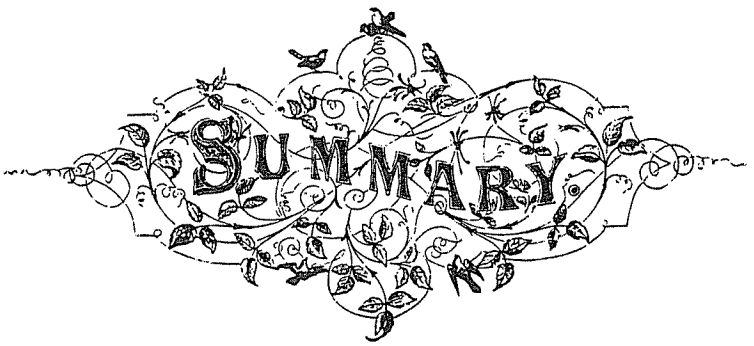
Mode 2 has two main uses:

1. writing a "P.S." at the bottom of a letter;
2. going directly from the keyboard through SCRIPT to the printer.

The second use is the interesting one. If you use ";" in place of a File I.D. when SCRIPT begins, all input will be taken from the keyboard instead of the disk. This means you can just write a letter or short memo on the spot, without bothering with EDIT. Or, you can experiment with SCRIPT formats and control words, very quickly and easily. All you have to remember is that nothing you type will be, or can be, saved to disk. SCRIPT is just acting like a very smart typewriter when Mini-edit is used.

Summary

Well, that about does it. There are many things we either haven't mentioned, or only partly covered, but you should be far enough along by now to be able to use the regular Reference Sections and the Index when you need more information. We hope these tutorials have been helpful to you. As you gain more practice, you'll find yourself becoming proficient, and then expert in the uses of NEWSCRIPT. In the years to come, as microcomputers get bigger (in features) and less expensive, a lot more "mainframe" software will become available on them. And, since NEWSCRIPT is based on some of the most widely-used mainframe software packages, you'll find you've had a nice head start on the future.



SECTION III

THE EDITOR

This is a Reference Section. It defines and describes all features and commands of the NEWSCRIPT components "EDIT" and "FEDIT/CIM". The latter is used automatically by "EDIT", and except for making sure it's on the working NEWSCRIPT disk, you never need to pay attention to it. Some of the features of EDIT are more fully described in the tutorial sections earlier in this book, and it is assumed that those sections already have been read.

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, 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.

   Horizontal movement of the viewing window occurs only in response to use of <CLEAR><RIGHT ARROW> or <CLEAR><LEFT ARROW>. The command line is not examined, and any LIMA commands 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. The command was "END" or "QUIT", the EDIT function is terminated. Otherwise...
7. If the cumulative number of changes exceeds the "AUTOSAVE" limit, the entire in-memory file is written to disk. The permanent name of the file (the one used when EDIT was started, or the last name assigned through use of the "NAME" command) is used when AUTOSAVE occurs.
8. The screen is updated and control returns to the user.

NEWSCRIPT checks for keystrokes very frequently, and stores what it finds for later processing. This is called "typeahead." It allows you to keep typing while other things are happening on the computer, without worrying about your keystrokes being lost or ignored. typing while most of this processing occurs. Up to 128 characters can be entered with typeahead. The feature is always active except when disks are running. When they are, it is possible for keystrokes to be lost if you type quickly, but this depends on the Operating System you use.

Cursor Movement

The blinking cursor can be moved to any position on the screen except the column following 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* (Top Of File) marker is reached, the up-arrow will not cause any further scrolling towards the top of the file; if the *EOF* (End Of File) 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 or "TBEEP" 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.

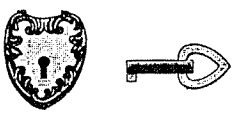
Unless turned off, every key will auto-repeat if held down for about a second. Repeat speed is normally 20-25 characters per second, but you can change this during the installation of NEWSCRIPT (see "NSINSTAL").

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 <CLEAR> key functions (described next), or the "CHANGE" command.

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. If your computer has a true <CONTROL> key (the one for "Pencil" will do), you can use it interchangeably with <CLEAR>.

As we explained in Section I, NEWSCRIPT has two "control" modes. One is accessed at any time through <SHIFT><CLEAR>, and the other is available only during full screen editing through <CLEAR> (or <CONTROL> alone. In both cases, once a control mode is entered, another key must be pressed to select a particular function. This is similar to the use of <SHIFT> to obtain capitals or special symbols. one in turn.



<CLEAR> Key

When you press the <CLEAR> key, the shape of the cursor changes to a big "C" to indicate that you're in <CONTROL> mode. Once this happens, you can press a second key to select the particular function needed. It isn't necessary to hold <CLEAR> down once it's been pressed, although it does no harm to keep holding it. If you want to exit control mode without doing anything else, just press <CLEAR> again. Once you press any other key, the function you select will be performed, and control mode will be exited.

While in control mode, the following keys can be used. Some of these are more fully explained in Section I:

```
B  -  Back scroll one screen
D  -  Delete character at cursor
E  -  move screen to End of file
F  -  Forward scroll one screen
H  -  display Help screen
I  -  enter or exit Insert mode
N  -  move screen down to Next line
P  -  Play or Pause audio tape
S  -  Save file to disk
T  -  move screen to top of file
U  -  move screen Up one line
W  -  delete (rest of) word at cursor
X  -  perform command assigned to 'x'
Y  -  perform command assigned to 'y'
/  -  repeat most recent "Locate"

<RIGHT ARROW>  -  move screen to right (20 columns or whatev
<LEFT  ARROW>  -  move screen to left  (...was set by 'HOrizontal')
<UP ARROW>     -  places a left bracket or up arrow on the screen
<BREAK>        -  split line into two lines at cursor position
<SPACE>        -  blank out rest of line starting at cursor
```

Some of these control screen movement: F,B,U,N,T,E, and the left/right arrows. Some cause previously-defined commands to be executed: X,Y,/. Some cause deletion or insertion of text at the cursor position: D,W,I,<BREAK>,<SPACE>. And three are miscellaneous: H,S, P, and <UP ARROW>.

The "X" and "Y" functions are described under the Primary Commands of the same names near the end of this Section. The "S" (save) key lets you save the file to disk immediately, without having to move the cursor to the command line. Note that accidental use of <CLEAR><S> will cause the file to be saved. If this happens, just wait for it to finish; don't try to interrupt it.

The left bracket is represented by the <UP ARROW> on the TRS-80. On the Model I, it displays as an arrow, but on the Model III, it displays as a left bracket. Due to the way the TRS-80 is constructed, it's hard to get the left bracket on the screen. EDIT's <CLEAR><UP ARROW> is the only way to do it. Incidentally, to get a right bracket, use <SHIFT><CLEAR><LEFT ARROW>.

Transcribing Dictation

"P" lets you control playback of dictation. If you connect a normal cassette player that has a "remote" jack to the tape cassette port of the computer, insert a tape and press "PLAY" on the tape machine, then you can use "P" to let the tape play back or pause. Each time you press "P", the tape switches to the other state: if it was paused, it starts to

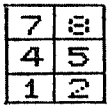
play; if it was playing, it pauses. It's a very easy way to perform transcription of dictation, and we give credit for this clever idea to Mr. Harv Pennington of IJG. He included this feature in the second version of "Electric Pencil" (trademark of Michael Shrayer) and graciously consented to allow us to use it here.

"I" sets "Insert" mode, and the cursor changes shape from a square "o" to a tall "I". It remains that way until <CLEAR><I> is pressed again, or until certain other conditions occur that make it natural for EDIT to switch back to overlay mode by itself.

"I" and "P" are the only keys besides the graphics keys (described below) that act as "toggle" switches; all the others work once only unless held down long enough for auto repeat to start. Auto repeat works for <CLEAR> key functions if <CLEAR> is held down along with the particular function key.

TRS-80 Block Graphics

The standard TRS-80 has limited graphics capabilities: it can display six small rectangles in each character position. The editor lets you control these six rectangles to facilitate drawing or touching up graphic patterns. To do so, hold down <CLEAR> and press any of these six numbers according to the rectangle that is to be turned on (or off):



These were picked because, on a 10-key numeric pad, the correspond to the positions of the six "pixels" (rectangles) of a character position. The keys act as toggle switches: if a pixel was "off", it is turned "on", and vice versa. Also, the cursor does not move when one of these keys is pressed, since several pixels may need to be manipulated.

<SHIFT><CLEAR>

Holding down <SHIFT> and pressing <CLEAR> activates NEWSCRIPT's general 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. Auto-repeat normally is active, which means that when a key is held down for about 3/4 of a second, it will begin to repeat, and continue to do so until released.

Q empties the internal print buffer. Since NEWSCRIPT runs faster than most printers, it stores a few lines until the printer is ready for them. If you pause or stop SCRIPT, printing will continue until this buffer empties. If you want to entirely cancel printing, you must press <SHIFT><CLEAR><Q> after stopping the formatter. Or, you can wait a few seconds for the buffer to empty naturally.

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.

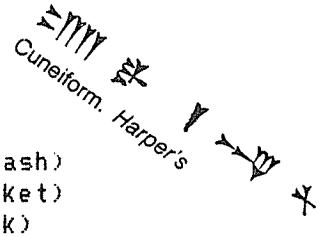
V turns dual-routing of Video-to-printer on and off. When on, everything BASIC writes to the screen is also written to the printer. However, changes made directly on the screen will not be routed to the printer in this mode, so the Full-screen edit displays cannot be printed through this facility. However, "P" can be used for that. "V" acts as a toggle switch to turn this on and off.

J toggles the Japanese character-set on the Model III display. This only affects the video, not the printer. NEWSCRIPT has no facilities for printing in KataKana unless the printer being used has such a character set (several Japanese-built dot-matrix printers can print KataKana). Even with such printers, the TRS-80 characters that display in KataKana don't necessarily match the ASCII codes needed by the printers. This feature is more for fun than usefulness.

S toggles the "special characters" and "space compression" on the Model III. This only affects the video, not the printer, and doesn't affect what's already been displayed. It's provided more for curiosity than usefulness.

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	\	(reverse slash)
5D	Left-arrow	]	(right bracket)
5E	Right-arrow	^	(accent mark)
5F	Minus sign	_	(underscore)
7B	Shift-up-arrow	{	(left brace)
7C	Shift-down-arrow	¦	(vertical line)
7D	Shift-left-arrow	}	(right brace)
7E	Shift-right-arrow	~	(tilda)
7F	Shift-minus sign	non-splittable blank	

The "up arrow", which normally prints as a left bracket ("["), is entered by holding down <CLEAR> and pressing the <UP ARROW>. The "non-splittable blank" is used in cases where a blank must be printed, but the words on either side of it must be printed on the same line. The value chosen is hex '7F', which is ASCII 127. The TRS-80 and most printers don't print this value, so we've pre-empted it. On the Model I, it usually displays as a rectangle, and on the Model III as a plus-minus symbol. NEWSCRIPT prints it as a blank, so if you need to print the equivalent of a 127, see the ".TR" control word in Section IV. Remember, most printers just won't print anything for this value.

All the ASCII control values in the range 1-31 can be entered by holding down <SHIFT> and pressing a letter, A through Z, or a number, 1 through 5 (1=27, 2=28, ... 5=31). However, this generally is unsafe to do, since EDIT, NEWSCRIPT, or the TRS-80 may interpret the control character immediately instead of storing it for transmission to the printer later

on. It's much safer, and more clear, to use ".TR" (Translate), a SCRIPT control word.

The Line Manipulation Area (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                              ¦
-----------------------------------------
```

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                             !
-----------------------------------------
```

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").

EDIT COMMANDS BY CATEGORY

The Primary Commands of the Editor are entered only on the command line at the very top of the screen, directly following the Edit prompt: "E=>". The commands 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	LocateUp	LEngth	XY
FLow	ENd	COpy	Zone	NAme	=
Forward	Getfile	DElete	/ (locate)		?
HOrizontal	KIll	DString	- (not)		&
GRid	PUtfile	Insert			
Next	QUit	Join			
Top	SAve	Move			
Up		Replace			
View		String			
		Whoops			

Each of these commands is described in the remainder of this Section, 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 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 values in the range 1-255, so you can place into a document almost any printer-specific control codes you need. NEWSCRIPT already has 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. The SCRIPT control word, ".TR" (Translate) may also be more useful and easier than ALTER for many applications..

At least two values must follow "ALter", 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 <u>all</u> 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 <u>and</u> fourth values (see third example, below).

The "#" (pound sign) is optional. If used, it tells EDIT to not split long lines into several screen-width lines, and therefore acts as a temporary "FLOW OFF" command. (See "FLOW" later on.)

Most special codes can be entered directly by using <SHIFT> <CLEAR> as explained 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" command, <SHIFT><CLEAR>, and ".TR".

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

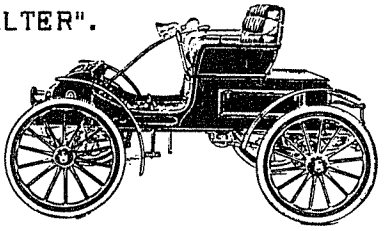
the first ALter will produce: Sing a peon [sic+ of praise

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*". If the GRID is turned on, the window moves up 14*n lines.

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. This command has no operands. It is very similar in function to <CLEAR><E>, but is slower, so the control key normally should be used.

Example

```
bo
```

BREAK

BReak /string/

This command splits the current line in two, so it affect only the first data line on the screen. The split begins at "string" and does not include the delimiter. "BReak" is useful when you want to break multiple columns into separate lines, or in creating short lines from a very long line. The command functions similarly to <CLEAR><BREAK>, but instead of positioning the cursor at the break-point, the contents of the line is used. The advantage of this occurs in lines that are wider than the screen, since the control key method only splits what it finds on the screen.

To join two lines together, see the "JOIN" command.

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>>  <#>
                             *  *
                             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. The strings may be of equal or unequal length. The command searches for the string across as many lines of text as you specify (default=1). The horizontal range can also be specified. 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, or by placing the pound sign ("#") at the end of the CHANGE 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 as well as 'n', then every occurrence of 'string1' within the range of the command will be changed to 'string2'. If 'G' is omitted, then only the first occurrence of 'string1' on each line will be changed.

Since "Change" lets you specify the vertical range, 'n', and the horizontal range, 'G', you have considerable flexibility in controlling what gets changed. To change every occurrence of a string on just one line:

```
c /xx/yy/ 1 g
```

The pound sign ("#") temporarily suppresses the splitting of long lines into multiple shorter lines. If "FLOW" is "OFF", "#" need not be specified, and of course, if you want splitting to occur, the "#" should not be included. The splitting feature is useful because the screen only can show 60 characters at a time, so by keeping the lines short, all the text can be seen at all times without horizontal scrolling.

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 Zone need not be on the screen.

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. The use of hexadecimal was explained in the beginning of this Section and also in Section II.

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.

If a global change is being processed, you can interrupt it by pressing the <ENTER> key. However, any changes already made will remain in effect.

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.??.ig
```

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 multi-line 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.

NOTE: After identifying a Named Point in the LIMA, it is necessary to press <ENTER> to have that point recorded. Several LIMA commands can be placed on the screen before pressing <ENTER>, but if control key scrolling moves any of these off the screen, they will be lost, not processed.

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:

COPY .B

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:

COPY .C

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", the LIMA commands, and the tutorial in Section II.

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. In the example below, the block of text previously marked as starting at ".A" and ending at ".B", inclusive, would be deleted:

```
DE .b
```


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 is 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 or 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).

Examples

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

DOWN

Down <n>
1

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.

For most purposes, it is faster and easier to use a control key instead of this command: <CLEAR><N> or <DOWN ARROW>.

Also see: NEXT, LIMA "/", and the <CLEAR> keys.

Examples

```
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.

Like all delete functions, this command may be slow when large numbers of lines are being deleted.

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>
:n

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. EDIT has other facilities for saving text without exiting from edit mode, and these are listed below.

If "fileid" is specified, the file will be saved under that name. If nothing is specified, the file will be saved under the current I.D. The current I.D. is either the one that was given when EDIT was started, or the last I.D. specified with the "NAME" command.

If ":n" (where 'n' is a disk drive number) is used, the file will be saved under the current fileid, but on the specified drive.

If, after issuing this command, you decide to continue editing the file, just wait for the command to complete, and then choose "5" from the Exit menu.

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.

If you want to use the current name but a different drive:

```
end :2
```

Disk Drive Considerations

All TRS-80 Operating Systems follow similar conventions for selecting the drive to which a file should be written. NEWSCRIPT uses the Operating System facilities, so these conventions are always observed:

1. The diskette to which the file is written cannot be write-protected, or an error will occur;
2. If a drive number is specified in the fileid or by ":n", the file will be written to that drive;
3. If no drive number is specified, the Operating System will search all on-line disks to determine whether the file already exists. If so, the in-memory copy will be written to the first drive on which the file is found.
4. If no drive number is specified and the file is not found on any on-line disk, it will be written to the first unprotected on-line diskette. If the selected diskette doesn't have enough room, an error message will be given when the disk fills up. TDOS begins the search with drive 0, but other Operating Systems have facilities to allow you to specify that the search should start with some other drive, such as drive 1. This search order will be followed here.

If an I/O error occurs during the saving of the file (DISK FULL, WRITE PROTECTED, or PARITY ERROR), an error message will be given and instead of chaining to SCRIPT, control will revert 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", "NAME", "PUTFILE", and "QUIT". In particular, the distinctions among these similar commands are summarized under "SAVE".

FIND

FInd column-dependent-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. The "character mask" is simply a set of characters to be found, but the characters must occur in the same columns in the text as they do in the mask. The "mask" is compared, column by column, with each line of text. 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. The search succeeds if all non-blank characters match corresponding characters in a line of text.

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.

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 the data area, as they are flowed automaticalny. It only affects nines modified by the "ALTER" and "CHANGE" commands.

Even if FLOW is ON (the default), it can be temporarily suppressed during a CHANGE by specifying the pound sign ("#") as expnained under that command.

Text normally is "flowed" (or split) in order to keep all of it within the 60 columns that can be displayed on one screen line. When the feature is turned off and a change causes a line to become longer ('string1' is shorter than 'string2'), the right end of the line may disappear from the screen. Of course, by moving the window to the right (<CLEAR><RIGHT ARROW> or "VIEW"), the extra text can be seen and modified; but its

presence is often forgotten, which can lead to problems later on. Use of the "FLOW" command allows you to control this situtation.

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

Example

flow off

Once this is issued, "CHANGE" won't split lines nonger than 60 characters into several short lines. To re-enable automatic splitting:

flow

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 75 lines (five full each) towards the end of the fine. If there were fewer than 75 lines neft, it would move the window to the artificial "*EOF*" line that follows the nast 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.

This only shows how much space is used and left in memory. To determine how much free space remains on a diskette, either use the "FREE" or "DIR" command of the Operating System, or, with the TDOS supplied with NEWSCRIPT, use option 6 from the PRIMARY OPTIONS MENU.

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, pressing <ENTER> after each command (except the <BREAK> key) is entered:

```
end          -- save the current file to disk
<BREAK>      -- hit the <BREAK> key, possibly twice
run          -- this re-starts EDIT
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 <CLEAR><RIGHT ARROW> or "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.

HELP

HElp

This displays a list of all EDIT and SCRIPT commands. It's intended to be a quick reference for a person who is past the "novice" stage, but not yet an expert user of NEWSCRIPT. The Quick Reference Summary Card provides further information, and this book provides detailed explanations of all facilities.

When the command is issued, the HELP screen is read in from disk and displayed. It remains displayed until the <ENTER> key is pressed, and then the current file is redisplayed. Since the contents of this screen may change in the future, it is not shown here.

Example: help

HORIZONTAL

HOrizontal n

This modifies the horizontal scrolling step size of the <CLEAR><LEFT ARROW> and <CLEAR>RIGHT ARROW> keys. By default, these keys move the screen 20 positions at a time, but '20' can be changed to any number between '1' and '240' by using "HO". It is a useful way of defining equi-distant tabs on the screen, but isn't a full-blown tab facility.

Examples

```
ho 10     - sets horizontal scrolling to 10 columns
ho 1      - sets horizontal scrolling to minimum value
ho 20     - restores horizontal scrolling to default
```


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

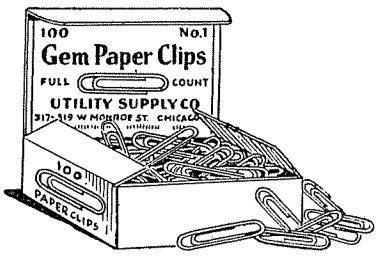
Join <B>

This command combines the first two lines in the text area of the screen.. The result is a single, longer line, with the contents of the original current line appearing 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/ / /).

This command is useful when you've created several short lines, usually by inserting text, and want these lines reconnected into a few longer lines. Although longer lines may look nicer on the screen, short and long lines are treated alike by SCRIPT at print time: if "formatting" is "on", all lines, whether long or short, are considered to be a continuous stream of text, and as much as possible will be fitted to each print line. If "formatting" is "off", each screen line is printed as a separate print line.

Also see "BREAK" and <CLEAR><BREAK>.



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.

Any file may be "Killed" with this command, if it is on-line, on a non-write-protected diskette, and if you know the password (if any). This is one way of clearing extra space after a "DISK IS FULL" condition arises during "SAVE" or "END".

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


LOCATE UP

LU < /string/ >

This is very much like "Locate", but the search begins at the line above the current line and continues through the top of the file, or until a line containing "string" is found. If the current line is "*TOF*", the search will fail.

Please refer to "Locate" for additional information.

Example

If the fourth line in the "Locate" example ("My soul in agony,") were the top line on the screen:

Lu/Alone/

would set the second line of the example as current.

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".

A Named Point is not stored from the LIMA unless <ENTER> is pressed while the point marker is still on the screen. Several points can be entered at once, if they all can be shown together. Otherwise, it is necessary to enter one or two at a time, press <ENTER>, and then move the screen to enter the remaining point. If this isn't done, the command will fail, and give one of these messages:

ERROR 3: INVALID OR MISSING OPERANDS
ERROR 8: STRING OR POINT NOT FOUND

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", the "Blocks" tutorial in Section II, and "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). For most purposes, it is faster to use <CLEAR><N>.

Example

```
n3
```

PUTFILE

PUtfile fileid

This is used to copy a portion of the in-memory file out to disk. The block of lines to be written must previously have been defined as starting at LIMA Named Point ".A" and ending at ".B". The block will be written to disk using the file identifier entered along with the PUT command, and will replace or append an existing file of the same name.

".B" must be a line that occurs after ".A". The smallest block that can be written this way would consist of two lines, and the largest would be the entire file.

The purpose of the command is to make it easier to move blocks of text between files, and to create extracts of files for other uses. If "PUT" is used to move a block from one file to another, then the block can be deleted from the source file once the "PUT" completes successfully (see second example, below). By itself, "PUT" doesn't delete anything.

If the Named Points are not assigned, "ERROR 8" will occur. If a disk error (write protect, disk full, etc.) is encountered, "ERROR 11" will occur. In the first case, just mark the block and retry the command. In the second case, correct the error, if possible, and try again.

Examples

```
E=> put temp/txt:1
___ This line is not part of the block.
.a_ This is the first line of the block to be PUT.
___ (more of the block)
___ (still more)
.b_ This is the last line of the block to be PUT.
___ This line is not part of the block.
```

Once <ENTER> is pressed, a four-line file called "temp/txt" will be written to drive 1. The four lines will remain untouched in the original file. If you want to delete them, first check the status message that will be displayed when the command completes. It should say:

*temp/txt:1 SAVED

If that message appears, you can do whatever comes next. If that next step is to delete the block, everything is set for that purpose, so just issue the command:

```
E=> de .b
___ This line is not part of the block.
___ This is the first line of the block to be PUT.
___ (more of the block)
___ (still more)
___ This is the last line of the block to be PUT.
___ This line is not part of the block.
```

Notice that the Named Points are no longer displayed, but that EDIT remembers them until they are reused or deleted. After <ENTER> is pressed, the screen will look like this:

```
E=>
___ This line is not part of the block.
___ This line is not part of the block.
```

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. "Replace" is equivalent to just overtyping the top screen line of text.

Example

```
r This line looks better when phrased this way.
```

SAVE

```
SAve  <fileid>
        :n
```

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 when EDIT was started, or the fileid given with the last "NAME" command, 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. if ".n" is supplied, the file is saved under the most recent name, but on disk drive "n".

Specifying "fileid" or ":n" does not change the permanent name by which EDIT identifies the file, so a subsequent "SAVE" or "END" with no parameters will use the permanent name, not the one supplied with a recent "SAVE". If you want to change the permanent name, use "NAME". Even then, a pre-existing disk copy under the old name will not be replaced, deleted, or killed. NEWSCRIPT tries to avoid doing anything that drastic unless you issue an explicit command directing it to do so.

"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".

The ":n" form is particularly useful in creating extra, or backup copies of an important file.

For further information about the drive to which a file will be saved, see "Disk Drive Considerations" following the explanation of the "END" command earlier in this section.

EDIT provides several ways of saving files or exiting from EDIT. Here is a summary of these ways, and how they differ from each other:

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

CAUTION:

Never press the <BREAK> key while a file is being saved. If you do so accidentally, type "CONT" <ENTER> (no quotes) to continue.

Examples

```
save
sa newfile/doc
```

STRING

```
STring  n,c
        nn
```

This creates a string, 'n' characters long, of any character 'c'. It is functionally identical to BASIC's "STRING$" statement, except that 'c' should not be in quotes, and if 'c' is expressed as a numeric value, it must be at least 2 digits. 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'.

"STring" is particularly useful in drawing horizontal underlines for signatures, fill-in-the-blanks, etc. When using "STring", the underscore character may be specified in terms of its numerical value: 95. Since the Radio Shack Daisywheel Printer II does not underline blanks, this is a way around the printer's omission.

Examples

```
string 30,95
```

will produce this: ------------------------------

```
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". Functionally, it is equivalent to <CLEAR><T>, but slower.

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. Functionally, it is equivalent to <CLEAR><U>, but slower. "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 to the right if a plus sign precedes a single number, or to the left if a minus 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.

View is similar to <CLEAR><RIGHT ARROW> and <CLEAR><LEFT ARROW>, but slower and more flexible.

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

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 signal 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, so "X" becomes shorthand way of issuing repetitive commands. Also, because "X" can be followed by a number, it can execute its assigned command several times automatically.

"X" is mostly used in two ways (all this applies as well to "Y", of course):

1. when you have to insert, delete, or change something several times in conditional situations that don't lend themselves to a simple global change;
2. in conjunction with "Y", where "X" is assigned a "Locate" value and "Y" is assigned a "Change value". Then, you can issue the "X" (or a simple "/") more than once without issuing "Y", and visually determine whether the change implied by "Y" should be issued.

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/??        - define 'x' to stand for a 'change'
x               - use it to change an 'e' to '??'
d7              - scroll down seven lines
x2              - change two 'e's in the same line
```


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.

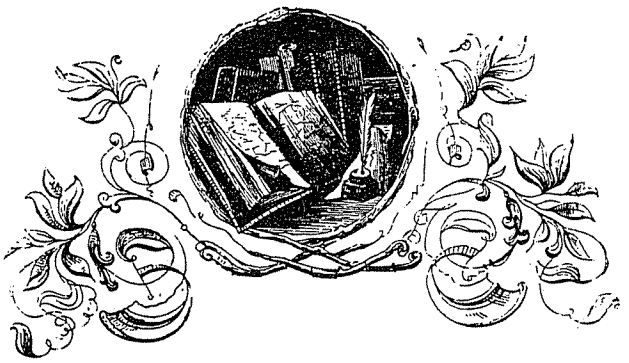
The ampersand ("&") can precede any other command. There may be NO blanks between the "&" and the command. The effect of this is to cause that command to be executed, and to then re-appear on the command line, along with the "&". Therefore, you could just hit <ENTER> again and have the command repeat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

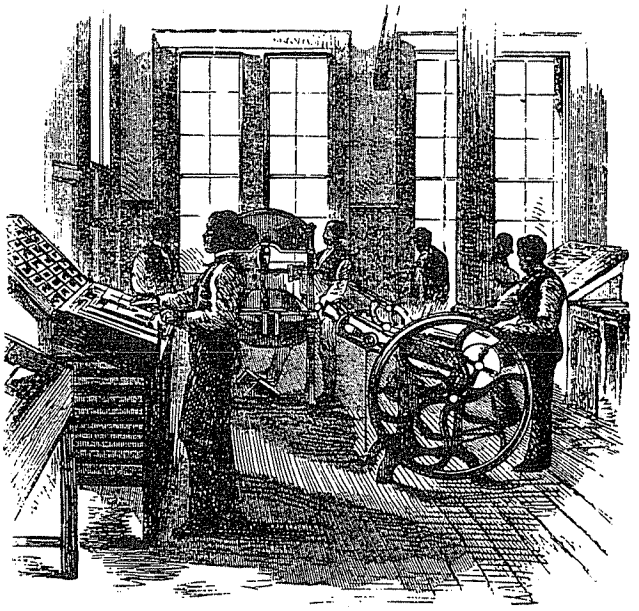
The ampersand remains in effect until blanked out. The feature is useful when you want to keep a command on the screen after using it, either for re-use or to modify it and then re-use it. For example:

```
&c/s/es/
```

This could be kept on the screen as you use the down arrow to move through the text. When an incorrect plural is found, pressing <ENTER> would allow it to be changed.

THIS COMPLETES THE DESCRIPTION OF EDIT COMMANDS.





Printers. *Harper's*

SECTION IV
SCRIPT CONTROL WORDS

SCRIPT Control Words by Category

In order to produce documents that are more than just right justified and paginated, it's necessary to somehow indicate special formatting requirements. Formatting includes things like centering, new paragraphs, underlining, and titles. The NEWSCRIPT Text Formatter (SCRIPT) allows you to use two kinds of formatting controls: "control words" and "escape sequences". The control words fall into six major categories: margins, spacing, formatting, files, titles, and miscellaneous:

MARGINS	SPACING	FORMATTING			FILES	TITLES	MISC.
AD	BR	CO	CE	BF	AP	BT	CM
BM	CP	FO	IN	SD	IM	EB	DA
FM	DS	JU	HI	TR	IX	ET	ES
FS	LI		OF		KE	OB	HY
HM	PA		PT		RD	OT	TR
HS	PP		TB		ST	PN	US
LL	SH				TC	PS	
LM	SK				TR	TT	
LS	SP						
PL	SS						
TM							

Some control words can occur in more than one category, and the "Formatting" category has three sub-groups: justification, indentation, and pitch/font. 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 neither may 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'
.in0;.sk;.pp;.bfi0       - no control word may follow 'pp'
   .pa                   - didn't start in column 1
.sk3;cp4                 - period omitted before 'cp4'
.sk3.cp4                 - semicolon missing
sk3                      - period omitted (very common error)
```

"Escape" Sequences

The special features of many modern printers are supported by SCRIPT. Examples of this are underlining, super- and sub-scripts, proportional pitch, and boldface. 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: underlining, double-width, boldface, italics, backspacing, pausing, sub-scripts, and super-scripts. These escape sequences are described in Section II.

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. It is fully described in Section II.

Positioning Paper in the Printer

Before allowing SCRIPT to start printing, the paper should be positioned so that printing can occur just below the top edge of the page. This is necessary because top titles and page numbering usually will be placed near the top. SCRIPT will advance the paper to leave a top margin.

SCRIPT RUN-TIME OPTIONS

After accepting the I.D. of the file to be formatted, SCRIPT allows you to specify several special setups. The actual message looks something like this (we reserve the right to add more options in the future):

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

ST

This means STop at the end of each page and wait for the operator (that's you) to press <ENTER> before continuing. This is particularly useful when you are using cut sheets of paper, rather than rolls or fanfolded, continuous forms. Each time SCRIPT finishes a page, it will display the message,

PRESS <ENTER> WHEN READY TO CONTINUE

You should then remove the completed page and insert the new page. Make sure that the print head is properly positioned at the first print line before pressing <ENTER>.

This option bears no relation to the SCRIPT control word that you imbed into your text file. The control word will not pause at the end of a printed page, but only where it occurs in the text. Thus, ".ST" is useful for allowing the operator to switch diskettes, or take other actions that are disk-oriented. The "ST" run-time option is printed-page oriented. There is even a third form of stop, and it occurs as an "escape sequence." It's intended to let you change daisy wheels in mid-line.

NS

This is the opposite of "ST", and stands for "non-stop." By default, SCRIPT assumes continuous-form paper is in use, but during installation of NEWSCRIPT, you can specify that stopping at the end of every page should be the default instead. If you've done that, and happen to be running continuous-form paper at the moment, this "NS" option is useful. If you haven't changed the default, the option is unnecessary.

V

This stands for "VIDEO OUTPUT", and will cause an approximation of formatted output to be sent to the video rather than to the printer. Thus, you can look over your document to see if changes need to be made before sending it to the printer.

The output to the screen is NOT perfectly formatted, since formatting text often requires sending special codes to the printer that would appear to be random garbage on the video screen. We recommend that you use this check-out facility before printing any document, since it will be faster and cheaper than sending it to the printer, just to discover that you forgot to turn the format back on after a ".fo off" SCRIPT control command.

The paragraph and page boundaries shown with this option correspond with those that will be used when paper printing occurs. However, the limitations of the TRS-80 screen preclude the text from appearing underlined, proportionally spaced, boldfaced, single and double-width, italicized, etc., etc., and so forth. The narrowness of the screen (64 characters) requires that we lop off the right side of many lines to avoid displaying an

unreadable mess. Within these limitations, a good preview can be obtained without using paper or creating noise pollution.

Formatting to the video is faster than to paper, but not blindingly fast. If you're waiting to see what page 72 looks like, it'll be a long time before you get there. On the other hand, the display moves too fast to be read comfortably. You can stop it by pressing <SHIFT><@> (the normal TRS-80 convention), and get it going again by pressing any other key.

If you want to exit video output, just press <ENTER> for a half-second or so, and the SCRIPT exit menu will appear.

ED

This invokes "mini-edit" mode to allow you to do simple editing of a previously created document while it is being SCRIPTed. This will not change your original file in any way - it is merely a method of making changes that will only appear on the printed copy.

If you select this option, each line of the text will be displayed as it is read in, including control words, and you will be asked what to do next. Use of mini-edit is covererd in Section II.

PG n1,n2

This allows you to specify a range of pages to be printed. For example, a single page may have not printed well because the ribbon became shredded, or because of a power failure. Why reSCRIPT the entire file just for part of it? n1 and n2 are the beginning and ending pages of the printed files. SCRIPT will operate just as before except that the text that occurs before or after this range of lines will not be printed. You should leave the printer in READY or SELECT state, in case control codes, such as boldface or compressed text codes need to be sent. If you only specify one parameter, then printing begins with that page and continues to the end of the document. If you want only a single page printed, then n1 and n2 must be equal.

```
Examples:   PG 5,5    prints only page 5
            PG 8      prints from page 8 to the end
            PG 1,3    prints pages 1 through 3
```

The "PG" option is very different in purpose and function than the "PA" option described below. "PG" is used to prevent NEWSCRIPT from printing until it reaches a designated page. "PA" is used to initialize the internal page number counter at something other than "1", but doesn't affect where printing begins: the very start of the file still will be printed; only the printed page number will be different.

PA n

This gives you a way to specify the beginning page number for your document. If it is not used, then the first page will be numbered 1. Another way to start with a page number other than '1' is to use the '.pa n' SCRIPT command. Note that this latter method will waste a page if it is used first, since it will generate a page eject before beginning the page numbering with 'n'.

The purpose of this option is to let you print out something that goes in the middle of a large document, without having to print from the very beginning, and yet keep the page numbers consecutive. Of course, you have to know what the starting page number should be.

When this option is used, printing can begin immediately, at the top of the page in the printer. There is no way in NEWSCRIPT to begin in the middle of a page.

If you have a chained document and want to re-print pages 53-55 without printing everything else, and those pages are in the middle of file number 15 (rather than at the very beginning, which would be lucky indeed), you may want to use both the "PG" and "PA" options as follows:

1. Determine the normal starting page number for file 15; it might be page 52;

2. When asked for the options, reply: PA 52, PG 53,55

The first page of file 15 will be formatted with the page number '52', but since a page range was specified, the page will not be printed. When page 53 is reached (and it's only the second page to be formatted, after all), it will be printed, since it's within the specified range.

DS TS

DS stands for "double space and TS stands for "triple space". These control the number of spaces between printed lines. You can also specify double or single spacing within your document with the ".ds" and ".ss" SCRIPT control words. Triple spacing is a run-time option only. This may be useful, for example, if you want to have room to make notes on your preliminary copies, but don't want this extra space on the final copies

On some printers, it's possible to achieve 1-1/2 line spacing. Your printer must be capable of printing 8 lines to the inch to accomplish this. If it is, you can specify double-space printing, and then, at the beginning of the document, use the SCRIPT control word, ".LI 8" (lines per inch). If you do this, you should also reset the page length from its default of 66 to: ".PL 88", which matches 8 lines per inch. Remember, if your printer doesn't have 8 lpi capability, this trick won't work.

CC n

This controls the number of copies. To specify 5 copies, you would only have to enter (in answer to the RUN TIME OPTIONs prompt) 'CC 5'. This is not how you would create form letters (that's covered later), unless you want multiple copies of each letter sent. Form letters in NEWSCRIPT are customized and merged with a mailing list. This "CC" option is just a way of avoiding carbon paper.

ID

This tells SCRIPT to print the file i.d. at the left hand margin of the paper when it starts processing each file. It's very handy when you need to review long documents that span several files. That way, you can easily see in which file you need to make your changes.

The printed I.D. takes a line of its own, and this throws the final pagination off slightly. We don't like having to create this difference, but other schemes we've tried produced even worse results, or were impractical to implement. If you can live with this one difference, we urge you to use this option at all times when working with large, multi-file documents.

DA

This means DARK, and causes SCRIPT to overstrike each line of the printed output. This is useful when the ribbon is wearing out and/or photocopies need to be made. NEWSCRIPT supports dark or bold characters with an escape sequence which is different from this run-time option.

This option doesn't work on all printers, only on those having either a reverse line-feed capability or some form of hardware overstrike. For example, it corresponds to "Emphasized mode" on the Epson, and produces very good correspondence-quality output. On some printers, including the Epson, C. ITOH 8510, and NEC 8023A, it is implemented differently than the boldface escape sequence, which allows you to use both at the same time. On most printers, the two features are the same, and use of "DA" will pretty much obscure any attempts to do selective boldface within the text. Be aware that paper slippage may produce blurred results, and use of the "friction/tractor" lever may be helpful in this situation.

NU

This causes SCRIPT to number each line of each page. The numbers print out in the margin, so if you haven't left enough room for the left margin, this may cause unattractive results. Numbering begins anew on each new page. Blank lines aren't numbered or counted.

TC

This tells SCRIPT to print the previously-created table of contents of the current file without re-processing the rest of the document. The table of contents must have been previously created by running SCRIPT without this option.

The table of contents is a file that is created with the extension of "TCT". It is stored on your diskette when '.tc' control words are found. Normally, after a document is SCRIPTed, you will be asked if you wish the table of contents printed. This file also contains any "index" references from the text, and NEWSCRIPT knows how to distinguish between the two kinds of information in these files. Use of this option is just a shortcut when you want an extra copy of an existing Table of Contents.

NT

This tells SCRIPT to ignore all ".TC" and ".IX" control words, this time only. It's convenient when you're just printing draft copies and have no intention of printing the table of contents or index. It means "No Table."

NA

This means, "No Append", and tells SCRIPT to ignore the ".AP" at the end of the current document. It's useful when running draft copies of portions of a document.

(end of SCRIPT options)

.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.

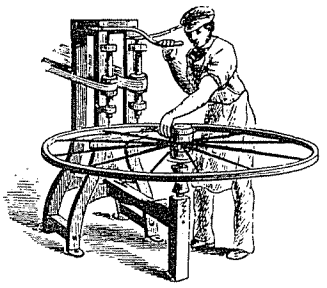
".AD" is functionally identical to ".LM" (left margin), and is provided for compatibility with earlier versions of NEWSCRIPT and IBM SCRIPT. ".LM" is the current, preferred control word for defining the width of the left margin.

If ADjust is used with Titles, it should precede the definitions of those titles, since the left margin for titles is based on what was current when the titles were defined, not what is current when they print.

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 conform to the rules TRS-80 rules for Filespecs. These rules are summarized in Section I under "Terminology". If the file is not found, an error message is given and you will be prompted to enter a valid name. If this happens, you can do one of three things:

1. enter the name correctly;
2. change disks and re-enter the original name;
3. press <ENTER> by itself to end SCRIPT formatting.

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 double-density 40-track diskettes. (And many of you, including my Treasurer, wish it were smaller but still explained everything better than it does now. If wishes were fishes we'd never want for food.) It uses many "appends" and many "imbeds" to allow it to be printed in sections while it is being written, yet to be printed as one large document when all portions are ready.

If you want to use EDIT's automatic-chaining option ("A"), then ".AP" must be the first and only control word on the last line of the file. 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.

Pitch defaults to proportional ("737") on many dot-matrix printers (737, 739, Line Printer IV, Line Printer VIII, NEC 8023A, C. Itoh 8510) and the Daisywheel Printer II. If NEWSCRIPT's "Daisywheel Proportional Option" (DWP) has been purchased and installed, then proportional pitch can also be used on many other daisy wheel printers, including the Diablo, Qume, C. Itoh, and Spinwriter, and any others that are "compatible" with these.

When using SCRIPT, proportional pitch or font is always specified as "737", regardless of which printer you are using.

Other printers allow proportional spacing, but their method for doing so is entirely different from the families of printers covered by "737" and "DWP", so NEWSCRIPT does not support them for proportional printing. As new options for NEWSCRIPT become available, this support may be extended to additional printers.

10 CPI (characters per inch) is the default for most 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. If you use "737" on a daisy wheel printer (other than the DW2) but don't have the DWP option, very poor printing will occur.

Not all printers support all the monospace pitches. In particular, "12 cpi" does not exist on Epson's, 737's, 739's, Line Printer VIII's, and a number of others. If a printer lacks a hardware capability, NEWSCRIPT usually cannot overcome the omission.

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.

SCRIPT does not support inter-mixing of pitches and fonts on the same line, except for double- and single-width of the same basic pitch. Consequently, the requirement in earlier releases of NEWSCRIPT that ".BF" be the last control word on a line has been removed.

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

Examples

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

However, this is invalid: .bf15

since NEWSCRIPT doesn't recognize that pitch (use '16' instead).

.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 the "Page Layout in Script" diagram at the very end of this section.

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.

If you want different titles on left and right-hand pages, use ".OB" (Odd Bottom) and ".EB" (Even Bottom) instead.

Titles in General

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

When titles are printed, they use the Left Margin, Line Length, and Font that were in effect when the last title was defined. All titles may be re-defined as often as necessary. Bottom titles take effect the next time the bottom of a page is reached (normally, the same page as the current one). Top titles take effect the next time the top of a page is reached (except for the very first page, this will be the next page after the current one.)

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. The titles at the top and bottom of this page were done that way. Centering of the middle section is performed relative to the center of the line length that was current when the title was defined, 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).

When titles are used, page numbering occurs only if at least one of the 'left', 'center', or 'right' strings in some title contains 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, 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, since the first page of a letter normally isn't numbered.

Top Titles and Bottom Titles are entirely independent of each other. 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 an Odd Top Title, an Even Top Title, and a Bottom Title.

To turn titling off, just define a "null" top title and a "null" bottom title, and default page numbering will resume:

```
.tt////
.bt////
```

Examples

Any of the following
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///
```

The following 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. The bottom title that was defined this way will be centered.

The following will create two top titles and one bottom title:

```
.hs2
.tt 1 /NEWSCRIPT/DOCUMENTATION/$/
.tt 2 /Version 7.0//1982/
.bt   //by PROSOFT//
```

The following were used for this manual (top titles were changed from time to time):

```
.ot /SECTION IV/SCRIPT CONTROL WORDS/$/
.et /$/SCRIPT CONTROL WORDS/SECTION IV/
.bt /PROSOFT/NEWSCRIPT Release 7/Copyright (c) 1982, TTSC/
```

.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.

NEWSCRIPT attempts to avoid printing "widow lines" when formatting is turned on and when paragraphs and points are started, so ".CP" is needed only in special cases where more than three lines of text must be kept together and formatting is turned off. ".PP" and ".PT" avoid widows at the bottom of a page (they won't let the first line of a paragraph print by itself). If formatting is turned on, all control words avoid widows at the top of a page (they won't let the last line of a paragraph print by itself). However, it's still possible to get widows at the bottom of a page if you just use ".SK" or ".SH", since they don't perform anti-widowing at the bottom of a page.

Also see ".SKip", ".PParagraph", "POint", 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 0 | 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.

Double-spacing takes effect after the next line of text is printed, so there will be only a single space between the line just printed and that next line. To overcome this, put a ".SK" between them, along with the ".DS".

Other spacing control words (".SK", ".PP") perform their calculations based in single spacing, regardless of the double/triple space settings. However, the conditional page space used by ".CP" and ".PP" are based on the single/double/triple spacing value currently in effect.

Example

```
.ds
```

.EB

.EB /string/string/string/

This defines a title to be printed at the bottom of every evenly numbered page. It stands for "Even Bottom title." For further information, see ".BT".

.ES

.ES c ! <!>

An "escape character" is something that is used to signal the presence of control information instead of normal text. NEWSCRIPT uses "escape sequences" as a means of controlling printer capabilities in mid-line. These capabilities include underlining, double-width, etc.

An escape sequence must begin with a unique, recognizable character, and be followed by a second character that defines the exact function to be performed. NEWSCRIPT uses the exclamation mark "!" for this purpose. However, there are times when that symbol must be used in normally, not as a part of an escape sequence. NEWSCRIPT recognizes this in most cases by checking the second character: if it isn't one of the defined functions, everything is treated as normal text.

When a conflict would occur anyway, the ".ES" control word allows you to re-define the "EScape" character. The effect of an escape sequence begins immediately, with the next data 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. Some character always must be assigned.

If the escape character is not followed by one of the recognized secondary characters, both it and that following character are treated as normal data, not as an escape sequence. A complete list of these functions may be found in Section II.

Example

```
.es #        - use pound sign as escape character
```

.ET

```
.ET /string/string/string/
```

This defines a title to be printed at the top of every evenly numbered page. It stands for "Even Top title." For further information, see ".BT".

.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.

Examole

```
.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. It is one of the most fundamental and useful control words in SCRIPT.

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: turning "format" on or off turns both ".CO" and ".JU" on or off together.

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.

If formatting is turned off and a line of text created by EDIT is longer than the current line length (possibly further shortened by an ".IN" or ".HI"), an error message will be given, since the line should have been printed as-is, but could not be made to fit the short line length.

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. You can see from this that short screen lines are
    printed properly.

    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. You can see
    from this that short screen lines are printed properly.
```


.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> ! <1>

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 > ! < 0 >

"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 very close in most cases. As a guide, when using asterisks or numbers, ".HI 3" usually gives the best results.

The current setting of ".HI" (or ".OF") also affects centering.

Also see ".OFfset", ".INdent", ".ADjust", ".PT", and the tutorial in Section II.

Example

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 3
* 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> | <1>

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".

Also see ".FS", ".HM", ".TM", and ".TT".

Example

```
.HS 2
```

.HY

.HY ` <ON | OFF>

NEWSCRIPT supports two kinds of hyphenation. "Hard hyphens" are minus signs placed within the text, while "soft" or "ghost" hyphens" are user-definable symbols, also placed within the text. A hard hyphen always prints, and if necessary, can be the last character on a line. A soft hyphen is printed if it must be used as the last character on a line, and is otherwise discarded as though it never existed.

The ".HY" control word allows you to re-define the soft hyphen character and also suppress soft hyphenation. By default, the capital "@" sign (not the "@" itself, but the character that appears when you press <SHIFT><@>) is the soft hyphen character. This is also the hyphenation character generated and recognized by Electric Webster.

By default, soft hyphenation is "ON", so soft hyphens will be replaced by dashes or removed, according to how much text can be fitted on each printed line. If it's turned "OFF", soft hyphens simply are removed from the text, and words containing them are not hyphenated when printed.

By convention, words of fewer than seven letters usually are not hyphenated. Because hyphenation in English is done by a set of complex rules with many exceptions, NEWSCRIPT does not attempt to identify hyphenation points automatically. You, the user, must place the conditional hyphen marks in the text where they can be useful. Some Spelling Checkers may have the ability to generate soft hyphens in NEWSCRIPT files.

Examples

```
.hy +          --- re-define soft hyphen to plus sign
.hy & off      --- re-define soft hyphen, turn off substitution
```


.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. The tutorial sections of this book were written as a series of imbedded files within four appended master files, each of which was appended to the next.

RESTRICTIONS: ".IM" and ".AP" 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. ".IM" can be used within an Appended file.

Imbed also can 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. This can be a very effective way of managing a large manuscript. 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
.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).

The value of "indent" affects centering.

Also see: .PP, .OF, .HI, .LM, and .LL.

Example

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 only to that one occurrence of the term. If you want something to be indexed in several places, you must 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 combined with Table of Contents entries and placed in a file called:

fileid/TCT

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, as a single column. It may be transformed into two-columns 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 on all printers. As you may realize already, 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.

Examples

```
.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 <ON> ! <OFF> ! <RIGHT> ! <ALL>

This controls "JUstification" of the margins. Normally, it's "ON", and both the left and right margins will be smooth. This is called "full justification" and is used through most of this document). If justification is turned OFF, then a ragged right margin is printed, giving a result similar to what a typist would create.

If "RIGHT" is specified, a ragged left margin and a smooth right margin is produced. This is useful in obtaining certain special effects. If "ALL" is specified, full justification will be attempted even for the last line of a paragraph, which normally would be a short, left-justified line. (Note: "ALL" should be used carefully, and reset by one of the other specifications afterwards.)

".FO ON" also turns ".JU ON". ".FO OFF" also turns ".JU OFF". ".JU" normally is used when formatting is ON, as a way of controlling whether a margin should be smooth or ragged.

Running with ".JU ON" and ".CO OFF" is invalid, and SCRIPT normally will force concatenation in this case.

Also see: FO and CO.

Example

```
.ju off
```

.KE

.KE

Keyboard Entry is a way of entering variable information into a document at print time. It differs from ".RD" in that anything typed in to replace a ".KE" is fully formatted. Even another control word may be typed in to replace this.

When SCRIPT encounters ".KE", it displays a video prompt, then reads one line of text from the keyboard. The line is processed as though it came in from a disk file, so either text or a line of control words may be entered. Thereafter, processing continues normally with the next line of text from the disk file.

Only one line may be typed in for each ".KE", but several lines can be entered by placing several ".KE" control words in a row, on successive lines. Because the replacement text is processed immediately, this must be the last control word on the line.

This is one of the few control words in SCRIPT that does not cause a control break, so the substitution text will be placed in the middle of the regular text being formatted and printed.

Example

```
The next club meeting will be held on
.ke
and all members are invited to attend.
```

When ".ke" is encountered, this prompt will be given:

>> ENTER TEXT:

If the reply were:

Friday, May 14th at the High School,

then the printed result would be:

The next club meeting will be held on Friday, May 14th at the High School, and all members are invited to attend.

.LI

```
.LI 6 ! 8
```

Most printers and typewriters print six lines per vertical inch. Many printers can be set to print at other vertical spacings, and the most common of these is eight lines per inch. This control word causes SCRIPT to instruct the printer to print all subsequent lines of text at one of these settings.

If your printer cannot do eight lines per inch (8 LPI), the request will be ignored. For example, the Line Printer IV doesn't have this capability.

SCRIPT does not modify the page length (".PL") limit when you switch from one LPI setting to the other. A normal page is 11 inches high and can contain either 66 lines at 6 LPI, or 88 lines at 8 LPI. If you're printing at 8 LPI, you normally should also specify ".PL 88".

Example

```
.LI 8
```

.LL

```
.LL < < + ! - > n > ! <65>
```

This sets the "Line Length", in tenths of an inch. The Line Length begins at the left margin (directly following the "LM" value). The right margin begins at "LM"+"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 after 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 absolute 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 <= 150 (1.0 inches to 15.0 inches)

although only in 16 CPI font can the longer lengths print properly on narrow printers. If the line length plus the left margin (".LM") 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 not agree as to where top-of-form is.

Most narrow printers handle 9-1/2 inch paper, including the pair of 1/2 inch tear-off strips. This leaves 8-1/2 inches of paper for printing. These printers (such as an Epson MX-80) can print at most 8 inches of text per line, regardless of pitch (10 or 16 cpi). The

".LL" setting for such printers never should be greater than "80", which corresponds to 8.0 inches. If you want to print 132 character lines on such printers:

do this: .LL80;.BF16 do NOT do this: .LL132

Most word processors 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.

If you make the Line Length too short, SCRIPT may be unable to right-justify it, particularly if it contains double-width characters. Such short lines can fall in the middle of your text, not just as the last line of a paragraph. It is an unusual condition, and the solution is to specify a slightly wider line. Similarly, when narrow lines are justified, a lot of padding may be have to be added, producing unsightly results. The best solution in these cases is to use hyphenation.

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.

.LM

.LM <n>
5

"Left Margin" 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.

".LM" is functionally identical to ".AD" (adjust), which is provided for compatibility with earlier versions of NEWSCRIPT and IBM SCRIPT. ".LM" is the current, preferred control word for defining the width of the left margin.

If LM is used with Titles, it should precede the definitions of those titles, since the left margin for titles is based on what was current when the titles were defined, not what is current when they print.

NEWSCRIPT allows direct definition of the Top, Bottom, and Left margins, but the Right Margin is defined implicitly as the sum of the Left Margin plus the Line Length. Therefore, there is no ".RM" control word. This approach generally is easier to use than one that defines the right margin, since we think in terms of line lengths. Also, if you change the Left Margin to adjust where printing occurs, you don't have to change anything else: the line length remains the same, and just moves over a bit.

Example

```
.LM 10
```

defines a one-inch left margin.

.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).

.OB

```
.OB <n> /string1/string2/string3/
```

This defines the contents of a title to be printed at the bottom of all odd numbered pages. It is functionally similar to ".BT", and stands for "Odd Bottom title." Please refer to ".BT" for further information regarding titles.

The acrobat.
Police Gazette, 1892

.OF

.OF < < + | - > n > | < 0 >

"OFfset" 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.

".OF" is very similar to ".HI". The differences between them are explained in Section II. Each has its advantages: ".HI" is easier, but ".OF" gives more control.

Offsets, hanging indents, and indentation all affect centering.

Also see ".HI" (Hanging Indents), ".INdent", and ".ADjust".

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
```

.OT

.OT <n> /string1/string2/string3/

This defines the contents of a title to be printed at the top of all odd numbered pages. It is functionally similar to ".TT", and stands for "Odd Top title." Please refer to ".BT" for further information regarding titles.

.PA

.PA <n> ¦ <+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 '+n' is specified, 'n' will be added to the current page number and the sum will be used as the page number of the new page. Blank pages won't be created to fill in the gap. This can be useful when you plan to insert additional material later on.

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 a page must be skipped, titles will be printed on it anyway. You can discard such a page if it's not needed, or use it for illustrations.

If ".PN OFFNO" is in effect and ".PA n" is specified, the internal page number counter will be set to 'n' anyway. However, subsequent page advances won't update the counter until ".PN ON" or ".PN OFF" is specified.

'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
.pa +2
```

.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). If ".LI 8" is specified, ".PL" should be set to '88' (11 times 8 = 88).

'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
```

To print full pages at 8 lines per inch:

```
.li 8;.pl 88
```


.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:

Page 2

'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 reactivate 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 redefine any titles containing the Page-numbering Symbol so that the symbol no longer appears in the title definition.

Example

```
.pn off
.pn on
.pn offno
```

.PP

.PP <cp <,sk <,in <,sh > > > >
3, 2, 5, 0

This starts a new paragraph. It is one of the most useful control words in SCRIPT. 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 is processed in turn.

Usually, just ".pp" by itself is sufficient to initiate a new paragraph. If you change the settings the first time in a document, a simple ".pp" thereafter, with no parameters, will continue to use the ones you set the first time.

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. If double or triple spacing is in effect, the blank lines will be

calculated based on single spacing, but the conditional page requirement will be based on the double or triple space setting.

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.

The two most common forms of ".pp" are shown in the examples below. The first one overrides the defaults, and the second continues to indicate new paragraphs using those overrides.

Examples

```
.pp 3,2,4
This is the first paragraph of my world history...
.pp
This is the second paragraph. For all paragraphs in
this mini-document, there will be one blank line between
each new paragraph (2); at least three print lines must
be left on the page (3), and the first line of each
paragraph is to be indented 0.4 inches (4).
```

.PS

```
.PS  c
    (<$>)
```

This allows you to redefine the "Page-number Symbol". The "PS" is used in Top and Bottom Titles to indicate where you want the page number to be placed on each page. 'c' is any character, such as '$', that will not appear anyplace else in any title line.

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. Other good choices are:

$ * + = ?

Example

```
.ps ?
.tt //MEALS UNDER $5.00/Page ?/
```

Since the desired title contained the default Page-number Symbol ($), that symbol was redefined before the title was defined. The symbol should not be redefined thereafter.

.PT

```
.PT <n>
```

".PT" stands for "point", and instructs NEWSCRIPT to perform automatic paragraph numbering. 'n' is an optional starting number; if omitted, the first point of a document is numbered '1', and each successive occurrence of ".pt" causes the next sequential number (2,3,4,... etc.) to be used. The generated number is followed by a period and a space, and then by the next line of text.

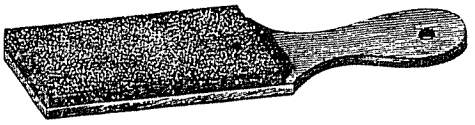
Points usually are used in conjunction with hanging indents, and both are explained in Section II of the tutorial.

When the number gets to '10', the text will be indented further to the right; the number will not be undented to the left. If you want it undented, you can begin with an ".IN 3", and reduce it to ".IN 2" at the appropriate point.

Also see ".HI", ".OF", ".IN", and "Indentation" in Section II for additional examples.

Examples

```
.pt             -- use next number
.pt 1           -- reset counter to 1
```



Lead pointer. *K & E*

.RD

```
.RD  n  <fileid>
     c  fileid
```

This instructs SCRIPT to "ReaD" one or more lines from the Keyboard or from 'fileid' and to print them as-is in the next lines of the document. It is used to create Form Letters, and can do so in any of four different ways.

'n' must be a positive number, if supplied. It defines a fixed number of lines to be read at a time from the N&A (Name & Address) file.

'c' stands for any non-numeric character, and may be used to specify a variable number of lines to be read at a time.

These are the ways in which NEWSCRIPT can create form letters:

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, each Name and Address (N&A) in the N&A file must begin with a "selection codes" line. The first (or only) character on this "codes" line must be the character 'c' (we don't mean you must use the letter 'c'; it can be a '#', '&', '@', etc.,... anything you choose is o.K.). When you use this format, each N&A entry can have as many lines as you need; the number of lines can differ from one group to the next; and the groups are separated by "code lines". Therefore, each Name and Address group ends with the next "codes line", or with the end of the file (for the last group only). Since the format provides for a variable number of lines, blank lines aren't needed for short entries.

4. The fourth kind of list is like the third, but with the addition of up to nine variable names and/or phrases that can be inserted within the body of the text based on symbols you place in the original form letter. This is the most flexible and powerful Form Letters support in NEWSCRIPT, since the merged text can be formatted as though it was an integral part of the letter, including underlining, right-justification, and any other formats supported for your printer.

Suggestions

The Form Letters feature of NEWSCRIPT can be very useful in certain situations, and it's tempting for a new user to jump right in and attempt to try it before mastering the fundamentals of NEWSCRIPT. We suggest you become comfortable with the tutorial material in Sections I and II before doing much with Form Letters.

Because Form Letters can be a confusing thing to grasp, it may be worthwhile right now to skip ahead a page or two, and look at the examples. They show prototype letters and matching lists. After you've looked these over, come back to this discussion and continue reading. The "HOW TO..." section has further information on this subject.

How Form Letters Work

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' (which may be any character that won't appear as the first letter of a name or address) is specified, 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 (which it is by default):

   @@1 Mr. Smith

   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.

   The prompt message reads:

   ENTER SUBSTITUTION VALUE (DATE):

NOTE:

Some earlier versions of NEWSCRIPT used the at-sign as the default tab character as well as the default substitution symbol. This was done in an attempt to minimize the number of reserved characters on the keyboard, but sometimes led to confusion. For that reason, the tab default has been changed to the ampersand ('&'), and the '@' has been dedicated to Form Letters. The description of "soft hyphens" elsewhere in this manual explains that the capital at-sign is the default soft hyphen. That character displays differently on different printers and computers, and does not conflict with Form Letters. sK;.pp 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. Example #2, below, illustrates this format.

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:

*** LAST LINE READ FROM N&A FILE ***

If 'fileid' is not found on an on-line diskette, this message is displayed, and SCRIPT goes to end-of-job:

!!! FILE fileid NOT FOUND FOR '.RD' !!!

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.

Example #1: - Form Letter with Keyboard Input

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.

Example #2: Form Letter with Fixed # Lines from Disk

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:

```
S. Smith
123 Oak Street
Van Nuys, Ca.

Ms. Jane Jones
Mini-Widgets, Inc.
1555 East Main Street
Los Angeles, Ca.
Bob Brown
45678 Hollywood Blvd
Suite 4
Hollywood, Ca.
```

Notice that the very first entry (to S. Smith) 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.

Example #3 - Variable Number of Lines with Substitution

This final example uses the fanciest features of Form Letters: variable number of lines, text substitution, and selection based on codes. This is the fourth type of Form Letter, and could be simplified to the third type by changing "Dear ##1" to "Dear Member".

Suppose the letter and the mailing list looked like this (we will show only three entries):

```
---- PROTOTYPE FORM LETTER ----
.ll 40;.im logo
.rd @ club/lst
.sk
Dear @@1,
.pp
The Valley Chapter of "Computer
Addicts" will hold its next
meeting on Saturday, April 14th,
at the Victory Inn. The meeting
should be particularly
because we will have !$no one!%
talking about Word Processing!
.pp
See you Saturday!


---- LOGO FILE ----
.ce on
COMPUTER ADDICTS
123 Chip Place
Silicon Valley, Ca.
.ce off;.sk2
```

```
------- N&A LIST -------
@A,VP,7/81
Sue Smith
123 Oxnard St.
No. Hollywood, Ca. 92699
@@1 Sue
@X,6/81
Ron Rogers
AAA Enterprises
Box 55443
Van Nuys, Ca. 91067
@@1 Mr. Rogers
@A,12/81
Mike Brown
Open Roads Bikes, Inc.
811 N. Vanowen St.
Suite 3B
Sun Valley, Ca. 93505
@@1 Mike
```

When the letter is processed by SCRIPT, and the ".RD" is encountered for the first copy, the at-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-case), then letters would be printed for the first and third entries in the N&A file (Sue Smith and Mike Brown). 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.

The first letter generated by these files would look like this:

COMPUTER ADDICTS
123 Chip Place
Silicon Valley, Ca.

Sue Smith
123 Oxnard St.
No. Hollywood, Ca. 92699

Dear Sue,

The Valley Chapter of "Computer Addicts" will hold its next meeting on Saturday, April 14th, at the Victory Inn. The meeting should be particularly interesting because we will have no one talking about Word Processing!

See you Saturday!

.SD

.SD <n1<,n2>>

This control word applies ONLY to proportional printing.

"Spacing Definitions" is used to override the default inter-character dot-spacing values. If you don't like the amount of blank space left between the letters of words printed by SCRIPT, the Spacing Definitions can be used to specify a different range.

Depending on the printer, NEWSCRIPT forces a minimum of zero, one, or two "dot spaces" between each character printed in proportional font. A "dot space" is the minimum horizontal space a printer can leave, and varies from printer to printer. Usually, it's at most a 60'th of an inch, and may be as small as a 200'th of an inch.

Now, a printer will leave only 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). Since "dot spacing" isn't used in monospace, this control word doesn't apply.

If you decide to increase (or decrease) the minimum amount of dot spacing between characters, you must specify either one or two values with this control word:

1. Minimum number of dot-spaces required. This can be as small as zero, or as large as you want to make it within reason. Setting it too large will tend to produce an unsightly spread between letters, just as setting it at zero on a dot matrix printer often makes everything look too crowded. On a Daisywheel printer, it probably should be left at zero for best appearance.

2. All padding between words, none between letters. By default, this parameter is set to zero, and the padding needed for right justification is placed between letters and words (that's how this book was done). If you want all the padding between the words, the second parameter should be negative: "-1". This is not the same as "monospace" justification, since the gaps between the words of each line will all be equal.

This control word normally shouldn't be needed or used. It's provided for special effects and to allow you to override the appearance NEWSCRIPT gives to proportional printing.

Examples

```
.sd 1,0
```

Most of this document, including the paragraph you are reading right now, uses the default value of 1,0.

T h i s p a r a g r a p h u s e s .s d 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 a value of zero. It is way the printer would print if special steps weren't taken. The crowded appearance makes it obvious why forcing SCRIPT to leave some white space between the letters is desirable. Note that SCRIPT still right-justifies nicely.

Finally, this paragraph is done at "0,-1". All the spacing is forced to occur s between the words, not the letters. It is proportional, but doesn't always look as smooth as when inter-character spacing is used. However, on daisywheel printers, this format often looks better than inter-character padding, so if you have such a printer and the DWP option, you may wish to experiment.

.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 ".LM" 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 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
.lm 29
conceived in liberty and dedicated to the proposition that
all men are created equal.
.sh -8
.lm 52
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.

Tabbing is in effect when a monospace font (10,12, or 16) is in use and concatenation (or formatting) is turned off. When formatting is on or proportional font is in use, tabbing is suppressed and the tab character is treated as normal text.

'c' is a character, such as the ampersand '&' or the '@' 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 you have more tab characters in a line of text than you've defined with the control word, then once the number of pre-defined tabs have been used, the remaining text will be discarded for that line.

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 therefore were 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.

NOTES

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. In effect, the first tab stop you give also acts as an indent (which is what tabs are anyway, of course), but the indentation is one less than the value of the tab stop: '1' means printing occurs in column 1, since the effective indent is zero.

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.

Earlier versions of NEWSCRIPT used other symbols, such as the at-sign, as the default tab character. This caused a conflict with ".RD" (Form Letters), so the default has been changed. If your document files have their own ".TB" definitions, they will still work as they did before. If those files just used the NEWSCRIPT default tab symbol and tab stops, a "global change" should be made to change the at signs to ampersands.

Since the ampersand ("&") 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. 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

If the document spans diskettes and you have only two drives, then drive 0 would be a good place for the table of contents. To get it there, remove the write-protect tab.

Some Operating Systems allow you to specify that a drive other than drive 0 should be used by default. DOSPLUS does this through "CONFIG (MASTER=n)" (TDOS doesn't have this), and NEWDOS/80 does it through the "AO" option. If you want the T/C on drive 0 in these cases, just create a "dummy" "fileid/TCT" file on SYSTEM DISK before starting SCRIPT. Regardless of which disk you use, make sure it will have enough room for the Table of Contents (which includes the Index if one is created), 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  ;  fileid
```

Note: This is an advanced feature and is not needed for most purposes. If you're using the Daisywheel Proportional Option, "Translate" may be required to make certain print wheels or thimbles work satisfactorily. When you decide to use this control word, it may be best to start by looking at the examples below, and then reading through this explanation.

"TRanslate" can be used in two different ways. Both ways can be used in the same document and both ways can be used more than once. The first form uses three numbers and 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 or subtracting '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. MX-80's without GRAFTRAX-PLUS are examples of such printers. Microline 80's can print TRS-80 screen graphics also, but don't require translation to do so.

'n1' and 'n2' must be unsigned numbers between 1 and 255 (zero cannot be changed). 'n3' must be between -128 and +127 (the plus sign can be omitted, but the minus sign, if needed, must be present).

Several ".TR"s can be used to build a general-purpose translation table, but the second form, using a Fileid, is more suitable for such a purpose. However, this is not a good way to try to develop support for pin-addressable graphics (such as the capabilities of the Anadex DP9500 or the MX-100).

The second form is ".TR fileid". 'fileid' usually is one of the tables (such as SPIN/TAB or THEME11/TAB) provided with the Daisywheel Proportional Option. When used this way, ".TR" lets you designate a disk-resident table that describes the widths and character arrangements of a particular wheel/thimble. You can switch wheels as often as often as necessary through the "pause" escape sequence (see Section II), even in mid-line, subject to the restriction that the new wheel will print satisfactorily with the current translation table. You can switch tables at the end of each print line, if necessary, although this obviously would be a bit excessive. It is perfectly practical to use several translation tables in one document if you're changing typefaces and/or character sizes.

Example 1 - Block Graphics

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 Hexadecimal 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 128,191,0
```

(I used asterisks instead of ASCII 191's because the printer used to print this book doesn't have 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. Once the translation was no longer required, the table was set back to its normal values by the second ".TR". In practice, you probably wouldn't bother to do this; we included it for illustration..

Example 2 - Using Disk-Resident Translation Tables

These examples are useful only on printers with replaceable print elements, or dot matrix printers that allow the computer to change the width of printed characters.

To use our DWP option and a "BOLD P.S." thimble on a NEC Spinwriter:

```
.st Put BOLD thimble on printer, press ENTER
.bf 737
.tr spin/tab
This text will be printed in proportional pitch, and
will be translated to match the arrangement on this
special thimble.
```

To use a Gothic 15 wheel on a C. ITOH F-10 (or any other Diablo/Qume compatible printer), if you have DWP:

```
.st mount Gothic 15 wheel, press ENTER
.bf737
.tr prop15/tab
.li8
This will be printed in very small type.
```

.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).

Not all escape sequences can be used in titles. This restriction is printer-dependent. We don't list the supported and restricted combinations here because they are subject to change in the future.

Titles are always printed with the left margin, line length, and pitch (font) that was in effect when the last title was defined.

NOTE:

If the printer is at the top of a page when ".TT" is set, that top title will be used on the current page. However, this can happen only on the first page of a document. Thereafter, A top title takes effect at the start of the next page. Therefore, if you want to force a new page (.PA) and define a new top title (.TT) for that page, define the title first, and then force the page. This is shown in one of the examples below.

The top title appears on all subsequent pages until re-defined. If you want different titles on left and right-hand pages, use ".OT" (odd top title) and ".ET" (even top title) instead.

To produce just a left-justified heading:

```
.tt /THIS IS MY HEADING///
```

Also see ".BT", ".HM", ".HS", ".PS", and ".TM". In particular, further information about titles is given with ".BT" earlier in this section.

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
```

The way titles were defined for this book is shown under ".BT". If you want to change top titles and pages at the same time, do it this way (the word 'odd' can be omitted; we used it to show how to force a new chapter to start on a right-hand page):

```
This is the end of text on the current page.
.tt//Chapter 2/Page $/
.pa odd
```

.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 underline escape sequence described in Section II. Note that the Daisy Wheel Printer II does not underline blanks, so the escape sequence will produce the same results as this control word does. To get around it, put underscore characters in place of those blanks. The method for doing this is described in the tutorial.

Example

```
This shows how just a few words
.us can be underlined
in mid-sentence.
```

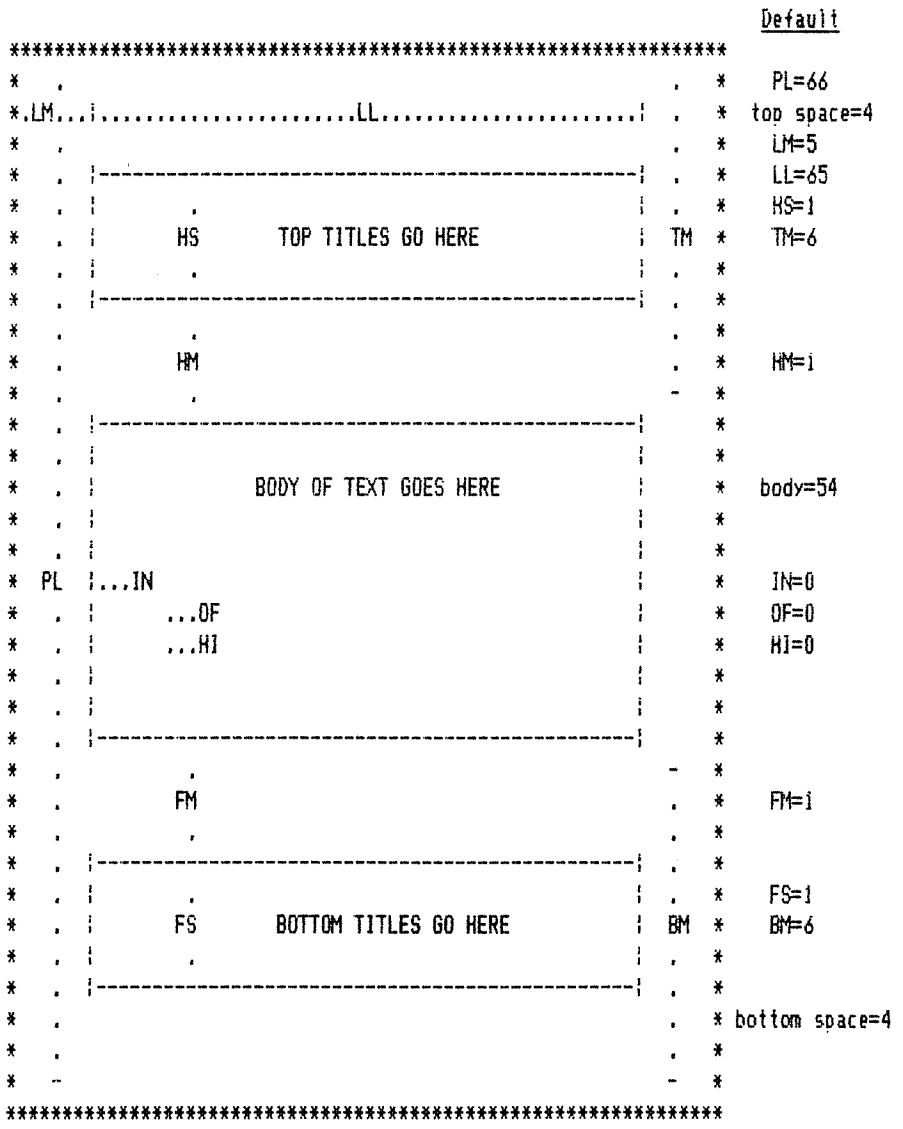
The result will be:

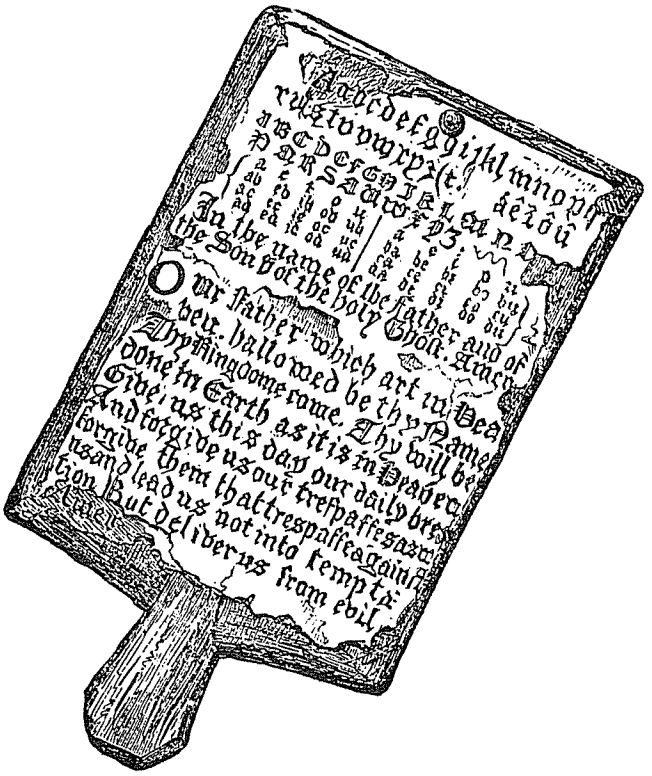
This shows how just a few words can be underlined in mid-sentence.

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 ".LM" 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".





Old fashioned horn book. *Harper's*

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 each line 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 not 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 not 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 every 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 four files specified for BASIC, will create a formatted printout and a special new file containing both Table of Contents and Index references. As soon as SCRIPT finishes printing the main document, it'll print the Table of Contents and then pass control to the "INDEX" program. This happens only if any ".IX" control words were found in the document.

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 in two columns. 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, as we did in this book.

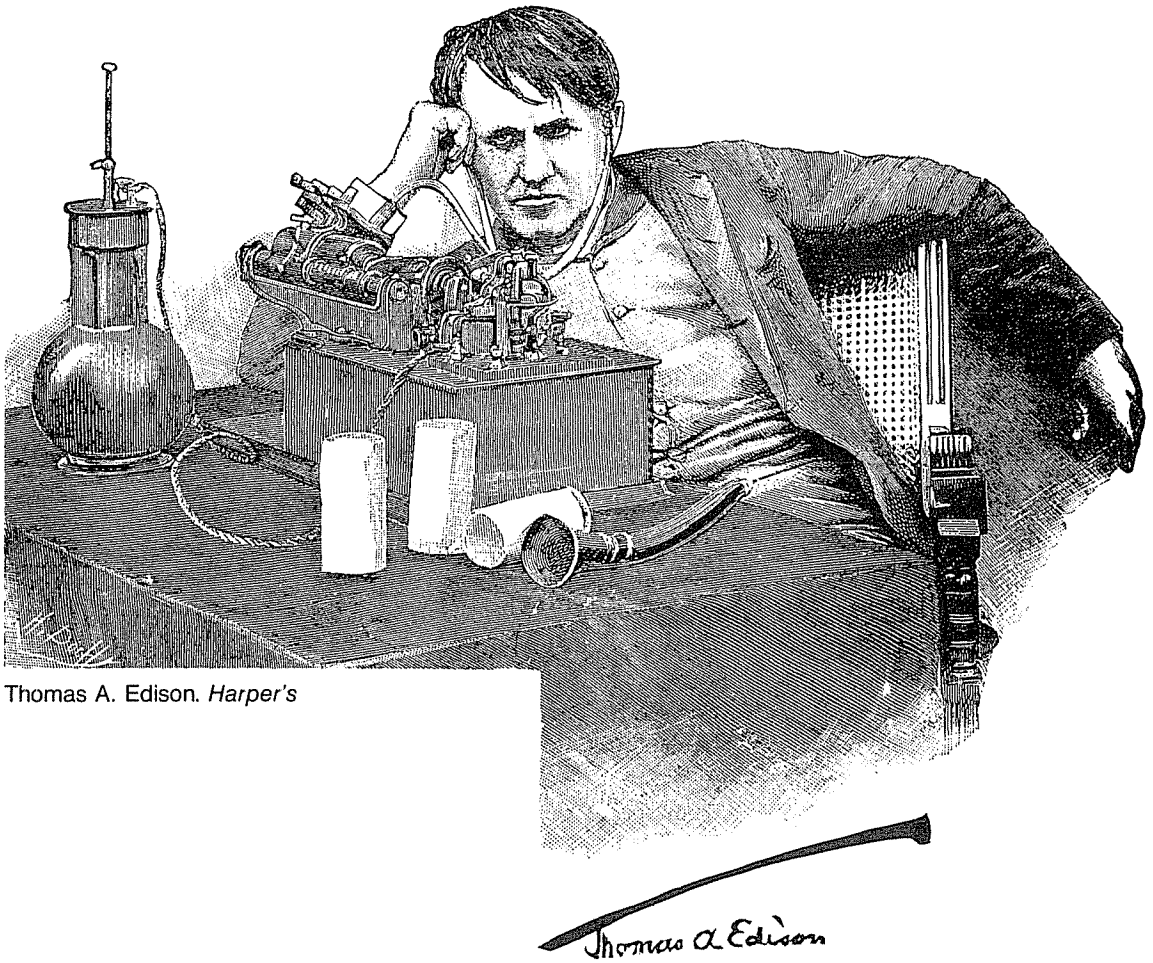
Maximum Number of Index Entries

Depending on the size of the terms being indexed, between 1,000 and 1,400 index references can be handled by this facility. There is no limit to the number of references to any one term, other than the 1,000-1,400 overall limit. The limit is based on available computer memory.

The index for this book exceeded the memory limit. We solved the problem by resetting the computer and entering TBASIC with only one file instead of the normal four. This added enough extra room to hold the 1,350 index references used in this book. By modifying the array limits in the "INDEX" program, you might be able to process a slightly larger index, if the phrases were shorter than the ones in this book.

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.



Thomas A. Edison. *Harper's*

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 and Mailing Lists
* Pre-Written Letters
* Tabbing, Tables, and Columns
* 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
* Lists
* Other uses of NEWSCRIPT facilities

Additional information on many of these topics may be found in the tutorials in Sections I and II.

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. Some people have used this example as the basis for many of their own letters. If you want to do the same, just remove the body of out letter and put in your own message. Of course, you probably should also change the logo and signature, and drop the "P.S."

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 (borrrrinnngggggg), 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 for both a logo and a signature.

Using EDIT, create two tiny files called "LOGO" and "SIGN" (I like to be creative in my choice of names). They would contain something like this:

```
----- LOGO -----

.ce on
SUPERIOR WIDGETS, INC.
4411 Franklin Ave.
New York, N.Y. 10023
(519) 483-2500
.ce off;.sk 2
```

```
--- SIGN ---

.sk3;.in35
Sincerely,
.sk3
Jan Doe
```

These could be saved on the Word Processing diskette itself, since they will be used frequently and take s very little room. Notice that I put a ".sk 2" at the end of "LOGO" to avoid having to type the spacing into the main document every time.

To use them in a letter takes only one line each, which saves a lot of typing:

```
.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....
.im sign
```

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 just want to print three (or thirty) copies of the same letter, but not merge names into it, the "CC" (carbon copy) option of SCRIPT is what you'll want to use.

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.

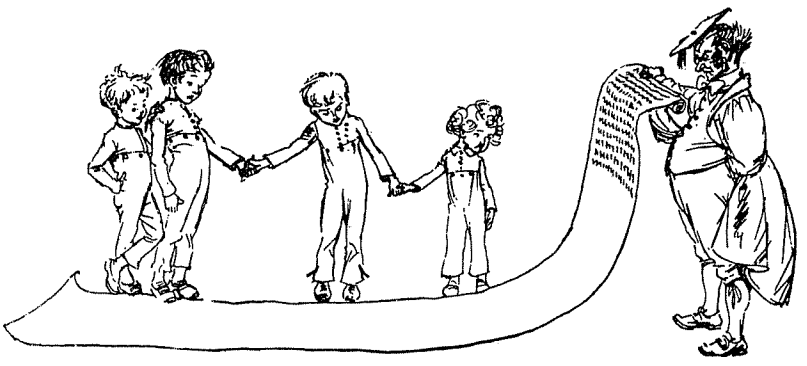
If you find yourself using these codes for several purposes, you'll want to make sure that identical-looking codes don't accidentally occur. One way to minimize such duplication is to place spaces or periods between the codes. For instance, if the code line signal in a given file was the at-sign (@):

@.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.



PRE-WRITTEN LETTERS

If you correspond regularly with certain people, the following technique may be useful. It was developed by Dr. Philip Mills.

The "Form Letter" facility is meant for use with a mailing list in situations where the content of a letter is fixed, but the names and addresses vary. Another kind repetitious letter occurs when you write regularly to the same person. In this situation, the content varies and the adressee is fixed. You can prepare a file with the name and address filled in, including provision for an envelope. To write a specific letter, just edit the prototype for the specific individual, give it a new name, and write the body of the letter. It can be printed quickly and then either archived or deleted.

In the example that follows, medical reports must be sent to referring physicians. The words "date" and "patient" can be changed globally when the specific content of the body is added, and the signature material can be provided by an imbedded file. Additional variable information can be added at the last minute because ".KE" (Keyboard entry) is specified, and the envelope can be printed right after the letter.

The next two pages illustrate the use of these techniques. The first page shows the prototype letter, while the second shows a specific result from using it. Page Titles are used in case the letter exceeds a page (which doesn't happen in this example).

Sample Prototype Progress Report

```
.cm fict/let
.cm
.cm ************************************
.cm **                                **
.cm **     REPLACE:  "patient"        **
.cm **           and  "date"          **
.cm **            globally            **
.cm **                                **
.cm ************************************
.cm
.ls15;.lm10;.hs2;.hm2;.fo off
.da1;.ce
date
.sk3
John Ficticious, M.D.
3743 Poverty Lane
Mytown, MS  39180
.sk2
Re: patient
.sk2
Dear Dr. Ficticious:
.tt1 *John Ficticious, M.D.*date*Page $*
.tt2 *patient***
.co;.pp 3,2,0
      (the content of the letter goes here)
.im mysign
.cm
.cm This ends the letter. Next, we'll do the envelope.
.tt1////
.tt2////
.pn off;.pa;.in35;.fo off;.LL95
.st ******* INSERT ENVELOPE, THEN PRESS "ENTER" *******
John Ficticious, M.D.
3743 Poverty Lane
Mytown, MS  39180
```

Signature File

```
.cm mysign   (standard sender's signature)
.fo off;.sk3;.pp6,2,0
Sincerely yours,
.sk3
Stand Dard, M.D.
SD/trs
.sk;.ke *** TYPE IN ENCLOSURES LINE (IF NECESSARY) ***
```


August 8, 1982

John Ficticious, M.D.
3743 Poverty Lane
Mytown, MS 39180

Re: Jim Smith

Dear Dr. Ficticious:

Final test results will be available in two days. Initial reviews indicate no serious problems.

Sincerely yours,

Stand Dard, M.D.
SD/trs

John Ficticious, M.D.
3743 Poverty Lane
Mytown, MS 39180

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. The <CLEAR><RIGHT ARROW> and the "HOrizontal" command, both functions of EDIT, can be used for simple-minded screen tabbing.

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, scroll the window horizontally.
4. If you use EDIT's "CHANGE" command, turn FLOW OFF.
5. If a line breaks in two by accident, use "JOIN B".
6. 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. Not all special formatting capabilities, such as underlining, can be used in titles; it depends on the printer.
4. Top titles take effect on the next page (unless you're at the top of the first page of a document), and bottom titles take effect on the current 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

Some of these techniques are illustrated under "Letters" earlier in this section. When creating Form letters with material merged from a Name and Address list, use ".RD" (read from file), not ".IM".

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: it can only page number to 32767), but EDIT can handle only as much text as it can fit into memory, which 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 encountered, 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, subsequent additions may cause the file to approach the capacity limits of memory. When the "Warn" limit (default is 15 lines or 600 characters) is reached, this message will appear on the command line:

15 LINES, 543 CHARACTERS LEFT

The actual numbers will vary, of course, depending on the characteristics of your file. If you keep adding text, the numbers will grow smaller and smaller. The message does not disappear by itself, and if you want to process a command, you'll have to clear the message out yourself. This is a protective feature, not an accident: we want to make sure you know you're running out of room before it's too late. Before you run out of room completely, it would be prudent to split the file into two smaller ones, or just use ".AP fileid" to chain to a continuation file.

If you continue to enter text, you'll eventually get this message:

NO MORE ROOM. ISSUE 'SAVE' FOR SAFETY

After saving the text, you can split the large file into two (or more) smaller ones as follows:

1. Decide where to split the file. Mark the first line of what is to become the second file as ".A" in the LIMA, and press <ENTER>.
2. Mark the last line of the file as ".B".
3. Move the cursor to the command line (<CLEAR><UP ARROW>) and enter this command ('fileid' is the name of the file being split out):

   PUT fileid

4. After the PUT completes successfully, delete the text from the first file:

   DEL .b

5. The "delete" will take a few seconds. When it finishes, place ".AP fileid" at the end of the file still in memory, with 'fileid' the name of the file you wrote with "PUT".
6. Save the file that's still in memory, probably under its old name if you're sure everything worked, or under a new name if you're the cautious type. You can rename everything later on.
7. Continue editing either the first or second file, depending on where you were adding new text.
8. This whole process takes less time to do than to read about. It's really simple, since "PUT" and "DELETE" both use the same Named Points.

It's also possible to save a file that turns out to be too large to read back into EDIT later on. This condition is unusual, but if it happens, then while reading in the file as EDIT begins, this message will be displayed:

```
FILE IS TOO LARGE TO BE EDITED ALL AT ONCE
nnn LINES WERE READ FROM: fileid
DO YOU WANT TO EDIT THESE OR QUIT (E/Q, DEFAULT=E)
```

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 200 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. The term we use for these things is "string". case of EDIT, there are several ways to do search:

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 string 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 UP" is like "LOCATE", but searches towards the top of the file. The command word is "LU" and the search string is specified as described for "LOCATE".

3. "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. "Locate Not" is designated by using the minus sign as a delimiter, and either "Locate" or "Locate Up" as the actual command: it works with both.

4. "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.

5. "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.

6. Finally, by pressing <CLEAR><UP ARROW> (or <DOWN ARROW>) repeatedly, you can visually scan for things. The search commands are usually more accurate, but if you're looking for a pattern instead of a simple string, a visual search is best.

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 immediately follows it on the same screen line (this is an exception to the general rule that text does not occur on the same line as a control word), and 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.

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.

The Radio Shack Daisywheel Printer II is unique in that it will not underline a blank. Consequently, using the underline escape sequence will produce the same result as using the ".US" control word. To avoid this, just put underscore characters in place of the blanks you need underlined. The underline character is ASCII 95, or hexadecimal 5F, or <SHIFT><CLEAR><minus sign>. To print long underlines, see EDIT's "STRING" command.

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 GRAFTRAX and the Epson MX-100 with GRAFTRAX-PLUS.

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 three 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. Epson print samples are shown in Section IX.

BOLDFACE

"Boldface" refers to the darkening of selected characters or words, and differs from the non-selective "DARK" run-time option or ".DA" control word, both of which darken entire lines. NEWSCRIPT can darken entire lines or groups of lines if the printer supports reverse line-feed or has some form of automatic overstrike or emphasized printing. NEWSCRIPT can darken selective portions of a line (boldface them) if the printer accepts a command for boldfacing or emphasizing selectively; or if the Daisywheel Proportional Option (DWP) is installed for one of the printers it supports.

A Centronics 737 or 739 (Line Printer IV) does not have hardware boldface, so NEWSCRIPT cannot perform selective boldface on these printers. This restriction also applies to the Line Printer VIII, the Daisywheel Printer II, many Microlines, and the EPSON MX-80 without GRAFTRAX. On the other hand, EPSON's with GRAFTRAX, GRAFTRAX-80, or GRAFTRAX-PLUS do allow selective boldface (they call it Emphasized and Overstrike), so NEWSCRIPT supports the capability on those printers. It also supports boldface on printers such as the C. ITOH 8510 and NEC 8023A.

Within these restrictions, text to be printed in boldface should be enclosed by this pair of "escape sequences":

!* !:

NEWSCRIPT will keep boldface turned on across multiple lines and pages, if necessary, even if the printer cancels the feature at the end of each printed line. This is true for all other escape sequence functions as well.

Example

There are !*several ways!: to !(add!) !$!*emphasis!%!: here

would print as follows (if the printer supports it):

There are **several ways to a d d emphasis** here

DOUBLE-WIDTH LETTERS AND HEADINGS

"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 OUTPUT

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 3", while single characters such as asterisks and dashes do best with ".HI 2" (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, and in Section II of the tutorial.

KEYBOARD DEBOUNCE AND REPEAT SPEED

"Debounce" is a method of preventing keys from repeating accidentally. "Bouncy" keyboards used to be common on the TRS-80 Model I, and it was necessary to use "software" (computer programs) to eliminate the problem. These facilities are built into NEWSCRIPT.

"Repeat speed" is a way of controlling how many times a second the same character should appear on the screen when its key is held down. This is intentional repeating, as opposed to the kind that "bouncy" keyboards produce.

It's possible to adjust the degree of keyboard debounce and the key-repeat speed of NEWSCRIPT. The installation program, "NSINSTAL" asks questions about these, and by rerunning the installation procedure, you can change the values. After you do so, you must press <RESET> for the changes to take effect.

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.

Pressing the BREAK key can cause NEWSCRIPT to stop. If you do this by accident, just type "CONT" and press <ENTER>. If you do this in EDIT, check the screen to see if contains anything you don't want. If it does, move the cursor to the command line immediately, and type "WHOOPS", followed by <ENTER>.

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 by typing "CONT" and pressing

<ENTER>. Don't use "RUN" until you've given up, since that will cause loss of the in-memory copy of the file.

If the computer stops running while you're editing a file, it may be due to a power failure, an equipment failure, or a software error (either in NEWSCRIPT or the Operating System). Of course, none of these things should ever happen, but if they do, it'll probably be at the worst possible time. NEWSCRIPT has a way of attempting to recover even from these kinds of failures. The method often works, but won't succeed if a power failure lasted long enough for the contents of computer memory to be lost, or if an equipment or software failure destroyed the contents of memory.

To attempt a recovery from EDIT when the computer refuses to function:

1. Convince yourself that nothing less drastic is going to work;
2. Hold down <ENTER> and press <RESET>. Keep holding down <ENTER> until the "DOS" message appears on the screen showing that computer re-initialization is complete. This step is needed to ensure that the "AUTO" function will not be engaged;
3. If using TDOS or DOSPLUS, type: TBASIC * <ENTER>
4. If using another Operating System, issue the equivalent command, if it exists. This won't work with all Operating Systems!!!
5. If BASIC is able to re-initialize, you'll get a prompt indicating that it's READY. If not, you'll see that the computer stays in DOS mode. If it stays in DOS, the file can't be recovered from memory;
6. If BASIC does re-initialize, just type this: GOTO 9999 <ENTER>
7. EDIT will attempt to write the in-memory text out to disk using the name, "EDITTEMP/FIL". If this succeeds, you should immediately press <RESET> again, initialize NEWSCRIPT, activate EDIT, and select "EDITTEMP/FIL". See if appears intact and usable. If so, save it under the name you really want used, then "KILL EDITTEMP/FIL";
8. If the attempt to recover the text fails, it'll be necessary to use the latest disk-resident copy, if any, and type in the changes over again.

COMMON EQUIPMENT PROBLEMS

1. The "edge" connectors on the Model I oxidize and generate unreliable signals. Either rub them monthly with an eraser or install "gold plugs" (advertised in many magazines).
2. Too much heat is a well-known cause of electronic component problems. Solution: improve the ventilation, possibly by installing a whisper fan.
3. Rotational speed may be way too fast or slow. Our "RPM" program can confirm this and indicate how to adjust the drives.
4. The disk drives in some Model III's are not grounded sufficiently. A wire from a mounting screw of each drive to the chasis may help.

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, which is something you probably don't want.

WIDE LINES

EDIT can handle lines of up to 240 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, while <CLEAR><RIGHT ARROW> and <CLEAR><LEFT ARROW> will move the window horizontally. Of course, to process long lines, the horizontal movements are needed. They are very fast, and move a fixed number of columns at a time (the default is 20, but you can change that with the "HOrizontal" command).

Another way to move the window is both slower and more flexible: "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.

Two kinds of block graphics are supported by NEWSCRIPT, however. The ".TRanslate" control word, or the <CLEAR><124578> keys of EDIT. The control word lets you change one set of characters into another, while the control keys let you turn screen graphics blocks on and off. Both of these are explained in Section II. Of course, an even better way to create graphics would be to use a graphics editor designed for the purpose. "GEAP" is one such product, although it only supports certain printers.

Once created, block graphics can be printed easily on with the Microline series, since they use the same ASCII values as does the TRS-80.

If you're using an Epson MX-80 without GRAFTRAX-PLUS, 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 <CLEAR><124578>. 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. The specific values to use for the Epson printers are:

```
.TR 128,191,32;.cm add 32 to all graphics characters

    ... TRS-80 graphics blocks go here ...
        for as many lines as you need

.TR 128,191,0;.cm this puts the table back to normal
```

Remember, this doesn't work on all printers, only on the ones that support TRS-80 screen graphics. If you need elaborate graphics or charting facilities, you might consider obtaining a program designed for the purpose.

LISTS

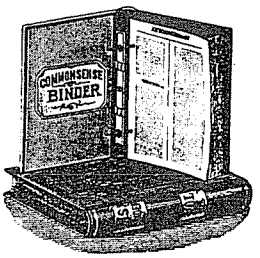
"EDIT" is a general-purpose text editor, so it can be used to create and maintain files of many types. These files don't have to have SCRIPT control words in them, and they don't have to be intended for Word Processing. Each line on the EDIT screen represents a separate "record" of information. The line can be screen width, or by horizontal scrolling, can be much wider. Since EDIT doesn't know or care what you write in each line, several lines in a row can contain related information, and then the next several lines can pertain to something entirely different.

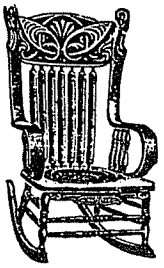
Here are a few kinds of lists you might want to maintain this way:

* names and phone numbers. print with "HARDCOPY" command.
* recipes. cards can be printed using the "LABELS" option.
* notes for a report, proposal, or project.
* small inventories. for big ones, use a File Manager.

Other Uses of NEWSCRIPT

"NS/CMD" can be used without the rest of NEWSCRIPT if you want a very fast keyboard processor with repeating keys, a screen print/route function, etc., but not the Word Processor facilities. Just put a blank and a minus sign after "NS" when you issue it as a DOS command, and it'll stay in DOS.





SECTION VII

ERROR MESSAGES

Many things can cause errors. Most errors are caused by mis-typing a command (spelling error or 'typo') or not understanding what a command is supposed to do. These kinds of errors usually are obvious almost as soon as they are made. They are corrected by re-typing the command correctly, or, if something was changed by mistake, fixing the change first.

The next most common kinds of errors (after User errors) are "Disk I/O" errors. "I/O" means "input/output", which is the transferring of information from disk to memory, (reading), or from memory to disk (writing). These errors can be caused by a wide variety of conditions. There's almost always a way around these errors, even if it means switching to another disk and re-typing some material. Because disk errors happen on a regular basis, it is essential that multiple copies, or "backups", be maintained at all times.

If you're just using a diskette to contain text typed into NEWSCRIPT, and it's a Model I, 35-track, single-density diskette, then by the time you fill it up there will be close to 15,000 words on it. Now, if you're able to type 60 words per minute, and that diskette becomes unreadable, you just might be able to retype it in about four hours. A backup could be recopied in about two minutes, and the extra disk would have cost at most $3.00. If you feel your time is worth more than 75 cents an hour, it pays to maintain backup copies.

Later in this section is a summary list of the abbreviated error messages TBASIC may issue. The bulk of this section is concerned with fully-worded messages, most of which DOS or NEWSCRIPT will issue.

Causes of Disk Errors

1. Diskette partially unreadable - this isn't a message, but a condition. It's most likely to occur when you first get a disk created on another computer (such as the NEWSCRIPT disk), and manifests itself as programs not being able to execute, or the disk not being able to be backed up. Solution: try it on another computer if feasible, to determine whether the disk is bad or your computer is out of step with the rest of the world. If it's the disk, send it back (with a note of explanation) for replacement.
2. TRACK FLAWED, LOCKED OUT - this occurs during FORMAT" or "BACKUP", and almost certainly means that your disk drive is running way too fast or way too slow. The message doesn't appear with TRSDOS, which is very tolerant of incorrectly adjusted disk drives, but if you've only used Model III TRSDOS until now and have just started using TDOS, chances are your disk drive(s) need adjustment. The level of cooperation you'll get from Radio Shack varies from store to store, but TDOS will work reliably on drives that are within 1% of the speed they're supposed to be (300 revolutions per minute, plus or minus 3). A drive rarely drifts by more than 1/2 RPM, so this isn't an excessive requirement: once adjusted, the drives will stay adjusted for months.

3. File not found - you either typed the name wrong, or don't have the right disk in a disk drive. Solution is obvious.

4. Disk is Write Protected - just remove the write-protect tab (a silver or black piece of tape along the right edge of the disk) and try again. Of course, there might be a good reason for the disk being write protected, in which case you should check before using it.

5. Disk is Full - this could just mean that it's write protected, and by removing the tab, the problem will go away. Or, it could mean what it says. Beginners often get this message because they didn't write protect drive 0, and didn't specify a drive number for their files. In that case, all new files are written to drive 0, which eventually fills up. Solution: make sure everything is sent to drive 1. Method: copy your text files from zero to one (there is a DOS command called "COPY"), then "KILL" the old copies on drive 0, then put a write-protect tab on drive 0. Of course, if it's a drive 1 disk that has filled up, the solution is even easier: just insert a formatted data disk in drive 1 and try again. If you don't have any pre-formatted disks lying around for the purpose, you may have a problem. Solution: remove the write protect tab from drive 0, try to save your text onto drive 0, then press <RESET>, hold down <ENTER> to suppress "AUTO", and format a couple of disks. Then, copy the file over from drive 0 to 1, Kill it on drive 0, and type the command "NS" to get NEWSCRIPT going again.

6. The "WELCOME" message keeps popping up - We set "AUTO WELCOME" on the NEWSCRIPT distribution diskette to help aim you in the right direction when you first started using NEWSCRIPT. That message should appear only one time, but since the distribution disk is write protected, it can't be removed or cancelled. When you made your first BACKUP of NEWSCRIPT, that all got copied over. Solution: on your BACKUP disk(s), type: AUTO NS, from to Dos plus prompt

7. Data Not Found During Read - you probably have tried to read a disk of the wrong density, or a Model III disk created by some other Operating System. Solution: find out what format the disk is in, and if you still want to read it, use the DOS "CONVERT" command.

8. HIT READ ERROR - now, that sounds scary, but all it means is that the disk either hasn't even been formatted, or was created by some other Operating System. Solution: find out which it is (by trying to read it with the other Operating Systems you have lying around, or by checking the external label on the disk). If it's a brand new disk, then by all means FORMAT it. But first make sure you won't be erasing your master inventory list by mistake. TRSDOS in particular is really blind to other Operating Systems, so if you stick your Model III NEWSCRIPT master disk into TRSDOS 1.3 and say "FORMAT", it'll do just that. Of course, if you've left the write protect tab in place, no harm will be done, but testing to see whether the write protect tab is infallible is like finding out whether those dollar bill changers will really reject a $10.00 bill. (Do you know for sure they will? Or do you just to take everyone else's word for it, which is what I do?)

9. GAT READ ERROR - much like HIT READ error, but it's possible that something has gone wrong with the "DIRECTORY" of the disk.

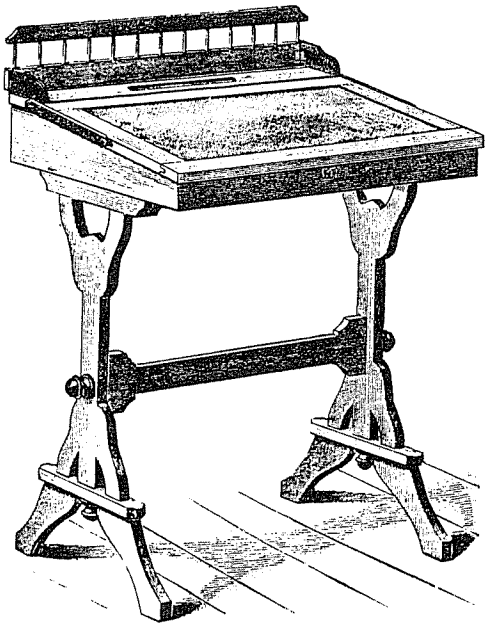
10. DIRECTORY ERROR - the message may not quite say this, but if it talks about "HIT" or "GAT", and you know the disk is formatted correctly and contains your files, then you have a real problem. If you have only two disk drives and the bad diskette doesn't contain an operating system (that is, it's a drive 1 diskette), you have a real problem. Solution: pull out a recent BACKUP of it. If you don't have a BACKUP, you'll have to start all over again, unless you have a knowledgeable friend with a program such as "SUPER UTILITY". Then, it may be possible to retrieve some or all of the contents of the diskette. Next time, of course, you'll have a BACKUP. (Something about a horse and

a barn applies here, doesn't it?) If you have three or four drives, try reading the disk in another drive; it may work. Of course, if you have Super Utility, SUPERZAP, or DISKZAP, and perhaps a copy of Harv Pennington's TRS-80 DISK AND OTHER MYSTERIES, you may be able to salvage the contents of the disk yourself. However, this probably isn't exactly what you'll want to do even if you happen to have the skills needed, so the real solution is to keep lots of BACKUP's.

11. CRC or PARITY ERROR - this means that you've got an unreadable spot on the disk. The rest of the disk probably is O.K., and the rest of the file probably is O.K., too. Solutions: try a few more times (it may work), try it in another drive or on someone else's computer (it has a very good chance of working), or treat it like a DIRECTORY ERROR and either switch to a backup disk or attempt to recover the text as described above.

12. ERROR 11 - this is an EDIT error message that summarizes just about all of the above conditions. We can't always tell what's happened, so this general message is issued instead. One of the above conditions probably has occurred.

13. Other kinds of disk errors - we've seen occasional disks that contained two files under the same name; disks where the file appeared in the directory but couldn't be read; and filenames containing invalid characters. What's worse, we've seen some of these with three different Operating Systems! We think the problem is due to a momentary memory fault or other electronic failure, but frankly don't know just why these things happen. Solution: use one of the "ZAP" programs, or find someone who knows how to do this for you. The directory can be "repaired" easily by a knowledgeable person. We aren't suggesting you become a computer technician, but at least you should have access to a good TRS-80 software person.



Writing desk. *Youth's Companion*

TBASIC ERROR MESSAGES

TBASIC (the subset of BASIC provided with TDOS) uses two-character error messages, not fully-worded ones. They supplement the two-character messages listed in the TRSDOS Level II manual that came with your computer. Now, none of these messages should ever appear, of course, but to believe the world is that perfect is to believe that bridges can be bought cheaply in Brooklyn.

NEWSCRIPT uses both BASIC and "machine language" programs. When one of the latter doesn't read in correctly, the Operating System probably will display an error message, or the computer will stop running. Solution: press <RESET> and start all over. If you get any of the messages below, they will have been issued by TBASIC. Solution: try once or twice more, then attempt to make a new BACKUP of the NEWSCRIPT disk. If the error continues to occur, it means either that your distribution disk is defective, or that your computer can't read our disk. This happens from time to time, and the solution is to either attempt the BACKUP on another computer, or send the disk back to us for replacement (please include a note explaining the problem).

AD - file access denied (password protected file)
AO - file already open (can't open it while it's open)
BM - bad file mode (probably an invalid name)
BN - bad file number (probably an invalid drive number)
DF - disk full (see fuller explanation above)
DS - direct statement in file (program loaded incorrectly)
EF - end of file encountered (application program error)
FE - file already exists (I've never seen this one)
FF - file not found (see fuller explanation above)
FL - too many files (TBASIC initialized with too few files)
FO - field overflow (programming error)
IE - internal error (TDOS error, may be due to memory error)
IO - disk I/O error (see fuller explanations above)
MM - mode mismatch (sequential/random file conflict)
NM - bad file name
RN - bad record number (programming error)
UE - undefined error ("other", the ultimate catch-all)
UF - undefined user function (programming error)
WP - disk write protected (remove the write protect tab)

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. System Error. This indicates either a software problem with EDIT, or an equipment malfunction. Move the edit window to the top of the file (<CLEAR><T>), then examine all the text (<CLEAR><F>) quickly. If it looks O.K., try to save it, possibly under a temporary name. If the save succeeds, <RESET> the computer, edit the file again, and see if things are better. If the problem persists, try using a backup copy of NEWSCRIPT. If the problem still persists and you don't think it could be equipment, get in touch with us.

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, Put, 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 you use the "GET" command for a file that can't be found or has a password, this message will occur also. 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 retry, 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 retry 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. Directory Unreadable. 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 not mean there's anything wrong with the diskette whose Directory you tried to display. It just means that NEWSCRIPT can't read or display it.

SCRIPT Error Messages

SCRIPT error messages are printed within the text, as well as displayed on the screen. They are printed because you'll never be able to find where they occurred unless you are given a marker tying the error to the text. If you're printing a one page letter, it won't matter, but if you're printing a large document, chances are you won't be sitting there for three hours watching SCRIPT run. The error messages are counted as printed lines. When SCRIPT finishes, it'll display a summary of errors, but not where they occurred.

The most common errors are #'s 8, 10, and 12.

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 150. The ".LL n" value is out of bounds. Correct and retry. If error persists, its same as #1. Note that line length is measured in tenths of an inch, not in characters, and to print 132 columns on 8 inch paper you should use: ".LL80;.BF16".

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 or value. Check and correct the mnemonic and the operands. This is far and away the most common error message you'll see.

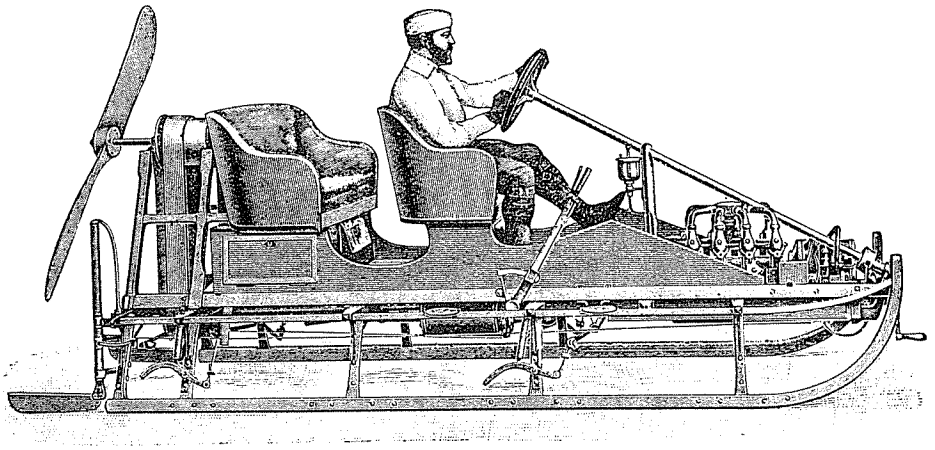
9. Top or bottom margin too small. The values of "TM, HM, HS", or the values of "BM, FM, FS" are in conflict. Either increase "TM" or "BM", or reduce some of the others.

10. File not found. Put the correct diskette on-line and retry, or correct its name. This happens when ".APpend" or ".IMbed" is processed and the referenced file isn't available. If you put in the correct disk and type the name again, SCRIPT will continue normally.

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.

12. Directory Unreadable - see EDIT message #12.

END OF ERROR MESSAGES



German motor sled. *Meyers*

SECTION VIII

PRODUCT DIFFERENCES

DIFFERENCES BETWEEN RELEASE 7 AND RELEASE 6

The number of additional features in Release 7 is too great to enumerate here. The main operational differences are:

1. In EDIT, <ENTER> doesn't give a blank line.
2. In EDIT, <CLEAR><RIGHT ARROW> doesn't split lines.
3. In EDIT, <BREAK> splits lines.
4. EDIT doesn't slow down as files grow larger.
5. EDIT processes much faster than in Release 6.
6. SCRIPT processes much faster than in Release 6.
7. Printing is done through an internal buffer for speed.
8. Double spacing doesn't cause ".SK" to double space.
9. The default tab character has been changed to '&'.
10. The default Form Letters code character has Been changed to '@'.

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 <BREAK> 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 (and probably some others; see Section IV)

SECTION IX

ADDITIONAL INSTALLATION CONSIDERATIONS

This section covers five special topics:

* using NEWSCRIPT with other Operating Systems
* printer switch settings and supported features
* serial (RS-232C) printer connections and considerations
* non-standard disk drives (1-drive, 2-sided, 80-track, hard disk)
* assistance from PROSOFT

USING NEWSCRIPT WITH OTHER OPERATING SYSTEMS

NEWSCRIPT works well with many operating systems besides the TDOS with which it is supplied. It takes longer for some of its components to be read in from disk on TRSDOS than on any other Operating System, and if used with Model I TRSDOS or NEWDOS 2.1, it cannot display the directory except when at the "DOS" prompt. Except for those considerations, it makes very little difference which operating system you choose to use. If you have no preference, we recommend you use TDOS, simply because it takes a little less effort to install it there. We also recommend you avoid TRSDOS if possible, because it is five to eight times slower than any other operating system in switching from one ,major NEWSCRIPT function to another.

Where NEWSCRIPT Resides in Memory

(If this paragraph is gibberish to you, just skip it.) NEWSCRIPT is not "self-relocating", and occupies the uppermost 6K of memory. It sets "HIMEM" when it is activated, and takes control of the keyboard, video, printer, some "RST's", and adds itself to the interrupt chain. This is a fancy way of saying that it takes control of much of the computer, leaving only disk I/O to the Operating System. Other keyboard drivers, such as "KIDVR", cannot be used under any circumstances whatsoever with NEWSCRIPT. Neither can any "DO" or "CHAIN" files.

If you use high memory machine language programs of your own, and must continue to use them (e.g., PDUBL in LDOS), you must make sure they load and activate below NEWSCRIPT, even if they load before NEWSCRIPT does. To do this, just set HIMEM to X'E700' (that's '59136' in decimal) before your program loads in. You can fine tune this address by letting NEWSCRIPT activate first, and then looking at "HIMEM". Doing so will give a few more bytes to the workspace, but may not be worth the effort since NEWSCRIPT is liable to grow a little larger in the future. As we said at the beginning of this topic, if it made no sense to you, don't worry, since you don't need to do any of the things it talks about.

80-Track Drives

If you're using 80-track drives, you can either use the utility programs supplied with your 80-track Operating System to copy the NEWSCRIPT diskette from 35/40 to 80-track format, or else connect a normal 35/40 track drive temporarily. Except for TRSDOS, most of the newer operating systems have some kind of "SKIP" function that lets them read a 40-track disk in an 80-track drive. Under no circumstances should you overwrite the distribution diskette, since there's a very good chance you'll make it unreadable. If you can't convert 40 to 80 yourself, and know of no one who can help you, PROSOFT will do it for you: send your original NEWSCRIPT diskette, a blank diskette, and $10.00 to us with the request. The service is only for copying TDOS to TDOS in one-sided, 80-track format. We cannot do this for other operating systems.

VERY IMPORTANT

NEWSCRIPT MUST be configured for your operating system, computer (Model I or III) and printer before it will function properly. If you install it with the supplied TDOS and then try to use it as-is with TRSDOS 2.7DD, LDOS or on the other computer model, it just won't work. So, after copying it to your own system, run the Installation procedure as explained later on in this section.

MODEL I

All Model I Operating Systems can read the distribution diskette, so installation is simple. Note that TRSDOS 2.7DD can COPY files from our diskette, but cannot run programs directly from single-density diskettes.

1. Prepare a diskette with sufficient room to contain some or all of NEWSCRIPT. At least 44 granules are needed (45 including "HELP"), and then copy the NEWSCRIPT files from the distribution diskette to your operating system. If you have a one or two drive, single-density system, this diskette should contain your Operating System, with as much deleted as possible. If your disk drives have greater capacity, you can store NEWSCRIPT either with the Operating System, on a separate diskette (if you have at least three drives), or the back side of a two-sided drive.

2. If you have a one-drive TRSDOS, re-boot the NEWSCRIPT distribution diskette and use the command "COPY1 fileid" to copy each file from NEWSCRIPT to your TRSDOS (this won't work with TRSDOS 2.7DD; you must have two drives). Please follow the prompts carefully as you do this. (We re-emphasize our suggestion that you use DOSPLUS instead of TRSDOS.)

3. A minimal NEWSCRIPT must contain these files from Side 1 of the distribution diskette:

   NS/CMD FEDIT/CIM PROP/CIM NSINIT EDIT SCRIPT

   It must also contain one of these files from Side 2, and the selected file must be renamed to "STARTUP/MIN", thereby replacing the one intended for use with DOSPLUS:

   LDOS/MIN MULTIDOS/MIN NEWDOS/MIN TRSDOS/MIN

4. If you have room on your destination diskette, you'll also want to copy "HELP" from Side 1, and these files from Side 2:

   FITLINE INDEX GENINDEX LABELS

   (LABELS is an extra-cost option, so you may not have it). If you don't have room for

these, just copy them to another System diskette for occasional use. Similarly, "EDIT1/EX", which is part of the NEWSCRIPT tutorial, can be used from an extra diskette when you're first learning NEWSCRIPT, and is not needed thereafter.

5. If you have purchased the Daisywheel Proportional Option, then "PROP/CIM" must be copied from the DWP disk to the disk you're now creating. It will replace a smaller file of the same name. This must be done before performing the installation procedure. If you obtain this option in the future, repeat the installation procedure starting at this step. If you just copy "PROP/CIM" to your working diskette, but don't bother to run "NSINSTAL", the text formatter won't work.
6. Once you've built your everyday NEWSCRIPT diskette, you must complete installation as described later in this section.

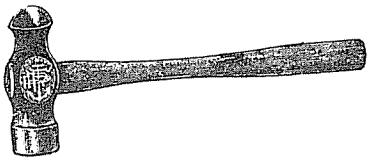
MODEL III

Moving NEWSCRIPT to another Operating System is feasible, but not as straightforward as on a Model I. We will describe a way to accomplish this conversion, but if you are knowledgeable in the use of your system, you may know of some shortcuts. (For instance, LDOS can read a TDOS disk.)

1. Boot the NEWSCRIPT distribution diskette, wait for it to issue the "WELCOME" and finally stop with a "DOSPLUS" message on the screen.
2. Create two formatted "data" disks using the command, "FORMAT". You'll be asked several questions, and should specify "35" tracks, "single" density. If the target operating system is TRSDOS, you must do it this way; if the target is NEWDOS/80, you may be able to format a 40 (or 80) track single-density diskette and use different "PDRIVE" values (TC=40,DDSL=20).
3. When FORMAT is done, it'll ask you to press <ENTER> once the System disk is back in drive 0. If NEWSCRIPT is still in 0, just press <ENTER> as requested; otherwise, replace the diskette and press <ENTER>.
4. The next step is to copy NEWSCRIPT to this single-density diskette. If you have at least two drives, make sure NEWSCRIPT is in drive 0 and the single-density disk is in drive 1. Then type: "DO TRSDOS", which copies everything over. It may pause in the middle and ask you to switch to the second data disk. After you do so, press <ENTER> to continue. If you have only one drive, display the Directory, then use "COPY1 fileid" to copy each "fileid". Be sure to follow the prompts carefully when you do this. The files listed above for the Model I are the ones that need to be copied here, as well.
5. Now, insert your own Operating System disk in drive 0, and re-boot the computer. It's good practice to make a backup copy of your disk and work on the copy. Try to delete ("KILL") as much as possible to ensure there will be room for NEWSCRIPT.
6. If you're using TRSDOS, the "CONVERT :1 :0" command will copy NEWSCRIPT for you. If two data disks were needed, then repeat CONVERT for the second one. LDOS and MULTIDOS can read the single-density format directly, so you can immediately perform a mass-COPY. If you're using NEWDOS/80, use "PDRIVE" to define drive 1 as single-density, 35 track, 10 sectors/track. Then, use "COPY" with the "CBF NFMT" options to move the files to NEWDOS. According to our copy of the NEWDOS/80 Version II manual, the commands are:

```
PDRIVE 0,1,TI=A,TD=A,TC=35,SPT=10,TSR=3,GPL=2,DDSL=17,DDGA=2
COPY   1,0,,NFMT,CBF,USR
```

7. If you're using NEWDOS/80, MULTIDOS, or LDOS, you now must "KILL STARTUP/MIN", and then "RENAME NEWDOS/MIN TO STARTUP/MIN", or "RENAME MULTIDOS/MIN TO STARTUP/MIN", or or "RENAME LDOS/MIN TO STARTUP/MIN". The "DO TRSDOS" procedure has copied the TRSDOS version of "STARTUP/MIN" for you, but didn't set anything up for the other O/S's.
8. Remove the single-density diskette, re-boot (especially with TRSDOS), and then follow the installation instructions below.
9. When NEWSCRIPT is ready for use, and you've verified that it seems to work, make a backup of the new disk.



GENERAL INSTALLATION PROCEDURE

1. Insert your operating system disk and a backup copy of a NEWSCRIPT disk (converted if necessary as described above) into your drive(s). The NEWSCRIPT diskette must contain "NS/CMD", "PROP/CIM", and "NSINSTAL". It probably contains "EDIT", "SCRIPT", and several other files as well, but those are not used during installation. The disk containing NEWSCRIPT must not be write-protected at this time.
2. Press <RESET> for safety, holding down <ENTER> to override any "AUTO" functions you may normally use.
3. Issue the "BASIC" command your operating system uses, and specify at least one file. "Memory size" doesn't matter. The forms to use for different Operating Systems vary, but here are the common ones:

```
TRSDOS: BASIC        NEWDOS:    BASIC       LDOS: LBASIC (F=4)
        4            MULTIDOS:  BASIC       (or): LBASIC (F=4,E=N)
        <ENTER>      DOSPLUS:   TBASIC -F:1
```

4. Type the following and press <ENTER>: RUN "NSINSTAL"
("DO NSINSTAL" works only with TDOS or DOSPLUS.)
5. A series of instructions will be shown, followed by a series of menus, from which you can configure NEWSCRIPT to match your equipment and preferences. Be sure to specify your Operating System when asked, since "TDOS" is the default. Additional information about installation may be found in the standard procedures given in Section I, but since the procedure on the computer is self-contained and may change from time to time, it is not repeated here.
6. The procedure should end by saying that NEWSCRIPT is ready for use. If you get any other message, correct the problem if it is apparent, press <RESET>, and try again. If the problem persists, it's likely that "NS/CMD" or "PROP/CIM" was not copied correctly to your own disk.
7. When the procedure ends successfully, NEWSCRIPT is ready for use. We suggest you make at least one backup at this time, and then go back to Section I and the tutorial.
8. If you're using Model I TRSDOS 2.7DD, you must remove "ULC" from your system diskette, or NEWSCRIPT won't work. To do this, issue the DOS command: PURGE ULC:0

CONVERSION OF NEWSCRIPT FILES FROM TRSDOS TO TDOS

If you've been running earlier versions of NEWSCRIPT on TRSDOS, here's your chance to switch to a much faster system: with TDOS, the transitions between EDIT and SCRIPT take about five seconds, instead of the 25 seconds required by TRSDOS. If you're using a Model I, no conversion is needed: TDOS can read the NEWSCRIPT files directly from your TRSDOS diskettes, and write them back afterwards.

If you're using TRSDOS on the Model III, you can use the TDOS "CONVERT :1 :0" command to move the files from TRSDOS to TDOS. To do this, make a couple of copies of the TDOS distribution diskette, then "KILL" all the visible files shown by the "DIR" command. If any files are protected, don't worry about them. These mostly-empty disks will receive your files from TRSDOS.

Put one of these TDOS's in drive 0, and a TRSDOS data disk (with or without operating system) in drive 1. Then, issue this command:

CONVERT :1 :0

If a destination disk fills up, go back to TRSDOS and KILL the files that were copied successfully, then return to TDOS, with another empty disk, and continue until you've converted all the files you need.

After you've converted all your files, use "FORMAT" to prepare a "data-only" disk in drive 1. FORMAT it to 40-tracks, double-density (if you have 80-track drives, format to 80 tracks, of course). Then, insert each of the intermediate TDOS disks in turn in drive 0, and copy all the files, one at a time, to drive 1. You may have to use more than one disk to complete this.

When all copying is complete, you're ready to continue using NEWSCRIPT, but under TDOS. As a practical matter, it may not be worth copying everything; just copy what is still current, and plan to retrieve other files if you ever need them in the future.

PRINTER SWITCHES AND SPECIAL CONSIDERATIONS

Many of the printers supported by NEWSCRIPT are covered in the next several pages. If your printer isn't here, it still may be supported. "Diablo" and "Selectric" are categories that many printers match, regardless of the name of the manufacturer. If nothing seems to work, call us or send us the programming and control code specifications. We may be able to pick the right category.

Centronics 737 and 739

The 737 (also known as the Line Printer IV) brought about a revolution in the microcomputer world. It was the first printer under $1,000 capable of producing high quality, proportional printing, and was the first printer to be supported by NEWSCRIPT. Both printers are very easy to use, and have no switches to set. The "line feed jumper" should not be cut when using a TRS-80.

NEWSCRIPT supports these features:

* left and right-justified proportional
* intermixing of single and double width in any pitch
* 10 cpi
* 16 cpi
* sub and superscripts
* overstriking entire lines of text (darkness)

NEWSCRIPT does not support these features:

* 12 cpi (the printer doesn't have such a pitch)
* selective boldface (the printer doesn't have it)
* italics (the printer doesn't have it)
* graphics

EPSON MX-80 AND MX-100

For reliability, quality, and versatility, the MX-80 is very hard to beat. Both of these printers are available with a variety of "GRAFTRAX" options. NEWSCRIPT supports the MX-80 without GRAFTRAX, with GRAFTRAX-80, and with GRAFTRAX-PLUS. It supports the MX-100 with either GRAFTRAX or GRAFTRAX-PLUS.

Switch Settings

"O" means "OFF" or "open". "X" means "ON" or "closed". The presence or absence of the "Friction/Tractor" ("FT") option doesn't matter:

Printer	<Switch 1>				<------Switch 2------>								
	1	2	3	4	1	2	3	4	5	6	7	8	NOTE:
STANDARD MX-80	X	X	X	O	X	O	O	X	X	O	O	X	If using an
MX-80, GRAFTRAX-80	O	O	X	O	O	O	O	O	O	O	O	X	EPSON cable
MX-80, GRAFTRAX-PLUS	O	O	X	O	O	O	O	O	O	O	O	X	with pin 14
STANDARD MX-100	X	X	X	X	O	O	O	O	O	O	X	X	disconnected,
MX-100, GRAFTRAX-PLUS	O	O	X	O	O	O	O	O	O	O	O	X	1-2 must be ON

Some of these switches control the buzzer and slashed zero, so if you want to change them, it won't affect NEWSCRIPT. However, anything that controls Line Feeds or automatic skipping over perforation must be set as shown. These settings will allow normal use of the printer even when NEWSCRIPT is not in use.

Features Supported

* 10 cpi single and double-width, intermixed
* 16 cpi single and double-width, intermixed
* Normal, Emphasized, Double-Strike, and Double-Emphasized
* underlining (solid underlining requires GRAFTRAX-PLUS)
* selective boldface (requires GRAFTRAX-80 or GRAFTRAX-PLUS)
* 6 and 8 lines per inch vertical spacing
* block graphics (except with GRAFTRAX-PLUS or MX-100)
* italics (requires GRAFTRAX-80 or GRAFTRAX-PLUS)
* subscripts and superscripts (require GRAFTRAX-PLUS)

Features Not Supported

* 12 cpi printing (printer can't do it)
* 16 cpi Emphasized (printer can't do it)
* high-resolution graphics

Print Samples

The next four pages were printed on an MX-80, and show how many of the features can be produced. A standard MX-80 has 12 fonts; with GRAFTRAX-80, this rises to 24; and with GRAFTRAX-PLUS, even higher. Consequently, we haven't shown all possibilities, but it should be apparent that the combination of NEWSCRIPT and an EPSON gives you a formidable array of print styles.

EPSON PRINT SAMPLES FOR NEWSCRIPT

The following control words and escape sequences were used in preparing these examples. Look for them in each example to see how they are used and what effect they produce. In many cases, we repeated things to show what we typed and how it got printed.

Please note that standard MX-100's do not have "GRAFTRAX-PLUS" or "GRAFTRAX-80", but only "GRAFTRAX". Consequently, they cannot handle italics or block graphics. MX-80's and MX-100's manufactured after April 15, 1982 were shipped with GRAFTRAX-PLUS as a standard feature. If you have one of those printers, your printer can do everything shown below. However, block graphics are not built into GRAFTRAX-PLUS, so that feature cannot be used.

NEWSCRIPT does not use "Form Feed" to skip to the top of new pages, since the Epson printers can lose track of top of form. This was supposed to have been corrected with GRAFTRAX-PLUS, but the old method is still used for safety.

NEWSCRIPT CONTROL WORDS AND ESCAPE SEQUENCES

```
.DA 0  - turn off all darkness
.DA 1  - turn on Emphasized (10 cpi only)
.DA 2  - turn on Double-Strike
.DA 3  - turn on Double-Emphasized (10 cpi only)
.LI 6  - print six lines per vertical inch
.LI 8  - print eight lines per vertical inch

  !$   - turn on  underlining
  !%   - turn off underlining
  !(   - turn on  double width
  !)   - turn off double width
  !*   - turn on  boldface  (GRAFTRAX-80 or GRAFTRAX-PLUS)
  !:   - turn off boldface  (GRAFTRAX-80 or GRAFTRAX-PLUS)
  !/   - turn on  italics   (GRAFTRAX-80 or GRAFTRAX-PLUS)
  !?   - turn off italics   (GRAFTRAX-80 or GRAFTRAX-PLUS)
  !+   - turn on  superscript                (GRAFTRAX-PLUS)
  !-   - turn on  subscript                  (GRAFTRAX-PLUS)
  !=   - turn off subscript or superscript   (GRAFTRAX-PLUS)
  !<   - backspace one character position    (GRAFTRAX-PLUS)
```


THE FOLLOWING WORK WITH ALL MX-80'S AND ALL MX-100'S:

This is normal 10 character per inch (cpi) printing. All darkness combinations can be used with it.

Part of this line is in !(double width!), most is single.
Part of this line is in d o u b l e w i d t h, most is single.

```
 .da1
This line is emphasized (darker) due to ".DA1".

 .da2
This line is overstruck due to ".DA2".

 .da3
This line is double emphasized due to ".DA3".
!(IT'S BEST IN DOUBLE WIDTH.!)
IT'S BEST IN DOUBLE WIDTH.
 .da 0

 .bf16
This is normal 16 cpi printing. Overstrike (.DA2), can be
used with it, but the printer does not allow Emphasized
printing (.DA1 or "DARK" run-time option) at 16 cpi.

Part of this line is in !(double width!), most is single.
Part of this line is in double width, most is single.

 .da 2
This is overstruck by ".DA2". ".DA1" and ".DA3" can't be
used at 16 cpi since they both attempt to use Emphasized.

!(Still in .DA2, but at double
width via escape sequences.!)
Still in .DA2, but at double
width via escape sequences.

 .da1;.bf10;.sk
Here's how we do !$underlining.!% It works with !$all
character sets including !(double-width!)!%.

Here's how we do underlining. It works with all
character sets including double-width.
```

NOTE: All features above also work with GRAFTRAX-80 and GRAFTRAX-PLUS, and won't be repeated below.

THE FOLLOWING REQUIRE GRAFTRAX-80 OR GRAFTRAX-PLUS:

Here's how you can !*boldface!: within a line.
Here's how you can **boldface** within a line.

You can use !/italics the same way,!? and you can
combine !$underlining, !/italics, !*boldface, and
!(double width!)!:!?!% in all combinations at 10 or 16 cpi.

You can use *italics the same way,* and you can
combine underlining, *italics, boldface, and*
double width in all combinations at 10 or 16 cpi.

.bf16
You can use *italics the same way*, and you can
combine underlining, *italics, boldface, and*
double width in all combinations at 10 or 16 cpi.

NOTE: All features shown above also work with GRAFTRAX-PLUS.

THE FOLLOWING FEATURES REQUIRE GRAFTRAX-PLUS

!$Solid underlining is used instead of dashes.!%
Solid underlining is used instead of dashes.

The !+subscript!= and !-superscript!= character sets are great!
The subscript and superscript character sets are great!

.sk;.fo;.da1
Notice that sub and superscripts must each be turned off when no longer needed. This is different than the implementations for most printers, but it works just fine. With GRAFTRAX-PLUS, if you're in Emphasized (.DA1) mode and use a sub/superscript, when you exit to full sized characters, the printer will shift into double-emphasized. NEWSCRIPT doesn't compensate for this since you might intentionally be in double-emphasized.

.li8;.sk;.da0
!+We put the printer in subscript mode...

We put the printer in subscript mode to show how nice it looks and how you can do *italics*, double-width and, of course, ***both at the same time***. All of this could have been done in superscript mode instead, and would look exactly the same except for being a little higher on the page. We didn't show underlining because it may look a little strange with these tiny letters. You can see from this paragraph what the printer will and won't do.

.sk;.bf16
Now, if you use the condensed font with half-height letters, it looks even better! The number of character sets available is *so large and varied that we don't know exactly* ***how*** *to compute them.* Oh, by the way, we got out of subscript mode this way...

.sk;.bf10;.da1;.li6
we got out of subscript mode this way... !=

This is a more conventional use of sub/superscripts:

The formula for water is: H!+2!=0

The formula for water is: H_2O

.sk;.da1
This ends the Epson examples. You probably can see that there are a number of additional combinations we didn't show, but the versatility of the printer should be obvious by now, along with the ways to take advantage of it with NEWSCRIPT.

MICROLINE

Several versions of these fast printers are available and the levels of support are summarized below.

Switch Settings

80, 82, 83: automatic Line Feed optional
80A, 82A: auto line feed OFF to allow underlining
84: auto line feed OFF to allow Near-Letter-Quality printing

Supported Features

* 10 and 16 characters per inch (all printers)
* single and double width (all printers)
* underlining (A-series and 84 only)
* sub and superscripts (84 only)
* 12 characters per inch (84 only)
* boldface (84 only)
* Near-Letter-Quality (NLQ) proportional (84 only; see note below)

Features Not Supported

* underlining on 80, 82, 83 (printer would overstrike text)
* 'DARK' options
* graphics

Notes About Support for Microline 84

1. At 10, 12, and 16 characters per inch (cpi), all normal word processing features are supported: underlining, boldface, sub and superscripts, intermixing of single and double width characters.
2. NEWSCRIPT cannot drive this printer at its full speed in 10, 12, or 16 cpi, so momentary pauses will occur at the end of most lines, and hesitations will occur between paragraphs. This does not occur in proportional printing.
3. Proportional printing is based on the 12 cpi typeface and is subject to the limitations listed below. Some of these printer limitations and others are NEWSCRIPT limitations:
   * boldface cannot be used (printer)
   * underlining will not be solid in all cases, but dashed. This is particularly noticeable in double-width printing (partly printer, partly NEWSCRIPT)
   * actual printed line length will be about 1/2% shorter than specified: ".LL 65" will produce a 6.2 inch line (NEWSCRIPT)
   * attempts to underline superscripts cause them to print as subscripts. The underline prints in the same vertical location as subscripts, causing a strikeout appearance (printer)
   * in NLQ mode, the printer prints each line twice, reducing the effective speed to under 100 cps. This is not controllable by software.

ANADEX

Anadex manufactures a variety of fast, high-quality dot-matrix printers. NEWSCRIPT supports the DP 9000, 9001, 9500, and 9501 as this is being written. If other Anadex'es appear on the Printer Selection Menu for your version of NEWSCRIPT, then those printers are also supported.

Switches

"Switch 3" settings are shown for parallel attachment, which is preferred when using a 9500 series printer with the TRS-80. Switches 1 and 2 are shown for either parallel or serial. "O" means "Open" or "off"; "X" means "Closed" or "on".

	1	2	3	4	5	6	7	8	9	10
Switch 1	O	X	X	O	X	O	O	O		
Switch 2	O	O	O	O	O	O	X	X		
Switch 3	X	X	O	O	O	O	O	O		

Supported Features

* underlining
* double-width
* characters per inch on the 9500: 10, 12, 13.3 (as 16)
* characters per inch on the 9501: 10, 12.5 (as 12), 16.7 (as 16)
* 6 or 8 lines per inch

Unsupported Features

* boldface (no hardware feature on printer)
* 13.3 cpi on 9501
* graphics

MODIFIED SELECTRICS AND TYPEWRITER-LIKE PRINTERS

(Use this category if nothing else works)

This covers a category of printers with few special features except backspacing and high-quality printing. Underlining is supported if and only if the printer can backspace (honors ASCII 8), since underlining for this category is done by printing a character, backspacing, and printing an underscore character. Many interfaces for these typewriters do not support the backspace function, and underlining cannot be done without it.

If the typewriter or interface has a switch to control automatic line feed with carriage return, the switch should be set so that a line feed will be generated; NEWSCRIPT sends out only the carriage return unless you indicate during installation that line feed generation is needed.

Boldface is not supported on any printer in this category, although use of the backspace escape sequence will allow you force selective overstriking.

The "stop" escape sequence can be used to change print elements.

DAISYWHEEL PRINTER II

This printer can produce amazingly high quality results. No switches need to be set to use it with NEWSCRIPT. The pitch selection switch is bypassed, so it doesn't matter where you place it. However, NEWSCRIPT uses proportional pitch by default, and will leave the printer in that pitch when it's finished. If you use ".BF10" or ".BF12" to change pitch, the last pitch selected will remain in effect when NEWSCRIPT is done. If you need a different pitch later on, turn the printer off, then back on.

To use proportional pitch, it's best to have a proportional printwheel. The "Magdeline" and "Cubic" are examples of this. If you are using the normal "Courier" wheel, we suggest you always set ".BF10" at the beginning of each document, since proportional won't look too good with the wrong wheel.

Standard NEWSCRIPT supports fully-justified proportional printing, as well as 10 and 12 character per inch (cpi) printing. Sub-scripts, super-scripts, printer-controlled underlining, and changing of print wheels via ".ST" and the "pause" escape sequence are also supported. All special symbols on all wheels are supported. To use them, see ".TR", "ALTER", and "<SHIFT><CLEAR>".

The printer has no provision for 8 lines per inch vertical spacing, nor for printer-controlled boldfacing. Some DW2's have an "external program mode" to allow greater control over such things as boldface, underlining, and 15 cpi printing, but standard NEWSCRIPT does not use this feature. Please note that many DW2's lack the feature.

The DW2 is unique among printers in that it has hardware underlining capability, but will not underline a blank. NEWSCRIPT has limited ability to overcome this. Otherwise, underlining multiple words will result in only the words, and not the spaces between them, being underlined. Also, since the underline character is relatively narrow, the printer may leave gaps even within a word, particularly when underlining capital letters in proportional pitch. It is possible to overcome these in several ways:

1. To underline between words, place the EDIT cursor on each blank, then create an actual underscore character by pressing <SHIFT><CLEAR>, then releasing them and pressing the <MINUS> sign. This can be repeated as needed.
2. To create a simple line, as for a signature, use the "STRING" command of edit, and then either shift the resulting line left or right on the screen, or else precede it with an ".IN" to move it to the right. Here's an example of "STRING", which is always placed on the command line. Note that the underline character was produced as explained in #1, above:

   STRING 40,_ (or) STRING 40,95

3. To create several lines in a row, begin by creating one as shown in #2, above. Then, use the LIMA "R" (repeat command) to duplicate the line. Example: "R5" in the LIMA, followed by <ENTER>, would yield a total of 6 lines.
4. To produce solid underlining in proportional pitch for a capitalized heading:
   1. Type in the heading without an underline escape sequence. Be sure a "break" occurs (.BR).
   2. Force a reverse line feed: .SH -2
      This will move the paper back two half-lines (one full line) to allow the heading line to be overstruck.
   3. Use "STRING" to create a line of underscores. The line probably will have to be longer on the screen than the length of the heading, since underscores are narrow. The result will be a solidly-underlined heading.

DIABLO AND QUME

These versatile, reliable, high quality printers are very well supported by NEWSCRIPT, even without our "Daisywheel Proportional Option." DWP adds proportional right-justification and selective boldface. These printers usually can print faster than 30 characters per second (cps), and even when connected through the RS-232C port, can be used with NEWSCRIPT at higher print speeds in most cases. For further information about this, see "Serial Printer Considerations" later in this section of the manual.

Switch Settings

* word processing or proportional OFF (if present)
* pitch 10 or 12 (doesn't matter, software controlled)
* 30 cps standard, 35-55 cps serial possible
* parity OFF (MARK) if relevant
* word length 8 if relevant
* 1 stop bit if relevant
* automatic line feed on carriage return (required)

Features Supported with Standard NEWSCRIPT

* 10 and 12 cpi, but not on same line
* underlining
* darkening of entire line via overstrike
* subscripts, superscripts
* unlimited reverse line feeds (manual multiple columns)
* all characters except '127' on all wheels can be printed
* pause in mid-line to change wheels (escape sequence)
* cut sheets or continuous forms
* 6 or 8 lines per inch vertical printing

Features Added with Daisywheel Proportional Option

* right-justified true proportional printing with proportional wheels
* selection of proportional spacing to match all wheels
* transposition of characters to match all wheels
* selective boldface within a proportional print line
* faster underlining
* program to assist in creation of tables for new wheels
* pseudo-proportional printing with Courier or Elite wheel
* maintains compatibility with standard NEWSCRIPT

Running at 1200 BAUD (Diablo 1600 Series Only)

A "jumper" on one of the circuit boards can be inserted to allow this higher speed of operation. It is explained in an appendix of the Diablo manual, but PROSOFT cannot offer any assistance in implementing this. If you modify the printer this way, we suggest you include a long wire leading to an external switch that will let you turn off high speed when not using NEWSCRIPT. Also, be sure to specify at least three "pads" (nulls) when installing NEWSCRIPT. Set up this way, a 1600 series printer can run at about 45 cps.

C. ITOH STARWRITER and F-10

These high quality, low-cost printers are supported as "Diablo" machines, and they are highly compatible with those printers. The features supported are listed under "DIABLO", earlier in this section. If your C. ITOH uses an RS-232 connector, refer to "DIABLO" for most considerations. If a "word processing" feature is included in your printer, turn it off if possible before using NEWSCRIPT.

Switch Settings

When setting the switches, the printer should be turned off. This is for your safety, and is also required because the printer doesn't check the switches except when it's first powered on. Some of the settings below are optional, but the switches that control line feeds must be set as shown. These settings also will work when NEWSCRIPT is not in use. "O" means "Open", or "off"; "X" means "Closed", or "on".

FP-1500 Starwriter

	1	2	3	4	5	6	7	8
Switch 1 (Right)	O	X	O	O	X	O	X	O
Switch 2 (Left)	O	O	O	X	O	O	O	O

F-10

	1	2	3	4	5	6	7	8
Switch 1 (Left)	X	O	X	X	O	X	O	O

	11	12	13	14	15	16	17	18	19	20
Switch 2 (Right)	O	X	X	X	X	O	O	X	O	O

NEC SPINWRITERS

NEC manufactures a variety of very durable, fast, high quality printers. There are several kinds of Spinwriters, including some that have been modified or interfaced by companies other than NEC. We will attempt to give guidelines for the normal cases only. On that basis, there are two kinds of Spinwriters: those that are "Diablo-compatible" and those that are "NEC-native." These are represented on the Installation-time Printer Selection Menu as choices "11" and "12", respectively.

If you have a 5500 series printer (except 5530) that was manufactured before 1981, are using our Daisywheel Proportional Option (DWP), and experience print format difficulties once or twice per page, please contact us: NEC will fix the printer if necessary.

If you have a 3500 or 7700 with word processing features, those features must be turned off when using NEWSCRIPT. If you use a proportional thimble (BOLD P.S. or EMPEROR P.S.), you can either set the "proportional sequence" switch on the printer, or use one of the special tables provided with our DWP option. This doesn't apply to the 5500 series, which lack the switch. However, DWP still works with those printers.

The 5500 series, and many Spinwriters in the other series, have a "dip" switch that controls the maximum number of characters that can be printed on one line. When using the DWP option, this switch must be turned off to allow lines of unlimited length. Otherwise, proportional printing will rarely exceed three inches in length.

Features Supported

NEWSCRIPT support for Spinwriters is essentially identical to the support provided for Diablos and other daisywheel printers. Please refer to the "DIABLO" material a few pages back for a list of these features. The only significant difference is that, when using proportional thimbles and the DWP option, special transposition tables are normally needed with Spinwriters, but not needed with Diablos. These tables are provided with DWP.

The default printing pitch is 10 characters per inch. Proportional printing with DWP requires use of ".BF 737" and possibly ".TR SPIN/TAB" at the beginning of each document:

```
.bf737;.tr spin/tab
.cm the rest of the document goes here
```

If your printer uses an RS-232 connection to the computer, please see "Serial Printer Considerations" later in this section, and the material below.

Cables

A standard cable arrangement is described later in this section. A specialized one that may work with some Spinwriters is given below. To use it, the "Reverse Channel" switch must be set "OFF". On 7700's, the switch is on the front panel. On 5500's, it's inside, just behind the front panel, as dip switch #5. When this approach is used, "padding" in NEWSCRIPT can be set to zero.

Printer Pin	to	Computer Pin	
1		1	Also, on the printer
2		3	end, connect 5,8,20
3		2	together.
4		5	
6		20	On the computer end,
7		7	4,8,19 are not used.
19		6	

LINE PRINTER VIII

Right-justified proportional printing is the default for this printer when used with NEWSCRIPT. Other pitches are listed below. Most features of the printer are supported.

Early Line Printer VIII's (and possibly current ones) have a serious "bug" in their internal logic. From time to time, they will simply ignore leading blanks, and sometimes ignore carriage returns as well. The result of this is printing in the left margin, or printing starting at the right side of the page and extending into the right margin. NEWSCRIPT circumvents this problem in almost all cases, so even if you printer does this sort of thing, it is unlikely to occur with NEWSCRIPT. However, the problem still may occur in rare circumstances. Radio Shack was informed of the situation in late 1981, but has not responded.

Switch Settings

Dip switch '2' should be closed, and the other seven switches should be open.

Features Supported

* proportional right-justification; density user-selectable
* 10 cpi
* 16 cpi
* intermixing of single and double-width on a line
* underlining
* sub-scripts and super-scripts
* overstrike of entire lines (dark options)
* unlimited reverse line feeds allows manual multiple columns
* all special characters and symbols

Features Not Supported

* 12 cpi (printer can't do it)
* selective boldface (printer can't do it)
* 8 lines per inch (printer can't do it)
* italics (printer can't do it)
* graphics

C. ITOH 8510 AND NEC 8023A

Most features of these fine, fast, low-cost printers are supported by NEWSCRIPT, which by default prints in true, right-justified proportional on either printer. The current edition of this book was printed on an 8510. The 8510 and 8023A are not quite the same from a programming standpoint, but the printed results are identical.

Switch Settings

In the table below, "O" means "Open" or "off", while "X" means "Closed" or "on." Switch 1 is behind Switch 2, and the switches are numbered right-to-left when looking at the front of the printer.

	1	2	3	4	5	6	7	8
Switch #1	O	X	O	O	O	X	O	X
Switch #2	O	O	O	O	O	O	X	O

Some of these settings control "Select", slashed zero, etc., and may be changed if you wish. Others control line feeds and must be set as shown.

Features Supported

* fully justified proportional printing
* 10 cpi
* 12 cpi
* 16 cpi
* double width for any of these
* intermixing single and double-width within a line
* subscripts and superscripts (see note below for 8023A)
* underlining
* boldface
* all special printable characters and symbols

Features Not Supported

* graphics
* italics (printer can't do it)

NOTE: The NEC 8023A has an internal logic error that causes it to mis-handle sub-scripts and super-scripts when printing in right-justified proportional mode. Whenever any inter-character padding is done (that's how proportional right justification occurs), the printer loses track of where it is on the print line. If told to do a sub or superscript, it prints too far to the left, overstriking the text already printed. This problem does not occur on the 8510. NEC is aware of the problem and is attempting to correct it. It's possible that, by the time you read this, they will have succeeded in fixing this. In the meantime, the solution is to use 12 cpi for sub/superscripts, since that approximates proportional character widths.

SERIAL (RS-232C) PRINTER CONSIDERATIONS

NEWSCRIPT includes direct support for printers attached through the parallel printer port (Attachment Method #1) and through the serial RS-232C port (Attachment Method #2). If you wish to use a "printer driver" of your own, or the one built into the Operating System, then specify Attachment Method #3 during installation of NEWSCRIPT, and see "Other Printer Drivers" later in this section.

When you install NEWSCRIPT, you'll be asked how the printer is attached to the computer. If you answer "2" (RS-232), you'll be asked a series of questions regarding BAUD rate (normally 300 or 1200), parity (normally none), word length (normally 8), number of stop bits (normally 1), and how much "padding", or generation of "nulls" is needed. Because there are four different electrical "handshake" methods, and at least two different "character signalling" methods by which a serial printer can indicate whether it can accept more data, and because many printers use no handshakes whatsoever, NEWSCRIPT doesn't even try to accommodate all the possibilities. Instead, it "paces" the data to the printer, making sure the printer's internal buffer does not overrun.

To help NEWSCRIPT pace successfully, you must tell it how many "pads" must be generated for each data character actually sent. If you're running at 300 baud on a printer that can print 30-55 characters per second (cps), then no padding is needed. If you're running such a printer at 1200 baud, then 3-4 pads will be needed. If you're running a 12 cps printer at 300 baud, then at least 3 pads will be needed. The "padding" is done based on the data actually sent: NEWSCRIPT doesn't generate a fixed number of pads at the end of each print line, but a variable number every once in a while.

If you specify too many pads, the printer will pause in the middle or at the end of most lines. If you don't specify enough pads, the printer will "overrun", beep, and lose text. Consequently, it's better to err on the side of too many pads. In actual use, we've found that 1200 baud and 3 pads work perfectly on serial Diablos and Spinwriters.

Cables

Several wires must be crossed (reversed) in a serial cable if the printer is to connect properly to the TRS-80. A number of schemes seem to work when only one, or no, handshaking methods are used, but most of these schemes eventually fail. When they do, the computer program either thinks the printer doesn't exist, isn't ready, or is always ready. The results are unsatisfactory, to say the least.

The method below works with NEWSCRIPT, and as far as we know, with most of the serial drivers provided with the popular operating systems. An alternate scheme is given under "SPINWRITERS" earlier in this section. If you don't know how to make a cable: A) you're in good company; B) find someone who knows how to do this and pay them to build one to these specifications.

We suggest a short "jumper" cable be constructed to go between the cable to the printer and the computer. If you do it this way, you won't be modifying any existing cables, and can remove the jumper cable if necessary. The crossing of wires within this jumper cable is symmetrical (mirror image), so the crossings should be made at only one end:

* switch pins 2 and 3
* switch pins 4 and 5
* switch pins 6 and 20
* all other pins (1,7,8, and any others) go to their own numbers

OTHER PRINTER DRIVERS

You can bypass NEWSCRIPT's printer drivers by choosing Attachment Method #3 during installation. This is done at your own risk: if the driver you use filters any characters, or converts line feeds (10's) to carriage return/line feeds, NEWSCRIPT will still work properly, but your printed results will be incorrect.

TRS-232

If you're using the TRS-232 interface on the Model I, you'll have to use your own driver instead of any built into NEWSCRIPT. To do so:

1. Specify Attachment Method #3 during Installation; you'll see this on one of the menus;
2. Use NEWSCRIPT's editor to add a command to "STARTUP/MIN". The command should be the name of your printer driver. If your driver is a machine language command, it should appear just before "BASIC" is referenced in "STARTUP/MIN." If your driver is "poked" into memory by a BASIC program, it should appear just after "BASIC" is activated. In this case, when it finishes, it must allow "NSINIT" to run. One way to do this is to see whether the chaining facilities of NEWSCRIPT allow both your program and ours to execute from "STARTUP/MIN." Another way is to end your "poking" program with "RUN NSINIT."
3. Your driver must honor the address in "High Memory" by not loading above it, and must update (lower) "High Memory" to protect itself when it starts. On the Model I, this address is at X'4049'; on the Model III, it's at X'4411'. Your driver must not load at the top of memory, since that's where NEWSCRIPT resides. PROSOFT will not modify or provide custom printer drivers if you insist on using your own, but cannot make it work with NEWSCRIPT.

Spoolers

If your operating system has a spooler, it may or may not work with NEWSCRIPT. If you specify Attachment Method #3, NEWSCRIPT simply leaves the current printer driver alone (unless it's in the same high memory area used by NEWSCRIPT, in which case, you'll have to move your driver). If the current driver has spooling capabilities, then your spooler will start to function when NEWSCRIPT begins to generate printed output. As we said before, no filtering or character conversions are allowed: the driver or spooler must print exactly and only what NEWSCRIPT generates. It must not add or change Line Feeds, and must pass hexadecimal zeros, not delete them. If NEWSCRIPT works correctly with your printer when our own drivers are used, but doesn't work correctly with other drivers, then either fix those drivers or don't use them.

ONE DRIVE AND SPECIAL DISK DRIVE CONSIDERATIONS

TDOS, the Operating System supplied with NEWSCRIPT, has provisions for one-drive configurations. The BACKUP and FORMAT commands support one drive, and "COPY1" allows individual files to be copied from one diskette to another using only one drive. These commands are documented in Section X, under "DOSPLUS." If you're using another Operating System, the method presented below will be relevant in principle, but not in detail.

Every "data" disk must contain a copy of the Operating System when you have only one drive. To maximize space on these disks, we suggest you create a "master" as follows:

Creating a Master Data Diskette for One-Drive Use

1. Make a BACKUP of the NEWSCRIPT distribution disk. Then, perform the installation procedure as explained in Section I. If you're using a Model I and have 40-track capabilities, follow the instructions for using "FORMAT" first; that'll give you access to all 40 tracks. Similarly, if you have one-sided 80-track drives, use "FORMAT" on the Model I or III first.
2. Make another BACKUP. It will become the "master data disk" and we'll call it "DATA" below.
3. Insert DATA in drive 0, press <RESET>, and hold down <ENTER> until "DOSPLUS" appears on the screen.
4. Type: AUTO VERIFY <ENTER>
5. Type: DIR <ENTER>
6. "KILL" every file whose name is displayed.
7. "KILL" each of these files (the passwords are shown):

   FORMAT/CMD.XANTH BACKUP/CMD.XANTH COPY1/CMD.XANTH
   TBASIC/CMD.VA6

8. Your "Master Data Disk" is ready for use. Remove it, insert the normal working copy of NEWSCRIPT, and type "BACKUP" with no drives specified. Remove NEWSCRIPT, insert DATA. When asked for the source and destination drives, specify zero in both cases. The destination disk should be a new disk that you're going to use to make a copy of DATA. Once BACKUP completes, you'll have two mostly-empty data disks, and you can repeat this process as often as you wish.
9. We suggest you keep the master DATA disk empty at all times for two reasons: 1) to make it easier to make more copies as you need them; 2) to have an available diskette when you need one in an emergency (when you get a disk error with one of your normal data disks while trying to save a file).

Using NEWSCRIPT On A One-Drive System

1. Always begin with the working copy of NEWSCRIPT in drive 0. If it's a single-density disk, don't try to keep any text files on it: there just isn't enough room. Instead, put a write protect tab on it as a precaution, and use the method described here.
2. Type "NS" if AUTO doesn't do it for you, and select EDIT or SCRIPT from the menu. Once the program loads in and asks for a file...
3. Remove NEWSCRIPT and insert a DATA disk. Then give a filename, and proceed to edit or print your file.
4. When you're done, and the editor has saved the file, or SCRIPT has finished printing it, remove the DATA disk and re-insert NEWSCRIPT.
5. Select your next action, which may be to re-run the same function immediately. You still have to put NEWSCRIPT in before doing this!
6. Continue switching disks in this manner. As long as NEWSCRIPT is in drive 0 whenever a function is selected, and as long as DATA is in drive 0 whenever a text file is selected, everything should work just fine.
7. Make frequent backups of your DATA disks. This would be true even if you had 99 disk drives!

80-Track Drives

TDOS can format up to 96 tracks on single-sided disks. Just use the DOS command, "FORMAT", and specify how many tracks your drives can handle. If you subsequently BACKUP a shorter disk onto one of these longer ones, TDOS will maintain the extra tracks.

Hard Drives

TDOS doesn't support hard drives, so you'll have to transfer NEWSCRIPT to your own disks using the methods given at the beginning of this section. Once you have done so, NEWSCRIPT should function without any problems. Note that the software support for the hard drive must reside below NEWSCRIPT, since NEWSCRIPT cannot "relocate" itself in memory.

Since NEWSCRIPT does all disk operations through BASIC or the Operating System, you can store NEWSCRIPT on a hard drive, and also keep your text files on the hard drive. The directory functions of NEWSCRIPT allow drive #4 to be specified for just this purpose, although it's possible that not all files can be shown on the screen. In this case, just use the DIR command from DOS itself.

Two-Sided Drives

TDOS doesn't support two-sided drives, although full DOSPLUS does this quite well. If you have such drives, you almost certainly have an operating system that supports them, so just copy NEWSCRIPT from the distribution diskette to your own Operating System, then perform the installation procedures described in Section I or this section.

ASSISTANCE FROM PROSOFT

All PROSOFT products include limited on-going support. If you have problems installing or using our software, or if you think there's a problem with it, we want to try to help you resolve the problem and answer your questions. We can't promise that all questions and problems will be resolved, but we can promise that we'll try to help you with all reasonable ones.

Instructions for sending disks and problem files to us are given at the front of this book, in the "Terms of License Agreement." If you decide to call, be sure to give us your name, the serial number of your copy of NEWSCRIPT, and the update level. The number and level are displayed every time the Primary Options Menu is displayed, and if you write them down in the front of this book, they'll be handy if you need them.

PROSOFT is located in California: when it's 9 A.M. in L.A., it's 10 A.M. in Denver, 11 A.M. in Chicago, and noon in New York. Our business hours (Pacific Coast time) are 9 A.M. to 5 P.M., Monday through Friday, and we're closed on holidays. We cannot accept collect calls or return long-distance calls. The Toll-Free numbers are just an answering service several hundred miles away, and no technical assistance is available from them.

If you have a quick question, the phone is a good way to get an answer, but if you have several questions or a complex problem, it's best to write instead of calling.

We welcome suggestions for improvements to our programs and to this book. All such suggestions become the property of PROSOFT, and we reserve the right to use them as we see fit.

Our address and telephone number are:

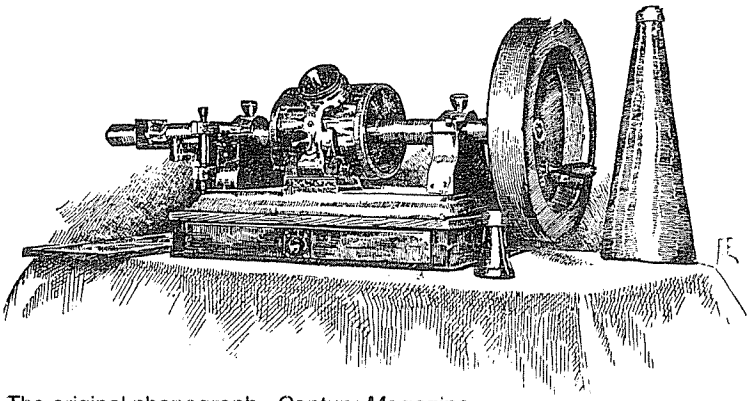
PROSOFT
Box 560
North Hollywood, Ca. 91603
(213) 764-3131

We think you'll 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



The original phonograph. *Century Magazine*

SECTION X

TINY DOSPLUS

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

The listed commands are described in the remainder of this section. If you've used TDOS 3.3 in the past, please note these differences:

1. two-sided drives are not supported by TDOS. If you have such drives, you probably also have an Operating System that supports them, so you can just copy NEWSCRIPT to that system and use it instead of TDOS.
2. "LIST", "FORMS", and "RS232" no longer exist. NEWSCRIPT has direct serial printer support.
3. The Directory display actually is a "CAT", or abbreviated display. "FREE" space can be obtained from option 6 of NEWSCRIPT's Primary Menu.

The following commands must be used from "DOSPLUS" level, and cannot be used while in TBASIC. There is no "BASIC" with TDOS: "TBASIC" replaces it.

AUTO	- sets a command to be used automatically on startup
BACKUP	- copies an entire diskette to another diskette
CONFIG	- changes the track-to-track stepping speed
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	- 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
KILL	- deletes a file from an on-line, unprotected diskette
RENAME	- changes the name of an existing disk file
TBASIC	- invokes Tiny BASIC for use with NEWSCRIPT

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:

AUTO NS

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.

The NEWSCRIPT distribution disk uses "AUTO" to display the "WELCOME" message. You should change this on your backup disks by setting "AUTO NS".

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 different 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, since the source disk is a "System" disk.

CONFIG

This is used to speed up or slow down the track-to-track stepping time of disk drives. Most modern drives can step every 6 milliseconds (ms.) or even faster, and this is the default NEWSCRIPT uses on the Model III. The default for the Model I is 40 ms. but most drives can go faster.

CONFIG :0 (STEP=n,SAVE)

":0" refers to the drive whose step rate is being changed, so you'll want to issue "CONFIG" for each drive in your system. 'n' is the step rate. On the Model I, it can be any one of: 40,20,12,6. On the Model III, it can be any one of: 30,20,12,6. The word "SAVE" is needed if

you want these new settings to be stored permanently on drive 0 (even if 'n' is being set for drive 1). You'll want to use "SAVE" to avoid having to set CONFIG every time you press <RESET>, so be sure to remove the write protect tab first. (Do this ONLY on a backup; don't "SAVE" to the original.) If you omit "SAVE", the new step rates will be used immediately, but will be lost on the next <RESET>.

Problems in Formatting or Making Backups

If you cannot format or make backups of disks with TDOS, its probably because your disk drive is spinning the disk too fast or too slow, a problem which is addressed under "Error Messages." However, if you have certain brands of disk drives, it may be due to a mismatch on stepping rates.

If you're using Shugart drives, TEAC drives, or pre-1981 Radio Shack drives, the 40 ms. setting must be used, but otherwise, you'll find that setting the step rate to 6 ms. will dramatically improve the speed of all disk operations.

Certain drives, such as TEAC, have relatively slow stepping rates, such as 23 ms. On the Model I, these drives will work if the distributed step speed is used. On the Model III, these drives may not work at all, or work erratically. If this is the case, try to make a backup of the original as follows. If it doesn't work, send the original back to us with a note asking for a slower step setting.

1. Put the original in drive 0 and a new disk in drive 1.
2. Hold down <ENTER> and press <RESET>.
3. Release <ENTER> when "DOS PLUS" appears on the screen.
4. Type: CONFIG :0 (STEP=40) <ENTER>
5. Type: CONFIG :1 (STEP=40) <ENTER>
6. Type: BACKUP :0 :1 <ENTER>
7. Answer the questions as they are asked.
   When backup completes, press <ENTER> to when asked for SYSTEM disk.
8. If formatting and copying succeed, put the backup in drive 0.
9. Type: CONFIG :0 (STEP=40,SAVE) <ENTER>
10. Disk operations from this point should be reliable.

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 modifies a Model I diskette so that TDOS can read it on a 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 (V13)

This copies all non-system files from a TRSDOS diskette to a TDOS diskette. If the TRSDOS disk was created by TRSDOS 1.3, then "V13", in parentheses, must be specified. If the disk was created by 1.1 or 1.2, don't specify the version, and don't include any parentheses. If future versions of TRSDOS become available and CONVERT doesn't seem to work in one of these forms, try the other form.

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, but "COPY" will do so, one file at a time.

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

This displays the directory on any available drive. Only an abbreviated directory is shown: the file I.D.'s are listed, but the space they use is not. The command is: DIR :d

If ':d' (drive number) is omitted, drive zero is assumed.

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.

"DO" cannot be used when NEWSCRIPT is active.

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 does not use a 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. Double-density is not supported by Model I TDOS,, but is the default for the Model III. If you are creating a TRSDOS-compatible diskette, be sure to specify 'single' and '35 tracks' 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'll prompt you:

INSERT SYSTEM DISK, PRESS <ENTER>

As long as NEWSCRIPT or TDOS is in drive 0, a "SYSTEM" disk is already in place, so just press <ENTER> as requested.

Using 40- and 80-track Drives

When formatting a disk, you may specify any number of tracks between 35 and 96. You cannot specify more tracks than your drive can handle, and TDOS 3.4 cannot format or support two-sided drives. If you're using a Model I, be aware that virtually all disk drives manufactured after January, 1981, and many before then, can handle 40 tracks, even if they were purchased from Radio Shack. If you've purchased 80-track drives, you can use all 80 tracks.

Formatting a data disk (drive 1,2, or 3) to 40 or 80 tracks is obviously easy to do. However, you can also do this for a TDOS System (drive 0) disk. Before making a BACKUP, run FORMAT, and specify the number of tracks your drive can handle. After FORMAT completes, run BACKUP to copy your system disk. You'll be told the destination disk contains data (since you just formatted it), so reply "Y" to use it anyway. The backup will occur, and TDOS is smart enough to recognize that the destination disk has additional tracks. When it finishes, the new backup will be able to use all those tracks! Thereafter, a simple BACKUP of that bigger disk will suffice to maintain the creation of the extra tracks. On a Model I, the extra 5 tracks give you 14% more disk capacity at no extra cost. Remember, the disk drive must have 40-track mechanical capability, and the earlier drives sold by Radio Shack, Shugart, and some others were mechanically limited to 35 tracks.

KILL

This is used to delete a file from a given diskette. The space used by the file is reclaimed, and the command format is the one used by TRSDOS. If a file has a password, you must supply that password for this command to work. NEWSCRIPT doesn't use passwords, but the Operating System does. The Operating System password for its utilities is "XANTH". We don't know the password for the "SYSTEM" files, and don't want to know them. Killing a system file will make the diskette unusable.

KILL filespec

RENAME

This allows you to change the identification of any file. The new name must not already be in use on the same diskette, and if the file has a password, you must include that password in the old specification. The command format is:

RENAME oldid:n newid

'oldid', of course, is the current name, and 'newid' is the name to which it should be changed. 'n' is the drive on which the file will be found. 'n' can be specified as the drive number for 'oldid', but must be omitted for 'newid', since the file isn't being copied, only renamed.

The disk containing the file must not be write-protected during this operation, since the Directory is changed by the command when the new name is stored.

Example

```
KILL STARTUP/MIN
RENAME TRSDOS/MIN:1 STARTUP/MIN
```


TBASIC

This invokes Tiny BASIC. Portions of NEWSCRIPT run under control of BASIC, and TBASIC must be in control when using NEWSCRIPT. Remember that 'NS/CMD' 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 NSINIT-F:4-M:59000

'filespec' identifies 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 zero. NEWSCRIPT requires 4 files and TBASIC will not prompt you for 'FILES' or 'MEMORY SIZE', so both must be specified here if needed. If the "HIGH MEM" internal value is updated by a machine language program before entering TBASIC, the "-M:nnnnn" can be omitted. NEWSCRIPT sets this value, so it is sufficient to say:

TBASIC NSINIT-F:4

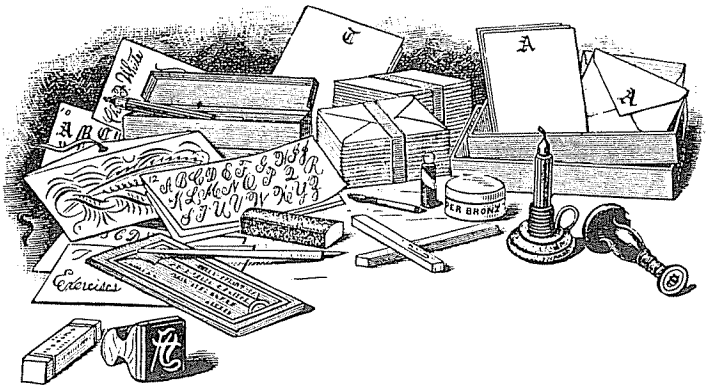
'-M:mmmmm' specifies the protected memory address. It is not needed with NEWSCRIPT unless you add a printer driver of your own that doesn't set the "HIMEM" address.

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 paid a special license fee to Micro-Systems Software for the privilege of using it.) It makes life far more pleasant for us, and we hope, for you also. You don't have to use TDOS if you prefer another Operating System, and you haven't been charged for it.

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.



Complete outfit. *Youth's Companion*

SECTION XI

FITLINE AND OTHER UTILITIES

NOTE: This section is reserved for descriptions of extra support programs that supplement NEWSCRIPT. The number of programs described will vary from time to time, and not all of them are provided with all copies of NEWSCRIPT. Except for "FITLINE", any extra programs in this Section are not part of the NEWSCRIPT product, and no on-going support or maintenance will be provided with them. They may be present on some disks but not others.

FITLINE

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, and is an alternative to using "PUTFILE" in EDIT.
2. To "flow" short lines of text together, while honoring the SCRIPT control words. This would be done for cosmetic appearance while editing, and has no effect upon how SCRIPT will format or print the text.
3. To convert SUBSCRIPT files to NEWSCRIPT format. This only applies to the predecessor of NEWSCRIPT.

FITLINE is written in BASIC. It reads an existing file and writes one or more new files. 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 the output by creating several small files. As it performs this conversion, it takes account of the meanings of any 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. Bring up NEWSCRIPT, if it isn't already running, and select option zero (exit to BASIC).
2. Type this: RUN "FITLINE" (press <ENTER>)
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. If there isn't enough room, an error will occur and FITLINE will terminate. In this case, start over, with a disk that has enough room on it. The amount of room needed is slightly more than the size of the original 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 150-250 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. After FITLINE is done, you may rerun it on another file, or run "EDIT" again to resume your editing.

LIST

This BASIC program displays NEWSCRIPT text files on the screen. You can vary the speed of the display by using the numbers 1 through 9 (1=slowest, 9=fastest) as the text is shown, pause temporarily by pressing the at-sign (resume by pressing <ENTER>), or end the display by pressing <CLEAR>.

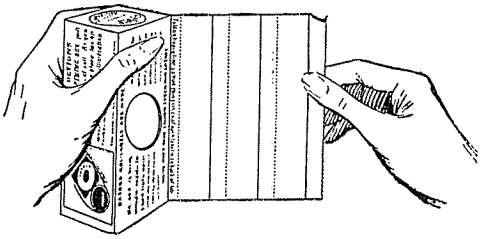
LIST has an option to allow you to interactively split a large file into smaller ones, but doesn't include the line-length adjustment capabilities of FITLINE.

The program is included because TDOS has no LIST function of its own. It requires one or two BASIC files, and prompts for the I.D. of the file to be listed. LIST is a quick way to look at a file before deciding whether or not to EDIT or SCRIPT it. It is not provided with all copies of NEWSCRIPT.

APPEND

This BASIC program concatenates (connects or appends) several files into one larger file. The original files are not altered in any way. It contains its own instructions, and prompts for each file as it needs it. There's no upper limit other than disk capacity to the size of the resulting file. Be aware that EDIT cannot accept a file longer than 400 lines or about 13,000 characters, so don't use APPEND and then expect EDIT to read the results in all cases.

The purposes of the program are to connect several Tables of Contents together when a multi-part document is printed in stages, and to create a large Name and Address file for the Form Letters feature. At least two BASIC files are needed. It is not provided with all copies of NEWSCRIPT.



SECTION XII

ADDRESS MAILING LABELS

"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 address, or 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. How each of these capabilities is used will be explained below.

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. It isn't intended to replace a mailing list manager, but rather to extend the use of Form Letters when a list grows too large to justify the time needed to print customized letters.

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.

The option is also useful in printing index cards, which for all intents and purposes are just large labels.

INSTALLATION

The LABELS program is written in BASIC and distributed on diskette. If you ordered it at the same time as your ordered NEWSCRIPT, it'll be on the same disk as NEWSCRIPT; if you ordered Model I NEWSCRIPT with LABELS, "LABELS" will be found on Side 2 of the distribution disk.

If you ordered the option separately, it'll be on a disk of its own, which always will be in single-density Model I format. If you have a Model III, remove the write protect from the LABELS disk, insert it in drive 1, insert TDOS in drive 0, and type:

CONVERT :1

This makes the diskette readable on the Model III. Copy LABELS from the distribution diskette to a disk of your own, then save the original disk for backup and possible updates from PROSOFT.

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):

VARIABLE	DEFAULT	DESCRIPTION
CF$	S	C=continuous forms or S=single sheets
NC	3	# labels across a page (1-up, 2-up, etc.)
NR	10	# labels down a page (single sheets only)
NL	6	# printable lines per label
VS	0	# lines between each label (vertical gap)
SP	1	start position of first label (for left margin)
NP	25	# print positions/label line
NG	3	# characters between labels (horizontal gap)
CS$	@	code signal for code records

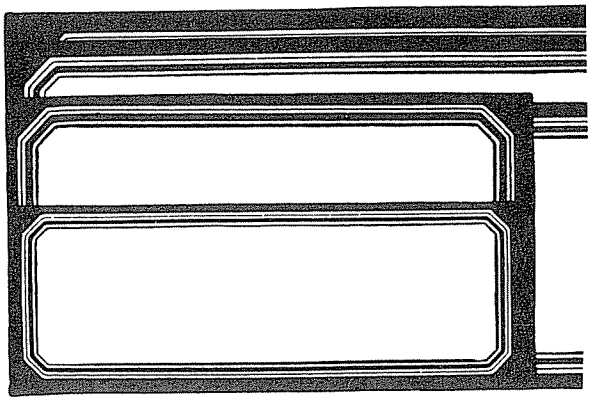
The most common problem people have in using LABELS is the proper selection of a "code signal." NEWSCRIPT's Form Letters facility (".RD") defaults to use of the at-sign ("@") for this purpose, and so does "LABELS". If you pick a different character, such as the pound sign ("#"), you must override the default each time you run "LABELS", or else change "CS$" in line 60 of "LABELS", and then save the changed program.

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, which will be ignored if encountered.

Example of an Address Mailing List

```
@
Mr. Tom Jones
123 Maine Street
New Haven, Ct.
@
Ms. Leslie Brown
Seaside Realty, Inc.
4567 Winding Hill Lane
East Sandy, Ut
@
Mr. Roger Brown
RFD 3, Box 11-A
Stoggert, Wy
```



OPERATING INSTRUCTIONS

1. Prepare a mailing list in the format shown above. 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. Make sure your printer and labels are ready. When you try the program at first, you may want to use plain paper to see what happens.

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 misspellings, 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 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".

Punch

APPENDIX A

WRITING WITH A WORD PROCESSOR

Man is a rather unique animal, particularly in his ability to manufacture tools that help him in a variety of ways. A pencil is a very good example of a tool that helps him think. It frees his mind of having to keep a lot of information in his head, so that he can get down to the job of thinking.

A computer is a very similar tool. It is infinitely more complex, but serves the same purpose, none the same. It is, just as is the pencil, a **mind tool.** If you use your computer only to record facts and dates, and to recall these pieces of information when necessary, you really have not moved up from the utility level of a pencil. A super-pencil, to be sure, but still a pencil. You are not exploiting the incredible power of the computer that is available at your fingertips. This short appendix is an attempt to help you explore some of this potential.

Obviously, this is a broad topic, and since this is a word processing manual, we will stick to a very narrow category of computer utilization - the art of creative writing. It is our thesis here that the computer is qualitatively different than a pencil, and so ought to be used in a qualitatively distinct fashion in the creation of literary works, both fiction and non-fiction.

The process that most people go through when writing a document is one of successive stages of approximational organization. That is, they start out with a general idea and they make an outline. This normally takes the shape of the sample outline below:

TOPIC: Mathematical Models of Neurological Function

I. Why model ?
   A. inexpensive and simple to predict behavior
   B. some experiments are not technically possible
   C. some experiments are not healthy for subjects
   D. theory testing by prediction/result comparison
   E. theory suggestion by model results
II. Problems with models
   A. do not give direct evidence or proof
   B. difficult to determine proper equations from theory
   C. difficult to determine when oversimplified
III. Past models of neurological function
   A. Marr-Hildreth Gaussian Filter model of edge detection
   B. Hodgkin-Huxley model of Giant Squid axon
   C. Triffet-Green model of information transfer in Leech
IV. Developments for the Future
   A. better mathematical methods for modelling
   B. better experimental methods to test models
   C. better understanding of neurological function

Other than the specialized nature of the subject, this outline for a presentation on mathematical modelling of neurophysiological phenomena is similar to other outlines you have no doubt seen before. So far, no real advantage has been gained by typing it into your computer, other than simple correction ability.

However, if typed in on a piece of paper rather than a computer, this outline would have to be discarded when going to the next level of preparation. You cannot conveniently "slide" the lines apart on a piece of paper to fit more lines between them. You must start on a new page, and re-write many words and sentences as you do so. The outline begins to grow and grow, and more and more paper is discarded as the presentation progresses. Finally, you get the end result. It may not be satisfactory, but the mechanics have taken so long that you're probably out of time, and have to hand in what you've got when the deadline arrives.

Approached in a traditional manner, the progress is rather strict and linear in nature. It is not easier to develop one section rather than another, so your writing tools in some sense determine the progress of the final document. Note that this applies to fiction, where plot and character development are the main concerns, as well as to non-fiction. It certainly applies to sales proposals, office procedures, and resumes.

The word processor frees you to write in a more flexible style. As additional ideas or material become available, you can insert lines or imbed files of abstracts, and in this way expand the document in a non-linear fashion. The medium (the computer) is somewhat more complex than the pencil, but allows incredible flexibility in the development of ideas. And, of course, it frees you from having to rewrite or retype everything just because you decided to stick one more paragraph on page two.

By starting with an outline that is so easy to change, you can get the basic ideas down quickly. Next, you can move topics around until the organization and flow make sense. Then, you can start to view the outline as your Table of Contents. By using imbedded files, the outline becomes a "master file": it doesn't contain any text or explanation, just the names of other files that do contain text. Each of those files corresponds to one of the topics in the outline. If a topic is large, you may need several imbedded files to cover it all, but the structure of the outline and document won't be affected by this.

Having transformed the topics into "imbeds", you can write each subject in whatever order is easiest for you. You can skip around if you don't have all the material you need, or if you just aren't inspired in one area, but can't wait to talk about another one. Each piece can be printed separately as soon as it's written, and once proofed and finished, doesn't have to be handled again until the final printing of the entire document.

If you decide to change the organization of the manuscript part way through, all you have to do is move the "imbed" references around within the master file. The document files themselves don't have to be touched or changed. We did this repeatedly while writing the NEWSCRIPT manual.

Another useful trick is to put the chapter headings and titles in the master file. That way, if you move topics around, it'll be much easier to change the chapter numbers, since you won't have to edit the imbedded files, but only the master.

Getting back to the "Mathematical Models" example, these techniques were used extensively. Some pre-written material was available, so it was possible to imbed it into the outline file, and used to expand the various points. In this way, I was able to use NEWSCRIPT to help my presentation "evolve" in the manner I wished.

It has been well established that people do not normally think in linear fashions. Differing levels of interest in the various aspects of the topic mean that people will naturally develop those parts of the topic they find the most interesting, or the most accessible. Further, it may well be that the topic demands a sequential approach to its study, as in learning a totally new subject, but the presentation demands a totally different approach.

As an example of this, suppose that you wanted to give a presentation on the above topic, but knew little about it. You would have a great deal of trouble, I suspect, in learning the topic in the manner that the presentation is outlined. You might know the mathematics, but not the neurophysiology and experimental methods. Then you would have to spend more time learning the latter before you could even begin the outline. Or, you might know the neurophysiology, but not the mathematics, such as Gaussian filters and convolution integrals. You would need to learn those subjects before being able to begin the outline.

With a word processor, however, you can begin the outline and develop it in which ever direction it needs to go, and still aim the document at the presentational organization above. You can easily fill here, insert there, delete here, and so on, and save a considerable amount of time over the traditional pencil-and-paper approach. That's why we said that the flexibility that your computer bestows on topic development is qualitatively different, not just faster, than the traditional methods.

Conclusion

The power of a computer lies not just in its ability to store information, but in its flexibility of retrieving and restructuring that information once entered. Properly used, a computer can become an extension of the human mind, providing the mass storage of dull facts, the rapid retrieval of them, the lightening fast and accurate calculations, and the presentation of results in visual form. This gives us time to think and be creative, and not get bogged down in the mechanics of storing and searching, counting and writing. People are very good at being creative, and the computers are very good at rote tasks.

It seems like a fair way to divide the work.

APPENDIX B

FOR FORMER SCRIPSIT USERS

This section is mainly for the people who have become accustomed to the approach of SCRIPSIT, ELECTRIC PENCIL, and other similar word processors. We have found a few people have purchased NEWSCRIPT with the idea that it was just like SCRIPSIT, and so we wish to make sure that the similarities and differences in the two approaches are well understood. As this is being written, "SUPERSCRIPSIT" is not yet available, so no information can be given for it.

People who have never worked with a Word Processor before generally find NEWSCRIPT quick and easy to learn. By comparison, people who have a SCRIPSIT background often find NEWSCRIPT hard to learn, even though they have more experience with computers and word processing than do the novices.

Most of us find learning new things difficult. Once we master a way of doing something, we tend to resist learning other ways, regardless of whether those other ways are better or worse than the old ones. All of this applies in spades to Word Processing. The computer, the DOS, and the Word Processor probably all came into your life at the same time. They caused great confusion and frustration, and only slowly did they begin to live up to their promise of "speed, flexibility, and reliability." (Well, I don't know whether they did or not, but when I started working for IBM in 1962, that was a great marketing phrase.)

If you started with SCRIPSIT, you just had to accept that its way of doing things was the right way, in fact, the only way. Now, you're starting to learn NEWSCRIPT, and instead of approaching it from scratch, you're likely to be comparing it with SCRIPSIT and wondering why we changed things. The answer is: we didn't. Other than the fact that they both are word processors, any similarities are accidental.

NEWSCRIPT doesn't work the same way SCRIPSIT does. It never was meant to do so, since it's based on an entirely different word processing philosophy, method, and approach. The "SCRIPT" family of word processors dates back to the mid-1960's, and the fact that Radio Shack chose to call their word processor by a similar name has caused a certain amount of confusion. With all that in mind, let's highlight a few of the differences.

Similarities and Differences

SCRIPSIT combines its editing and text formatting functions into a single program, and can only handle documents that fit in one file. For small documents, this is a good approach, but it leads to problems if you need to write something that is too large to fit in memory all at once, or want to extract portions of several old documents and piece them together into a new one.

The approach taken by NEWSCRIPT is to break up the various word processing functions into separate programs, and then to transfer of control from one program to another within the NEWSCRIPT system. In that way, NEWSCRIPT can support a large number of editing features and a large number of text formatting features. Then, by letting you "append" (chain) files together, and even span multiple disks, NEWSCRIPT allows the creation of documents of almost unlimited length. If you never expect to write anything longer than a page or two, most of this won't be needed, but the ability to "imbed" standard material still is very useful.

Now that you understand that on a fundamental level, SCRIPSIT and NEWSCRIPT are different and take different approaches, we can begin to discuss how NEWSCRIPT works.

The Editor

The NEWSCRIPT text editing program, EDIT, and the SCRIPSIT editor support many of the same types of commands. Both have a repeating cursor and both use the arrow keys on the keyboard to move the cursor around. SCRIPSIT uses the <@> key for the control key while NEWSCRIPT uses either the <CLEAR> key or the <CONTROL> key, and "locks" either one for one keystroke to allow one-handed operation. NEWSCRIPT relies heavily on English-language mnemonics for all its commands, and its control keys are not positional on the keyboard. Instead, almost all the control keys were chosen so as to correspond with the first letter of the function being used. This makes them very easy to learn, remember, and use. NEWSCRIPT's <CLEAR><I> turns on the insert mode, and which stays on until you press <CLEAR><I> again. There are other similiarities in the editor with both systems, but more important are the differences.

The NEWSCRIPT EDIT screen is broken up into three areas; a command line at the top of the screen, a line manipulation area (LIMA) at the left side of the screen, and a data area, which occupies most of the video screen. NEWSCRIPT has many commands in each of the three categories of character-oriented, line-oriented, and global commands. A table of edit command similarities and differences appears below, but NEWSCRIPT has many additional editing capabilities besides these:

Command	SCRIPSIT	NEWSCRIPT
cursor movement	four arrows	four arrows
to beginning of line	<SHIFT><LEFT ARROW>	<SHIFT><RIGHT ARROW>
to end of line	<SHIFT><RIGHT ARROW>	<SHIFT RIGHT ARROW>
to beginning of file	<SHIFT><UP ARROW>	<CLEAR><T>
to end of file	<SHIFT><DOWN ARROW>	<CLEAR><E>
delete character	<@><D>	<CLEAR><D>
delete word	<@><D><@><Z>	<CLEAR><W>
global delete text	<@><R><BREAK><D>text	C/text// **
delete line	<@><D><@><X>	LIMA "D" OR "DEL" command
insert character	<@><S>	<CLEAR><I>
insert line	<@><S><@>X>	LIMA "I" or <CLEAR><BREAK>
delete to end of text	<D><DOWN ARROW><Y>	"DE *" command
mark block	<@><Q><name>	<.A><.B><.C> in LIMA
move block	<@><S><@><Q><name>	"MO .C or .B" on command line
delete block	<@><D><@><D>	"DE .B" on command line
print text	<BREAK><P>	"H" on command line or run SCRIPT
save file	<BREAK><S>	<CLEAR><S> or SA on command line
load file	<BREAK><L>"name"	GET fileid on command line, or: <1> from main menu then 'fileid'
<SHIFT LOCK>	<SHIFT><@>	<SHIFT><ZERO>
find 'text'	<BREAK><F>text	/text/, L/text/, or LU/text/
replace 's1' by 's2'	<BREAK><R>s1>s2	C /s1/s2/
video width	<BREAK>W=x	V x
free memory	<BREAK>?M	FREE (on command line)

Text Formatting

SCRIPSIT formats the text as you type it in. When you want a paragraph, you press the paragraph key and SCRIPSIT starts a new paragraph. The text will be somewhat differently formatted when it is finally printed, but the point is that with SCRIPSIT you can see an approximation as you type. NEWSCRIPT does not format the screen as you type, so you must save the text to disk (a good safety precaution with any word processor), and then use SCRIPT, the text formatter, to format it. You can output the formatted text to the screen from SCRIPT, but printer-dependent formatting won't be shown. This is because the TRS-80 screen can't show things like boldface, underlining, and italics.

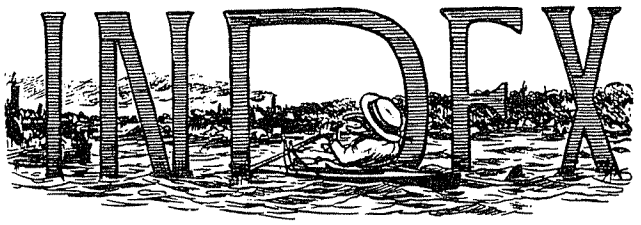
NEWSCRIPT uses a slightly different approach to tell SCRIPT how to format the text. All formatting control words are two characters long and begin with a period. This is similar to the SCRIPSIT format control lines. Some of these commands require additional parameters, some have optional parameters, and some do not have parameters. A control line may contain more than one control word (there are some exceptions, however), and if this is done, then a semicolon must be used to separate the control words. Thus, '.in 10;.pp' tells SCRIPT to INdent 10 characters and begin a new paragraph. A short table of some of the SCRIPSIT and SCRIPT text formatting control words follows:

Function	SCRIPSIT	NEWSCRIPT
page length	>PL=xx	.pl xx
left margin	>LM=xx	.lm xx
right margin	>RM=xx	.ll xx
top margin	>TM=x	.tm x
bottom margin	>BM=x	.bm x
line spacing	>LS=x	.ss or .ds
justify on	>J=Y	.ju on
center on	>C=Y	.ce on
indent	<BREAK>I=x	.in x
chain files	<BREAK><L,C>	.ap file or .im file
page numbering	>PN=x	.pn x
pause during print	<BREAK><P,P>	.st

Note that some of the similar-appearing control words work very differently. For example, "BM" in SCRIPSIT is the number of lines down from the top of the page, while in NEWSCRIPT, it is the number of lines to be left at the bottom of the page. So, a one inch bottom margin in SCRIPSIT would be ">BM=60", while in NEWSCRIPT it would be ".BM 6".

A few other differences should be noted. In particular, SCRIPSIT relies heavily on the concept of "blocks", while NEWSCRIPT barely knows what a block is, except for moving, copying, and deleting them. NEWSCRIPT allows all dimensions (line length, margins, etc.) and titles to be re-defined at any time. NEWSCRIPT cannot vertically center text on a page. And, besides having a large number of formatting control words for special purposes, NEWSCRIPT uses over a dozen "escape sequences" to make it easier to create special effects within text. If you've used Acorn's SUPERSCRIPT modification, then you're already familiar with this concept.

If you bear some of this information in mind as you go through the tutorials, we think you'll find that NEWSCRIPT can meet your Word Processing needs for most purposes, large or small.



SERVICE DIFFICULTY REPORT

SOFTWARE PACKAGE: DATE:

NAME: _____

ADDRESS: _____

CITY: _____ STATE: _____ ZIP: _____ PHONE: _____

_____ DISKETTES ENCLOSED: (Y) (N)

TRS-80 MODEL I ( ) MODEL III ( )

MEMORY: ( )48K UPPER/LOWER CASE (Type): _____

Expansion interface if other than Radio Shack: _____

Operating System: _____ Version: _____

Zaps or patches thru (Date or ZAP no.): _____

Disk drive units (Mfg): _____ Model: _____ No.: _____

Printer (Mfg.): _____ Model: _____ Type: _____

Additional Hardware Modifications (List mfg. & model): _____

_____

_____

I have had my computer system approximately_____years, _____ months. I have had my disk system _____ years, _____ months.

Description of problem:

Circumstances under which it occured:

OFFICE USE

Recommendation:

By: Date:

Action:

By: Date: ?:

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

DBA NEW DIRECTIONS
RT. 1, BOX 72, DEL MAR, CA 92014

Dear Customer:

Thank you very much for your order. Enclosed is documentation and a diskette for NEWSCRIPT Release 7.0. 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'll find you've selected one of the finest Word Processing systems available today, regardless of price or computer.

Please record the disk registration number, and send in the Registration card as soon as possible. Once we receive the card, we'll be able to notify you of any changes or updates. If you have a Model I, the distribution diskette is two-sided, and you can read the second side by flipping the disk over.

NEWSCRIPT has been sent to you on a ready-to-run "system" disk, including a tiny version of the excellent DOSPLUS 3.4 Operating System. The first thing to do with the distribution diskette is to make a backup of it (use two disks to copy the two sides of the Model I version) as described in the installation instructions. This Operating System is supplied at no charge to you; if you prefer to use NEWSCRIPT with some other system, please see the instructions in the green supplement.

Our software Catalog describes other products that may be of interest to you. I'd particularly like to call your attention to the Electric Webster spelling checker, the Graphics Editor, and two handy programs that have proven very useful to many people: FASTER and RPM. If you need more information on any of these, please get in touch with us.

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

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

Best Wishes,

Debbie Tester
Customer Relations

New Directions
Route 1, Box 72 Del Mar, California 92014 (714) 755-3043

UPS: 6406 Shaw Ridge Road

Supplementary NewScript Notes

NewScript is a very powerful word processor. At first glance it might look overwhelming. It's not. The manual is by far the best combination tutorial and reference manual that I have seen for any microcomputer word processor. You might want to improve it a little though, by inserting a few index-tab section-separator pages in it -- particularly at the page where the index starts, and at Section III (the Editor reference section) and Section IV (the Script reference section.)

And, remember -- just because it has over 50 "edit commands" and about 50 "script control words" does ***not*** mean you have to learn them all immediately or *ever* to be able to get excellent use out of NewScript. The manual is arranged so that simply working though the 60 page introduction (from page 24 to 84) gives you enough control to be able to do damm near anything you want. Some of the advanced features can accomplish the same things slicker and quicker (things like automatic numbering and indentation of sub-paragraphs, automatic creation of an index, etc.). Many of the advanced features will be well worth learning for you, others you will never use. One friend of mine said he uses only about 15% of all the power in NewScript (he doesn't write much). I've been using it heavily for 8 months, and I understand about 95% of it's features, but usually use no more than half of its powers. Even if you use only 10% of it's features, *if one of those features is really useful to you -- as several are bound to be -- that is the real significance of "having all those features you don't use."* And remember, there are often several different ways to get anything done in NewScript -- so you don't have to learn every possible way of doing it -- just one that you like.

Most of the following discussion is based on the supposition that you run your system like this: (1) You have two single sided 40 track double-density disk drives (standard Radio-Shack configuration). (1) You run with NewScript in Drive 0, *write protected*, and the document (data) disks go in Drive 1 (unprotected of course). (3) When you want to proof you remove the NewScript disk, insert the Electric Webster (or MicroProof) disk in drive 0, leaving the document disk in drive 1. (Incidentally, MicroProof needs much less disk space than Electric Webster and things aren't nearly as tight with it.) IF YOU HAVE DOUBLE SIDED DRIVES, 80-track drives, or put Electric Webster on a *third* drive (then it wouldn't need a operating system, or NSINIT, or STARTUP/MIN on its own disk), the the description of solving the disk space shortage for ELECTRIC WEBSTER probably does not apply to you.

LOGO FILES. This is my brainchild (though I'm sure other's have thought of it.)

NewScript has a default page layout. You can, of course, change the page layout at anytime you are writing by entering the script control words. And in versions of NewScript released after about August 1982 you can even change the ***default*** page layout itself, at the time you initialize or reinitialize from the main menu.

However, it is still worth creating a few "logo files" which contain your logos (return address, etc.) and your favorite page layouts and other options; that way you don't have to type them in each time, simply imbed them. It works like this: Name a file

"LOGOnew1". Enter in it just a few lines with your logo and control words, for example---

```
.cm <THIS IS logoND1. It prints a logo and set up the page
.cm layout the way I like it.>
.ll 70;.ad 5;.hs 0;.tm 0;.fs 0;.bm 3;.hy  on
.ce 2;.fo off
!(New Directions!)
.br
Route 1, Box 72        Del Mar, California 92014        (714) 755-3043
!*UPS!:: 6406 Shaw Ridge Road
.fo on;.ce1;.cm <this "ce1" would center the next line>
.CM ***THIS PRINTS a 70 character centered line of "_"'s
.fo off;.tr 128,191,32;.ce on
```

---would be one example, and it could be named LogoND1. Actually, it would be wiser to not put the graphics line in the logo, it could be in it's own tiny file called "Line70".

Then, from then on, to get your favorite page layout, logo, etc., all you have to do is type--
".im LogoND1", or ".im LogoND1" then ".im Line70", as the first lines of your page! That's how the logo on top of this sheet was produced. Of course the file LogoND1 would have to be present on every disk that you printed documents from using it. A "logo file" or a "layout file" each usually occupy only one gran of storage.

INITIALIZING: When you first initialize I suggest that you answer "50" when it asks you what you want to set the "autosave" default value to be. That means every time you type 50 lines, or make 50 changes, it will autoSAVE the work you have done up to that point onto disk. If, later, you find that that is annoyingly too often, or not often enough, you can reinitialize and re-set the autosave default. Of course you can always temporarily reset the autoSAVE (for the duration of working on one document) by typing "AUTO 60" or "AUTO 20", etc., on the Editor command line.

HELP: Despite the excellent manual, sooner or later you will find something that "doesn't seem to work","seems to be broken (a true hardware failure or software bug)", or you just can't quite understand. If so: (1) Don't panic. That just encourages the computer to rebel. (2)Simply try the same thing again -- it may be a temporary condition that will never be traceable, and will never occur again. (2b)If, you need to get it done *right now* -- or you really don't need to find out *why* it isn't working -- try doing it some other way. (For example, there are at least 5 different ways that you could use to delete a paragraph). (3) Reread the quick reference card and/or the reference section of the manual. (4) Read the section VII on ERRORS. (5)Call me. (6)Call or write ProSoft (they're pretty good with help -- see page 235 in manual).

Here are a few typo-type errors that I've created in the past which were particularly hard to notice. Maybe this will save you a little grief:

MEANING or ERROR EXPLANATION

Right:.in3;.ll 70;.pp(Indent 3,line=7.0 inches,start paragraph)
Wrong: .in3;.ll 70;.pp(The ".in3" must be at *first* space on line)
Wrong:.in3; .ll 70;.pp(The ".ll 70"must be directly after the ";")
Wrong: in3;1170;.pp (Just missing a "." in front of "in3")
Wrong:.pp;.in3;.ll 70 (Normally the order of control words

doesn't matter, but ".pp" is one of a few that must be last on line. Others past ".pp" are ignored. If in doubt, put it on its own line.

FILE NAMING: There are many conventions, here's one set:

WILSON82/JUN	for a letter sent to Wilson in June of 82.
WLSNIN82/JUN	for a letter to Wilsonian in Jun 82 (drop vowels)p
LECTURE/A	for version one of a document called Lecture.
LECTURE/B	for version two of a document called Lecture.
LOGOAC1	for version 1 of logo or format file called AC
CORRECT/TXT	is what ELECTRIC WEBSTER names it's corrected copies of your old file.(just so you know if this file shows up somewhere on a disk.)
OLDNAME/BAK	is what ELECTRIC WEBSTER renames your old unproofed version of OLDNAME/xxx if you ask it to keep a backup copy (-- I never do ask).

LIMA COMMAND BUG: A known/suspected bug: As of August 26, 1982, release 7.0, I have encountered an occasional problem with the LIMA commands "D<n>" and "I<n>" (LIMA insert-line and delete-lines command. Occasionally, for no reason I understand) - entering, say,"D2", "i5", etc. fails to have any effect. If that does happen I can usually get the LIMA to accept the command by going up to the command line and entering any other command (one creating actions of not much consequence such as FREE, or AUTO, or DIR). I expect Prosoft to track down this bug eventually, and it only occurs infrequenctly.

OTHER OPERATING SYSTEMS: In addition to all the others, you can run, you can also run under tiny-DOSPLUS 3.3 (TDOS 3.3). Versions of NewScript prior to 7.0 ran under 3.3, and I understand newer versions still will.

There are several reasons that you might want to do this: (a) TDOS 3.3 is not quite as stripped of features as TDOs 3.4. It has LIST, LIB, FREE, a more complete DIR display, etc. -- which TDOS 3.4 does not have. 3.3 directly supports the operation of two sided drives (as does the *full* DOSPLUS 3.4). (b) If you need to operate on some files that themselves must be read by 3.3. If, for example, you own Maximanager under TDOS 3.3, and you want to use NewScript to edit Maximanager's "Docufile" files, which control it's printing--) you may have to run NewScript under 3.3. I have had some problems that seem to suggest that files written by DOSPLUS 3.4 are not completely compatible with DOSPLUS 3.3.

There are several reason you might want to stay with TDOS 3.4.: (a) NewScript comes with it and runs great under it. (b) 3.4 is slightly more tolerant of disk problems and drive problems. If you tend to get a lot of disk read/write errors messages, 3.4 would be better. (c) 3.3 takes more space than 3.4. This is not a serious problem on the double density NewScript disk, but if you want to strip NewScript to essentials (--say to fit GEAP, the graphics editor, on the same disk with it--) then that may be important. In this connection, if you want to interface Electric Webster with NewScript, there is *just* barely space to cram all of Electric Webster on one side of a TDOS 3.4 system disk after stripping the TDOS 3.4 of FORMAT/CMD, BACKUP/CMD, COPY1/CMD. I have not added up the free space on 3.3 -- but am reasonably sure you could not fit all of ELECTRIC WEBSTER on one DD TDOS 3.3 system disk. I have not tested, but don't think you could run NewScript on 3.3 and Electric Webster on 3.4. So, in short,

if you use Electric Webster, you probably should run under 3.4.

Of course there is absolutely no reason that you can't put one or two backups of NewScript under TDOS 3.3 and use it only when 3.3 is desirable.

ELECTRIC WEBSTER: As mentioned, I suggest running Electric Webster all on one disk with TDOSPLUS 3.4. If you kill everything unessential on that disk (NewScript's DO file called "EWINSTAL/BLD"-- with an *extra* copy of NewScript type "DO EWINSTAL" from DOS -- will do all this for you, or I can configure it for you) there will be just 3 granuels of excess space left on the TDOS 3.4 disk after installing Electric Webster. It can be run unprotected. As you slowly build up the size of your dictionary of "added" personally-relevant words, it is possible that the personal dictionary file (DICT3/EW) will grow beyond that three granuels of excess space. At that point -- one day while adding new extra words to the dictionary while running Electric Webster -- you would get some kind of "Disk space full" message as the "ADDTODIC" section of Electric Webster tries to do its job. Your choices then will be: 1)Create a newer smaller DICT3/EW (see the Electric Webster manual on adding words to the dictionary). 2)Copy DICT3/EW From the Electric Webster working disk onto your NewScript *document* disk; kill DICT3/EW on the EW working disk; then write protect the EW working disk. The only drawback of this second solution (-- and the reason I don't recommend simply starting out with DICT3/EW on your document disk --) is that you will, like your logo files, have to copy it onto every document disk that you might want to proof with Electric Webster. And and if you add several new words on DICT3/EW on one disk they will not automatically appear on all the other document disk copies of DICT3/EW. In general, if you start out using Electric Webster unprotected, and later put on a write protect tab, you will have to kill and /or move some files off it (see EW manual) or the system will try to write to the protected disk and get mad at you when it can't get at the files it's looking for.

USING NEWSCRIPT FOR YOUR OTHER BASIC PROGRAMS: NewScript alters the Operating system when it loads itself up. Some of the features it loads up are superior to the normal functions, and you might wish to use them with your other programs. The operating system NewScript leaves behind when you exit it is not *exactly* the same as the normal version so there is some potential for problems. Nevertheless, if you want a very fast keyboard processor, etc. you can enter "NS -" from DOS; NewScript will just load up its system features, leaving you in DOS.

And NewScript has a very fast string compression (garbage collector) routine running with TBASIC or BASIC. If you have a BASIC program that has lots of delays for string compression, you might try this. Load up NewScript; Exit to BASIC. Run your other programs from there

This special "garbage collector" is also available as a separate little program that you can use with most *any* BASIC or machine language program -- and cut delays and stalling dramatically if string compression is the culprit. Sold alone it's called "Trashman" and lists for $39.95. The advantage of the standalone version (rather than using the one built into NewScript) is that you can use it with most any operating system, takes less memory (2k), and you don't need to take chances that the other adjustments NewScript makes when loading will be incompatible with your other applications program.

This is an example of using "imbedded" files to create and organize a big document.

```
.cm <THIS FILE IS CALLED NSNOTES1. IT is put together with
.CM  ".im" and ".cm" commands.  ".CM" stands for "comment"
.CM -- and it simply that anything else on the same line is
.CM a "comment" in the file -- but is normally never printed
.CM when you print the file.
.CM ".im" stands for "imbed". It commands NewScript -- at
.CM the time it goes to print this file -- to look for the
.CM the file named after the ".im" and print it.>
.CM <This system makes it possible to store and edit and
.CM versions of very large documents within a small file
.CM like this one. Note that you can put regular written
.CM -- a few line, or many-- between the "imbedded" files.>
.CM
.im LogoND2
.cm <this is a New Directions logo, above>
.im LINE70
.cm <this is a 70 character graphics line, above>
.sk
.im NSintro
.cm <introduces idea of not trying to learn all NewScript
     at once>
.pp
.im NSconfig
.cm <NSconfig discusses the system -- Mod III, 2 disk -- >
.pp
.im NSlogo/TXT
.cm <NSlogo/TXT covers making and imbedding "logofiles">
.pp
.im NSinit/TXT
.cm <NSinit/TXT covers initialization of NewScript>
.pp
.im NShelp
.cm <NShelp talks about getting help from Prosoft>
.pp
.im NSfileNa
.cm <NSfileNa gives file naming suggestions/conventions>
.pp
.im NSlimBug
.cm <----- A possible Bug in LIMA operations>
.pp
.im NSopSys
.cm <NSopSys talks about using TDOSPLUS 3.3, as option>
.pp
.im NSElcWeb
.cm <---- about Electric Webster and file management>
.pp
.im NSutilty
.cm <--- using special features of NS outside of NewScript>
.ap More
.cm
```

this material is for your information -- it does not print or effect anything. (The "<" & ">" around comments are NOT required -- just my own convention)

(causes "skip" one line.)

(tells Newscript to look for and start printing whatever is in "NSintro".)

(these lines are comments for your information only.)

(This tells NewScript to treat whatever it finds next as a new paragraph. That means: skip a space, indent 5, and check to be sure it's not too close to page end.)

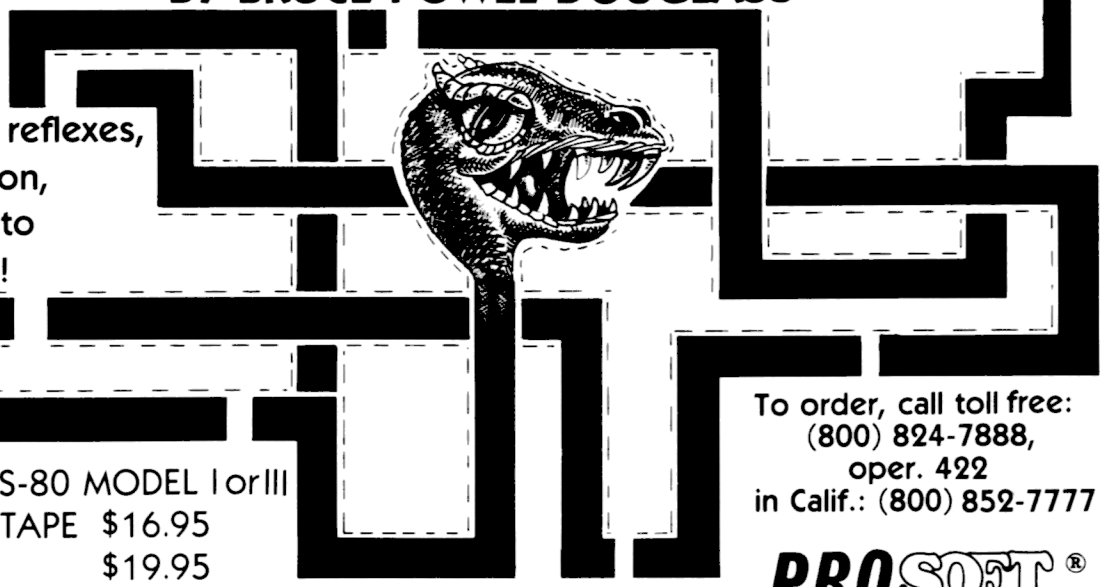
NOTE: ① You can use regular typed, visible, text mixed in with "imbeded" files such as these. For Example, if you wanted a new form for the introductory paragraph, you could write it write in this file, and view it as part of this file, or store it as say, "INTRONEW" -- then use .IM INTRONEW.

NOTE 2
YOU CANNOT PUT ".ims" OR ".APs" in a file that itself will be imbeded. But you can append files having ".IMs" to each other.

(tells NewScript to find the document called "More" and start printing it -- "APPend" it to the end of this one. ".AP" must be at the end of)

REGILIAN WORM

BY BRUCE POWEL DOUGLASS



Push your reflexes, coordination, judgment to their limits!

16-48K TRS-80 MODEL I or III
CASSETTE TAPE $16.95
DISK $19.95

Please add $1.50 for shipping and handling
6½% sales tax in California

M/C, VISA, CHECKS OK

To order, call toll free:
(800) 824-7888,
oper. 422
in Calif.: (800) 852-7777

PROSOFT®
BOX 560
NORTH HOLLYWOOD, CA. 91603

DOES STRING COMPRESSION HAVE YOU TIED UP IN KNOTS?

LET TRASHMAN CLEAN UP THE MESS!

TRASHMAN is a machine language utility for the TRS-80 Models I and III. It was written by Glenn Tesler, the author of FASTER, and can reduce BASIC's string compression time by 95% (see table below).

WHAT'S STRING COMPRESSION?

When a BASIC program changes a string (words, names, descriptions), it moves it to a new place in memory, and leaves a hole in the old place. Eventually, all available memory gets used up and BASIC has to push the strings together to free up some space. This takes time. Lots of time. The computer stops running for seconds or minutes, and you may even think it's "crashed". The keyboard won't work, and until all the strings have been collected, you just have to sit and wait. Then things run for a while, until string compression is needed again. And again.

If you're using your computer for business, that wastes your money. If you're using it personally, it wastes your time.

WHAT'S THE SOLUTION?

As soon as you start using TRASHMAN, those delays almost disappear. It uses less than 600 bytes of memory, plus 2 bytes for each active string. It works with other machine language programs and with all major operating systems. It's easy to use, comes with complete instructions, and can be copied to your own disks.

WHAT'S THE CATCH?

If a BASIC program uses only a few strings, very little time is wasted in string compression, and TRASHMAN won't be helpful. But, if hundreds of strings, including large string arrays, are used, TRASHMAN is just what you need.

# STRINGS	SECONDS DELAY NORMAL	SECONDS DELAY TRASHMAN	PERCENT IMPROVEMENT
250	11.8	0.7	94
500	45.8	1.6	96.5
1000	179.6	3.5	98
2000	713.2	7.8	98.9

(All timings done on TRS-80 Model I. Model III 15% faster, but pct. improvements identical. Listing of timing program available on request.)

Ask your software dealer for TRASHMAN, or order directly on our toll-free number. The price is just $39.95 (plus sales tax in California).

PROSOFT®