```
 SSSSSSSSSS   TTTTTTTTTTTT  EEEEEEEEEEEE  TTTTTTTTTTTT
SSSSSSSSSSSS  TTTTTTTTTTTT  EEEEEEEEEEEE  TTTTTTTTTTTT
SS        SS       TT       EE                 TT
SS                 TT       EE                 TT
SSS                TT       EE                 TT
 SSSSSSSSSS        TT       EEEEEEEE           TT
  SSSSSSSSSS       TT       EEEEEEEE           TT
         SSS       TT       EE                 TT
          SS       TT       EE                 TT
SS        SS       TT       EE                 TT
SSSSSSSSSSSS       TT       EEEEEEEEEEEE       TT
 SSSSSSSSSS        TT       EEEEEEEEEEEE       TT
```

- A STructured Editing Tool.

Description and Users Guide.

Version 2: 28th Dec 1980 (Up to STET Version 4.45)

S.Davies (DAVIES at VENTA)
M.F.Cowlishaw (MFC at VENTA)
Mail Point 137
UK Scientific Centre
Hursley.

```
 SSSSSSSSSS   TTTTTTTTTTTT  EEEEEEEEEEEE  TTTTTTTTTTTT
SSSSSSSSSSSS  TTTTTTTTTTTT  EEEEEEEEEEEE  TTTTTTTTTTTT
SS        SS       TT       EE                 TT
SS                 TT       EE                 TT
SSS                TT       EE                 TT
 SSSSSSSSSS        TT       EEEEEEEE           TT
  SSSSSSSSSS       TT       EEEEEEEE           TT
         SSS       TT       EE                 TT
          SS       TT       EE                 TT
SS        SS       TT       EE                 TT
SSSSSSSSSSSS       TT       EEEEEEEEEEEE       TT
 SSSSSSSSSS        TT       EEEEEEEEEEEE       TT
```

- A STructured Editing Tool.

Description and Users Guide.

Version 2: 28th Dec 1980 (Up to STET Version 4.45)

S.Davies (DAVIES at VENTA)
M.F.Cowlishaw (MFC at VENTA)
Mail Point 137
UK Scientific Centre
Hursley.

Table of Contents

1. Introduction

Conventional text editors present data to the user as though it were written on an ancient Egyptian scroll - not an approach that might be expected to promote fast and efficient work. Various means have been tried to circumvent the problems caused by the length of a large file: the most successful use 'windows' to allow the user to see two or more parts of a file simultaneously or to switch between two parts at the press of a key.

In the last few years a lot of work has been carried out on the concepts of structured programming and top-down design - a variety of methods have been developed to speed the process of producing 'debugged' programs. Much of this work has shown that the human mind works well when a program is designed 'from the top' and all blocks of code are kept to a limited size (often one page). Work in other fields also shows that it is advantageous to use a top-down approach when producing, for example, articles or documentation.

To date, structured programs have been generated very much from a language point of view: e.g. PL/1 structured programming is based on SPF (Structured Programming Facility) and the PL/1 Preprocessor; Assembler and Microcode structured programming is based on other macro processors; and no facility exists (to my knowledge) for producing other physically structured text forms, such as SCRIPT files.

STET is a first attempt to take the structure out of the domain of languages, and into the domain of editors. In addition to conventional editing facilities, STET gives the user a third dimension: a tree structure that may be traversed using Program Function Keys much as scrolling is normally implemented. The possibilities of this approach have only been partly explored, but experience to date shows the tool to be both effective and powerful.

In addition to having the unique structured facilities, STET is also an efficient tool for normal editing of sequential files: its concept of a single "Group" of lines, and the commands that manipulate it, are especially powerful.

2. Sequential editing with STET

STET is a powerful editor in its own right, even if only conventional files are to be edited. Most types of files (up to 32767 bytes wide, and 32767 lines long) can be successfully edited.

Let us suppose that we are to edit a normal sequential file, without using the structured facilities of STET. The program is called up using the EXEC procedure called STET using the normal CMS format: "STET Fn Ft Fm". Only as many of Fn/Ft/Fm have to be given as are required to uniquely identify the file - usually only the Fn has to be specified. Asterisks may be used for the Ft or Fm fields.

Assuming the file previously existed, the user will be presented with a screen showing the first 22 lines of the file to be edited.

The user can then control the editing of the file by using:

* **Line Commands** - typed in columns 2 through 6 of each line on the screen, allow one to insert, move, copy, and delete groups of lines etc. The line command area may be moved to the right hand side of the screen if desired.

* **Top-line Commands** - entered in the command area on the top line: general control commands, e.g. locate string; leave program, etc.

* **Attention Keys** - to scroll, repeat commands, etc.

These are obeyed strictly in order: first all line commands are executed (from the bottom up), then any top-line command is executed; finally (if there were no top-line commands) Attention Keys are obeyed.

Any non-displayable characters will be displayed on the screen as a '"'. STET will not corrupt these characters provided the **"** on the screen is not altered to any other character, or moved using the 3277 Delete/Insert keys. Lines with non-displayable characters may be shifted, moved, copied, etc. (using line commands or top line commands) without loss of information.

The editing screen could appear like this: (but 80 X 24 chars!)

```
----->                           10 Lines  DEMO PLI    11:06  FsM
     2 <-9-----+----2----+----3----+----4----+----5--..  ..--+-->80
        DEMO: PROC OPTIONS(MAIN);
              Dcl  in char(72) var;

              IN='Type in line to be echoed, or "END" to end';
              DO UNTIL(IN='END');
                DISPLAY(IN) REPLY(IN);
                END;
              DISPLAY('Bye for now');
              END DEMO;




       <-9-----+----2----+----3----+----4----+----5--..  ..--+-->80
```

The various indicators on the top line are:

"----->"	Points to top-line command entry area. Changes to "t" if hardware tabs are enabled. Also indicates if any group is defined (e.g. "<5").
"10 Lines"	Number of lines in file.
"DEMO PLI"	Filename/Type of sequential file, otherwise Block Name
"11:06"	Time when screen was last written
"FsM"	Three indicator flags: a) F/V Record Format that file will have on filing b) s/' ' "s" if file will be serialised c) U/M/L Upper/Mixed/Lower case translation in effect

On line 2, the number ("2") in the line command area is the number of the line currently at the top of the screen.

Also on line 2, the numbers at the left and right hand ends of the column rack show the current left and right margins in use (in this case the leftmost column is column 9 of the data, and the rightmost is column 80).

2.1 Line Commands

Line commands are entered in the line command area, which is an unprotected field normally at the left of each line. It may be moved to the right hand side if desired.

The current Zone (normally the full width of the data) defines the limits of all changes made by line commands, except for 'Split'.

The concept of a 'Group' of lines has been expanded and rationalised in STET to give the user a powerful set of operations:

< One or two of these symbols are used to define a Group: if one has been selected a '<' will appear at the beginning of the first line; once a second has been selected, the number of lines in the group will also be displayed. Group definitions are cleared using CLEAR (q.v. Page 15); following a Move line command or Merge top-line command, or by defining a new group with '<'s. If only one line is to be manipulated, a second '<' need not be entered.

Cn Causes n copies of the current group to be inserted after the line with this command. Only the data in the current Zone will be copied to the new lines (see the "Zone" command, page 12).

Mn Move: as Cn, except that the original group is deleted (it may be restored using the "GRoup" command, or 'RG' line command).

Dn Deletes n lines including this line.

/ Make this line the top line of the screen.

>n Insert n blank lines after this line.

***n** (Where * is any non-alphabetic character other than '<','>', or '/') will cause n lines consisting of that character to be inserted. The specified characters will only be inserted in the current zones. <u>Note:</u> To enter a line of "<"'s (or "/"'s) enter "<<" (or "//") as the line command.

In Insert n copies of the current mask after this line (the mask is set using the top-line command "MASK", q.v. page 12). Characters are only inserted within the zone.

Rn Repeat the line n times. Only data in the current zone is copied. Any current Group definition is unaffected.

On Overlay n lines with the current mask. A blank in the mask will not change the line(s) being overlaid, a '_' in the mask will force a blank in the line(s) being overlaid. Only characters within the current Zone will be altered.

Un Convert n lines (within zone) to Upper Case (All Capitals)

Ln Convert n lines (within zone) to Lower Case (No Capitals)

S Split the line at the column indicated by the cursor.

Sn If the cursor is not in the data area, SPLIT the line at column n. Otherwise, split the line at the cursor position, and move the new line to column n. The Zone setting has **NO** effect.

2.1.1 Group orientated line commands:

These commands are recognised by the suffix 'G' on the line command. The entered command will apply to the whole of the defined group (e.g. all lines in the group translated to upper case). With the exception of 'D' and 'O' which must be placed on a line of the group, the command may be placed on any line of the block within which the group is located.

UG Translate group to upper-case

LG Translate group to lower-case

OG Overlay group with currently defined mask.

MG Merge/overlay group onto the next n lines, or until the end of the block (n is group length).

The above commands **<u>only</u>** affect data within the defined zones. The following two commands affect all the data.

DG Delete the group (must be entered on a line within the group).

RG Restore group after this line.

2.1.2 Notes on line commands:

You are allowed to enter line commands in the input area on line 2: this allows lines to be inserted at the top of the file. (the 'D' line command and the '<' symbol may not be used on line 2).

If n is not specified on any line command, 1 is assumed.

The '<' and '/' symbols may occur on the same line as other commands. e.g. "<M4" will replace a line by four copies of itself.

Any combination of line commands will be obeyed, starting from the bottom line of the screen and working up. (Except that all '<'s and '/' are handled first.)

There is a very useful additional mechanism for entering blank lines: if the cursor is moved to the left of the screen (e.g. by using the 'return' key above the right hand SHIFT key) and ENTER is hit, then a new blank line is inserted. Since this is most useful when using the hardware tab or return keys, the line is inserted ABOVE the line with the cursor if the line command area is on the left, or BELOW if the line command area is on the right.

2.2 Top Line Commands.

A fairly conventional selection of top-line commands are available: many have unusual facilities. Multiple commands may be put in the command area (or be assigned to a PF Key) if there is space, by separating each command by the character '#'. The linend character (# by default) may be changed by the 'LE' (see page 12) command.

2.2.1 Location commands

General note on locate commands: For all of the following locate commands the search columns are normally those defined in the ZONE command (q.v. Page 12). A means exists, however, by which the columns to be searched can be specified when entering the command. This is done by specifying the search columns following a second delimiter. For example 'L/the/1 10' will only search columns 1 through 10.

L This is a most useful command which is used to locate the next occurrence of a specified string. All lines on the screen containing the string are brightened, so allowing 22 lines to be visually scanned at a time. Typical formats are "L/String/" or "L String". The final delimiter need only be specified if the string to be located ends in one or more blanks, or if you wish to specify the columns for the search. If the Locate command is repeated, the search will start from the current cursor position. The located line will be placed **on the line of the screen which holds the cursor,** or failing that, the 9th line. If the cursor is moved down as far as one of the four bottom lines, the next located line is re-centred on line 9. In multicase files, all occurrences of the specified string are found. e.g. "L the" will find the strings "THE" and "The" as well as "the". (But see the comments below about the effects of certain de-limiters.)

(L') Using a single quote (') as the delimiter forces an exact an exact match to be made in order for the search to be successful; i.e. case translation is NOT done.

(LX) If the string delimiter used is "X" then the search string may be specified in Hex - e.g: "LxA302". Again an exact match is required for the search to succeed.

The L command will only search the data in the columns selected by the Zone command, unless you specify the actual columns to be searched.

¬ "Negative locate" Search until a match is <u>not</u> found. The search is only ef-fective between zones; thus the combination of "Z 1 1#¬/ /" will find any non-blank character in column one. (Also "¬/ /1 1".)

/ "/" generates "L/" (see last paragraph). The final delimiter (if specified) must be "/".

G As "L", except that a global locate is carried out. With simple editing, a complete search of the file is made (ie the search will not halt at the bottom of the file). (See also structured editing, page 18.) If the delimiter is "¬", a search is made for a non-match. The "X" and "'" delimiters have the same effect as with "L".

LG Locate Group. The top line of the current group (if one is defined) is displayed using the same strategy as the locate command (see above), i.e. line with cursor, or line 9 etc..

RL Repeat the last issued locate ("L") command (regardless of the contents of the command stack).

2.2.2 Major editing commands

C Change. Allows change of one or more occurrences of a string to some other string. 2 or 3 fields must be specified, separated by delimiters e.g. "C/aaa/bbbb/5*" . The first field ('aaa') is the string to be searched for, and the second ('bbbb') is the string to replace it. The third field controls the extent of the change: i.e the number of lines over which the change may take place (starting at the current cursor position), with * meaning "all remaining lines", and the default being one line. The final asterisk (if present) means "all occurrences on each line", rather than just the first. A number (preceded by a separator blank) will define a fixed number of occurrences on each line. e.g. "c/ABC/cba/5 3" i.e the first 3 times on each of the next 5 lines. The change command will ONLY affect data which is in the columns selected by the Zone command. No other data will be affected.

(CX) If the delimiter specified is "X" then a Hex search and change will be carried out: both the first two fields must be valid hex digits. e.g. "Cx838Fx01C672x" NOTE: Null hex strings are valid.

(C') If the delimiter specified is "'" then an "exact" change will be carried out: (normally in an uppercase file the entire change command is translated to uppercase before being executed).

Following the change command you are told the number of lines affected, and the number of changed strings.

D Delete down to line with string. All criteria for the search are the same as the "L" locate command: all lines from the current line to the line above that containing the given string are deleted. e.g. "D RABBIT", "D'the elephant". "X" is a valid delimiter for Hex location.

B Break lines in group after the specified string of characters. e.g. if the current group consists of the one line: " A=B; C=1; D=12; " then the command "B ;" will cause the group to be converted into three lines (No blank lines are generated). The string may be delimited if required: thus "B/ /" will cause a group to be separated so there is one word per line. See also the "S" (Split) Line command, which will be found useful in some situations.

ME Merge lines in group. All the lines in the current group are overlaid onto a new blank line, and the original group is deleted (it can be restored using the "GRoup" command, see below). The lines are overlaid from top to bottom of the group, so the last to be overlaid (i.e. the 'top' line) will be the bottom line of the group. See also the "MG" (Merge group) line command on page 5.

GRoup Delete/Restore current group. Entered once, the currently defined group is deleted. If entered again, the group will be restored as it was. No restore is possible once a new group is defined, but it is possible to restore following line inserts, deletions, etc. **Note:** The 'GEtfile' command (q.v. page 10) defines a new group. "GR" will also restore the group following a "Move" line command, or following the ME (Merge) top-line command. See also RG line command on page 5.)

Both of the following shift commands will only affect the data within the columns specified by the Zone command. If any non-blank data is shifted outside the Zone, it is truncated and a warning is given.

>n Shift lines in current Group n columns to the right.
<n Shift lines in current Group n columns to the left.

2.2.3 Scrolling (window movement) commands

Un Fine Scrolling: go Up the file n lines.

Nn Fine Scrolling: go Down the file n lines.

TOP Go to Top of File

BOT Go to Bottom of File

nn <cc> Where nn is a valid number will place line nn on the line currently containing the cursor (if possible). If a second number is given it is treated as a column request, and the cursor will be placed in column cc of the file. If cc is omitted then the cursor will default to the first visible data column if it started from outside the data area (e.g. the top line command area), otherwise it is left alone. If necessary the file will be scrolled left/right to bring the column requested onto the screen.

Onn Set the window left-hand Origin. Sets the column of the data which will be lined up with the left edge of the window area. "O*" will set the origin to the maximum allowed (so the right hand edge of the data will be at the right of the window). If the command "O" alone is entered, the column in which the cursor is positioned will be moved to the left edge of the screen (if possible). If "O" alone is entered, and the cursor is still on line 1, then the possible bounds for the left margin will be displayed. The option "G" or "Global" may be added to the command to cause the origin to be set in all blocks of a structured file. For example "O10 G" will set the origin to 10 in all blocks of the file.

+n Fine Right Scrolling: increment the window Origin by 'n'.

-n Fine Left Scrolling: decrement the window Origin by 'n'.

W Create a window ('bookmark'). (See under PF1 on page 15.)

SKIP Skip around list of bookmarks (if any).

RESET Clears any windows set up by the "W" command.

2.2.4 File Handling/Exit Commands

The various file handling commands allow one to control saving of the file, add CMS files to the file being edited, etc.

FILE This is used to stop editing a file, and save all changes made. (STET is an in-core editor, so changes are not written away to disk until initiated by the user or by Autosave). Fn, Ft, Fm may be specified if desired - any omitted are taken to be that of the original file. (e.g: "FILE", "FI = PLI", "FI = = D" etc.).

FQUIT as file above, except leave STET when the file completes.

SAVE as FILE, except that editing is continued afterwards.

QUIT Leave STET, WITHOUT filing changes made. "QUit *" will cause STET to stop execution, without the normal 'second chance', UNLESS the change count is non-zero, in which case you are given the chance to re-edit the file.

QQuit 'Quick Quit' - leave STET immediately.

PQuit 'Protected Quit' - as normal 'QUIT' unless the change count is non-zero, in which case the current change count is displayed, and the quit is prevented.

**** Note: immediately after 'QUIT' or 'FILE' you are asked to enter a filename, filetype, and filemode, unless you have requested an immediate exit.

You may then:

a) Enter a null line to halt execution of STET.

b) Get a new file for editing (or re-fetch the last one from disk) by entering Fn, Ft, and Fm. In this case, you need only specify as many of Fn, Ft, and Fm as are necessary to identify the file, and '=' may be used to imply the Fn etc. of the last file edited. The "N" & width options may also be specified (see under the 'STET' CMS Command, page 14)

c) Return to edit mode (re-edit the file) by entering '*'. (The file will not then be fetched from disk first. You can recover from an 'accidental' QUIT this way)

AUTO The command "A" will tell you the current autosave point, and also controls the Autosave facility. By default, the file is automatically saved after every 30 changes. If a number is specified as an argument, (e.g. "A 50") the save point will be changed to that specified. If the argument is 0 or 1, autosaving will be switched off.

BUILD When editing sequential files - same as SAVE. For the effect in STET structured files see page 18.

PG Write the current group to file. If a file of the same name exists, the group will be appended. The "RF" command (see below) can be used to find out what format file will be written by this command. N.B. No build is done.

RF This may be used to query or set the record format of the file. "RF" will cause the record format (and length for fixed files) to be displayed. "RF V" will set the record format to "Variable", and "RF F" will set it to "Fixed". The RF command must be issued before the FILE or SAVE command(s) to which it is to apply. If the format specified is 'F', then the record length may also be given. E.g. "RF F 80" The maximum record length specifiable is 32767 bytes.

RS Set Record format of STET file (rather than the file to be built from it). "RS V" or "RS F" - the latter takes more space on disk, but the file can be fetched VERY much faster.

DB Switch Auto-Deblanking On or Off. If on, any blank lines will be automatically deleted whenever a sequential file is written to disk. If no argument is specified, Deblanking will be toggled on/off. use "DB ON" to force deblanking ON, or "DB anything" to switch it off explicitly.

GET Adds the specified CMS file to the end of the current file and defines the added lines as a Group (so they may easily be moved or copied elsewhere). Fn must be specified, Ft & Fm may be *'s, ='s, or omitted. A standard search criterion is followed to find the file to be added. Two options are possible '(from-no for-no' to allow you to add parts of a file (by record number).

PRINT Prints a copy of the file being edited. If no argument is given, the printer class will remain unchanged: if a single letter is given, the printer class will be changed to that specified. Only the first 133 bytes of data will be printed. A warning will be issued if this results in data not being printed.

PUNCH Punches a copy of the file being edited (useful if you run out of space on your disk!). **Only the first 80 characters of data are punched** and a warning will be given if this resulted in data not being punched. In a STET structured file you may specify one of two arguments. 'PU B' will do a build before punching, 'PU P' will do a partial build.

SER Serialise the file when SAVEd or FILEd. The first three characters of the serialisation field will be defaulted to the chars in the serialisation field when the file was read in (if none, 3 chars are taken from the Filename). The chars may also be specified as an argument to the SER command: e.g. "SER NEW". Once this option is set, an "s" will be displayed next to the case char at the top right of the screen. If "s" is set it may be switched off by the SER command (without an argument), or by using "SER OFF". If the argument is numeric (e.g. 'SER 50') the increment value (default of 10) is set to the number given. If you are editing a structured file, SER takes effect for BUILD only. Note that the SER command is only valid for F 80 files.

FN Set Filename that will be used as default for SAVE/FILE/AUTOSAVE. You may also specify the filetype and filemode after the new filename.

FT Set Filetype that will be used as default for SAVE/FILE/AUTOSAVE. You may also specify the filemode after the new filetype.

FM Set Filemode that will be used as default for SAVE/FILE/AUTOSAVE

2.2.5 Miscellaneous Commands

***** (Prefix to any command: e.g. "*TOP"). If a command is prefixed with asterisk, it will not be added to the save stack. This means that the command that would previously be executed by hitting PA2 (for example) will not be displaced by the new command. This is especially useful for commands that are assigned to the PFKeys.

Zone The Zone command is used to define vertical columns to enable restricted portions of the data to be manipulated. Two numbers may be specified, namely the left and right margins of the Zone. e.g. "Z 30 50". Once a zone has been set up, then certain of the edit commands will only act on data within the zone. The commands affected are: Change; Locate; < and > (shifts); and all line commands.

PFxx Set up the specified PFKey with the command given. e.g: "PF3 c/ABC/abc/**". If no argument is given, the PFKey is reset to its default value. Note that multiple commands may be assigned by separating them with '#'.

PF PF Key settings may be displayed and edited with the PF command. The current settings of all the PF Keys are shown, and you may change any of them. If a field is left blank, the PF-key will revert to it's default value. STET supports the use of 24 PF-keys; by default PF13-24 = PF1-12. To set PF13-24, enter 'PF', and press one of PF10-12. A count of the number of times each key has been hit is also shown: the counts may be reset to 0 by specifying the argument 'Reset'. Further statistics displayed include: editing time, edit session count, space for editing, file format, editor version number. The File name, type, and mode are also displayed and may be overtyped to change any or all of them.

SF Screen format. When entered without an argument, the line command area is toggled between the left and right side of the data. "SF R" or "SF L" explicitly set which side it is to be. "SF Z" will set the screen width = to the zone width, and the the origin to the zone start; "SF nn" will set the screen data width to nn.

H Hexadecimal: switch hex mode on/off. The origin will be changed as necessary to keep the cursor in the same position on the screen; and the data is then displayed in Hex. The hex may be edited as desired: any invalid letters (not 0-9 or A-F) will cause that byte to be changed to X'6F' which is displayed as '?'. H on its own will toggle the mode, "ON" or "OFF" may be used to set the mode explicitly. If adding to the end of the file the lines you enter (in HEX) will be padded with nulls (x'00').

LE Change the Line-end character to that specified. (The default line-end character is "#", so if you want to locate a string containing a "#" you must first use this command to change it.)

MAsk Used to set up the current overlay/insert mask (see line commands, page 4): following this command the user is prompted to type in the mask on the top line: any data is permitted. ('_' is used to force blanks during overlaying (q.v. "O" Line Command, page 5)) If an argument is given STET will set the mask from one of two places. If the first letter of the argument is 'S' the stack will be checked, and if a line is found there it will be used for the new mask (note: you can use this to set up default masks); if the stack is empty, or the letter is not 'S', the mask is set from the top line of the current group.

TAbs Like MASK, the user is prompted to enter the positions for tabs using the character 't' wherever a tab is desired. Alternatively the tab positions may be specified absolutely; e.g: "TAB 10 40". You may also selectively add ('+') or delete ('-') tabs.
For example:

```
   'TA + 30 40'      will add tabs in columns 30 and 40.
   'TA - 10 15'      will remove tabs from columns 10 and 15.
```

See the section on 'Tabs' (page 16) for information on using tab stops.

T Switch tabs mode between hardware and software tabs. (See the section on 'Tabs', page 16). If entered without an argument, the command toggles between the two modes. Use "T H" or "T ON" to explicitly select hardware mode, or "T anything" for software mode.

?,?? Displays the last or penultimate top-line command stacked.

=,== Repeats the last (like PA2), or penultimate command stacked.

HElp Invoke IOS3270 to display an online tutorial.

SHow Show/Don't show a count of the number of changes to each line. (Toggles on/off each time you execute the command, or may be specified with "ON" or "OFF").

CAse Used to control the translation of alphabetic characters in the file. Options M, L, or U are accepted, causing respectively no translation, translation of upper to lower, and lower to upper case. The current case is shown in the top right hand corner of the screen (U, M, or L). In Upper case files, lines which start with "*" in column 1 will not be translated. This is intended to improve readability of Assembler and EXEC files. STET will select an appropriate CASE setting on reading in a file: if a file is all uppercase (or has multicase only on lines beginning with '*') then CASE U will be assumed. Otherwise CASE M will be used.

CMs Issue CMS command. e.g. "CMS ERASE TEMP FILE A". Only modules that will run in the transient area will be executed. If no argument is given, this command will put you in CMS SUBSET mode (use the "RETURN" command to get back into STET). An untokenised PLIST is provided to the CMS command for it to use.

. (period) is used to execute EXEC files specifically. For example ".HELLO" is equivalent to the command "CMS EXEC HELLO". Note that the EXEC will in effect run in a 'SUBSET' environment.

CP Issue CP command. (e.g. "CP Q N", "CP MSG OPERATOR HELP!!"). If no CP command is specified (i.e. just "CP"), enters CP mode (like TEST REQ/PA1). Then use "B" to get back into STET. CP commands are executed via diagnose 8.

MSg Display this command as an error message. This is intended as a means whereby an EXEC can stack a message for display. If the first character is '!', it is removed, and the alarm (bleep) will be sounded when the message is displayed. e.g. "MSG !STOP"

TRace Trace execution of commands from the CMS Stack. Each command is displayed before it is executed, and may then be altered or erased. Hit ENTER to continue normal execution. (Toggles on/off each time you execute the command, or may be specified with "ON" or "OFF").

2.3 The 'STET' CMS Command.

The format of the STET command has been briefly covered in the introduction; this is a more formal description.

The format is:

STET <Fn <Ft <Fm> <(options >>>>

If none of Fn,Ft,Fm are given; STET will prompt you for them. Only as many of these three need be specified as will uniquely identify the file. In addition, STET will follow a built-in search order if the specification is not unique. Files with type PLI, ASSEMBLE, SCRIPT, EXEC, etc will be loaded in preference to TEXT, MODULEs, etc.

Two options are currently available:

Option "N" means load this file without running STETDEF EXEC to set up user defaults. (I.e. use the STET built-in defaults).

Option "count" will specify the width of the data that is to be available to the user.

For example, if you are about to read in a F 80 file, but you know that you want to increase its width to 100 bytes during the editing, then the option "(100" will cause STET to allocate 100 bytes for each data record (instead of 80). This option will also set the width for new files (default 80).

If both options are to be specified; the "N" must come first,

e.g: STET FRED (N 121

2.4 Attention Keys.

For simple editing, all the attention keys can be used. Some of the attention keys have a fixed function; however all the PF (Program Function) keys are user definable. This documentation describes the DEFAULT settings for the PF Keys.

ENTER is used to register changes and cause the activation of line and top-line commands. When Software tabs are enabled (see section on tabs, page 16) and the cursor is in the data area, ENTER also acts as the 'Tab' Key and causes cursor movement.

CLEAR is used to recover from an error (such as an accidental "ERASE INPUT"):- changes on the current screen are not registered. Also any current group definition is cleared.

PA1 Will execute the command "?". That is, the last command you used will be displayed on the top line.
Note: with release 5 of CP/370, PA1 will put you into CP mode: TEST REQ therefore takes over the '?' function. This change from earlier versions of STET is due to the Release 5 CMS/CP changes.

PA2 Is the "=" key: the last top-line command is repeated (e.g. especially for Locate, Change, etc).

PF1 ("W") Is used to create a new window ('bookmark') in the file. A bookmark is a line in the file that will be remembered by the editor as a place that you may want to get back to easily: the bookmarks you set up may be quickly reviewed using the "SKIP" (see below) command, which will take you to each in turn.

PF2 ("SKIP") Up to 20 windows may be set within a file by using the "W" command (default for PF1). PF2 allows the user to scroll between these windows in a circular fashion. The windows may all be reset using the 'RESET' command. (In a structured file, "Out" will reset the current window.)

PF3 ("Home") Homes the cursor. Hit once, the cursor is moved to the command line. If hit again, the cursor will be returned to its last position on the screen. Automatic return is caused by many top line commands (such as "Save" and "H" (Hex)). This change from earlier versions of STET is due to the Release 5 CMS/CP changes.

PF4 ("-36") Scroll Left (decrements the window Origin).

PF5 ("RL") Repeats the last-used Locate Command.

PF6 ("+36") Scroll Right (increments the window Origin)

Keys PF7 through PF12 are used for scrolling:

PF7	("U8")	Scroll up 8 lines (within block)
PF8	("U20")	Up 20 lines
PF9	("TOP")	Goto TOP of block
PF10	("N8")	Down 8 lines
PF11	("N20")	Down 20 lines
PF12	("BOT")	Goto BOTTOM of block

2.5 Tabs.

STET allows both hardware (on-screen) and software (via CPU) tabs: switching between the two modes is achieved by entering the command "T" (q.v. Page 12). If hardware tabs are enabled, a 't' is shown near the beginning of line 1.

2.5.1 Hardware Tabs.

STET hardware tabs are almost identical to those in SPF. Tab positions are set up using the TABS top-line command, and entering a "t" for every column to which one wishes to tabulate.

Since each hardware tab needs a screen attribute byte, there are two mechanisms to permit the user to enter data where attribute characters are present.

Tab attributes are only inserted when hardware tabs are enabled and there is no character where the attribute byte is to go. All tab attribute bytes on the screen can therefore be removed by disabling Hardware Tabs: this is done by entering the command "T" which switches between Hardware and Software tab modes. When in Hardware tabs mode, a "t" is shown near the beginning of the first line. A line beginning with an asterisk in column one is treated as a special case and will not have any tab attributes inserted.

To remove a single attribute byte, positioning the cursor under a tab attribute character and then hitting ENTER will cause it to be changed to a Null, so enabling typing to procede.

2.5.2 Software Tabs.

Whenever hardware tabs are disabled and the cursor is in the data area, the cursor will skip to the next tab column if ENTER is hit. If no tabs are set up, the cursor will be positioned at the first non-blank character of the next line. If hitting ENTER would logically cause the cursor to tab off the bottom of the screen the file will be auto-scrolled up 8 lines.
If the next logical tab position in not on the screen, the window Origin will be shifted so that the cursor can be placed in the right position. This can be used for left <--> right movement within a file by setting up tabs near the right and left edges of the file.
For example, in a 121-byte wide Listing file, the command "tab 10 100" would allow one to scroll left and right in the file just by hitting the ENTER key.

The same tab positions are used as during hardware tabs mode.

3. Structured Editing with STET

STET was originally designed and written in order to give the user a third dimension while editing. The various top-line commands and PF Key defaults described in this section give the user the ability to manipulate and edit collections of lines that are named and known as "Blocks".

In general, Blocks may be considered to be groups of lines that may be referenced by name and included at one or more points.

A reference to a block is made by making the first non-blank character in a line be a (user definable) special character, the default being ")". The next group of non-blank characters (maximum 16) is then taken to be the name of a block which would be inserted at this point if the file were sequential.

You can use the references within the data (together with three of the default PF Keys) to traverse the structure; rather like normal scrolling. This is the real innovation within STET that makes it such a powerful Editor. Other features, like the 'LIST x' command (that displays the Structure of the file graphically) and 'Global Locate', make this approach practical and easy to use. Data can be conveniently moved from one block to another using the normal Move and Copy line commands.

Once within a block, the user is in an environment similar top that of editing a simple sequential file, and all the commands described in section 2 are available. The edit (locate, change, break and merge) commands are restricted in scope to the block currently being edited, except:

LG Locate Group (there is only one group, and this command will find it regardless of which block it is in).

G Global Locate. (The entire file is searched).

Note that a normal (sequential) file will be converted to a STET special format file if any of the special structured editing commands are issued. To prevent this happening inadvertently, the user is asked to confirm such a conversion before it takes place.

There are three advantages to the special format STET files (in addition to the structured nature of the program or text):

a) Information stored with the file, including:

* Tab settings and overlay mask
* Change counts, Attention Key Usage, and date information
* Autosave trigger point
* Character translation mode (Upper/Multi/Lower Case)
* Build and STET file record formats
* Block within the file being edited
* Status of Deblank Flag
* Current Zone settings
* Elapsed time & session count
* Screen format and margins

b) STET files may be RECFM V, so Assembler files held in STET format (for example) would take up less disk space, yet still be immediately editable.

c) STET files are held in core-image format, so fetching a STET file is significantly faster than fetching a normal file. If STET files are held in Fixed format, a further considerable reduction in real fetch time will be realised, together with a 8:1 reduction in the CPU time used during the fetch. (This is due to the entire file being read in one FSREAD operation).

| The format in which STET structured files are stored on disk is described in detail
| in section 3.3 on page 21.

3.1 Structured Editing Commands.

The structured editing commands allow the user to create, delete, list, and explicitly edit structure blocks; together with other functions.

Whenever one of these commands (e.g. FORM) uses a block name, the symbol "=" may be included to cause the substitution of the name of the current block. e.g: If the current block is named 'FRED', then the command "DEL =DY" would delete the block called 'FREDDY'.

FORM This command converts the currently defined group of lines into a named block. e.g. "FORM xyz" will create a new block called 'xyz' and replace the group by a reference to xyz (e.g. ")XYZ"). The group definition is preserved within the new block.

XPAND Will expand all references in the current group. i.e. will replace any block references by the lines in that block. After expansion the group will remain defined, so a further level of block references may be expanded if desired by hitting PA2.

LIST Lists all blocks in the file and gives information about them. If an argument is specified it is first checked to see if it is numeric. If so, the display starts from the 'n'th alphabetical block. If the argument is the name of a block, the display starts from that block. Finally, if the argument is unrecognised (e.g. "LIST *") the display starts from the top of the list. The total number of blocks in the file, and the number of the first block on display is shown on the bottom line of the screen.

LS List the structure of a file in a graphical manner. If no argument is given the structure is displayed from the root. If the argument is numeric, n levels are displayed. If the argument is the name of a block, the sub-tree headed by that block is displayed. Finally, if the argument is unrecognised (e.g. "LIST *") the entire structure is presented. In the structure display, unre-solved references are highlighted; recursive references are highlighted and flagged with '**'. The total number of blocks in the file is shown on the bottom line of the screen.

DELETE Deletes the block specified (e.g. "DEL xyz")

NAME Renames the current block. (e.g. "NAME xyz"). Note that references to the block will not be changed by this command.

EDIT Allows one to explicitly define the next block to be edited. (Instead of using PF2, PF5, PF6 defaults). (e.g. "E xyz")

RESET (or "R") If there are any 'bookmark' windows, these are cleared. If none, then any history is cleared and the current window is set to the ROOT of the file.

CHAR Sets the block reference character e.g. "CHAR !". Default is ')'. If no ar-gument is given, the current block reference character is displayed.

BUild Builds a sequential file from the structure by expanding all block references - will always build from the root. Fn must be specified, Ft and Fm may be specified if desired. The user is warned if there are any unresolved or re-cursive references. The generated file may be serialised (SER command, page 11). '=' may be used in Fn Ft Fm: e.g. "BU = PLI" The built file will have a fixed record format unless "RF V" has been issued. To protect users against accidental overwrite of STET files, a filetype of 'STET' is invalid for BUILD.

PArt Partial Build. Same as BUILD, except that only blocks from the current one 'downwards' will be included.

G As "L" (see section 2.2.1, page 7). A global locate is carried out - the entire file will be searched starting at the top of the current window. Blocks are searched in Alphabetical order starting at the current block so each is only scanned once.

Out Go out a level in the structure (see PF4 description, page 20).

IN Go in a level (see PF5 default description, page 20).

ALong Go along to the next block at the same level (if possible). (See PF6 default description, page 20.)

3.2 Structure manipulation PF Keys.

When editing a structured file three PF Keys are defaulted to the commands ("OUT", "IN", and "ALONG") used for traversing the structure. Their function is analogous to scrolling, but in the third dimension.

A history of editing is kept that allows you to retrace your steps during the editing process. (This history is not currently held between runs like Tabs etc).

PF4 is dual function: if editing the current block was begun by using the PF5 key, then PF4 will transfer the user back to editing the block 'above'. If the EDIT command was used to reach the current block (e.g. "E xyz") then the user will be transfered back to the last block being edited.

PF5 This key takes one down (in) a level. It carries out a function similar to locate, in that the next block reference (searching from the line currently holding the cursor) is located, and you are then presented with that block for editing.

PF6 Will transfer the user to the 'Next' block logically at the same structure level. i.e. hitting PF6 is equivalent to hitting PF4 ('Out') followed by PF5 ('In').

3.3 Storing structured files

STET structured files are saved in core image format; this ensures a fast start-up time. Details of the core image format are given below; this is the format that ALL files being edited have, and is maintained when storing structured files. The first four records of the incore array are always special control records, the fifth is the first block header record. The remainder of the records are data and/or block header records.

3.3.2 Special records: ROCK

Byte	Length	Description
1	1	'R' - always the first record in the file array.
2	1	Version flag. 'V' in files edited with the later versions of STET.
3	2	Version number * 100
5	2	Free chain forward pointer; zero outside edit sessions.
7	2	Free chain backwards pointer; zero outside edit sessions.
9	2	X'0000'
11	2	The alpabetical number of the block currently being edited.
13	16	STET identification field. Has undergone several changes with the different levels of STET.
29	2	Record number of the header of the block being edited.
31	2	File length
33	2	Count of records on the free chain. (Zero outside edit sessions.)
35	1	Case flag of file. (U/M/L)
36	1	Current block reference character. (Default=')')
37	1	't'==> hardware tabs on. ' '==> software tabs on
38	1	'*'==> some tabs stop exist, else they do not.
39	1	Serialisation flag on/off ('s'/' ')
40	1	SAVE/BUILD record format
41	2	Point at which to autosave
43	1	De blank flag: 'D'==> ON
44	1	Format to store STET files in. (F or V)
45	2	Total number of blocks
47	2	Zone start column
49	2	Zone LENGTH (Note, NOT the end column)
51	2	Screen width (columns) Currently unused
53	2	Screen depth (rows) Currently unused
55	1	Position of line command area (L/R)
56	1	Command separator (default='#')
57	3	Serialisation increment (default=010)
60	13	Unused (set to nulls for posterity)
73	2	LRECL for build/file commands
75	2	Count of EDIT sessions
77	4	time count (seconds) of this edit session.

3.3.2 Special records: TABS

Byte	Length	Description
1	1	'T' - the SECOND record in the array defines the tab mask
2	3	Unused
5	2	Unused (=X'0002' in older STET files)
7	2	Unused (=X'0002' in older STET files)
9	rest	Contains a 't' in the columns for which a TAB stop exists. A null (X'00') is stored in all other columns.

3.3.3 Special records: MASK

Byte	Length	Description
1	1	'M' - the THIRD record in the array defines the mask used in overlays/inserts.
2	3	Unused
5	2	Unused (=X'0003' in older STET files)
7	2	Unused (=X'0003' in older STET files)
9	rest	Mask data........ see page 12 for details on using a mask.

3.3.4 Special records: PF-key log

Byte	Length	Description
1	1	Type = 'Z' - the FOURTH record in the array. The log of attention and PF key usage.
2	3	Unused
5	64	An array of 32 fixed bin(15) numbers. This maps onto the attention keys in the following way:

Index	Attention Key
0	ENTER
1	PA1
2	PA2
3	PA3
4	Light-pen
5-6	Test Request (dependent on READ MODIFIED)
7	CLEAR
8-32	PF1 - PF24

3.3.5 Special records: BLOCK HEADERS

Byte	Length	Description
1	1	'H' - indicates a BLOCK header record.
2	1	Build recursion flag. Only used during BUILD commands; otherwise is blank.
3	2	Unused
5	2	Pointer to first data record in this block.
7	2	Pointer to last data record in this block. Note: In a block without data records, these two pointers point to the header record itself.
9	2	Block position in the alphabetical list of blocks.
11	2	Unused.
13	2	Length of the block - may be zero.
15	16	Block name.
31	7	Date when the block was last altered. (e.g. 17Dec80)
38	7	Date when the block was created.
45	2	Number of the record which is currently displayed on line three of the screen.
47	2	Cursor row on the screen.
49	2	Cursor column on the screen.
51	2	Current left hand margin.
53	2	Flag to say whether the name to be used in a partial build (see next definition) is set up. When the name has been set, is X'FFFF'.
55	16	File-id associated with this block, to be used when doing a partial build of this block. NOT YET FULLY IMPLEMENTED.

3.3.6 Structure of data record

Byte	Length	Description
1	1	'D' - data record.
2	1	'B' ==> block reference on this line, otherwise unused.
3	2	Count of the number of changes made to this line.
5	2	Forward pointer (index into array) to next record.
7	2	Backwards pointer (index into array) to previous record.
9	rest	File data........what you actually see on the screen.

4. CMS interface - EXEC macros and user defaults.

STET has a very simple interface with CMS which is surprisingly powerful. Using it a number of useful EXEC's and edit Macros may be simply generated, without having to learn a new macro language. The same mechanism is used for setting up user defaults where required.

When STET is running, it will 'poll' the CMS Stack just before writing each new screen: if it finds there is data in the stack, it will be read in (up to 30 characters) and executed as a STET top line command. A new STET command may therefore be written which is in effect a normal CMS 'EXEC' file, whose function is to stack a number of top-line commands. The EXEC may be simply invoked using the '.' command (See page 13).

The same mechanism is used for defaults: as soon as a file is loaded, the EXEC named 'STETDEF' will be executed by STET, with the following arguments:

```
&1 = File Name
&2 = File Type
&3 = File Mode
&4 = File Format (F/V)
&5 = File Logical Record length
&6 = 'NEW' or 'OLD'
&7 = 'SEQ' or 'STET'
```

The EXEC may then determine which commands are to be executed for this type of file, and stack them for STET to execute. The execution of STETDEF may be inhibited by adding the option " (N " (see page 14) to the original list of Filename etc given to STET.

A trace facility ("TR") allows you to trace the execution of commands as they are read off the stack: each command is displayed before it is executed (you may then alter or delete it if necessary before hitting ENTER).

5. Summary of top-line commands & abbreviations.

Comm.	**Ab.**	**Function.**
ALONG	AL	Move to next block at same level (if possible)
AUTO	A	Query/set Autosave trigger count, or inhibit Autosave
B		Break all lines in group at specified character string
BOT	BO	Go to bottom of block
BUILD	BU	Build a sequential file from tree structure
C		Change occurrence(s) of string to another string
CASE	CA	Set translation mode: UPPER, Multi or lower case.
CHAR	CH	Query or set character that identifies block references.
CMS	CM	Issue CMS command or enter CMS SUBSET
CP		Issue CP command / Enter CP mode
D		Delete down to specified string (Careful!)
DBLANK	DB	Switch Deblanking On/Off
DELETE	DE	Delete named block
EDIT	E	Edit named block
FILE	FI	File edited data on disk, and leave STET
FM		Set the default Filemode
FN		Set the default Filename
FORM	F	Form a new named block from group, replace group with ref.
FT		Set the default Filetype
G		Global (all-file) locate string
GET	GE	Get CMS (sequential) file & add to end of block as a group
GROUP	GR	Delete/Restore Group
H		Hex mode on/off
HELP	HE	Show Help (Then hit PA2 for further information)
HOME	HO	Home the Cursor to the command area
IN	I	Go in a level of the structure
L		Locate string
LE		Change or display the current Line-end character
LG		Locate top of group
LIST	LI	List block names in alphabetical order. <n>
LS		List block structure map (in STET-type files).
MASK	M	Set up insertion/overlay mask
MERGE	ME	Merge lines in group to form 1 line (group can be restored)
MSG	MS	Display a user-defined error message
N		Scroll down file n lines
NAME	NA	Rename current block
O		Set window Origin (left margin)
OUT	OU	Go out a level of the structure
PART	PA	Partial BUILD
PRINT	PR	Print file (Printer Class may be specified)
PG	PG	Write lines within the current group to disk
PUNCH	PU	Punch file (Build or partial build may be requested)
QUIT	QU	Leave STET, WITHOUT saving changes
RESET	R	1) Clear all Windows. 2) Return to structure root.
RF		Set Record Format (for SAVE, FILE, or BUILD) to V or F.
RL		Repeat last Locate command
RS		Set Record Format (for STET files) to V or F
SAVE	SA	Save file on disk, do not leave STET
SERIAL	S	Serialise sequential file (eg: for SAVE, FILE or BUILD)
SF		Screen format Left/Right/Zone/nn (nn is width)
SHOW	SH	Show change counts by line
SKIP	SK	Move to next bookmark in the list (if any set by "W")
T		Switch between hardware and software tabs
TABS	TA	Set up tab positions
TOP	TO	Go to top of block
TRACE	TR	Trace execution of commands from the CMS stack
U		Scroll up file n lines
WINDOW	W	Create a window (bookmark) in the file
XPAND	X	Expand all block references in group
ZONE	Z	Set current zone margins
n		Move to line n of file <cc and column cc>
+		Scroll right (Increment Window Origin)
-		Scroll left (Decrement Window Origin)
?		Display last command
??		Display penultimate command
=		Repeat last command (= PA2)
==		Repeat penultimate command

>	Indent lines in group n columns
<	Shift lines in group n columns to the left
.	Execute an EXEC file (equivalent to "CMS EXEC ..."
*	(Command prefix) Do not add this command to stack.

6. Summary of line commands & their function.

For all line commands if a repetition factor is allowed, but not specified, a default of 1 is assumed. '*' means 'do to the end of the block/file'.

Command	**Function.**
Cn	Copy group n times after this line (and leave group definition)
Dn	Delete n lines (including this one).
In	Insert n copies of the current mask.
Ln	Translate n lines to lower-case.
Mn	Move group n times after this line (and delete original group)
On	Overlay n lines with the current mask.
Rn	Repeat the current line n times. (i.e. R3 inserts three copies)
Un	Translate n lines to upper-case.
S	Split line at column given by the cursor.
Sn	Split line at column n, except that if the cursor is in the data area, the new line will be split at the cursor and begin in column n.

"Group" commands - recognised by a 'g' as the second letter. (Glen is group length.)

DG	Delete this group (must be entered on a line of the group).
LG	Translate all line(s) in the group to lowercase.
MG	Merge/overlay group onto the next glen lines.
OG	Overlay all line(s) of the group with the current mask.
RG	Restore the group after this line.
UG	Translate all line(s) in the group to uppercase.

7. Summary of Attention Key Functions, and PF Key defaults.

```
PA1     (CP Release 5): Enter CP mode. Earlier releases: "?".
PA2     "=" (Repeat last command)
ENTER   Register changes/commands, also Software tab key

CLEAR   Ignore any changes just entered, and clear any current group definition
TEST REQ (CP Release 5): "?". Earlier releases: Enter CP mode.

PF1     Window - Create Bookmark
PF2     Skip - Scroll around windows
PF3     Home - Move the cursor to the command line
PF4     Out  - go back up a level in the structure  / Scroll left
PF5     In   - go down a level in the structure     / Repeat Locate
PF6     Along - go across to the next block         / Scroll right
PF7     Scroll up 8 lines within block
PF8     Scroll up 20 lines
PF9     Go to TOP of block
PF10    Scroll down 8 lines
PF11    Scroll down 20 lines
PF12    Go to BOTTOM of block
```

```
œ††††††††††††††††††††††††††††††††††††††††††††††††’
|                                                |
|  ==== STET 4 ====   œ†††††††††††††††††††††††’  |
|  PF. Key Template   |CREATE |SKIP   | HOME  |  |
|                     |WINDOW |ROUND  | CURSOR|  |
|  ENTER for tabs etc |       |WINDOWS|       |  |
|  CLEAR to restore    ††††††† ††††††† †††††††|  |
|        changes      | OUT   | IN    | ALONG |  |
|  TEST REQ = "?"/CP  | BLOCK | BLOCK | BLOCK |  |
|                     | † † †   † † †   † † † |  |
|  "HELP" top†line    |  <==  |REP LOC|  ==>  |  |
|  command for help    ††††††† ††††††† †††††††|  |
|  "PF" to redefine   |       |       | TOP   |  |
|  key meanings       |       | Up 20 |       |  |
|                     | Up 8  |       |       |  |
|  œ†††††††††††††††’   ††††††† ††††††† †††††††|  |
|  |       |       |  |Down 8 |       |       |  |
|  |"?"/CP |   =   |  |       |Down 20|       |  |
|  |       |       |  |       |       |BOTTOM |  |
|  ¿†††††††††††††††’  ¿†††††††††††††††††††††††’  |
¿††††††††††††††††††††††††††††††††††††††††††††††††’
```

S.Davies/M.F.Cowlishaw
28th Dec 1980

--- End of Document ---