SuperSoft
P.O. Box 1628
Champaign, Illinois 61820

THE TFS C-SPECS AND SYSTEM COMMANDS AT A GLANCE

TFS CONTROL SPECIFICATIONS

```
APND <APPEND DISC FILE> [FILE NAME]
ASIS <AS IS>
BR <BREAK PAGE>
BS <BACK SPACE>
CH n <CHAPTER> (CHAPTER #> [TEXT]
COPY n <COPY> [# OF TIMES]
CR <CARRIAGE RETURN / LINEFEED>
C <CENTER> [TEXT]
ENDC <ENDCOPY>
ENDM <END MACRO>
HEAD <PAGE HEADING> [TEXT]
LMAR n <LEFT MARGINE> [NUMBER]
LLEN n <LINE LENGTH > [NUMBER]
LINE n m <# TEXT, # PAGE> [# #]
MAC <name> <DEFINE MACRO> [NAME]
N <NUMBER, AS IN A LIST>
P <PARAGRAPH>
PX <EXTENDED PARAGRAPH>
PAGE <FORMFEED>
PAR n m <DEF. PAR.> [INDENT,SKIP]
PARX n m <DEF. XPAR> [INDENT,SKIP]
PGON n <PAGE NUMBERING ON> [TAB #]
PGOF <PAGE NUMBERING OFF>
PNUM n <1st PAGE #> [#]
REM <REMARK>
RES <RESTORE SAV<ed> INVIRONMENT>
SAV <SAVE MACRO>
SETN <RESET N TO 1>
SKIP n <ADVANCE # LINES>
SP n <BLANK LINES BETWEEN TEXT> [#]
TP n <TEST PAGE> [# LINES]
UL <UNDERLINE>
```

TFS SYSTEM SPECIFICATIONS

<NOTE: '*' INDICATES A LEVEL 3 DISC BASED COMMAND>

```
APND <LINE #>
CMND <LEVEL 3 TOGGLE>
DDEL <FILENAME>
DEL <LINE # (#,#)>
DDEL <FILENAME>
DNAM <FILENAME> *
EDIT <LINE #>
EXIT
FCHK <FILECHECK>
FCPY <FILENAME> <LINE #> *
FILE <FILENAME> <ADDRESS>
FIND <SEARCH TEXT>
FORM *
INS <LINE #>
LDIR <FILENAME>
LDEL <FILE NAME>
LIST <LINE #>
LOAD <FILENAME>
LSCR <SCRATCH LOCAL DIRECTORY>
LNAM <FILENAME> *
MOVE <BEGIN#,END#,DESTINATION#> *
RCVR <FILENAME> <ADRS, ADRS>
RESE <RESET>
RNUM <LINE#> <INCREMENT>
SETC <RESET I/O>
SUBS <SUBSTITUTE> *
SAVE <FILENAME>
WORK <ADDRESS, ADDRESS>
```

NOTE: LEVEL 3 COMMANDS ARE ONLY ACCESSABLE ONCE THE LEVEL 3 COMMAND TOGGLE HAS BEEN 'TURNED ON'. A FULL DESCRIPTION OF THE COMMANDS IS FOUND IN THE USER'S MANUAL.

STATEMENT OF WARRANTY

SUPERSOFT DISCLAIMS ALL WARRANTIES WITH REGARD TO THE SOFTWARE CONTAINED ON DISC OR LISTED IN MANUAL, INCLUDING ALL WARRANTIES OF MERCHANTABILITY AND FITNESS ; AND ANY STATED EXPRESS WARRANTIES ARE IN LIEU OF ALL OBLIGATIONS OR LIABILITY ON THE PART OF SUPERSOFT FOR DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF PERORMANCE OF THE SOFTWARE LICENSED.

TRANSFERABILITY

SUPERSOFT SOFTWARE AND MANUALS ARE SOLD ON AN INDIVIDUAL BASIS AND NO RIGHTS FOR DUPLICATION ARE GRANTED.

TITLE AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN WITH SUPERSOFT.

IT IS AND HAS ALWAYS BEEN SUPERSOFT'S BELIEF AND INTENTION TO PROVIDE EXCELLENCE IN BOTH DESIGN AND SERVICE. IT IS TO THESE ENDS WHICH WE DEDICATE OURSELVES.

WELCOME TO THE TFS - TEXT FORMATTING SYSTEM

YOU HAVE PURCHASED WHAT WE AT SUPERSOFT BELIEVE TO BE ONE OF THE FINEST TEXT PROCESSORS AVAILABLE FOR THE NORTH STAR SYSTEM. IN THE PAGES THAT FOLLOW, YOU WILL LEARN HOW TO USE THE FORMATTER, THE INPUT LINE EDITOR AND HOW TO CUSTOMIZE TFS FOR YOUR OWN SYSTEM.

AS WITH ALL SUPERSOFT PRODUCTS, TFS IS 'LOAD AND GO', WITH THE EXCEPTION OF A TWO BYTE PATCH THAT MUST BE MADE TO THE USERS PRINTER DRIVERS.

BEFORE YOU BEGIN , LET'S SEE JUST HOW THIS MANUAL IS ORGANIZED. TFS HAS THREE BASIC FEATURES; THE INPUT LINE EDITOR, THIS IS HOW YOU ENTER TEXT, SECOND, THE FORMAT COMMAND MODULE, AND THIRD THE SYSTEM EXECUTIVE WHICH PERFORMS ALL FILE MAINTANCE, ETC. THERE ARE VARIOUS COMMANDS ASSOCIATED WITH EACH OF THESE SECTIONS, AND YOU WILL HAVE TO BECOME FAMILIAR WITH THEM TO FULLY USE TFS, HOWEVER, MOST OF THE COMMANDS ARE EXTREMLY EASY TO LEARN AND MOST INPORTANT EASY TO REMEMBER, SO YOU WILL BE 'UP AND RUNNING' TFS IN A VERY SHORT TIME.

THE LAST SECTION OF THIS MANUAL IS LISTING OF TFS 'SOURCE' TO CHAPTER ONE. THIS WILL ENABLE YOU TO SEE JUST HOW THINGS ARE 'DONE' IN TFS.

WE RECOMMEND THAT YOU READ THIS ENTIRE MANUAL THROUGH, AND THAT YOU FOLLOW THE INSTRUCTIONS IN 'BRINGING UP TFS'. WE ARE CONFIDENT THAT YOU WILL FIND TFS BOTH A USEFUL AND ENJOYABLE ADDITION TO YOUR COMPUTER'S LIFE.

N O T E

REMEMBER, IN ORDER TO USE THE 'FORM' COMMAND OR ANY OTHER LEVEL 3 OR DISK BASED COMMAND, THE LEVEL 3 COMMAND SWITCH MUST BE IN THE 'ON' POSITION. THIS IS ACOMPLISHED BY THE 'CMND' COMMAND IN 'TFS'. THIS COMMAND ACTS AS A TOGGLE: ON-OFF-ON-OFF, ETC. IT IS OFTEN DESIREABLE TO TURN OFF THE DISK COMMANDS WHEN NOT NEEDED IN ORDER TO AVOID UN-NECESSARY DISK ACCESSES.

BRINGING UP 'TFS'

ARE TWO METHODS YOU CAN USE TO CUSTOMIZE 'TFS' FOR Y YSTEM AND SPECIFIC DESIRES. YOU CAN USE THE 'SGEN MAND, OR YOU CAN DO IT 'MANUALLY', BOTH METHODS ARE D BED BELOW.

HOW TO USE 'SGEN'

'TFS' DISK COMES COMPLETE WITH A PROGRAM CALLED 'SGEN ' ALREADY ON THE DISK, WHICH WILL MAKE THE CUSTO ION OF 'TFS' AS EASY AS POSSIBLE. ALSO, IT CAN BE US) MAKE SPECIALIZED COPIES OF 'TFS'

STEP BY STEP WITH 'SGEN'

1). BRIN THE HOST COMPUTER WITH YOUR 'DOS'. COPY YOU DISK!

2). TYP TFS'.

3). TYP AD FORM.CMD', WHILE IN 'TFS'

4). TYP ND' (THE LEVEL 3, <DISK COMMAND> ACTIVATOR).

5). TYP EN', AND FOLLOW THE INSTRUCTIONS PRINTED ON YOUR SCR

THE 'MANUAL' VERSION

BRINGING UP TFS THE FIRST TIME

1). MAKE A COPY OF YOUR TFS DISC!

2). TFS IS ALREADY PATCHED FOR THE NORMAL NORTH STAR I/O JUMPS, SO ALL YOU HAVE TO DO IS LOAD YOUR VERSION OF DOS INTO YOU COMPUTER, INSERT THE TFS DISC, AND TYPE 'GO TFS', YOU SHOULD RECIEVE THE OPENING MESSAGE AT THAT TIME. IF YOU DON'T THERE COULD BE A PROBLEM WITH YOU DISC, OR YOU MAY HAVE A NON-STATNDARD DOS.

3). BUT, EVEN THOUGH YOU HAVE TFS 'ON YOUR SCREEN', YOU WILL STILL HAVE TO PATCH THE JUMP TO YOU PRINTER I/O DRIVERS. THIS IS NECCESARY BECUASE, WHEN YOU WISH TO EXECUTE THE FORMAT COMMAND, TFS MUST KNOW WHERE TO SEND THE DATA TO BE PRINTED.

TO MAKE THIS PATCH, YOU MUST LOAD THE FORMAT MODULE INTO MEMORY, MAKE THE CHANGE AND SAVE IT BACK TO DISC. LETS 'DO THAT NOW'.

1) YOU MUST KNOW WHERE YOUR PRINTER I/O STARTS, MAKE SURE THAT YOU GET THIS RIGHT! SECOND, YOU MUST HAVE A MONITOR,

OR ASSEMBLER IN ORDER TO AFFECT THE CHANGE. OK?

2). NOW, FIRST, FROM DOS, TYPE 'LF FORM.CMD' XXXX (THE FORMAT MODULE IS CALLED 'FORM.CMD' ON YOUR DISC) ITS NORMAL RAM LOCATION IS 3500 HEX, AND WE WILL ASSUME THAT THAT IS WHERE YOU WILL PUT IT, IF YOU CANNOT PUT IT THERE FOR ANY REASON, THEN REMEMBER TO USE THE OFF SET.

3). NOW, AT LOCATION 3E2A AND 3E2B HEX, ARE THE TWO BYTES YOU MUST FILL IN, REMEMBER LOW - HI ORDER! THIS IS A CALL TO YOUR PRINTER I/O DRIVERS, YOU PRINTER DIRVERS MUST NOT AFFECT ANY REGISTERS EXCEPT A, AND MUST END WITH A RETURN. ALSO, THE DATA IS IN THE B REGISTER, THUS IT IS EXACTLY LIKE NORTH STAR.

THOSE OF YOU THAT WANT MAY WRITE THEIR OWN I/O DIRECTLY INTO THE FORMAT MODULE STARTING AT LOCATION 3E25 HEX. IF YOU DO THIS YOU HAVE 30 BYTES FREE THERE, AND YOU CAN AFFECT NO (THAT MEANS NO !) REGISTERS, AND YOU MUST END WITH A RETURN.

4). ONCE YOU HAVE MADE THE PATCH, YOU MUST 'SF FORM.CMD' BACK TO DISC. (AND PLEASE, DON'T MODIFY YOUR 'ONLY' COPY, USE A BACKUP) AT THIS POINT IN TIME YOU WILL HAVE A FULLY OPERATIONAL TFS SYSTEM!

5). TO TRY IT OUT, BRING UP TFS, FROM TFS TYPE : 'LOAD TEST' (THIS IS PROVIDED ON YOU DISC), YOU CAN 'LIST' THE FILE, ETC. NOW, TO FORMAT AND PRINT THIS, YOU MUST DO TWO THINGS: FIRST, YOU MUST 'TURN ON' LEVEL THREE COMMANDS (THESE ARE THE COMMANDS ON DISC, INSTEAD OF RESIDENT IN RAM, THE 'FORM.CMD' COMMAND IS A LEVEL THREE COMMAND) TO TURN THESE COMMANDS ONE YOU SIMPLY TYPE : 'CMND'. THEN TO PRINT IN FORMATED FASHION TYPE : 'FORM'. YOU SHOULD

HEAR THE DISC START UP AND THEN YOU WILL SEE:

+TFS+
SET TOF

THIS MEANS TO SET THE TOP OF FORM, ONCE THAT IS DONE, SIMPLY HIT ANY KEY ON THE KEY BOARD. YOU SHOULD THEN SEE THE 'TEST' FILE BEING PRINTED OUT ON YOUR SCREEN IN FORMATED FASHION. IF YOU DON'T YOU DID SOMETHING WRONG.

AT THIS POINT IN TIME, YOU SHOULD READ THE MANUAL, AND THEN START PLAYING WITH TFS. AFTER A FEW HOURS WITH TFS, YOU WILL FEEL QUITE COMFORTABLE WITH THE SYSTEM.

ERRATA, UPDATES, AND CORRECTIONS

(A NOTE FROM THE DISTRIBUTOR) IT SEEMS LIKE EVERY TIME YOU GET A PIECE OF SOFTWARE THESE DAYS THERE ARE FAMILIAR 'ERRATA' SHEETS! WELL, LET ME TELL YOU, OUR SOFTWARE DEVELOPMENT DIVISION HAS BEEN SO BUSY THAT I CAN'T KEEP UP WITH THEM! CASE IN POINT: THE MANUAL THAT YOU ARE RECIEVING IS THE THIRD (3) RE-WRITE IN AS MANY MONTHS! NOT BECUASE OF ERRORS, BUT BECUASE OF IMPROVMENTS!

SO.... WHAT HAPPENS, I NO SOONER GET THIS MANUAL COMPLETE, BUT THE SOFTWARE GUYS BRING ME A 'NEW, GREAT, YOU GOT TO INCLUDE THIS RIGHT NOW' IMPROVEMENT. (I JUST HAD TO SIT DOWN A WHILE..) I REVIEWED THEIR NEW COMMANDS ('DRVE' & 'VIEW') AND DECIDED THAT THEY WERE RIGHT; HAD TO BE ON ALL DISCS STARTING YESTERDAY, BUT I REALLY WASN'T UP TO A FULL RE-WRITE AND PRINT JOB, SO I HAVE ENCLOSED THIS 'ERRATA' SHEET. HOPE YOU DON'T MIND TOO MUCH, AND HOPE YOU LIKE THE NEW COMMANDS.

THE 'DRVE' COMMAND

A NEW COMMAND HAS BEEN ADDED TO YOUR 'TFS' SYSTEM, THE 'DRVE' COMMAND. THIS COMMAND MAKES IT POSSIBLE FOR YOU TO USE UP TO THE THREE DRIVES THAT NORTH STAR ALLOWS! 'TFS' DEFAULTS TO DRIVE #1, AS IT SHOULD, BUT ONCE YOU HAVE TURNED ON THE DISC COMMANDS (LEVEL 3) BY USE OF THE 'CMND' COMMAND, YOU CAN 'LOG' INTO WHAT EVER DRIVE YOU WANT. YOU SIMPLY TYPE 'DRVE'. 'TFS' GOES TO DISC, FINDS THE COMMAND, AND THEN PROMPTLY TELLS YOU WHAT YOUR CURRENT DRIVE LOG IN IS, AND THEN ASKS YOU FOR A NEW DRIVE LOG IN. AT THIS POINT YOU MAY EITHER 'PASS' BY HITTING <CR> OR ENTER THE NUMBER OF THE DRIVE SUCH AS 2. YOU MUST HIT ONLY ONE KEY, AND IT MUST BE A NUMBER BETWEEN 1 AND 4. 'TFS' THEN LOGS YOU INTO THAT DRIVE, AND ALL DRIVE ACCESSES ARE TO DRIVE 2, UNLESS YOU WANT TO GO BACK TO ONE.

THIS PROCESS IS VERY MUCH SIMILIAR TO THE WAY 'CP/M' WORKS, YOU LOG IN TO A DRIVE, AND CAN LOG IN TO ANY DRIVE IN YOU SYSTEM.

---> BUT, THERE IS A LITTLE TINY THING TO REMEMBER, YOU MUST HAVE THE 'DRVE.CMD' FILE ON THE DISCS WHICH ARE IN THE DRIVES YOU MAY LOG IN TO. HERES WHY: LETS SAY THAT YOU ARE IN DRIVE 1, YOU TYPE 'DRVE' AND LOG INTO DRIVE 2, TO GET BACK TO DRIVE 1, YOU MUST TYPE 'DRVE' AGAIN, HOWEVER, IF THE DISC IN DRIVE 2 DOES NOT HAVE THAT COMMAND ON IT, IT WILL JUST GIVE YOU AN ERROR, AND NO WAY BACK. HENCE, THIS IS ONE UTILITY THAT SHOULD BE ON ALL YOU 'TFS' DISCS!

THE ABILITY TO LOG INTO THE OTHER DRIVES IS INDISPENSABLE FOR INSTANT BACK UP AND FOR HAVING LOTS OF TEXT ON LINE. BY THE WAY, THE 'APND' C-SPEC WILL ASSUME THE NEXT FILE IS ON THE CURRENTLY LOGGED IN DRIVE.

THE 'VIEW' COMMAND

THE 'VIEW' COMMAND IS AN EXACT SIMULATION OF THE 'FORM' COMMAND, EXCEPT THAT THE USER CAN SEE ON THE CRT SCREEN ('VIEW') THE FORMATTING BEFORE IT IS PRINTED. TO USE THIS COMMAND ONE TYPES: 'VIEW'. ONCE THIS HAPPENS THE 'VIEW' MODULE IS LOADED FROM DISC AND PLACE IN MEMORY. IT WILL PROMPT THE USER WITH :

```
+ TFS +
SET TOF
```

THIS IS BECUASE IT IS AN EXACT SIMULATION OF THE 'FORM' COMMAND. AT THIS THE USER HITS RETURN AND THE THE TEXT WILL APPEAR ON THE SCREEN ONE (1) LINE AT A TIME. THE USER MUST PROMPT 'VIEW' WITH A CARRIAGE RETURN FOR EACH NEW LINE, HOWEVER, THE TEXT IS NOT BORKEN UP BY THIS, IT WILL LOOK AS IT WOULD IF IT WERE PRINTED. THE USER MAY TERMINATE THE 'VIEW' COMMAND BY HITTING A CONTROL-C.

THE 'VIEW' COMMAND IS A DISC COMMAND, HENCE THE DISC COMMANDS MUST BE TURNED ON. THIS COMMAND IS EXTREMLY USEFULL FOR TRYING THOSE 'UNIQUE' PRINT OUTS BEFORE THEY ARE FORMATTED.

CONTENTS

PART ONE

THE C - SPECS

INTRODUCTION TO THE FORMATOR 1

OUTPUT CONTROL SPECS 4

PARAMETER SET C-SPECS 10

MISCELLANEOUS C-SPECS 12

PART TWO

THE OPERATING SYSTEM

THE SYSTEM 14

TFS SYSTEM COMMANDS 21

INTRA-LINE EDITOR 25

APPENDIX

SAMPLE SESSION APPENDIX 1

SOURCE TO CHAPTER 1 APPENDIX 2

CHAPTER 1

Introduction to the Text Formatting System

The Text Formatting System, hereafter referred to as TFS , is a program which produces a modified printer listing of the contents of an TFS file. This listing is formatted, that is, it is modified according to control specifications contained within the file itself.

TFS is also a complete text processing operating system. This operating system has its own set of commands and is discussed at length in section two of this manual. TFS is a program which interprets these control specifications and produces the formatted listing.

These control specifications, hereafter referred to as C-specs , are commands of the form '%NAME', where '%' designates that a command name follows and 'NAME', a character string of up to four characters, is the name of the C-spec. For example, the C-spec which turns on underlining in TFS is '%UL'; every word which follows a %UL is underlined until another %UL is encountered.

Naturally, as one may notice at this point, a string starting with a '%' designates a C-spec, and this seems to prohibit the use of the '%' character as the first character of a word. This assumption, however, is not the case. An escape sequence, '%%', has been designated to eliminate this problem. Any word starting with a '%%' is displayed as that word beginning with a single '%'; for example, '%%this' is printed by TFS as '%this'.

A word , as defined by TFS, is a string of characters delimited by either two blanks or a blank and a carriage return. Hence, a word is any string like 'string', where 'string' is delimited by blanks like ' string ', and words may not cross line boundaries in an TFS file, since TFS terminates each line of its files with a carriage return character (ASCII 0D hexadecimal). TFS is a word-oriented text formatter. It places words in an output line until the line is filled, and then it prints the line. TFS analyzes each word as it reads the word from the TFS file, checks to see if the word is a C-spec, formats the word if it is not a C-spec and executes the C-spec if it is, and then continues by reading the next available word in the file. TFS continues until it reaches the end of the file.

TFS operates in either (1) ASIS mode (see the %ASIS command) and (2) normal mode. In normal

mode, TFS will read words from the TFS file, building the current output line as it goes. All output lines are of a definite length, defined either by default or explicitly by the user, and TFS puts the words it reads into the current output line buffer until it reads a word too large to fit in the remaining space. If there are no words in the line at this time, it will force the word into the line and print the line; otherwise, it will "fill" the line. In filling a line, TFS inserts blanks between words in the output line buffer until there is a specified number of characters in the line (the length of the line). It will insert these blanks starting at the right end of the line, going to the left. As a result, the left and right margins of a page produced by TFS line up.

While creating the output line, TFS follows the convention that all words ending in a special character ('.','!',':', or '?') are followed by two blanks; all other words are followed by one blank. It is felt that this convention improves readability of the line.

TFS C-SPECS

As mentioned earlier, TFS recognizes many commands (or C-specs) during the formatting of an TFS file. These C-specs all take the form of '%NAME', where '%' indicates that a C-spec name follows, and 'NAME' is the name of the C-spec. The following is a list of all C-specs recognized by TFS; the chapters of this manual will describe these C-specs in detail.

Type of C-spec: Output Control
UL, ASIS, BR, CR, P, PX, PAGE, TP, SKIP n, HEAD text, C text, CH n text, COPY n, ENDC, BS, and N

Type of C-spec: Parameter Set
PAR n m, PARX n m, LMAR n, LLEN n, LINE n m, PGON n, PGOF, PNUM n, SP n, and SETN

Type of C-spec: Miscellaneous
SAV, RES, REM, APND, MAC, and ENDM

TFS is invoked by typing the FORM (format) command. This command causes the FORMat program to be loaded into memory and executed. %P The FORM command has two options. These options permit the user to: (1) have TFS stop after printing each page, thereby allowing the user to change the paper, and (2) to have TFS skip to a specified page and start printing the report at that page. 'FORMP' invokes the first option, while 'FORM n' and 'FORMP n', where 'n' is the page number (1-99) is the second. If the command 'FORM' is given with no argument, then printing begins at the start of the file and continues through to the end.

CHAPTER 2

Output Control C-specs

The output control C-specs control the output of TFS directly. They include:

1) Line control C-specs, such as %ASIS, %BR, %CR, and %SKIP.
2) Page control C-specs, such as %PAGE, %TP, and %CH.
3) Format control C-specs, such as %UL, %HEAD, %C, %BS, and %N.
4) Paragraph control C-specs, such as %P and %PX.
5) and the Copy control C-specs, %COPY and %ENDC.

The %ASIS C-spec instructs TFS to display the following lines exactly as they are without filling the output lines. Only the spacing control is carried over the %ASIS C-spec; for instance, if the output is double-spaced when the %ASIS is encountered, the lines within the %ASIS block are also double-spaced.

An %ASIS block consists of a line of the TFS file to be formatted which contains the C-spec '%ASIS' followed by an optional comment, the lines to be printed "asis", and a terminating line beginning with the character '%'. The terminating line must start with the '%' character in the first valid character position of the line. For example, a typical %ASIS block would be:

```
0100 %ASIS
0200 [text to be displayed without formatting]
0300 %ASIS
```

The only restriction placed on an '%ASIS' block is that the '%' character may not be placed in the first valid character position of a line without terminating the '%ASIS' block.

The '%UL' C-spec is used to start and stop the underlining process. The group of characters to be underlined are enclosed in '%UL' C-specs. For example, a

typical use of %UL is:

```
0100 This text will not be underlined. %UL This text will be
0200 underlined. %UL This text will not.
```

LINE OUTPUT

Lines may be terminated in one of three ways: (1) automatically when TFS determines that the next word will not fit in the current output line, (2) in an implied way when certain TFS C-specs, such as %P, force the output of the current line by the nature of their function, and (3) explicitly in the use of the %BR and %CR C-specs.

%BR (break) and %CR (carriage return) force the output of whatever is in the output line buffer. If more than 10 characters are required to fill the line to make the margins line up, the line is not filled -- it is output exactly as it exists in the buffer. Otherwise, the line is filled as described earlier.

%BR breaks the output line. If the output line buffer is empty, nothing happens on the printer; if it contains something, it is printed, and a carriage return - line feed is output. %CR always performs a carriage return - line feed. If the output buffer is not empty, it acts like a %BR; if the buffer is empty, it outputs just a carriage return - line feed. Hence, if the user wishes to skip down one or two lines, he may insert two %CR's at the appropriate place.

Along the same lines as %BR and %CR, the %SKIP C-spec also can be used to terminate a line. %SKIP is always to be followed by a number 1-99; it acts like the specified number of %CR's. Unlike %CR, however, if the end of the page is encountered before the skipping is completed, a page eject is done and the skipping is stopped. For example, if only three lines are left on the page and a %SKIP 10 is encountered, then only a page eject will be done.

If the user wishes to skip down for the purpose of inserting a diagram in the text, %SKIP alone may not be adequate. If the diagram is to require ten lines and only five lines are left on the page, %SKIP will leave only five lines for the diagram.

To set around this problem, the %TP (test page) C-spec is implemented. This C-spec, which is also always followed by a number from 1-99, tests to see if the specified number of physical lines is left on the current page, and, if such is not the case, it forces a page eject. Here's an example [illegible] ten lines for the diagram. A %TP is followed by a %SKIP to mark the end. If the ten lines are not available on the current page, a

page eject is done followed by the skip; otherwise, just the skip is done.

Another C-spec available to the user is the %PAGE C-spec, which forces a page eject. The advantages of this C-spec are obvious.

PARAGRAPHS

TFS supports basically two types of paragraphs -- normal, indented paragraphs and exdented paragraphs. Indented paragraphs are as one would expect a paragraph to be. The paragraph starts with the first word indented a specified number of characters in from the left margin. TFS permits indentation of from 0 to 99 characters.

Exdented paragraphs are displayed as the first line extending a specified number of characters to the left of the left margin. The lists presented in this manual are formed using exdented paragraphs. Obviously, the left margin must be greater in length than the number of characters to be extended; if such is not the case, an error message is printed in the output. It makes no sense to extend a line beyond the first character space the printer can print in.

The %PAR and %PARX C-specs define the characteristics of the paragraphs to follow. These C-specs, which are described in detail later, set the number of characters to be indented and exdented.

The %P and %PX C-specs tell TFS that an indented or exdented paragraph starts with the next word. The current output line is broken (%BR) and the new paragraph is started when these C-specs are encountered.

HEADINGS and CHAPTERS

The %HEAD C-spec permits the user to place a headings at the top of each page if the page numbering option is on (see %PGON). This C-spec takes the form of '%HEAD text' on one line of the file. No C-specs may be placed in the text following the %HEAD C-spec, and all of the text following this C-spec to the end of the line in the file is used as the headings.

When a page eject occurs and %PGON and %HEAD are in effect, the first printed line contains the page number. This is followed by the number of blank lines specified by the last previous %SKIP C-spec and then the specified heading followed by one additional blank spaced line. For

instance, if the output is double spaced, one blank line follows the line containing the page number and three blank lines (1 blank, 1 blank for the next normal line in spacing, and 1 blank to follow that line) follow the heading. Note the headings on the pages of this manual as examples.

The chapter C-spec, %CH, is of the form '%CH n text', where 'n' is the chapter number 1-99 and 'text' is the chapter title. %CH forces a page eject, skips down 10 physical lines, centers the word 'CHAPTER' and the chapter number, skips down two blank spaced lines (3 physical lines if double spaced, 1 physical line if single spaced, 5 if triple, etc.), and centers the text of the chapter title. As with all centering, the next word after the %CH C-spec must be a C-spec which breaks the output line with a carriage return (see the section on centering). For example, such a C-spec may be %CR, %P, or %PX.

CENTERING

Centering is done explicitly in TFS by using the %C C-spec and implicitly by using the %CH C-spec. Centering always involves breaking the current line and starting a new line. The %C C-spec is of the form '%C text', where text is terminated by the end of the current line in the TFS file. When %C is encountered the output buffer is broken, and the centering is done on the next physical line of the printed output. Since spacing is done after a line is printed, centered lines in a single-spaced section of text will be on the next physical line and centered lines on double-spaced sections of text will be on the second physical line following the broken line.

NUMBERING and BACKSPACING

Two C-specs included at this time are %BS and %N. These C-specs give the user some additional control over the format of the output that is particularly advantageous.

%BS is the backspace C-spec. It has no arguments, and its function is to perform a backspace in the output line buffer. As each word is stored in the output line buffer, it is followed by one or two blanks. If it does not end in a terminating character like '.' (specified earlier), it is followed by one blank; if it ends in such a character it is followed by two blanks. %BS backs up the pointer which points to the next available character position in the buffer; hence, it erases a blank following the last word placed in the buffer. %BS effectively concatenates the next word to be placed in the buffer with the last word placed in the buffer. This is particularly useful in cases where a C-spec which outputs its arguments globally must be restrained.

As a case in point, %UL is such a C-spec. If the user wishes to underline a word, for instance, and terminate the word with a piece of punctuation which is not underlined, %BS makes this possible. For example,

```
0100 %UL TFS %UL %BS .
```

makes the string 'IES.' possible.

Two problems with using %BS should be noted at this time. The first problem is that if the string to be concatenated to the word in the buffer would result in a line overflow, TFS will not permit the concatenation to occur. For example, if 'stand' is concatenated to 'under' and there are only two spaces left in the output line buffer (i.e., 'understand' will overflow the right margin by three spaces), concatenation will not occur and 'stand' will appear as the first word of the next line. Hence, as a general rule, it is best to use %BS to append a single character to the last word in the output buffer. One may safely append larger strings only if he is sure that an overflow will not occur.

The second problem is that %BS will not work if the output buffer is empty. No significant error will occur, but the concatenation to the desired word may not occur. Specifically, if the word to be appended to was the last word of the line just printed, then the backspace will be ineffective.

Numbering is the second item covered in this section. It was included primarily because backspacing often accompanies numbering in TFS, such as in the numbering of list items.

The %N C-spec is used to enable automatic numbering. TFS supplies a number buffer to the user; this buffer is initialized to 1 when TFS is first invoked and whenever the %SETN C-spec (see later) is encountered. Whenever %N is encountered the value in this buffer (1-99) is placed in the output buffer as a two-character word. If the value is between 1 and 9, the first character is a blank. After the word is placed in the output buffer, the value of the number buffer is incremented. Hence, successive occurances of %N result in successive numbers being placed in the output line, like '%SETN %N %N %N' results in 1 2 3. This is particularly useful in producing numbered lists, where the user may wish to insert an element into a list at a later time and does not wish to manually renumber the lists in the TFS file. All lists in this manual are produced by using %N followed by a %BS and a ')' as the first words in an extended paragraph (see the list at the beginning of this chapter).

The last C-specs to be described in this charter are %COPY and %ENDC. These C-specs allow the user to copy sections of the file up to 99 times. With these C-specs, all or selected sections of the TFS TFS file may be duplicated in the output.

%COPY takes the form '%COPY n', where 'n' is the number of copies (1-99) to be made. The first word of the block to be copied is the first word of the line following the line containing the %COPY C-spec. For example,

```
0100 %COPY 2 %P This
0200 will be copied twice. %ENDC This once.
```

For instance, this paragraph is copied twice by the %COPY and %ENDC C-specs.

For instance, this paragraph is copied twice by the %COPY and %ENDC C-specs.

The entire file may be copied by placing %COPY after the last macro definition in the file (see the section on macros later) and by placing %ENDC as the last word in the file.

CHAPTER 3

Parameter Set C-specs

The Parameter Set C-specs are used to assign values to the various control settings used by TFS. These C-specs include:

1) Paragraph parameter C-specs, such as %PAR and %PARX.
2) Line parameter C-specs, such as %LMAR and %LLEN.
3) Page parameter C-specs, such as %LINE, %PGON, %PGOF, and %PNUM.
4) the Spacing parameter C-spec, %SP.
5) and the N parameter C-spec, %SETN.

All Parameter Set C-specs except %PNUM (see below) are effective immediately. For example, once %PAR is used, all subsequent paragraphs created by %P are affected; additionally, the current paragraph is also affected immediately.

The paragraph parameter C-specs set the indentation or exdentation number and the number of spaced lines to be placed between paragraphs. Both %PAR and %PARX have two numeric arguments -- the first sets the indentation or exdentation and the second sets the number of spaced lines to be placed between paragraphs. As in most numeric parameters, these may take on the values from 1 to 99. For example, '%PAR 5 1' establishes an indentation of 5 spaces and the number of spaced lines to 1. If the output is double spaced, three blank lines (1 associated with the last line of the paragraph and 2 associated with the skipped line) are placed between each paragraph. Indented and exdented paragraphs are discussed in the previous chapter.

The %LMAR and %LLEN C-specs set the location of the left margin and the length of the output line. '%LMAR n' sets the left margin at 'n' characters right of the physical left end of the carriage. '%LLEN n' sets the length of the line to 'n' characters, starting at the current position of the left margin.

The effects of these C-specs are order-dependent to some extent. Since %LLEN sets the position of the right margin based upon its argument and the position of the left margin and %LMAR ignores the length of the line, if an %LMAR C-spec is executed after a %LLEN C-spec, the new line length is the difference between the old line length and the new left margin. For example, if '%LLEN 80 %LMAR 10' is encountered in the text, the new line length would be 70. However, if '%LMAR 10 %LLEN 80' is encountered, the new line length would be 80. The %LLEN C-spec adds its argument to the current position of the left margin.

The %LINE C-spec sets the format of the output page. %LINE has two arguments -- the number of physical text lines on the output page and the number of physical lines on the output page. For instance, '%LINE 40 51' specifies that there are to be 40 physical text lines (including spacing) on a page and 51 physical lines on a page. Hence, when TFS is started and this instruction is encountered, TFS starts counting down from 40 with the line on which the user set the Top-of-Form, and, when it has counted 40 lines, it skips down 11 (51-40) for the next page.

The %PGON and %PGOF turn on and off the page numbering and heading facilities of TFS. %PGON has one argument, the column number from which the page numbers will be right-justified. For example, '%PGON 60' right-justifies the page numbers in column 60, so page 1 will print the 1 in column 60 and page 10 will print the 0 in column 60 and the 1 in 59. The heading facility described earlier is turned on by this command, so the heading buffer should be loaded by the %HEAD command (described earlier) before %PGON is encountered. %PGOF turns off the page numbering and heading facility. Note that paging itself is always engaged with TFS.

%PNUM sets the number of the next page to be printed to the value of its argument. Unlike the other C-specs, %PNUM is not effective immediately -- its value applies to the next page rather than the current page. Hence, '%PNUM 5 %PAGE' would result in the page supplied by the page eject (%PAGE) command having a number of 5. %PGON must be in effect for this C-spec to work.

%SP sets the spacing of the line. This C-spec is of the form '%SP n', where 'n' may take on values from 1-99. '%SP 2' sets double spacing, '%SP 1' sets single spacing, etc. '%SP 0' is not recommended since results are sometimes unpredictable.

Finally, %SETN sets the value of the number buffer to 1. This C-spec is used to initialize the number buffer for subsequent uses of the %N C-spec (described earlier).

CHAPTER 4

Miscellaneous C-specs, including Macros

The following are the miscellaneous C-specs supported by TFS:

1) The environmental control C-specs, %SAV and %RES,
2) The comment C-spec, %REM,
3) The append C-spec, APND, and
4) The Macro C-specs, %MAC and %ENDM.

The %SAV and %RES C-specs are used to save and restore the TFS environment, respectively. The TFS environment consists of the following:

1) The number of lines to skip between paragraphs,
2) The current page number,
3) The pasing flag, which indicates if page numbering is currently turned on,
4) The indent and exdent counts,
5) The left margin setting,
6) The length of the output line,
7) The underline flag, which tells if underlining is engaged,
8) The heading flag, which tells if a heading is currently set,
9) The line spacing, and
10) The current value of the number buffer.

Whenever %SAV is encountered, these values are saved in a reserve buffer. Their values remain unchanged. At this time, the user may change whichever values he wishes. When he wishes to restore the saved environment, he simply enters the %RES command.

The %REM C-spec allows the user to enter a remark, or comment, into the file. The comment starts with the first character following the word '%REM' and continues to the end of the line in the file. The next line is then interpreted normally.

Appending Files

In some cases, the user may wish to load another file and continue formatting in the same environment as the previous file (i.e., he may wish to have the same line length, the same paragraph settings, etc.). The %APND C-spec was created to permit the user to do this. This command gives the TFS user the additional flexability of producing a report which spans over several files.

The %APND C-spec is of the form '%APND FILENAME'. Upon encountering this C-spec, TFS clears the local file space and loads the specified file. It then continues formatting at the first word of the loaded file.

All the environmental attributes of the previous file are preserved, but the macros need to be redefined. Hence, the %APND C-spec easily allows the user to chain several files into one formatted listing.

MACROs

Macros are essentially subroutines placed within the TFS formatting file. Each macro is of the form of the %MAC C-spec followed by the name of the macro; this name must be four characters or less -- any additional characters will be discarded. The first word following the macro name is the first word of the macro. The contents of the macro include all words following its name up to and including the %ENDM termination word.

Macros are not executed when they are defined. They are executed only when their names are referenced. For example, if the TFS file contained:

```
0100 %MAC TFS %UL TFS %UL %BS , %ENDM
...
1000 %TFS you see
```

the phrase 'IES,' would only be printed when line 1000 was encountered.

All macros must be defined before they are first referenced. TFS is a one-pass formatter, so the table of macro names is only formed as the macros are defined.

The nesting of macros is permitted. Macro definitions, however, may not be nested, but one macro may call another macro which in turn calls another. This type of nesting is permitted up to ten levels deep. An indirect recursion is not checked for by TFS, and such a situation could be disasterous. Up to twenty macros may be defined.

The best way to discover exactly what macros can do is to try them. The macro facility of TFS is rather powerful, and it can be of enormous value in creating formatted text.

CHAPTER 5

THE 'TFS' OPERATING SYSTEM

Like many other operating systems, one learns how to use TFS by working and playing with it. This chapter is designed as an introduction to TFS and, with the aid of this chapter, this manual, and TFS itself, the user should easily be able to learn how to write and run programs on any microcomputer system which supports TFS.

How to Execute TFS

Since TFS is designed to run using Northstar's standard DOS for support, execution of TFS is very simple. Once the user has booted in DOS, he need only type 'GO TFS'. TFS should then be loaded and executed immediately. TFS will print its opening message and give the user its command prompt ('>').

NOTE: You must have 8K bytes of memory starting at location 0 in order to run TFS.

The TFS Input Line Editor

One of the first things the user should know about TFS is how to give it a command. This is done by typing the command on the user's keyboard. Once TFS has given the '>' prompt, it is in command mode: TFS is ready for the user to type a command to it. For a listing and explanation of the commands, see Chapter 7.

In almost all cases, typing done while in TFS is processed by the TFS Input Line Editor. This editor collects each character as the user types it and allows the user to correct any typing errors he has made. As each character is typed, it is checked to see if it is a special control character. If it is, the function of the control character is executed; if it is not, the character is saved in TFS's input line buffer and printed on the user's terminal. When the user has finished typing his line [illegible] a carriage return, which is a special control character that tells the line editor to finish inputing the line and to give the line to TFS to interpret and execute.

The following is a list of all the control characters

recognized by the TFS Input Line Editor:

1. the Escape (<ESC>) key. When typed, <ESC> is printed on the user's terminal as a dollar sign ('$') followed by a carriage return (<CR>). This key tells the editor to delete the line typed in so far and start over with a new line.
2. the Line Feed (<LF>) key. This key echoes as a <CR> and does not affect the line contained in the input line buffer. The sole purpose of this function is to allow the user to continue typing his line on the next physical line of his terminal.
3. the Backspace (<BS>) or Ctrl-H key. This key allows the user to delete the last character he typed. It echoes as the cursor backing up to the previous position on the screen. For example, if the user has typed "ABCD<BS>", only "ABC" is in the input line buffer; the "D" has been deleted. The user cannot delete beyond the beginning of his line; if he does, an <ESC> is processed, echoing as a '$' and <CR>.
4. the Delete (<DEL>) or Rubout key. This key performs the same function that <BS> does, but it echoes differently. The deleted characters are enclosed in backslashes. (<DEL> is for hard-copy terminals.) For instance, if the user typed "ABC<DEL>E", this would be echoed as "ABCD/D/E", indicating that the D was deleted and the string in the input buffer is now "ABCE". If the user types more that one <DEL> in a row, all the deleted characters are enclosed in one set of backslashes. For example, if the user types "ABCDE<DEL><DEL><DEL>ABE<DEL>C", this would appear on his terminal as "ABCDE/EDC/ABE/EC", indicating that "EDC" and then "E" were deleted and the resulting string is "ABABC". This feature is provided primarily to permit the use of a device that does not support hardware backspace to be used as a principal I/O device.
5. the Carriage Return (<CR>) key. Again, the <CR> key always instructs the editor to terminate the input of the line and give the line to TFS to interpret.

The input line editor is used in every aspect of TFS except for the intra-line editing mode (see the EDIT command), and these commands are in effect whenever the user is typing something. This editor is an extremely useful tool, and with practice it will soon become very easy and natural to use.

Files in TFS

TFS supports up to ten text files in memory (the local files) and 64 files (text, binary, or other) on disk. Whenever the user lists a file, assembles a file, edits a file, or uses one of the file modification commands, he operates on the primary file.

The primary file is one of the local files in memory. It is the last file loaded or the last file referenced by the FILE command. The FILE command (discussed later) is

used to create files and make a specified local file primary. The FILE <filename> command will create a local file if a ocal file of the specified name does not already exist or it will make the specified local file primary. Once a file is primary, it may be edited and assembled by the user.

Example: If the specified file does not exist, FILE will create it. The output looks like:

```
>FILE TEST
      >
```

Result: The address range specified by TFS shows that the file contains no data. (It exists from 4000 to 4000 hexadecimal.)

Example: If the specified file does exist, FILE will make it primary. The output looks like:

```
>FILE TEST2
TEST2 4570 4589
>
```

Result: Since there is a non-zero range, the user can see that the specified local file is now primary.

How to Create a Local File

The most common use of TFS is text processing and formatting. In order to do this, he must know how to create a file. This is done in a number of ways.

The first and easiest way is to use the FILE command. By typing FILE MYTEXT, the userr can let TFS create a file for him. In response to this command, TFS will automatically place the file in memory, initialize the file, and respond with something like

MYTEXT 4000 4000

This indicates that TFS has initialized the file and set it to start at location 4000 in memory. Now, to type his program into this file, the user need only use the APND command. By typing APND, the user tells TFS that he wants to add a block of lines to the end of the current primary file (the file he just created). TFS will then respond with a "?" prompt and permit the user to type the lines of his program. All text files in TFS must be numbered, but the APND command puts the user in block line entry mode, which automatically numbers the line for the user (line numbers do not appear while in the mode). At this point, the user simply types the program text, and, when he has finished, types a Control-C followed by a <CR>. TFS will the renumber the file and place the block of lines just entered [illegible].

[illegible] the following is an example of a short program entered by the user. From now on, underlined phrases or symbols in the examples presented indicate that

these were typed by TFS, and the rest is typed by the user.

```
>FILE TEST
TEST 4000 4000
? %P THIS IS A TEST PARAGRAPH. AS YOU CAN SEE, ALL ONE HAS TO DO IS
? TYPE TEXT WITH NO REGARD TO FORMATTING, AND SIMPLY LET 'TFS' DO
? ALL THE WORK!
? ^C
 >LIST
0010 %P THIS IS A TEST PARAGRAPH. AS YOU CAN SEE, ALL ONE HAS TO DO IS
0020 TYPE TEXT WITH NO REGARD RO FORMATTING, AND SIMPLY LET 'TFS' DO
0030 ALL THE WORK!
>
```

Result: The FILE command created the file TEST at location 4000 and the user then entered lines into this file using the APND command. Note that he terminated the entering of these lines by typing a Control-C (^C) followed by a <CR> (carriage returns are not shown in the above example). He then instructed TFS to list the file, and it did, showing the line numbers it assigned to the lines of the file. TFS automatically inserts a space between the line number and the first character typed in each line.

The FILE command is one method of creating a file, and the LOAD command is another. The LOAD command simply loads a file from disk and makes it primary. See the description of the LOAD command for more details.

How to Save and Load Programs on Disk

Saving and loading the test of programs with TFS is exceptionally easy because of TFS's dynamic file capabilities, and it is a vey useful feature, expecially in cases when the user is going to execute an untested program which may crash the system. He can save the text for the program by simply typing "SAVE PROG", assemble and execute the program, and if it crashes the system, reboot TFS and type "LOAD PROG" to get the text of the program back. That way, if the program destroys TFS, he can recover easily. Also, the user may wish to save his programs when he has finished with them or he has to go away and shut down his microcomputer for some reason.

Saving, and later loading, programs is done very easily in TFS. In order to save a program, the user may simply type the SAVE <filename> command, like SAVE MYTEXT. TFS will then save the primary file, regardless of what its local name is, on disk under the name specified ("MYTEXT"). Later, when the user returns to the system and wishes to reload his program, he may simply type the LOAD <filename> command, like LOAD MYTEXT. The specified file is made primary, and he may go on using it as he normally would.

Example: Saving and loading programs follows.

```
>SAVE IT
$ FILE SAVED
>LOAD IT
IT          683F 683F
>
```

Result: In the above example, the primary file, MYTEXT, was saved on disk under the name "IT" and reloaded. Two files now exist in memory -- IT and MYTEXT. Both files are exactly the same, but IT is the primary file. The addresses given above indicate the creation of another file upon the load. If a local file with the name "IT" already existed, TFS would prompt the user with "REPLACE?", to which he would respond with "Y" to load over the local file and "N" to abort the load.

CHAPTER 6

CUSTOMIZING TFS FOR THE INDIVIDUAL USER

There are several user-defined parameters built into TFS which the user may set to customize TFS for his particular microcomputer system. Briefly, these parameters are:

1. turning pasing of the display on and off
2. setting the number of lines to display per page
3. setting the number of nulls to be output after each <CR>
4. setting the end of the user's text workspace
5. setting the address to be branched to by the EXIT command
6. customizing the disk communication utility (especially for interrupt-driven systems)

Turn pasing on and off. At address 000c hexadecimal is the switch for pasing. It is active zero. For CRT use, pasing, is usually desired; however, if a long printout is desired or if the text editor is being used as a text processor, no pasing may be desired. Simply deposit a 01 hexadecimal at this address to turn off pasing and a 00 hexadecimal to turn on pasing.

Number of lines per page. At address 000B hexadecimal is the switch for setting the number of lines to be displayed per page. Page length can be from 1 to 255. Your disk comes with a default of 15 (0F hexadecimal), and a convenient change for a 24 line display is 17 hexadecimal. It is recommended that the user set this display for 1 less than the number of lines displayable by his CRT.

Number of nulls. At address 000F hexadecimal is the switch for setting the number of nulls output by TFS's output driver to the user's terminal. The value may be from 0 to 255; your disk comes with this value set at 0. Generally, a CRT should have a value of 0 and a teletype, like an ASR-33, should have 2 or 3 nulls.

End of workspace. At address 000D to 000E hexadecimal is the switch for setting the end of the text file workspace. Your disk comes with this set at 5FFF hexadecimal (assuming you have 24K bytes of memory from 0 to 5FFF) The workspace is where all text files reside, and the system sets [illegible] Byte 000D is the low-order of this address, usually FF hexadecimal, and byte 000E is the high-order (67 in the

above example).

The EXIT branch address. At address 1DAA to 1DAB hexadecimal is the switch for setting the address to branch to when the EXIT command is given. 1DAA is the low-order part of the address and 1DAB is the high-order part. This is set to 2028 hexadecimal (the entry point of DOS) when you receive TFS on disk. 28 is stored at 1DAA and 20 at 1DAB.

Customizing the disk communication return point. At address 0A2B to 0A33 hexadecimal is the switch for entering a customized reset into TFS. This is primarily for systems using interrupt-driven I/O. For example, such a user may wish to put an EI instruction followed by a call to reset I/O at this point.

CHAPTER 7

THE TFS COMMANDS

This chapter of the users' manual described all of the TFS commands in detail and how to use them. The followins is a list of these commands (parentheses mean the enclosed item is optional):

1. CMND turns on of off level 3 (disc resident) commands - acts as a toggle switch.
2. FILE <filename>
3. RESE
4. LIST (<line or starting line number>) (<ending line number>)
5. DEL (<line or starting line number>) (<ending line number>)
6. RNUM (<starting line number>) (<increment>)
7. APND (<line to append after>)
8. INS (<line to insert in front of>)
9. FIND (<line to start search at>)
10. EDIT <line to edit>
11. DDIR
12. LDIR
13. DNAM <filename> (LEVEL 3)
14. LNAM <filename> (LEVEL 3)
15. DDEL <filename>
16. LDEL <filename>
17. LSCR
18. FCHK (<filename>)
19. RCVR <filename> <starting address of file>
20. SAVE <filename>
21. LOAD <filename>
22. SETC (<address>)
23. WORK <starting address> <ending address>
24. EXIT
25. FORM n FORMP n (LEVEL 3)
26. SUBS (SUBS, SUBSQ, SUBSV) (LEVEL 3)
27. MOVE <begin> <end> <line number> (LEVEL 3)
28. FCPY <filename> <line number> (LEVEL 3)

The TFS Commands in Detail

The most basic of the TFS commands is <line> <text>.

This command, consisting of a line number, a space, and some text, enters that line into the primary file at the

correct place. Following are the rest of the commands in detail.

FILE FILE <filename>

FILE <filename> <address>

The file command allows the user to create a primary file or make a secondary local file primary. If the file specified does not already exist, it is created; otherwise, the specified file is made primary.

If an address is specified, the new primary file is located at the given address.

LIST LIST

LIST (<line or starting line number> (<ending line number>)

LISTF (<line or starting line number> (<ending line number>)

LISTN (<line or starting line number> (<ending line number>)

The LIST command allows the user to list all or parts of the primary file through the redirectable I/O driver (see SETC command). If no line numbers are specified, the entire file will be listed; if one line number is specified, just that line is listed; and, if two line numbers are specified, that range of lines is listed. LIST lists the file exactly as the user typed it (with line numbers added, of course). LISTF formats the listing (assuming it is an assembly language program). To format properly, all op code must start in column 2 if there is no label and each section of the line (label, op code, operand, comment) must be separated by only one space. LISTN lists like LIST does, but line numbers and the extra space between the line number and the text are not included in the listing.

Example: LIST

Result: The entire primary file is listed.

Example: LIST 100 200

Result: Line 100 to 200, inclusive, of the primary file are listed.

Example: LIST 100

Result: Only line 100 is listed.

Example: LISTF 300 456

Result: Lines 300 to 456, inclusive, of the primary file are file are listed in formatted form.

Example: LISTN 200

Result: Line 200 is listed without its line number nor the space after the line number.

DEL DEL <line or start line> (<end line>)

The DEL command deleted the lines specified from the primary file. The first line deleted is the first line number; if there is no line with this number, the line following this line number is deleted. At least one line number must be specified.

Example: DEL 100

Result: Line 100 is deleted from the primary file. If no line was labelled 100, and, for instance, say the lines around 100 were 90, 95, 101, 105, line 101 would have been deleted.

Example: DEL 100 200

Result: Lines 100 to 200, inclusive, are deleted. If lines 101, 120, 145, 195, 199, 210, 205 were the only lines in the file around this range, lines 101 to 199 would be deleted.

RNUM RNUM

RNUM (<new first line number> (<increment>)

The RNUM command renumbers the primary file. If no arguments are specified with RNUM, the file is numbered starting at 0010 and incrementing by 10. The first argument gives the number to start at and the second gives the increment.

Example: RNUM

Result: The primary file is renumbered, starting at 10 and incrementing by 10.

Example: RNUM 100

Result: The primary file is renumbered, starting at 100 and incrementing by 10.

Example: RNUM 100 5

Result: The primary file is renumbered, starting at 100 and incrementing by 5 (100, 105, 110...).

Warning: If wraparound occurs during renumbering, i.e., if a line number exceed 9999, the message "LINE NUMBER OVERFLOW" will be printed and the user must then renumber the file with a smaller increment and/or starting line number. He may destroy his file if he tries

APND APND

APND (<line number>)

The APND command allows the user to append a block of lines to the end of his file (just APND) or insert a block of lines after a specified line (APND <line number>). While in this block line entry mode, the user need only type the text of the lines; TFS will place a line number and extra space on the front of each line. The user is prompted with a "?" at the beginning of each line, and he then types the line. The input line editor is in effect, and he may use it to correct typing mistakes. When finished, he simply types a Control-C immediately followed by a <cr>. If the Control-C is the first character of a new line, the previous line becomes the last line of the block to be entered; if the Control-C is the last character of a text line, the Control-C is ignored and that line without the Control-C is entered as the last line of the block. See the sample TFS session to view an example of enterins lines throush block line entry mode with APND.

When block line entry mode is exited, the entire primary file is renumbered with the default starting line number of 0010 and an increment of 10. If the "LINE NUMBER OVERFLOW" message is printed, the user must immediately use the RNUM command to renumber the file until this message does not occur. "RNUM 5 5" is recommended as the command to use (renumber starting at line 5 and incrementing by 5).

Example: APND 100

Result: The following block of lines is inserted after line 100 and before the next line of the file.

Example: APND

Result: The following block of lines is appended to the end of the file.

INS INS <line number>

The INS command is exactly the same as APND, but the block of lines is inserted in front of the specified line. This command is used to allow the user to insert a block of lines in front of the first line of the file. Block line entry mode and renumbering is the same in INS

as it is in APND.

Example: INS 200

Result: The following block of lines typed by the user is inserted in front of line 200.

FIND FIND

FIND <starting line number of search>

The FIND command searches over the primary file for a string of characters specified by the user and prints every line which this string occurs in. FIND by itself will search over the entire primary file and FIND <line number> will search starting at the specified line and continue to the end of the file.

In response to the FIND command, TFS responds with "SEARCH STRING?", to which the user may simply type a <CR> to abort the command or a string or characters followed by a <CR> to execute the search. The <CR> is not a part of the string. See the sample TFS session for an example of the use of the FIND command.

Example: FIND 500

Result: Search for the string specified by the user starting at line 500 and search to the end of the file.

EDIT EDIT <line number>

The EDIT command invokes the TFS intra-line editor. This editor allows the user to edit a line that has already been typed without retyping the entire line. If a line number is not specified, the first line of the file will be edited; if a line number is specified, that line, if it exists, or the line that would follow it if it does exist, will be edited.

The intra-line editor is a dynamic editor which permits the user to see the effects of his editing commands immediately after he types them. When a line is edited, it is copied into the editor's old line buffer and then displayed to the user. The editor then does a carriage return and prompts the user with a question mark. As the user edits this line, each character of the new line that is created is place into the editor's new line buffer. The [illegible] line in the old line buffer [illegible] affected. Finally, when editing is finished, the user may type a carriage return to terminate the editing process

and replace the original line in the file with the line as it exists in the new line buffer.

The intra-line editor responds to a host of subcommands. The following is a complete list of these commands and their functions.

1. <sp> -- copy the character pointed to by the old line pointer into the character position pointed to by the new line pointer and advance the old line and the new line pointers by one. The space bar, therefore, will simply copy the next character from the old line buffer into the new line buffer. After the copy is done, the copied character will be displayed to the user.

2. E -- skip to the end of the line. The rest of the characters in the old line buffer are copied into the new line buffer and both pointers are advanced to point to the non-existent character after the last character copied. The copied characters are displayed to the user as they are copied.

3. D -- delete the character pointed to by the old line pointer (delete the next character in the old line). The character is deleted by advancing the old line pointer by one character position and not affecting the new line pointer. The deletion is displayed to the user as a backslash ("\") followed by the deleted character. If the next command typed by the user is another D, the next deleted character is displayed (without the backslash). This will continue until the user types some other command, in which case a closing backslash will be displayed. In effect, the deleted characters are enclosed in backslashes when displayed to the user.

4. I -- insert a string of characters in front of the character currently pointed to by the old line pointer. In response to the I typed by the user, the editor types a slash ("/"). The user may then type any string of characters he wishes except for an escape or a carriage return. These characters will be copied into the new line buffer, the new line pointer will be advanced, and each character will be echoed to the user as he types it. The escape and carriage return characters are special characters to the insert subcommand to end the insertion. The editor then types another slash to indicate that the insertion is finished and allows the user to continue editing normally. <CR> instructs the editor to terminate creation of the new line, copy the new line into the primary file, and return to TFS command mode. The <CR> is echoed as a slash, a carriage return, and a system prompt (">"), indicating that the user is now in TFS command mode.

5. R -- replace the characters pointed to the old line pointer

with the following string. Both pointers are advanced and the new characters are echoed to the user. No special character is typed to the user after he types an R, and the <ESC> and <CR> characters respond as the user types his string, each character he types replaces the corresponding character in the old line buffer.

6. S<letter> -- skip to the specified letter. This is the only two-character command in the editor; it consists of the letter S followed by a single character. When this command is typed both the old and new line pointers are advanced and the corresponding charracters are typed and copied into the new line buffer until the specified character is encountered or the end of the line is reached. Once the specified character is found, the old line pointer will point to it and this character will not be printed; it will be the next character in the line. The S and the specified letter are not echped to the user when the command is typed. This command is very useful, particularly when the user wishes to insert, delete, or replace at a specified character; he does not have to space over to that character with this command.

7. <DEL> -- the delete key back up the new line pointer. The characters backed over are enclosed in "<" and ">" (like they are enclosed in slashes in the I command) and deleted from the new line. Only the new line pointer is affected by this command.

8. <CR> -- terminate creation of the new line. This command terminates editing of the line and replaces the original line in the primary file with the line that currently exists in the new line buffer. If <CR> is the first editing character typed, the edit is aborted and no replacement occurs.

9. A -- abort the editing of the old line. This command may be typed whenever the editor is ready to receive a command (i.e., the editor is not in the middle of an insertion or replacement), and it terminates the edit and returns control to TFS without affecting the original line.

10. P -- print the new line and edit it. This command will terminate the new line at the current position of the new line pointer, copy the new line buffer into the old line buffer, print the new line, and restart the editing sequence with this new line instead of the original line. The original line as it exists in the primary file is not affected.

11. X -- exit and re-edit the old line. The X command terminates the editing done so far and restarts the edit [illegible] typed, the last line placed into the old line buffer is re-edited.

The editor has three error messages that it may display. These messages are:

1. ?? -- invalid command. A double question mark indicates that an invalid command has been typed.

2. ** -- end of edit line. A double asterisk indicates that the user has tried to go beyond the end of the original line illegally while editing. This error most commonly occurs while using the <SP> command to copy characters beyond the end of the line.

3. *EOL* -- end of line buffer. The length of the new line has just reached the limits of the new line buffer, and the user must re-edit the original line.

Example: Line 200 is printed and the user is prompted with a "?". The user may now edit line 200 using the intra-line editing commands. One useful aspect of this command not yet discussed is that line 200 may be copied as line 201 or any other desired line number by editing only the line number (such as changing the second zero in 200 to a 1, and typing the E (skip to end of line) command followed by a <CR>. Line 200 in the primary file will be unchanged and line 201 will be created; line 201 will be a copy of line 200.

DDIR DDIR

The DDIR command gives a directory of the files stored on disk. The directory listing is paged, which makes it easier to read than the normal DOS listing because no entries go over the top of the page if there are more files on disk than lines on the user's CRT.

This listing contains from 4 to 5 elements per line, depending on the type of file being listed. The name of the file is given first, followed by the starting disk address of the file in hexadecimal, a hexadecimal value for the length of the file in 256-byte blocks, the type of the file (see the DOS manual), and, if the file is binary (type 1), the execution address of the binary file. TFS will only work with type 0 (text) and type 1 (binary) files.

LDIR LDIR

The LDIR command gives a directory of the local disk files currently residing in memory. The primary file is

named first, and the secondary text files follow. The name of the file and the inclusive memory address limits of the file are given for each local text file. See the sample ARIAN session for an example.

DNAM DNAM <filename>

The DNAMe command allows the user to rename any disk file. The file name specified in the command is the name of the disk file as it currently resides on disk, and, in response to this command, ARIAN prompts the user with "NEW NAME?", to which he responds with a <CR> to abort the renaming process or the characters of the new name to do the actual renaming. These characters must number from 1 to 8 (8characters maximum for a file name) and be followed immediately by a <CR>.

Example: DNAM TEXT
.
NEW NAME?MYFILE

Result: The disk file TEXT is renamed MYFILE.

LNAM LNAM <filename>

LNAMe is exactly like DNAM, but the specified local text is renamed.

DDEL DDEL <filename>

DDEL deletes the specified disk file. Only the disk directory is affected; no disk file management is done by this command.

Example: DDEL MYFILE

Result: The file "MYFILE" is deleted from the disk.

LDEL LDEL <filename>

LDEL is exactly like DDEL, but it deletes the local text file specified. Also, the memory manager is invoked after the deletion and the remaining local files are packed together. The memory manager always makes sure that the primary file is physically the last file in the file workspace so the primary file can grow as the user modifies it. The memory manager also monitors the growth of the primary file which it is being modified.

LSCR LSCR

The LSCR command scratches the local file directory. All file entries are deleted, and there is no primary file after the command is executed. It effectively clears the file workspace. The files themselves, however, are not touched by this command, and they may be recovered by the RCVR command if the user knows the starting address of the file he wants to recover. See the RCVR command (following).

FCHK FCHK <filename>

The FCHK command checks the validity of the specified local file. It performs an error check on the internal structure of the specified file, and it does not do anything to alter that file. FCHK is used to check to make sure that the specified file is intact after a user error may have affected it, such as a user program running wild.

RCVR RCVR <filename> <starting address>

The RCVR command, as mentioned under LSCR, recovers the specified file after it has been deleted. This command starts at the address specified, does an internal format check on each line of the file, and looks for an end-of-file mark. If the file checks out as valid, it will make a directory entry under the specified name and make the recovered file primary.

Example: RCVR LOSTFILE 3500

Result: A recovery is attempted on the data starting at 3500 hexadecimal, and, if the data forms a valid file, the file LOSTFILE is created in the local file directory and made primary. The user may now edit the file like any other local file.

SAVE SAVE <filename>

SAVEB <file name> <start address> <end address>

The SAVE command saves a file on disk. SAVE <file name> saves the primary file on disk under the specified name. If another file already exists with the specified name, the user will be prompted with "REPLACE?", to which he responds with "N" to abort or any other character to do the replacement. The disk file manager is invoked by this command, and this is all the user need do to save the primary file on disk. Note that the response to "REPLACE?" is case and character and a <CR> is not necessary.

SAVEB saves the specified section of memory on disk under the specified file name. Again, the disk file manager is invoked and the "REPLACE?" option may be given if a file already exists with the specified name. See the sample ARIAN session for examples of the SAVE command.

LOAD LOAD <file name>

The LOAD command loads the specified file from disk into memory. If the file is a text file (type 0), it will be loaded into memory at a location chosen by the memory manager and it will be made into the primary file. This primary file will have the same name as the corresponding disk file.

If the file is a binary file (type 1), it will be loaded into memory at its execution address only. The user must the know what the file's execution address is in order to execute it. This can be discovered by using the DDIR command and reading this address from the directory entry for the loaded binary file.

SETC SETC

SETCI <address>

SETCO <address>

The SETC command controls redirectable I/O in TFS SETC by itself resets I/O to the I/O routines in Northstar's DOS. SETCI tells TFS that all further input is handled by the subroutine starting at the specified address; SETCO tells ARIAN that all further output is handled by the subroutine starting at the specified address. I/O is set or reset immediately after the <CR> is typed on the appropriate SETC command.

I/O is always reset to the DOS I/O routines upon initial entry into TFS and by the RESEt command. The entry point at location 4 hexadecimal also resets I/O.

The redirectable I/O drivers written by the user (the routines addressed by the SETCI and SETCO commands) must conform to the following rules:

1. No register may be altered by these routines. It is recommended that the user PUSH all registers, including the A register, onto the stack at the beginning of these routines and POP them at the end.

2. These routines must do a simple (or conditional RET when they are finished.

3. <CR> must be handled as just described (carriage return in ASCII). The output driver interface in TFS

always checks for a <CR> and will always send out <CR> <LF> and the specified number of nulls in response to a <CR>. The TFS output driver outputs all characters it receives exactly as it receives them -- except for <CR>. <CR> is always output as <CR> <LF> followed by the required number of nulls. Hence, "A" (41 hexadecimal) is output as "A" (41 hexadecimal); <CR> (0D hexadecimal) is output as <CR> <LF> (0A hexadecimal) and the required number of nulls (0 hexadecimal).

WORK WORK

WORK <start address> <end address>

The WORK command allows the user to set and display the boundaries of the text file workspace in TFS. This workspace is where the manager places and plays with all the local text files. The WORK command by itself just displays the boundaries currently set. WORK followed by the two addresses resets these boundaries. The starting address should never by less than 4000 hexadecimal.

Example: WORK 4000 67FF

Result: The workspace is set to 4000 to 67FF hexadecimal.

EXIT EXIT

The EXIT command simply branches to Northstar's DOS (location 2028 hexadecimal). The user may reset this branch address if he desires (see the page on customizing

1 . CMND : This command turns on or off the level three toggle. Each time it is executed the level 3 switch is switched.

LEVEL 3 (DISK BASED COMMANDS)

2 . FORM n : This is the command file which invokes formatter; after creating the file in TFS, turn on command level 3 and type 'FORM' as a command. 'TFS' takes over from there and proceeds to format the current file in TFS. in other words: when you wish to have the formated print out, you simple turn on command level 3 and invoke the 'FORM' command. Control passes back to TFS when the print out is completed.

There are two forms for the FORM command: FORM n and FORMP n. FORM n will format the text in a continous fashion suited for continuous feed paper, FORMP n will prompt the user, as his terminal, at each FORM FEED so that a new piece of paper can be inserted, such as in a DIABLO type printer. The argument (n) is the begins page number, if no argument is given, printing will start at the top of the file. This is especially usefull for long files where an error is found say on page 30 , one could simply correct the error and resume the formatting at page 30 with the command 'FORM 30'.

3 . SUBS : (SUBS, SUBSV, SUBSQ this command allow the substitution of one word, character or phrase for another. The 'SUBS' substitutes without displaying the results, 'SUBSV' displays the results, and 'SUBSQ' queries the user before each substitution. 0, 1 or 2 arguments are allowed, if 0, then the substitution is done over the entire file, if 1 argument, then just that line, and if two, the substitution is done with in the range of the arguments.

4 . MOVE : This moves groups of lines from one place to another within the current file. There are two forms: 'MOVE' and 'MOVEB', both forms always have three arguments. 'MOVE 10 30 100' moves lines 10 through 30 to AFTER 100, while 'MOVEB 10 40 100' moves 10 through 40 BEFORE line 100. (Remember: always 3 arguments %cr

5 . FCPY : This will copy the contents of a specified local file into the the current file (both files must be in ram. There are two forms: 'FCPY' and 'FCPYB' where <filename> is the file with the current AFTER line. 100, while 'FCPYB <fn> 100' copies the specified BEFORE line 100.

APPENDIX 1

THE FOLLOWING IS A SAMPLE LISTING OF TFS ENCODED TEXT. IT IS THE ENTIRE TEXT OF CHAPTER ONE, AND MUCH CAN LEARNED FROM STUDING IT AND THE FORMATED VERSION.

```
LIST 10 500
0010 %HEAD                    The Users' Manual for the Text Formatting Syste
0020 %SP 2 %REM Double space (manual is single spaced)
0025 %LINE 58 66
0030 %PAR 5 0 %PARX 5 0
0040 %PGON 70
0050 %PNUM 1
0060 %MAC LIST
0070 %BR
0080 %LMAR 15 %LLEN 45 %REM Set up for inverted paragraphs
0090 %SETN %REM Set N=1 for item count
0100 %ENDM
0110 %MAC ELST
0120 %LMAR 0 %LLEN 60
0130 %ENDM
0140 %CH 1 Introduction to the Text Formatting System
0145 %LMAR 10 %LLEN 60
0150 %P The Text Formatting System, hereafter referred to as
0160 TFS , is a program which produces a modified
0170 printer listing of the contents of an TFS file.  This
0180 listing is formatted, that is, it is modified according
0190 to control specifications contained within the file
0200 itself.
0210 TFS is a program which interprets these control
0220 specifications and produces the formatted listing.
0230 %P These control specifications, hereafter referred to as
0240  C-specs  , are commands of the form '%NAME',
0250 where '%' designates that a command name follows and
0260 'NAME', a character string of up to four characters,
0270 is the name of the C-spec. For example, the C-spec
0280 which turns on underlining in TFS is '%UL'; every word
0290 which follows a %%UL is underlined until another %%UL
0300 is encountered.
0310 %P Naturally, as one may notice at this point, a string
0320 starting with a '%' designates a C-spec, and this seems
0330 to prohibit the use of the '%' character as the first
0340 character of a word. This assumption, however, is
0350 not the case.
0360 An escape sequence, '%%', has been designated to
0370 eliminate this problem. Any word starting with a '%%'
0380 is displayed as that word beginning with a single '%';
0390 for example, '%%this' is printed by TFS as '%this'.
0400 %P A %UL word %UL %BS , as defined by TFS, is a string
0410 of characters delimited by either two blanks or a
0420 blank and a carriage return.  Hence, a word is any
0430 string like 'string', where 'string' is delimited by
0440 blanks like ' string ',
0450 and words may not cross line boundaries
0460 in an TFS file, since TFS terminates each line of its
0470 files with a carriage return character (ASCII 0D
0480 hexadecimal).
0490 TFS is a word-oriented text formatter. It places words in
0500 an output line until the line is filled, and then it
>
```

```
LIST 510 1000
0510 prints the line. TFS analyzes each word as it reads the
0520 word from the TFS file, checks to see if the word is
0530 a C-spec, formats the word if it is not a C-spec and
0540 executes the C-spec if it is, and then continues by
0550 reading
0560 the next available word in the file. TFS continues until
0570 it reaches the end of the file.
0580 %P TFS operates in basically two modes: (1) ASIS mode (see
0590 the %%ASIS command) and (2) normal mode.  In normal mode,
0600 TFS will read words from the TFS file, building the
0610 current output line as it soes.  All output lines are of a
0620 definite lensth, defined either by default or explicitly
0630 by the user, and TFS puts the words it reads into the
0640 current output line buffer until it reads a word too
0650 larse to fit in the remaining space.  If there are no words
0660 in the line at this time, it will force the word into the
0670 line and print the line; otherwise, it will "fill" the
0680 line. In filling a line, TFS inserts blanks between words
0690 in the output line buffer until there is a specified number
0700 of characters in the line (the lensth of the line).
0710 It will insert these blanks starting at the right end
0720 of the line, going to the left. As a result, the left and
0730 right margins of a page produced by TFS line up.
0740 %P While creating the output line, TFS follows the
0750 convention that all words ending in a special character
0760 ('.',',','!',':', or '?') are followed by two blanks; all other
0770 words are followed by one blank. It is felt that this
0780 convention improves readability of the line.
0790 %TP 10 %SKIP 5
0800 %C TFS C-SPECS
0810 %CR %CR %P As mentioned earlier, TFS recognizes many commands
0820 (or C-specs) during the formatting of an TFS file. These
0830 C-specs all take the form of '%NAME', where '%' indicates
0840 that a C-spec name follows, and 'NAME' is the name of the
0850 C-spec. The following is a list of all C-specs recognized
0860 by TFS; the chapters of this manual will describe these
0870 C-specs in detail.
0880 %TP 15
0890 %ASIS
0900
0910 Type of C-spec:  Output Control
0920    UL, ASIS, BR, CR, P, PX, PAGE, TP, SKIP n, HEAD text,
0930         C text, CH n text, COPY n, ENDC, BS, and N
0940
0950 Type of C-spec:  Parameter Set
0960    PAR n m, PARX n m, LMAR n, LLEN n, LINE n m, PGON n,
0970         PGOF, PNUM n, SP n, and SETN
0980
0990 Type of C-spec:  Miscellaneous
1000    SAV, [illegible], [illegible], [illegible], MAC, and [illegible]
1010
1020 %ASIS
1030 %APND TFSUM2
>
```

APPENDIX 2

SAMPLE SESSION

```
LIST
0010 %PAR 5 1 %PARX 5 1
0020 %LINE 58 66
0030 %SP 1
0040 %LMAR 10 %LLEN 55
0050 %C 'TFS' Sample Session
0060 %CR
0070 %P TO CREATE A FILE ONE TYPES 'FILE <filename>', THEN TO BEGIN
0080 ENTERING TEXT ONE TYPES 'APND'. FROM THIS POINT TEXT CAN BE ENTERED
0090 NON-STOP WITH OUT THE NEED OF LINE NUMBERS.
0100 %PX ALL FORMMATTING IS DONE WIL RESPECT TO THE C-SPECS. THE ABOVE
0110 SET OF C-SPECS ARE THE GUIDES FOR THIS SAMPLE SESSION. THIS PARAGRAPH N
0120 WIL BE EXDENTED.
0130 %P TO STOP ENTERING TEXT, ONE TYPE A CONTROL C. THIS WE WILL DO NOW
0140 AND THEN LIST THE FILE
>
>LETS SAVE THIS FILE NOW
>SAVE SAMP
$ FILE SAVED
>NOW I SEE AN ERROR IN LINE 100
>LETS EDIT THAT LINE
EDIT 100
 0100 %PX ALL FORMMATTING IS DONE WIL RESPECT TO THE C-SPECS. THE ABOVE
?0100 %PX ALL FORMMATTING IS DONE WIL/L/ RESPECT TO THE C-SPECS. THE ABOVE
>NOW LETS FORMAT THIS
>CMND
$ COMMAND LEVEL 3 ON
>FORM
  + TFS +
SET TOF
```

'TFS' Sample Session

TO CREATE A FILE ONE TYPES 'FILE <filename>'. THEN TO BEGIN ENTERING TEXT ONE TYPES 'APND'. FROM THIS POINT TEXT CAN BE ENTERED NON-STOP WITH OUT THE NEED OF LINE NUMBERS.

ALL FORMMATTING IS DONE WILL RESPECT TO THE C-SPECS. THE ABOVE SET OF C-SPECS ARE THE GUIDES FOR THIS SAMPLE SESSION. THIS PARAGRAPH WIL BE EXDENTED.

TO STOP ENTERING TEXT, ONE TYPE A CONTROL C. THIS WE WILL DO NOW AND THEN LIST THE FILE

```
: NOW LETS PLAY WITH THE SUBS COMMAND

>SUBSV
SEARCH STRING? C-SPCE
REPLACEMENT STRING? <THIS IS A TEST>

>OOPS SPELLED IT WORNG, TRY ABAIN

>SUBSV
SEARCH STRING? C-SPECS
REPLACEMENT STRING? <THIS IS A TEST>
0100 %PX ALL FORMMATTING IS DONE WILL RESPECT TO THE <THIS IS A TEST>. THE
0110 SET OF <THIS IS A TEST> ARE THE GUIDES FOR THIS SAMPLE SESSION. THIS P

>LIST
0010 %PAR 5 1 %PARX 5 1
0020 %LINE 58 66
0030 %SP 1
0040 %LMAR 10 %LLEN 55
0050 %C 'TFS' Sample Session
0060 %CR
0070 %P TO CREATE A FILE ONE TYPES 'FILE <filename>', THEN TO BEGIN
0080 ENTERING TEXT ONE TYPES 'APND', FROM THIS POINT TEXT CAN BE ENTERED
0090 NON-STOP WITH OUT THE NEED OF LINE NUMBERS.
0100 %PX ALL FORMMATTING IS DONE WILL RESPECT TO THE <THIS IS A TEST>. THE
0110 SET OF <THIS IS A TEST> ARE THE GUIDES FOR THIS SAMPLE SESSION. THIS P
0120 WIL BE EXDENTED.
0130 %P TO STOP ENTERING TEXT, ONE TYPE A CONTROL C. THIS WE WILL DO NOW
0140 AND THEN LIST THE FILE
>LETS PUT IT BACK

>SUBSV
SEARCH STRING? <THIS IS A TEST>
REPLACEMENT STRING? C-SPECS
0100 %PX ALL FORMMATTING IS DONE WILL RESPECT TO THE C-SPECS. THE ABOVE
0110 SET OF C-SPECS ARE THE GUIDES FOR THIS SAMPLE SESSION. THIS PARAGRAPH

>LETS MOVE THE TEXT AROUND

>MOVE 70 90 120
>LIST
0010 %PAR 5 1 %PARX 5 1
0020 %LINE 58 66
0030 %SP 1
0040 %LMAR 10 %LLEN 55
0050 %C 'TFS' Sample Session
0060 %CR
0070 %PX ALL FORMMATTING IS DONE WILL RESPECT TO THE C-SPECS. THE ABOVE
0080 SET OF C-SPECS ARE THE GUIDES FOR THIS SAMPLE SESSION. THIS PARAGRAPH
0090 WIL BE EXDENTED.
0100 %P TO CREATE A FILE ONE TYPES 'FILE <filename>', THEN TO BEGIN
0110 ENTERING TEXT ONE TYPES 'APND', FROM THIS POINT TEXT CAN BE ENTERED NO
0120 NON-STOP WITH OUT THE NEED OF LINE NUMBERS.
0130 %P TO STOP ENTERING TEXT, ONE TYPE A CONTROL C. THIS WE WILL DO NOW
0140 AND THEN LIST THE FILE
>
```

```
  LETS FORMAT THIS NEW ARRANGEMNT

>FORM
  + TFS +
SET TOF
                          'TFS' Sample Session


      ALL FORMMATTING IS DONE WILL RESPECT TO THE  C-SPECS.    THE
           ABOVE SET OF C-SPECS ARE THE  GUIDES  FOR  THIS  SAMPLE
           SESSION.  THIS PARAGRAPH WIL BE EXDENTED.

                TO CREATE A  FILE  ONE  TYPES  'FILE  <filename>'.
           THEN TO BEGIN ENTERING TEXT ONE  TYPES  'APND'.    FROM
           THIS POINT TEXT CAN BE ENTERED NON-STOP  WITH  OUT  THE
           NEED OF LINE NUMBERS.

                TO STOP ENTERING TEXT, ONE TYPE A CONTROL C.  THIS
           WE WILL DO NOW AND THEN LIST THE FILE



  TO ADD MORE TEXT WE TYPE APND AGAIN

>APND
?%P THIS IS NOW THE NEW LAST LINE OF OUR TEXT.
^C
>LIST
0010 %PAR 5 1 %PARX 5 1
0020 %LINE 58 66
0030 %SP 1
0040 %LMAR 10 %LLEN 55
0050 %C 'TFS' Sample Session
0060 %CR
0070 %PX ALL FORMMATTING IS DONE WILL RESPECT TO THE C-SPECS. THE ABOVE
0080 SET OF C-SPECS ARE THE GUIDES FOR THIS SAMPLE SESSION. THIS PARAGRAPH
0090 WIL BE EXDENTED.
0100 %P TO CREATE A FILE ONE TYPES 'FILE <filename>'. THEN TO BEGIN
0110 ENTERING TEXT ONE TYPES 'APND'. FROM THIS POINT TEXT CAN BE ENTERED MO
0120 NON-STOP WITH OUT THE NEED OF LINE NUMBERS.
0130 %P TO STOP ENTERING TEXT, ONE TYPE A CONTROL C. THIS WE WILL DO NOW
0140 AND THEN LIST THE FILE
0150 %P THIS IS NOW THE NEW LAST LINE OF OUR TEXT.
>FORM
```

```
FORM
  + TFS +
SET TOF
```

'TFS' Sample Session

ALL FORMMATTING IS DONE WILL RESPECT TO THE C-SPECS. THE ABOVE SET OF C-SPECS ARE THE GUIDES FOR THIS SAMPLE SESSION. THIS PARAGRAPH WIL BE EXDENTED.

TO CREATE A FILE ONE TYPES 'FILE <filename>'. THEN TO BEGIN ENTERING TEXT ONE TYPES 'APND'. FROM THIS POINT TEXT CAN BE ENTERED NON-STOP WITH OUT THE NEED OF LINE NUMBERS.

TO STOP ENTERING TEXT, ONE TYPE A CONTROL C. THIS WE WILL DO NOW AND THEN LIST THE FILE

THIS IS NOW THE NEW LAST LINE OF OUR TEXT.