4.00.00 THE PL/9 EDITOR

One of the strongest features of PL/9 is its built-in editor, which cuts out much of the loading and saving that makes using other compilers a time-consuming business. The editor is broadly similar to, and compatible with, the TSC text editing system, although the commands are not identical. Anyone familiar with the latter should have little difficulty in using the PL/9 editor.

The user is encouraged to make free use of spaces to improve the readability of a program, since these are stored in a compacted form and do not take up any extra room in memory or on disc.

The editor prints line numbers while listing the source file; these numbers are not part of the file, however, and are not saved on disc. It also allows editing of a line as it is being entered, by means of the following control characters:

```
TAB.........generates three spaces.
BACKSPACE...moves the cursor to the left destructively one place.
CANCEL......erases the entire line.
ESCAPE......in the left column terminates the insert session.
RETURN......generates a new line.
```

The default key values supplied may not suit your terminal. The SETPL9 program, described in the previous section, provides a convenient method of altering the keycodes PL/9 recognizes to those available on your terminal.

PL/9 will accept ANY text file that is stored on disk in the TSC TEXT EDITOR FORMAT. Many cursor oriented text editors/word processors, e.g. SCREDITOR III, save text in this format.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                       *
*   IF YOU USE AN EDITOR OTHER THAN THE PL/9 RESIDENT EDITOR  YOU       *
*   MUST ENSURE THAT THE LINE LENGTH NEVER EXCEEDS 127 CHARACTERS       *
*                                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

The PL/9 editor will not allow you to enter a source line longer than 127 characters, nor will it allow you to use the 'Z' command to append two lines that will create a line longer than 127 characters, nor will it allow you to use the 'C' command to change a string that will create a line longer than 127 characters.

If you have created a source file in another editor that contains lines with more than 127 characters you will be greeted with the message 'LINE TOO LONG' whenever you use the 'L' (LOAD) or 'R' (READ) commands of the editor. The offending line will be truncated to 127 characters with the last four characters in the line set to "%%%%". The four percent symbols present in the offending line(s) will enable you to use the 'F' (FIND) command to locate the lines.

If you use an editor other than the TSC TEXT EDITOR or SCREDITOR III to enter your programs and have problems with PL/9's editor don't blame us! The fault lies with the file format produced by your editor. The STYLOGRAPH disk file format is typical example of a non-standard format that will be absolutely useless to PL/9 unless you are very careful when you are using it, i.e. ensure that each and every line terminates in a carriage return!

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                       *
*  All PL/9 PROGRAMS may be entered in UPPER-CASE or lower-case letters.*
*                                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

There are two programs presented in the 'USERS GUIDE' (section ten) that will convert a text file from lower-case to UPPER-CASE and vice versa but will leave the letters inside of strings and comments alone.

4.00.01 DETAILED DESCRIPTION OF EDITOR COMMANDS

Each of the editor commands is described fully in the following paragraphs. This section groups the various editor commands by function. A command summary can also be found in a separate document.

Generally speaking the PL/9 editor does not support multiple command entry. Edit commands must be entered singly i.e. the command followed by a carriage return. There are a few exceptions to this rule which will be described as they are encountered.

4.00.02 EDITOR COMMAND SYMBOLS

The following symbols will be used as part of the definition of the editing, compiler and tracer commands:

<CR> represents a carriage return.

<> symbols are used to enclose a variable.

<NUMBER> represents a decimal number such as 36 or 192, and which defaults to one if it is omitted.

<TARGET> represents the decimal number of lines specified by the command, and defaults to one if none is given.

<#TARGET> represents the decimal line number specified by the command.

[] symbols indicate that the enclosed data is optional and may be omitted, in which case a default value is usually supplied by PL/9.

The TRACE (T) command and COMPILE (A) command are called from within the editor. Each of these commands is described in their own sections.

M O D E C O N T R O L

The following commands are only active when the (#) prompt is present.

I

INSERT lines into the file. The editor will change its prompt from (#) to (+) to remind you that you are now in the insert mode. Every line you type from now on will be added to the file immediately ABOVE the current line. This may seem strange at first if you are used to the TSC editor, but it has the advantage that having added a line to the file the current line is still the same one as before. It also enables you to insert a line above line number 1, something which is very awkward with the TSC editor. The current line and the rest of the file will be moved to make room for the new line, so the file will always be in sequence, i.e. it is automatically re-numbered each time lines are added.

When you have finished adding lines, ensure that the cursor is at the left margin and press the <ESCAPE> key. You will then be returned to the editor command level, with the prompt restored to a (#). Your current line will now have a new (larger) number because of the extra lines inserted above it.

X

EXIT to FLEX. You will be prompted 'IS TEXT SECURE?' which must be answered 'Y' to enter FLEX. Any other response will return you to the editor.

M

MONITOR. Enter the ROM System Monitor. A warning message will be posted along with instructions on how to re-enter PL/9 through the warm start address.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                         *
*  NEVER RE-ENTER PL/9 AT THE COLD START ADDRESS $0000.   *
*                                                         *
*  Doing so will cause the existing file to be erased!    *
*                                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

See the section on file start and file end markers at the end of this section of the manual for techniques to use in the event that you accidentally loose a file.

/<COMMAND>

Execute a FLEX command. Warning: Only use commands that reside in the Utility Command Area, or you may risk bombing PL/9, with possibly disastrous results.

The only commands we recommend are: CAT, DIR, LIST, DELETE, RENAME, ZAP, TTYSET, and ASN. Using COPY, for example, is a definite no-no!

LINE POSITIONING COMMANDS

The following commands are active only when the (#) prompt is present.

<NUMBER>[COMMAND]

Make line <NUMBER> the current line, then execute the command (optional) that follows the number.

1 or ^

Go to the first line in the file.

B or !

Go to the bottom (/EOF) of the file.

+<NUMBER>

Move down <NUMBER> lines from the current position.

<NUMBER>

Move up <NUMBER> lines from the current position. This command may be followed by the print command, i.e. -23P23<CR> would back up 23 lines and print 23 lines.

<CR>

Display the current line.

<ESCAPE>

Hitting the escape key will cause the next line in the file to be displayed and to become the current line. This provides a convenient method of 'stepping' through your file a line at a time. If you are already at the bottom of the file then you will get /EOF printed every time.

FILE ORIENTED COMMANDS

N

NEW file. This command erases the file currently in memory to make room for a new file. PL/9 will prompt with "ARE YOU SURE?" to prevent you from inadvertently erasing your file.

The file is not actually deleted from memory when 'N' is used. The file is 'erased' by setting the end of file marker to beginning of the text buffer. See the section on file start and file end markers at the end of this section of the manual for techniques to use in the event that you accidentally loose a file.

P<TARGET>

PRINT a number of lines of the file on the terminal, starting at the current line. The last line printed becomes the current line. Examples:

#P50<CR>	Print 50 lines, starting at the current line.
#142P8<CR>	Print 8 lines, starting with line 142.
#P<CR>	Print some more lines.

In the last example, the number of lines printed will be the same as that specified by the last P command. When you start up PL/9, this number will be preset to one less than that given by the TTYSET DP count (see your FLEX manual). This facility allows you to scan your file N lines at a time, by pressing P then <CR> repeatedly.

D<TARGET> or D<#TARGET>

DELETE line(s). The first form will delete the requested NUMBER of lines from the file starting with the current line. Lines above the current line are unaffected and it does not matter if you specify a target that is beyond the bottom of the file.

The second form deletes all of the lines starting with the current line TO AND INCLUDING, the line number followed by the #

If D is typed by itself then only the current line is deleted. After deletion, the editor displays the new current line. Examples:

#D15	Delete 15 lines, starting at the current line.
#81D3	Delete 3 lines, starting at line 81. The line that was previously line 84 will become the current line, which will still be numbered 81.
#43D#72	Delete from line 43 to 72 inclusive. The line that was previously line 73 will become the current line.

CAUTION: Be very careful when deleting multiple lines as the file will automatically be renumbered after EACH line is deleted. When you wish to delete several lines simply start at the highest line number and work toward the lowest.

L I N E E D I T I N G

The following commands are active only when the (#) prompt is present.

<CHAR>

OVERLAY the current line. This command is useful when a change has to be made near the start of a line. PL/9 displays the line in question, then prompts immediately underneath with a > symbol. You can then type in a new line containing only the characters you wish to change, in their correct positions under the displayed line. If <CHAR> is omitted, then spaces typed in the new line indicate characters that should be left alone; if <CHAR> is supplied then it becomes the character that you type to leave the corresponding one in the original as it was. The following example first shows the current line which is then overlayed twice:

```
0123    IF COUNT > 5 THEN CHAR = 'X      /* TEST CASE */
    #0
0123    IF COUNT > 5 THEN CHAR = 'X      /* TEST CASE */
    >           =                ;
0123    IF COUNT = 5 THEN CHAR = 'X;     /* TEST CASE */

    #0-
0123    IF COUNT = 5 THEN CHAR = 'X;     /* TEST CASE */
    >-------------------------------------SPECIAL CASE */
0123    IF COUNT = 5 THEN CHAR = 'X;     /* SPECIAL CASE */
    #
```

E

EDIT the current line. This command causes PL/9 to display the line and to leave the cursor at the end, as if you had just typed it in but had not yet pressed <CR>. The line can then be altered by backspacing or by adding more text.

=<TEXT>

Delete the current line and put in its place the remainder of the command line. Example:

```
    #52=   REPEAT COUNT=COUNT-1 UNTIL COUNT=0;
    #52
0052    REPEAT COUNT=COUNT-1 UNTIL COUNT=0;

    #
```

\

SPLIT up the current line. Every semicolon in the line will be taken as a line delimiter. Text from that point on will be made into a new line, indented the same as the current line. An example should make this clear; suppose that this is the current line:

```
0095    COUNT=0; FLAG=FALSE; BUFFER_POINTER=2;
    #\
0095    COUNT=0;
```

```
    #P3
0095    COUNT=0;
0096    FLAG=FALSE;
0097    BUFFER_POINTER=2;
              #
```

Z

CONCATENATE two lines, i.e. remove the carriage return from the end of the current line. If there are leading spaces at the start of the following line then these will be reduced to a single space. This command performs the inverse function of the \ command above, restoring the line to its original state. The two commands together help the programmer to produce a neat and readable source file.

N O T E

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                         *
*   IF  YOU  ATTEMPT TO USE THIS COMMAND TO CONCATENATE TWO LINES         *
*   THAT WILL CREATE A LINE THAT IS GREATER THAN  127  CHARACTERS         *
*   LONG PL/9 WILL REFUSE TO DO IT AND COMPLAINS WITH THE MESSAGE         *
*   'LINE TOO LONG'.                                                      *
*                                                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

G L O B A L E D I T I N G

The following commands are active only when the (#) prompt is present.

F<NUMBER>/<STRING1>

FIND the next <NUMBER> occurrences of <STRING1>, starting with the line following
the current line. If <NUMBER> is omitted it defaults to one. Any delimiter can be
used in place of the / symbol. Examples:

#F20/IF	Find the next 20 occurrences of IF.
#F,/What?/	Find the next occurrence of /What?/.
#^F!/HELLO/	Find every occurrence of HELLO from the top of the file (^) but excluding the top line, to the bottom of the file (!).

C<NUMBER>/<STRING1>/<STRING2>

CHANGE the next <NUMBER> occurrences of <STRING1> into <STRING2>, starting with the current line. Only the first occurrence of <STRING1> will be changed on any one line. Again any delimiter is allowed. Examples:

#C/THIS/THAT	Change THIS into THAT in the current line.
#93C;WILE;WHILE	Change WILE in line 93 into WHILE.
#^C!/THESE/THOSE	Change every occurrence of THESE to THOSE.

NOTE 1: If more than one occurrence of <STRING1> in is present on a given line only the first occurrence will be altered by CHANGE, the second occurrence will be ignored.

NOTE 2: (F)ind and (C)hange operate by setting up two buffers, one for <STRING1> and the other for <STRING2>, every time either command is called. If an incomplete command line is typed, only the specified data will be updated. For example, F23 instructs PL/9 to find the next 23 occurrences of the previously defined <STRING1>, while C8;VAR (note the missing second delimiter) will change the next 8 occurrences of VAR into whatever <STRING2> had been previously set to.

This facility simplifies making global changes where the same string occurs more than once on a line.

NOTE 3: <STRING1> may contain one or more "?" as "don't care" characters. For example, ^F!/VAR? finds all occurrences of VAR1, VAR2, VAR3 etc.

NOTE 4: The reason that 'F' does not operate on the current line is so that you can use 'F' and 'C' in succession to selectively change strings.

D I S K F I L E H A N D L I N G

The following commands are active only when the (#) prompt is present.

Q or ?

QUERY the default filenames. PL/9 maintains a table of filenames which are used in conjunction with the LOAD (L), SAVE (S), READ (R), WRITE (W), COMPILE to OBJECT (A:O) and COMPILE to LISTING (A:L) commands. If QUERY is used after invoking PL/9 from the FLEX command line before loading any source file the defaults defined by the configuration program SETPL9 will be presented thus:

```
    Present defaults are:
    ====================

    L/S = 1.         .PL9
    R/W = 1.SCRATCH .SCR
    A:O = 1.         .BIN
    A:L = 1.         .OUT
```

Now suppose that you wish to load a source file named 'TEST.PL9' that resides on drive #1. Since drive #1 is the default drive and '.PL9' is the default extension all you would have to do is issue the command:

```
L=TEST<CR>
```

If you then QUERY the default filenames you will get:

```
    Present defaults are:
    ====================

    L/S = 1.TEST     .PL9
    R/W = 1.SCRATCH .SCR
    A:O = 1.TEST     .BIN
    A:L = 1.TEST     .OUT
```

Note that the filename for the L/S, A:O and the A:L commands now echo the filename just specified. This GLOBAL filename updating will occur whenever you specify a filename when using the L, S, and A:O command but will not occur when using the A:L command. The the default drive number and default extension are INDIVIDUALLY updated whenever the L, S, or A:O commands are issued.

In summary whenever you specify a filename you can give any combination of drive number, filename and extension, and PL/9 will take whatever you omit from the current default for the command you are using. When you specify a new value for any of the defaults for L, S, and A:O the existing default will be replaced with the new value. Using SAVE (S) as an example:

```
    #S<CR>                  Save to 1.TEST.PL9
    #S=0<CR>                Save to 0.TEST.PL9
    #S=.TXT.1               Save to 1.TEST.TXT (note the order)
    #S=0.JUNK<CR>           Save to 0.JUNK.TXT
    #S=.PL9<CR>             Save to 0.JUNK.PL9
    #S<CR>                  Save to 0.JUNK.PL9
```

D I S K F I L E H A N D L I N G

Q or ? (continued)

If we now used the Q command the default values would be as follows:

```
    Present defaults are:
    ====================

    L/S = 0.JUNK    .PL9
    R/W = 1.SCRATCH .SCR
    A:O = 1.JUNK    .BIN
    A:L = 1.JUNK    .OUT
```

Note that the default filename has been GLOBALLY upated even though we only used the SAVE (S) command. Note also that the default drive number and file extension have been INDIVIDUALLY updated. The same thing will happen whenever you use the LOAD (L) command or the COMPILE to OBJECT (A:O) command.

The A:L command can be used to direct listing output to an alternative drive, and alternative filename or an alternative file extension. For example:

```
    A:L=0.TEMP.TXT<CR>
```

This command, does not alter any of the default values however.

The default values of the READ (R) and WRITE (W) command are 100% alterable and will be updated whenever any new information is given after either of these commands. For example:

```
    #W23<>                Write to 1.SCRATCH.SCR
    #W23=0<CR>            Write to 0.SCRATCH.SCR  (default drive now 0)
    #W23=TEMP<CR>         Write to 0.TEMP.SCR     (default filename now TEMP)
    #W23=.TMP<CR>         Write to 0.TEMP.TMP     (default extension now .TMP)

    #R=1.TEST.TXT         Read file 1.TEST.TXT (the default drive, file name,
                          and file extension will be updated accordingly.
```

NOTE 1: Whenever using L, S, R, W, A:O, or A:L the filename can be expressed fully in either of the two FLEX standard forms. For example:

```
    S=1.MYFILE.TXT

          or

    S=MYFILE.TXT.1
```

NOTE 2: QUERY does not show the default drive number of file extension for the INCLUDE directive as determined by the configuration program SETPL9.

D I S K F I L E H A N D L I N G

The following commands are active only when the (#) prompt is present.

L[=<FILENAME>]

LOAD a disc file. The default drive and file extension specified when configuring PL/9 with the SETPL9 command need not be supplied. The filename supplied will become the default name to be used by further Load, Save, Assemble to Object or Assemble to Listing file commands.

S[=<FILENAME>]

SAVE the file on disc in TSC editor format. The editor will over-write any existing file of the same name. Filenames have the default extension ".PL9" (but this can be changed using SETPL9). If the filename is omitted then the name of the file that was loaded will be used again, allowing files to be loaded, modified and re-saved without the name having to be typed more than once.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                       *
*  PL/9 does not make backup copies of files; if you require a          *
*  backup you must create it explicitly (e.g. S=FILE.BAK).              *
*                                                                       *
*  Alternatively you can invoke the FLEX RENAME command from            *
*  within PL9 and rename the file just loaded into the editor           *
*  (e.g. /RENAME,1.TEST.PL9,1.TEST.BAK).                                *
*                                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

W<TARGET>[=<FILENAME>] or W<#TARGET>...

WRITE part of a file to disc. As for (S) except that PL/9 writes only the specified number of lines, starting at the current line in the first form, and writes from the current line to the specified line in the second form. The default filename in this case is 1.SCRATCH.SCR, but this may be changed using SETPL9.

R[=<FILENAME>]

READ in a file, inserting it into the buffer immediately above the current line. The default filename is 1.SCRATCH.SCR, as for (W). These two commands enable block moves to be made safely, by writing part of the file to disc and then re-loading it at the new position. This technique for block copy-move operations may be a bit inconvenient at times but it does away with the overhead of reserving a large chunk of memory for a seldom used text buffer.

RECOVERING A FILE IN MEMORY

If your System Monitor has a memory dump facility that also displays the contents of memory in ASCII on the VDU screen you stand a 50-50 chance of recovering a file that has been lost though an accidental use of the 'N' command or re-entering PL/9 through the cold start entry point at $0000.

This same technique can also save a file in memory when a system crash occurs, but this time the odds are about 1 in 10 that you will be successful.

The first case concerns an accidental use of the 'N' command or a cold start of PL/9. In both of these cases you can be confident that the original file is still present in memory and intact. What you have to do is enter your system monitor and dump the memory contents out to your VDU starting at the memory location CONTAINED in $4000/1. This is the beginning of file marker. As you work your way through the file you should recognize the text of your source file. Keep searching until you find the last line of the file. The memory location that you are interested in is the location of the first byte past the carriage return ($0D) in the last line. Once you locate this position in the file make a note of the memory location. Use your system monitor memory examine and change facility to alter the contents of $4002/3 to the memory address just noted. Now warm start PL/9 by a JUMP to $0003. Your file should be back to normal.

The second case concerns recovering a file when a system crash has occurred. In these circumstances the following course of action should be followed to the letter.

(1) Hit hardware RESET.

(2) Examine the contents of memory location $4002/3 and make a note of the address pointed to.

(3) Re-boot FLEX using a disk that does not have a STARTUP file on it. This is very important unless you are absolutely 100% positive that your STARTUP file does not cause the memory below say $B800 to be altered.

(4) Use the 'GET' command to load PL/9, i.e. +++GET,PL9.CMD<CR>

(5) Enter your system monitor. Use the system monitor dump memory command to display the contents of memory starting about 500 bytes or so before the address noted in step (2). Work your way up to the end of the file as described in the earlier recovery instructions and verify that the address noted does in fact point to the end of the text file. If it doesn't then go back to the beginning of the file and start working your way up it until the text becomes junk. Make a note of the address of the byte following the address of the last sensible line in the file. Insert this address into to memory location $4002.

(6) Open both disk drive doors, unless you like to live dangerously!

(7) Warm start PL/9 by jumping to $0003.

(8) With a bit of luck you will have recovered your file or at least a reasonable part of it. Save it out to disk with a full file specification, i.e. #S=1.CRASH.SAV<CR>

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK