APPENDIX K

FUTIL
OS/8 FILE UTILITY PROGRAM

FUTIL was originally developed by Jim Crapuchettes of Menlo Computer Associates, Inc., Menlo Park, CA. It is now included within the OS/8 Extension Kit for the convenience of PDP-8 users.

K.1 INTRODUCTION

K.1.1 Description

FUTIL enables a user to examine and modify the contents of mass storage devices. It is the only program currently available that can be used to patch programs containing overlays (F4/LOAD outputs). Other possible uses include examination and repair of OS/8 directories; bad block checking and correction; decimal/octal conversion of double precision numbers; output of the Core Control Block (CCB) of ".SV" files and the HEADER of ".LD" files; and the creation of special directories. Supporting these functions is signed double-precision arithmetic expression evaluation that can be used in the command syntax whenever a numeric value is needed.

FUTIL's command set is divided into two groups of commands. The first group uses single letters to direct the program in the examination and modification of single words on the device specified. The second group of commands uses command words to direct the program in the dumping, listing, modifying and searching of the device on a block-by-block basis. Also included in this group is a series of commands to direct the program in some auxiliary functions including setting and resetting switches and variables within the program, showing current FUTIL parameters.

Several examples are given in Section K.4. The first two examples, especially, are simple and well-documented to acquaint you with the features of FUTIL. You may want to look at them at this point to get a better understanding of the material that follows.

K.1.2 Special Characters Used in This Appendix

To help increase clarity, the characters single quote ('), double quote ("), angle brackets (< and >) and square brackets ([and]) have been used to help separate special items from the words around them. The single quote character is used to surround a word-type command; for example, the 'FORMAT' option 'SET's up the format in which output is to be done. The double quote is used to surround an item whose actual name is being used; for example, the "RETURN" key is the key on the Teletype that has that word printed on it. The angle brackets are used to surround the name of a type of item (a syntactical type); for example, "<n>" means that a NUMERIC ITEM is to be used. The square brackets are used to surround optional items; for example, "w[ord]" would indicate that the characters "ord" may be supplied optionally.

K.1.3 Special Characters Used in FUTIL

Several characters, when keyed, cause immediate action from the program. Typing either CTRL/P (which prints "^P") or CTRL/C (which prints "^C") will immediately cause the program to stop whatever it is doing. CTRL/P then causes the program to go back to command input mode and wait for you, while CTRL/C calls the OS/8 Monitor (as it does with most system programs). CTRL/S and CTRL/Q control program execution (including all I/O). Typing CTRL/S at any time will cause the program to pause and wait for either CTRL/C, CTRL/P or CTRL/Q. Typing CTRL/Q will then allow program execution to resume. Any other characters entered at this point will be simply ignored. If a CTRL/Q is typed by itself at any time, it is simply ignored.

NOTE

CTRL/S and CTRL/Q are active at all times, not just during console output. The result is that both input from the console and program execution with no console interaction (such as 'SCAN', 'WORD' and 'STRING' command execution) can and will be paused and restarted with these keys.

During console terminal input, three other keys can be used to help with editing the input string of characters. These keys are RUBOUT, CTRL/U (which prints "^U") and CTRL/R (which prints "^R"). The action of RUBOUT and CTRL/U is exactly the same as it is for the OS/8 Monitor and Command Decoder (including usage of "scope mode" operation to change the action of the RUBOUT key from echoing the rubbed out characters between backslashes to erasing the characters from the screen). The action of CTRL/R is the same as that of the LINE-FEED key for the Monitor and Command Decoder.

For those users with upper-lower case terminals, the program translates all lower case characters received from the keyboard to upper case. The characters are echoed and handled internally as upper case characters. While this makes use easier, it does not allow any lower-case characters to be input directly. In those cases where lower-case codes are needed in the modification of a file, either use the codes directly or use a text editor. Note that this translation occurs only on input. Lower case characters in a file will be printed to the best ability of the output device. This may produce incorrect results on upper-lower case line printers.

All of the commands are taken in context, that is, many of the characters which are used in the single character command set will not be considered to be commands if they are included in a line which begins with a command word or if they are embedded within expressions.

The carriage-return ("RETURN") always starts command execution, and is the terminator for all word-type command lines.

K.1.4 Running FUTIL

FUTIL is run using the OS/8 Monitor command "R FUTIL" (or "RU dev:FUTIL").

When started, FUTIL is set up to access the system device, the 'ERROR' message output mode is set to 'LONG', the access 'MODE' is set to 'NORMAL' and no file is known. To access some other device, give the command 'SET DEVICE dev'. To set the 'ERROR' mode to 'SHORT', give the command 'SET ERROR SHORT'. To use some other access mode, give a 'SET MODE <mode>' command with a <mode> of 'LOAD', 'OFFSET' or 'SAVE'. When in 'OFFSET' mode, the 'OFFSET' to be used can be specified by the command 'SET OFFSET nnnn'. Lastly, a file lookup can be performed by giving a 'FILE' command (with three default extensions).

K.1.5 Access Method

The program accesses the OS/8 device one OS/8 block (256 words) at a time. For every location specified, the real block and word are determined and compared with the current block in memory. If the desired block and current block are not the same, the <something-changed> flag is checked to see if anything has been changed in the current block. If nothing has been changed, the new block is read in. If something has been changed, the current (modified) block is first written out and then the new block is read in. This action happens correctly even when the access mode is changed because it is done at the level of the OS/8 block number right before calling the current 'DEVICE' handler. The status of the <something-changed> flag can be determined by simply 'SHOW'ing 'ABS', 'REL' or 'ODT' locations. If the flag is set, the word "MOD" will be output following location information.

The contents of the OS/8 device are therefore not changed unless the block in which changes are made is written out either implicitly, as described above, or explicitly, using the 'WRITE' command (which is discussed near the end of the section on word-type commands). The result is that typing CTRL/C before writing out the current block (assuming it has been modified) will return to the Monitor without actually modifying the contents of the device

itself. Note, also, that only one implicit write attempt is ever made by the program. Should an error occur when the write is attempted (for example, write-locked device), an explicit 'WRITE' command must be given to actually write out the block.

If the words within some blocks are changed accidentally, the <something-changed> flag can be reset by using the 'SET' command to reset the 'DEVICE' (described further along in this writeup) to the same device currently being used. This will reset the <something-changed> flag, the current block in memory, and the file start block and core-control-block/header-block (if they had been set by a 'FILE' command). The resetting of the current block in memory will cause the next access to the device to read in the block desired. The resetting of the file information will require a new 'FILE' command to be given to set it back up. If you can't remember what is the current setting of the 'DEVICE', use 'SHOW DEVICE' first and then 'SET' it the same.

Files stored on an OS/8 mass-storage device generally fall into one of four categories. The program has four corresponding modes for accessing the device. The current 'MODE' of the program can be set by the 'SET' command or by chaining (as described previously) and examined by the 'SHOW' command (to be described later).

The four categories and their corresponding modes are:

1. General (binary, ASCII and data) files – 'NORMAL' mode
2. Core image (save) files – 'SAVE' mode
3. FORTRAN IV load modules – 'LOAD' mode
4. System overlays – 'OFFSET' mode

The actual operation of the program for each of these modes is as follows:

'NORMAL' The high order 7 bits of the 15 bit address are added to the current block number to get the actual block number. The low 8 bits of the 15 bit address are used to specify the desired word within that block.

'SAVE' The file to be examined must be set up by a 'FILE' command. "Block" numbers are used to specify an overlay number (future MACREL/LINK support) and must be exactly zero ("0") for files without overlays (generated by the monitor "SAVE" command). The core segment data (pages and fields) from the file's CCB (core-control-block) is used to determine where on the device the desired word is to be found. This is done by first determining the correct block from the file's CCB and then using the low 8 bits of the address to specify the desired word within that block. Specifying a nonexistent address or overlay for one of the single-character (ODT) commands will cause an error. Specifying a nonexistent address or overlay for any of the word-type commands will cause the program to ignore the address and access no data.

'LOAD' The file to be examined must be set up by a 'FILE' command. Block number specifications are actually taken as FORTRAN IV overlay specifications and must be contained within the file. The information from the OIT (overlay-information-table) in the header block of the file is used to determine where on the device the desired word is to be found. Nonexistent addresses are handled the same way as for 'SAVE' mode.

NOTE

Because the "block" part of the location specification changes definition depending on the mode in use, it is recommended that the first operation following a switch to 'SAVE' or 'LOAD' mode explicitly specify a "block" part of 0. Otherwise a previously specified "block" part will be taken to mean a non-existent overlay number, causing an error.

'OFFSET' The 12-bit 'OFFSET' (which is set by the 'SET' command and examined by the 'SHOW' command) is subtracted from the low order 12 bits of the address and then the same arithmetic as with the 'NORMAL' mode is used. This mode is used mostly with system overlays whose start block number and actual loading address is known. By setting the 'OFFSET' to the loading address (which can only be a 12 bit number), the 12 bit "actual" addresses of the overlay can be used.

The 'SAVE' and 'LOAD' modes are mentioned together throughout this appendix as MAPPED modes because their method of address translation uses a descriptor block from the file of interest to control access to the file in a non-contiguous manner.

NOTE

For all access modes, the OS/8 "actual" block number for the block to be read is stored (for display) in the computer MQ register (if present). The value is stored before checking if the current block needs to be written. It is particularly useful for following the progress of the 'SCAN' command.

K.1.6 Referencing Words on the Device

The words on the OS/8 device are referenced by their <location> (often abbreviated as <1>). This <location> consists of an optional <block> or <overlay> number (which must be followed by a "." if present), and an <address> or <displacement>. The <block>/<overlay> number is a 12-bit number which must be in the range 0 thru 7776 (octal), or 4094 (decimal). Block number 7777 (or 4095, decimal) does not exist under OS/8, and the program will ignore this number. The <overlay> number is further limited to the number of overlays at a given address. Whenever the <block>/<overlay> part of the <location> is not used, the program will use the last specified value. The <address>/<displacement> is a 15 bit number (5 octal digits), but leading 0's need not be specified. Thus, the forms and their corresponding examples are as follows:

Form	Example
<block>.<displacement>	1201.37524
<overlay>.<address>	3.57633
<address>	15721
<displacement>	223

CAUTION

Neither this program nor the OS/8 device handlers generally include checking for legal block numbers! It is simply assumed that all accesses to the device will be done after checking with the directory for legal file start blocks and lengths, which is the normal mode of operation under OS/8. This can have very interesting results with this program; for example, the RK8/E handler, given a block number greater than 6257 (octal) on device RKA0, will simply continue on into device RKB0.

For the rest of this document, unless otherwise stated, block will mean <block> or <overlay> and address will mean <address> or <displacement>, depending on usage. Therefore the definition will be:

[block.] address=<location> = <1>

Since these location references are numeric input, all of the characteristics described next can also be used when specifying locations.

K.1.7 Numeric Items (Or Numbers)

Two "switches" are used by the program to allow the input of either octal, decimal or mixed numeric input where ever numeric input is used. Each new command line always resets the input mode to octal. The character CTRL/D (printed as "^D") switches the input mode for any following input to decimal. The character CTRL/K (printed as "^K") switches the input mode back to octal. These two switches may be located anywhere in numeric input.

For example, when inputting a string of numbers, the input would be alternately decimal and octal if it were

^D100,^K100,^D200,^K200,^D300,^K300

Two other characters, the double quote (") and apostrophe ('), may be used for numeric input. The double quote functions the same way in this program as it does in PAL8 in that the 8-bit ASCII value of the following character is used as a number. As with all character input, the special characters described earlier cannot be used. The apostrophe functions in a way similar to the way that the "TEXT" pseudo-op operates in PAL8 in that the following two characters are masked to 6-bits each and packed into a 12-bit word. There must always be exactly two characters following the single quote. If it is desired to pack one half of the word with a 6-bit 00, use the character "@". For example, a string equivalent to the file-name "PIP.SV" would be represented by the string

'PI,'P@,0,'SV

Expressions may also be used for numeric input when enclosed in parentheses. The parentheses pair "(and)" must surround the expression. When this is so, all the options of the 'EVAL' command are available for numeric input. For example, the contents of the switch register can be used for a number by the expression "(S)", or the current block number +5 could be used by the expression "(B+5)". See the discussion of the 'EVAL' command for the other options available.

NOTE

The opening and closing parentheses must completely surround the expression. Neither digits nor the switch characters may be outside of the parentheses or an error will result. This is required because many of the non-alphabetic characters have multiple meanings (commands or operators) so the use of the parentheses pair "(. . .)" provides the necessary context to remove ambiguity.

K.1.8 Errors (And Error Messages)

Whenever the program recognizes an error of some type, it outputs an error message to inform you what went wrong. The message tells both what went wrong and where in the command line the error was made. Depending on the setting of the 'ERROR' mode switch, either 'SHORT' or 'LONG' messages are output.

The error messages have the forms:

"?<ee> at <cc> <error message>" -- 'LONG'

or

"?<ee> at <cc>" – 'SHORT'

where <ee> is the error code, <cc> is the number of the column in the command line where the program stopped scanning and <error message> is the message itself. There are currently 45 error conditions with corresponding codes and messages to assist the user of this program. The error codes and their messages can be printed out by the 'SHOW' 'ERRORS' command. The 'ERROR' mode is set by the 'SET' command.

The error messages are swapped with the USR, but not in the normal manner, allowing write locked startup with the loss of the message text (see the section on program execution for more information).

K.2 SINGLE CHARACTER (ODT-LIKE) COMMANDS

These commands allow the examination and modification of words on an OS/8 device in the same way that ODT allows the examination and modification of the memory in the computer.

In all of the following commands where <n> – a numeric item – is specified, the operation of "closing" the location is to place the value of <n> into the word if it is open. If the current location is not open, or if <n> is not specified, no change takes place. Refer to the "Introduction to Programming" and the OS/8 Handbook section on ODT for more information if needed. Note that (as mentioned previously) "[<n>]" with the following commands means that a numeric item may be optionally supplied.

<1>/	Open and output the contents of location <1> in the current 'OUTPUT' mode.
/	Reopen the last location opened by one of these commands and output its contents in the current 'OUTPUT' mode.
[<n>] #	Close the current location, reopen it and output its contents in 'BCD' (3 digit binary-coded decimal).
[<n>] $ (dollar sign)	Close the current location, reopen it and output its contents in 'OS/8' ASCII.
[<n>] %	Close the current location, reopen it and output its contents in 'BYTE' octal (8 bits with OS/8 packing).
[<n>] &	Close the current location, reopen it and output its contents in 'XS240' format packed ASCII.
[<n>] :	Close the current location, reopen it and output its contents in 'SIGNED' decimal.
[<n>] <	Close the current location, reopen it and output its contents in 'OCTAL'.
[<n>] =	Close the current location, reopen it and output its contents in 'UNSIGNED' decimal.
[<n>] >	Close the current location, reopen it and output its contents in 'PDP' (symbolic).
[<n>] ?	Close the current location, reopen it and output its contents in 'DIRECTORY' format [negated DECIMAL, DATE (see "@" next) and PACKED (ASCII)].
[<n>] @	Close the current location, reopen it and output its contents in 'DATE' format: dd-mmm-yy 2 digits each for the day and year and 3 alphabetic characters for the month (except for illegal month numbers, which are output as a space and 2 decimal digits).
[<n>] [	Close the current location, reopen it and output its contents in 'ASCII'.

[<n>]\ Close the current location, reopen it and output its contents in 'FPP' (symbolic).

[<n>]] Close the current location, reopen it and output its contents in 'PACKED' ASCII.

[<n>]$ ("ALTMODE" or "ESCAPE" key) Close the current location, reopen it and type its contents as specified by the current 'FORMAT'.

[<n>]<cr> Close the current location.

[<n>]; Close the current location and open the next sequential location. Neither address nor contents are output, but one space is echoed.

NOTE

The ";" command can be used to advance through addresses without outputting their value in octal when some other format is really more helpful. For example, when examining a directory, the file name and extension can be output using the "]" command (PACKED ASCII), the date can be output using the "@" command and the file length can be output using the ":" command and all of this information can be made to appear on one line by simply using the ";" command to do the incrementing between each of the output commands. The result would look something like this:

2.5/ 2317]SO ;]UR ;]CE ;]PA ; @30-AUG-72 ; : - 0071

For the following commands, the location of the newly opened word is output before the contents are output. This location is composed of the 12-bit block number (4 octal digits), a "." for a separator, and the 15 bit address (5 octal digits). This is immediately followed by "/" to separate the contents from the address.

[<n>]<line feed> Close the current location, open and output the contents of the next sequential location in the current 'OUTPUT' mode.

[<n>]! Close the current location, open and output the contents of the previous sequential location in the current 'OUTPUT' mode.

[<n>]^ (circumflex or up-arrow) Close current location, open the location that would have been referenced if the contents were a PDP-8 memory reference instruction, and output the contents of the new location in the current 'OUTPUT' mode. Note: this command works like the stand-alone version of ODT, not like the OS/8 version. Even if bit 3 of the word (the indirect bit of a PDP-8 instruction) is a 1, this command will not do the equivalent of an indirect reference.

[<n>]_ (backarrow or underline) Close the current location, take its contents as an address, open that location and print its contents in the current 'OUTPUT' mode. This operates as an indirect address into the current field would. The field currently being examined (the high octal digit of the 5 digit location) will not be changed by this operation.

<1>+ Open the location <1> locations forward from the current location and output its contents in the current 'OUTPUT' mode. 15 bit arithmetic is used and the block part is ignored, so this will operate across field boundaries, i.e., within a 32K area.

<1>- Open the location <1> locations backward from the current location and output its contents in the current 'OUTPUT' mode. Same restrictions as with the '+' command.

The "current 'OUTPUT' mode" has been mentioned several times above. The program will output the contents of a location either as a four-digit octal number, or as a four-digit octal number with two spaces and the "symbolic" representation ('PDP' or 'FPP') of the word. See the 'SET' and 'SHOW' commands as well as the following section.

K.2.1 "Symbolic" Output Formats

The "symbolic" typeout is in approximately the format that input to an assembler would need to be in order to generate the contents of the current location. It is assumed, of course, that these contents are either a PDP-8 or an FPP-12/8A instruction, depending on the output selected. If the word to be output is not an instruction, as is the case for the second word of all 2-word instructions (EAE and FPP), the decoding will obviously be meaningless.

For PDP-8 instructions decoding into mnemonics is done for all memory reference instructions, for all legal operate instructions (including 8/E EAE instructions except for "SWAB"), for all 8/E processor, extended memory and memory parity IOTs, for teletype and high-speed paper-tape IOTs, for 8/E redundancy check option IOTs, for programmable real-time clock IOTs and for FPP IOTs. There are currently a total of 96 IOTs and space has been provided in the program for an additional 32 IOT codes and their mnemonics. These can be patched directly into the program using itself. The first word of each four-word entry is the exact IOT code (for example, 6221 for "CDF 20"), followed by 3 words containing up to 6 packed ASCII characters padded with trailing 0's. No attempt is made to decode any micro-coded IOTs. Either an exact match for the current contents will be found in the table or the program will output "IOT nnnn where nnnn is the octal typeout of the low 9 bits of the code. The next free location in the table (which is in field 1) is pointed to by the contents of location 10000. The table is terminated by the first 0 for an IOT code, so additions must be contiguous and added directly at the current end of the table.

For FPP instructions, the full FPP-8/A instruction set is decoded except for "IMUL", which is actually an integer mode "LEA". For the data manipulation instructions, the op-code mnemonic is followed by a "#" for the long-indexed format, by a "%" for the indirect-indexed format and by a space for the base addressing format. For the indirect-indexed and base addressing formats, the operand address is output as "B+nnn", where nnn is the 3 digit octal value of the displacement (3 or 7 bits) multiplied by 3. These formats are those used by the RALF assembler. This is also true for "LEA" instructions (i.e., "LEAI" is decoded as "LEA%"). Both jump and load-truth instruction decoding is done as a single mnemonic whose last two characters indicate the specified condition. All instructions which use 2 words are decoded with an "*" in the location in the normal assembler format where the value of the second word would go. Index register number and "+" for auto-increment (if used) are also shown in the assembler format. Any combinations which are not in the FPP-8/A instruction definitions are output as "UNUSED".

NOTE

For both of these output formats, the use of the mapped access modes (and the 'OFFSET' mode for PDP decoding) allow the use of the "actual" addresses when decoding the instruction.

K.3 WORD-TYPE COMMANDS

These commands are grouped by function, as follows:

Group 1:

DUMP	type/list out the contents of one or more blocks.
LIST	type/list out the contents of one or more locations.
MODIFY	modify one or more locations.

Group 2:

WORD	word search
STRING	string search
SMASK	set up string search mask

Group 3:

SET	set up program switches & variables
SHOW	show settings of program switches & variables
FILE	look up file(s) on device
WRITE	write out current buffer
SCAN	scan for bad blocks
REWIND	move device to block 1 & reset directory segment

Group 4:

OPEN	open an output file on a file-structured device
CLOSE	close the open output file

Group 5:

IF	cause command skipping based on expression value
END	resume command execution after unsatisfied 'IF'
COMMENT	pass user commentary to output device
EXIT	exit to OS/8 (same as CTRL/C)

Group 6:

EVAL	evaluate a signed, double-precision expression.

Command words may always be abbreviated to their first two characters, as with the Monitor and BUILD, and some of the commands and their options may also be abbreviated to only one letter. When this is true, the command forms given will include the one-letter form, and the option forms will give the one-letter form directly under the full word form.

NOTE

In many cases, two or more words start with the same letter. In these cases, only one of these words may be abbreviated to one letter.

The descriptions for each command include each of the possible forms of the command, with an example of that form following it on the same line.

K.3.1 Output Formats

The 'FORMAT' option is used to 'SET' up the output format for the "$" ('ALTMODE' or 'ESCAPE') command (single-character) described earlier and the default format for the 'DUMP', 'LIST' and 'MODIFY' commands described below. The syntax of this command is shown with the other 'SET' commands but is described here to make the descriptions of the following three commands more understandable. The <format> may be one of the following:

ASCII A	output each word as a single ASCII character.
PACKED P	Output each word as two 6-bit trimmed and packed ASCII characters. This is the format of PAL8 TEXT strings.
OS	Output each word as 1 or 2 OS/8 packed ASCII characters. The even address words output 1 character and the odd address words output 2 characters.

XS240	Output each word as two 6-bit packed ASCII characters by adding a space (240 octal) to the contents of each 6-bit byte. This is the format of PAL12 SIXBIT strings.
BYTE	Output each word as 1 or 2 OS/8 packed bytes of 8 bits each as a 3-digit octal numbers. The even address words output 1 number and the odd address words output 2 numbers.
UNSIGNED U	Output each word as an unsigned decimal number.
SIGNED S	Output each word as a signed decimal number.
OCTAL O	Output each word as a 4 digit octal number.
BCD B	Output each word as 3 bcd digits. The digits 0 through 9 are followed by ":" (10), ";" (11), "<" (12), "=" (13), ">" (14) and "?" (15).
PDP FPP	Output each word as an octal number, followed by 2 spaces and its mnemonic representation, assuming it to be a PDP-8 or an FPP-8A instruction. See the "symbolic" output description.
DIRECTORY	Output each word in octal, decimal (signed), date (see "@" command) and packed ASCII formats.

The 'FORMAT' is initialized to 'PACKED' ASCII.

The output from the 'DUMP' and 'LIST' commands for each of these formats is set up as follows:

1. At the beginning of each line the current location is output in <location> format with a 4 digit block number and a 5 digit address, both in octal, as

   <block>.<address>:

   For example, "1271.17205: "--location 17205(8) relative to block 1271(8).
2. The maximum number of words per line is set up as follows:

   A. The four character formats output 16 words per line with no extra characters.
   B. The five numeric formats output 8 words per line with 2 spaces between each number.
   C. The "symbolic" and directory formats output 1 word per line.

For 'LIST' with A or B, the first line may be shorter than succeeding lines to force the second and following address outputs to be even multiples of 10 (octal).

K.3.1.1 DUMP – The 'DUMP' command is used to output one or more whole 256 word device blocks in the default or an optionally supplied format. This command has the following forms:

DUMP [<format>] <block string>.

DUMP <block string>	DU 100,200-213,250
D <block string>	D (B)-(B+10),(S)
DUMP <format> <block string>	DU PA 212
D <format> <block string>	D OS 514

where the optional <format> is one of those given for the 'FORMAT' option above, and the <block string> is one or more numeric items separated by ","s and "–"s. The "–" is used when it is desired to dump a group of blocks, and is used as

<start block> - <end block>

the "," is used to separate single blocks or groups of blocks if there is more than one per line.

NOTE

When in a mapped ('SAVE' or 'LOAD') mode, the 'DUMP' command cannot dump any block except the block containing location 0. To eliminate the confusion that this would produce, the command will simply output an error message reminding the user that the proper command to use in a mapped mode is the 'LIST' command.

The output from the 'DUMP' command is sent to the 'DDEV' ("dump" device), which can be either the console terminal, the line printer, or a file. See the 'SET' command for setting the "dump" device and output mode.

K.3.1.2 **LIST** – The 'LIST' command is used to output the contents of one or more words on the device in the default or an optionally supplied format. This command has the following forms:

LIST [<format>] <location string>

LIST <location string>	LI 123.200-517,200.0
L <location string>	L 312.10-17,100-117,176
LIST <format> <location string>	LI UN 200-227
L <format> <location string>	L SI 200-277

where the optional <format> is one of those given for the "FORMAT" option above, and the <location string> is one or more <location>s, separated by ","s. When it is desired to list a group of words, the "-" is used to separate the start and end addresses as

[<block>.]<start address>[-<end address>]

If the block part is not specified, the last block number specified to the program will be used. If an end address is specified, the start address is assumed to be in the same field as the end address (i.e. the highest octal digit of the 5-digit address), so a maximum of 4096 words can be specified by each group.

As with the 'DUMP' command, the output from the 'LIST' command is sent to the 'DDEV'. For more information see the last paragraph of the 'DUMP' command, the 'SET' command, and the miscellaneous information section.

K.3.1.3 **MODIFY** – The 'MODIFY' command allows a string of locations on the device to be changed in an easy way. This is done by specifying the format of the input and letting the program do the work of storing the data properly. This command has the following forms:

MODIFY [<format>] <location string>

MODIFY <location string>	MO 200.0-17,35-43
M <location string>	M 32745-32777
MODIFY <format> <location string>	MO PA 12342-12360
M <format> <location string>	M AS 367.7261-7275

where the <location string> has exactly the same format as for the 'LIST' command and the <format> options are shown below. If the <format> is not specified (as with the first form), the program will pick the one of the formats below which corresponds to the current setting of the 'FORMAT' option. The corresponding formats are shown below.

'MODIFY' format	'FORMAT' setting and 'MODIFY' action.
ASCII A	ASCII – one character of input is stored in each word to be modified.
PACKED P	PACKED – two characters of input are packed as trimmed 6-bit characters, padded with trailing 00's. Control characters (those with codes less than 240 octal) are packed as a 6-bit 77 (flag) and the low-order 6-bits of the character. Note that this means that "@" is packed as a terminator (00) and that "?" is not unique.
OS	OS – three characters of input are packed into two words to be modified. When using this format, the start address must be even and the end address must be odd.
XS240	XS240 – a space (240 octal) is subtracted from each character and then it is packed as 6-bit bytes. Control characters are handled as with 'PACKED' format.
NUMERIC N	SIGNED & UNSIGNED decimal, BCD, OCTAL, BYTE, PDP, FPP and DIRECTORY formats – the input is a string of numeric items which are stored one per 12 bit word. See the section on numeric items. Note that bcd, byte, directory and "symbolic" are not included, that decimal or octal input are determined by the "CTRL"-"D" and "CTRL"-"K" switches and that signed numbers must be input enclosed in parentheses, e.g., 17, (-10), ^D200, (-^K312), 40, (-^D35*129).

For each location or group of locations specified by the <location string>, the program will prompt for the input by printing the start location in the same format as described under the output format options above.

CAUTION

The program always modifies exactly the number of words specified by each item in the <location string>. If you input extra characters for the character formats or extra numeric items for the numeric format, they will be ignored. If you input not enough characters or items, the rest of the words to be modified will be set to the 'FILLER' value (see the 'SET' command). The program will not output any message if either of these things takes place. This does, however, make it possible to fill from 1 to 16 blocks on a device with zero or some other value by specifying all the words to be filled in 'NUMERIC' format and then responding to the prompt with a single "(F)" (the value of the 'FILLER') and "RETURN".

Input to the program is always terminated by a carriage-return ("RETURN"). It is therefore not possible to insert a carriage-return into a word using this command. All of the editing keys are available for use during input, therefore the CTRL/C, CTRL/Q, CTRL/S, CTRL/R, CTRL/P, CTRL/U and "RUBOUT" characters cannot be entered using this command either. For all of the character input formats, spaces (excluding leading spaces, which are ignored) and tabs in the input string are packed as they are seen. For numeric input, spaces are ignored and the numeric items must be separated by commas.

The command can always be aborted by CTRL/P if you change your mind before the "RETURN" key is pressed.

K.3.2 Search Limits:

There are two search commands in the program, the 'WORD' search and the 'STRING' search. They both search from a lower to an upper limit. The limits are either the 'LOWER' and 'UPPER' limits set by the 'SET' command (the default) or the limits set up by the " 'FROM' <1>" (which overrides the 'LOWER' limit) and/or " 'TO' <1>" (which overrides the 'UPPER' limit) clauses which can optionally follow the command word. Leaving out the block parts of either of the two temporary limits will cause the program to use the block part of the corresponding default limit set by the 'SET' command. When in a mapped ('SAVE' or 'LOAD') access mode, searching through non-existent locations or overlays will never produce a match. Whenever a match is found, the program outputs the location where the match occurred, followed by the word or string that matched.

NOTE

It is not possible to search through more than one overlay per search command. To do so would require different and separate handling of the "block" and "address" parts of the limits when in the mapped modes including the resetting of the "address" part. The result is that in the mapped modes the "block" parts are used to set the overlay to be searched (lower limit only) and only the "address" parts are used in the determination of the number of words to be searched.

K.3.2.1 WORD (Search) – The 'WORD' search command is used to search for a word for words which, masked by the 'MASK' (which is set by the 'SET' command), will match the search word (also masked). This command has five options and therefore has the forms:

WORD [UNEQ] [ABS] [MEM] [FROM <1>] [TO <1>] <n>

WORD <n>	WO 217
W <n>	W (S)
WORD UNEQUAL <n>	W UN 0
WO U <n>	WO U (C&377)
WORD ABSOLUTE <n>	WO AB 7402
W A <n>	W A 7000
WORD MEMREF <n>	WOR MEM 41
WO M <n>	WO M 40
WORD FROM <1> <n>	WO FR 213.0 2317
W F <1> <n>	W F 1.35 (S)
WORD TO <1> <n>	W TO 213.345 1111
W T <1> <n>	WORD T 6257.377 7777

. . . and any combination and order of the above options.

where <n> is the bit pattern being searched for, 'UNEQUAL' means that all words which are not equal to <n> under the mask do match, the temporary limits clause is as described above, 'ABSOLUTE' means that the location where the match occurred is to be output as an absolute block number and displacement rather than as a relative location, and 'MEMREF' means that only words whose high-order octal digit is 0 thru 5 (i.e. the PDP-8 memory reference op-codes) are allowed to match, independent of the setting of the 'MASK'.

When you want to search for those words which reference a specific location, 'SET' the 'MASK' to 377 (octal) and then use the 'MEMREF' option. This will exclude all Operate (op-code 7) and IOT (op-code 6) "instructions" from the output and can make it considerably easier to find the desired information (e.g. you will not output the location of every "CIA", 7041 octal, when you are looking for references to location 41 octal).

NOTE

'UNEQUAL' has a higher priority than 'MEMREF', so first each word is tested under the mask for equal/'UNEQUAL' and if the specified condition is true, then the word is tested for the 'MEMREF' condition.

K.3.2.2 STRING (Search) – The 'STRING' search command is used to search for a string of numbers (bit patterns) under an optional string mask. This command has four options and therefore has the forms:

STRING [MASKED] [ABS] [FROM <1>] [TO <1>] <numeric string>

STRING <numeric string>	ST 4557,0,0
STRING MASKED <numeric string>	ST MA 4577,0,1203
ST M <numeric string>	ST M 5566,0
STRING ABSOLUTE <numeric string>	ST AB 'PI,'P@
ST A <numeric string>	ST A "A, "B
STRING FROM <1><numeric string>	STR FR 100 1,4000,2
STR F <1> <numeric string>	ST F 123.4567 (S),(-S)
STRING TO <1> <numeric string>	STR T 7577 'ER, 'RO, 'R@
ST F <1> T <1> <numeric string>	ST F 1.0 T 7.0 'FO,'TP

. . . and any combination and order of the above options.

where the <numeric string> is simply a string of numeric items separated by commas, 'MASKED' specifies that the search is to be done under the string mask, 'ABSOLUTE' is as for the 'WORD' search, and the temporary limits clause is as described above.

When the 'MASKED' option is used, each item of the <numeric string> is masked by a separate mask word from the string mask. If the string mask is shorter than the search string, it is used in a circular fashion (the first word follows the last) as many times as necessary to mask all of the items of the search string. If the string mask is longer than the search string, the extra words are not used. This feature allows for very complex searches to be done.

For example: Suppose it is desired to find all calls to a certain subroutine in a file and also see their arguments. This could be done as follows:

FILE FUTIL	– look up file to be searched
FUTIL.SV 6070-6120 ^P	– you stop typeout
SE MODE SAVE	– set access mode to mapped
SMASK (-1),0,0	– set mask for 2 arguments per call
ST M 4547,0,0	– search for 4547 and 2 dummies

The output will give the address of the subroutine call (which requires an exact match due to the mask of 7777) and the contents of the two following words (which can be anything, since they are masked by 0).

Using the mask specified above, a search could be made for an exact match, 2 "don't care words" and another exact match by simply specifying a search string with 4 arguments. The first item of the string mask will be used to mask both the first and the last items of the search string.

This command can be particularly useful when trying to find certain kinds of references in programs for which no CREF listing (or perhaps no listing at all) is available.

K.3.2.3 SMASK – The 'SMASK' command is used to set up the string mask. It has the following form:

SMASK <numeric string>	SM (-1),0,0,7000,0

where the <numeric string> is the same as for the 'STRING' search command above. The current contents of the string mask may be examined by the 'SHOW' command.

K.3.2.4 SET – The 'SET' command is used to set up various switches and variables within the program. It has many options, each of which is the name of the switch or variable and is always followed by a word or number describing how it is to be set. All items are separated by spaces. The command has the following two forms:

SET <option(s)>	SE OU PDP ERR LONG MODE SAV
S <option(s)>	S LO 100.0 UP 123.377 1DEV LPT

where the options are as follows:

OUTPUT OCTAL OUTPUT O O PDP O P OUT FPP O F	Set the output mode for the single-character commands. Initialized to 'OCTAL'.
ERROR SHORT E S E LONG ERROR L	Set the mode for error message output. The 'SHOW' 'ERRORS' command will list all error messages. Initialized to 'LONG'. Also set to 'SHORT' by write-locking system device.
FORMAT <format>	Set output format for 'LIST', 'DUMP', etc. The formats have been described previously. Initialized to 'PACKED' ASCII.
OFFSET <1>	Set the offset to the low 12 bits of <1>. Initialized to 0.
FILLER <n>	Set the filler to the low 12 bits of <n>. Initialized to 0.
LOWER <1>	Set the lower search limit. Initialized to 0.200.
UPPER <1>	Set the upper search limit. Initialized to 0.17577.
DEVICE <device name[:]>	Set up the OS/8 device for access. The handler is fetched at this time. Initialized to "SYS" (device 01). ":" In <device name[:]> is optional. <device name> is an assigned or permanent OS/8 mass storage device name.
DDEV <device name[:]>	Set up the "dump" device. Initialized to 'SYS'. See also 'DMODE' below and 'OPEN' and 'CLOSE' commands.
MODE NORMAL MODE N MODE SAVE MODE S MO LOAD MO L MO OFFSET MO O	Set up the device access mode. These have been described previously. Initialized to 'NORMAL'.
DMODE NONE DMODE PART DMODE ALL	Set up the "dump" output mode. Initialized to "NONE", which sends all output to console only. 'PART' sends 'DUMP', 'LIST' and 'SHOW ERRORS' output to the 'DDEV' (perhaps to a file). 'ALL' sends all output to both the console device and to the 'DDEV'. (See section on file output.)

MASK <n> M <n>	Set the 'WORD' search mask to the low 12 bits of <n>. Initialized to 7777.
TEMP <n>	Set the 'TEMP' storage to the 24-bit value of <n>. Value is returned by subsequent use of the 'T' in expressions.

As many options as desired may be specified on one command line, separated by spaces. In the event of an error, none of the options past the point where the error occurred will have been set. If you have any question, use the 'SHOW' command.

K.3.2.5 SHOW – The 'SHOW' command is used to list the current setting of any of the program switches and variables set by the 'SET' command and other information. The program outputs either words or numbers to best describe the current settings. As with the 'SET' command, as many of the options for this command as desired may be specified on a single command line, separated by spaces. This command has the form:

SHOW <option(s)>	SH BL CCB LOW UP ODT REL ABS

where the <options> are as follows:

BLOCK B	Output in octal the start block number of the last file specified by the last 'FILE' command.
CCB C	Output the core control block of the last file specified by the 'FILE' command. If the file is not a 'SAVE' file, an error will occur. The start address of the file is output as a 5-digit octal number, the job status word (JSW) is output in octal, and the core segments are output as 5-digit octal addresses.
HEADER H	Output the header block information for the last file specified by the last 'FILE' command. If the file is not a 'LOAD' file, an error will occur. The start address is output as a 5-digit octal number, followed by the next free address as a 5-digit octal number, the loader version number in octal and a message if Extended Precision is required. Then, for each level, a line is output with the number of overlays, the 5-digit start address, the relative start block and the length of the overlays (in blocks) for this level.
ABSOLUTE A	Output the absolute location of the last word accessed on the device in <location> format (a 4 digit octal block number, a "." and a 5-digit octal address) and the word "MOD" if the current block has been changed (the <something-changed> flag is set).
RELATIVE R	Output the relative location (what you specified) of the last word accessed on the device in <1> format and the word "MOD" if the current block has been changed.
ODT	Output the relative location of the last word accessed by one of the special-character commands in <1> format and the word "MOD" if the current block has been changed . .
LOWER	Output the search lower limit in <1> format.

UPPER	Output the search upper limit in <1> format.
FILLER	Output the value of the filler in octal.
MASK M	Output the 'WORD' search mask in octal.
SMASK	Output the current contents of the 'STRING' search mask as a string of octal numbers.
OFFSET	Output the value of the offset in octal.
MODE	Output the name of the current setting of the device access mode switch ('NORMAL', 'SAVE', 'LOAD' or 'OFFSET').
DEVICE	Output the OS/8 device name and number.
DDEV	Output the name of the "dump" device.
OUTPUT O	Output the name of the current single-character (ODT) command 'OUTPUT' mode (OCTAL, PDP or FPP).
FORMAT F	Output the name of the current output format.
VERSION	Output the current version number of FUTIL.
ERRORS E	Output a complete list of all error codes and their corresponding messages. Note: This list is output to the 'DDEV' ("dump" device) so that it can be output using the "LPT" handler for your system. Note that Version number is also output with errors.

K.3.2.6 FILE – The 'FILE' command is used to locate files on the OS/8 device and to set up the start block of a file for the mapped access modes, 'SHOW CCB', etc. This command has the forms:

FILE <file name string>	FI FUTIL PIP.SV
F <file name string>	F MICRO.LD

where the <file name string> is a string of one or more OS/8 file names, separated by spaces. Any other characters except "." will be taken as part of the file names. The program assumes extensions of ".SV", ".LD" and null (in this order) when looking up the file. This can lead to a substantial amount of time when a large directory is searched three times for a file that does not exist. Specifying an extension will cause only one lookup attempt to be made. A null extension, if desired, may be specified by making the "." the last character of the file name. The program does one (or more) separate lookup(s) for each file name specified and outputs either

<file name> ssss-eeee oooo (dddd) b.111 dd-mmm-yr

or

<file name> ssss-eeee oooo (dddd) b.111

or

<file name> LOOKUP FAILED

where "ssss" is the start block of the file in octal, "eeee" is the last block of the file in octal, "oooo" is the length of the file in octal, "dddd" is the length of the file in decimal, "b.111" is the block (segment) and location within that block of the first word of the file entry (the first two characters of the name) in the directory, and dd-mmm-yy

is the file date. If the directory does not contain the extra word required for the date or the date word of the file is 0, the second form with no date will be output rather than the first form. The "LOOKUP FAILED" message means either that the file name was not found on the device or that the device is a write-only device.

The actual lookup operation is performed by the OS/8 USR, which is swapped as needed (see section on program execution). Since the USR keeps track of the current device once the first 'FILE' command is given, it will have the wrong directory in memory if the medium (tape or disk) is changed on the physical device. This can be solved one of three ways:

1. Use the 'REWIND' command to rewind the device being removed and clear the directory segment from the USR.
2. Do a 'SHOW ERRORS' and abort the output when the message output begins. This will have swapped out the USR. If messages are not available, use 1 or 3.
3. Use EXIT or CTRL/C to return to OS/8 and then directly restart FUTIL with the OS/8 START command. This will have swapped out both error messages and USR from memory.

Any of these methods should be followed by a 'SET' command to reset the 'DEVICE' and the rest of the I/O parameters desired.

The last file name specified that did not have a LOOKUP FAIL will be the file used in the mapped access modes, 'SHOW' 'CCB', etc. The program is initialized with no known file, so attempting to access any location in a mapped access mode or attempting to 'SHOW' 'CCB' or 'SHOW' 'HEADER' without giving a valid 'FILE' command will cause an error.

K.3.2.7 WRITE – The 'WRITE' command is used to force the program to write out the block currently in memory. It has the form:

WRITE [<block>]

where the optional <block> overrides the default number of the block that was read to specify where the current block is to be written. This obviously dangerous operation does allow a limited amount of copying in a special situation, e.g., allowing a directory to be backed up by moving a copy to the end of the device (see the examples section) or copying a single block from one device to another by changing the 'DEVICE' and then doing a 'WRITE' (with or without an argument). Again, as stated in the section on accessing the device, caution must be used because attempting to write beyond the end of a device may not be checked by the handler.

K.3.2.8 SCAN – The 'SCAN' command is used to do a rapid scan for read errors on the current 'DEVICE'. It has the form:

SCAN <block string> SC 0-6257

where the <block string> is of the same form as for the 'DUMP' command. Each block is simply read. If an error occurs, it is reported as:

oooo BAD BLOCK

where "oooo" is the block number in octal, and the scan continues. This is the only FUTIL command that will continue on a read error. Should the current block have been changed, and any other blocks be included in the scan, an implicit write will be attempted by FUTIL. An error on this implicit write will be reported and then the command will be aborted. This is the only time that this command will attempt a write. The command can then be repeated if it is desired and it will execute (only one implicit write attempt is ever made by FUTIL).

NOTE

The OS/8 "actual" block number for the block to be read is stored (for display) in the computer MQ register (if present). It is particularly useful for following the progress of this command. The value is stored before checking if the current block needs to be written.

K.3.2.9 REWIND – The 'REWIND' command is used to move a tape back to block 1 and to reset the USR directory segment. It has the form:

REWIND

and must be terminated by the "RETURN" key. It causes a read of block 1 of the device and resets the directory segment in the USR (if in memory). Any subsequent 'FILE' command will cause the directory to be read.

K.3.3 File Output

Output to file-structured or non-file-structured "dump" devices is provided through two commands, 'OPEN' and 'CLOSE', and two 'SET' options, 'DDEV' and 'DMODE'. They can be used to simply make fast hard copy output from the 'DUMP', 'LIST' and 'SHOW ERRORS' commands, to provide a hard copy log of all operations carried out with a video terminal, to provide an ASCII file output of some data for later processing by another program, etc.

Output to file-structured and to non-file-structured devices (serial devices) is handled in two separate ways. Output to the file-structured device is done by first setting the 'DDEV' and 'DMODE' and then 'OPEN'ing an output file. No output to the device will be done until the file is open (to protect your directories), and then output will be done one block at a time. When output to the file is complete, 'CLOSE' your file to make it a permanent file (properly terminated with a CTRL-Z and padded with nulls).

Output to a non-file-structured device is done by simply setting the 'DDEV' and 'DMODE'. Output to the device will be done one line at a time, as soon as specified by the 'DMODE', and neither the 'OPEN' nor the 'CLOSE' commands are needed. The output is done by padding the buffer with nulls after each line is ready and then calling the output device handler, so the handler used should ignore nulls (which leaves out the "PTR:" handler, for example).

K.3.3.1 OPEN – The 'OPEN' command is used to open an output file on file-structured devices for partial or total output from the program. It has the form:

OPEN <file name> OPEN OUT.DA

where the <file name> should be a standard OS/8 file name. The extension defaults to ".DU" (for "dump") if none is supplied.

WARNING

FUTIL gives significance only to the characters space, carriage-return and "." when scanning file names. It is therefore the responsibility of the user not to include characters that are not legal to other OS/8 programs or the files will be able to be accessed only through FUTIL or the CCL command decoder.

This command must be given after the "dump" device is 'SET' by the 'DDEV' option. The output specified by the 'DMODE' will then be sent to this file, one block at a time (packed only 8 bits per word), until either the 'DMODE' is changed or the file is 'CLOSE'd.

Files can be opened at will without closing any previous file. This gives the user additional flexibility, but at the expense of possibly losing an output file if it is not 'CLOSE'd.

Should an error occur on the output device while doing output, the file is simply thrown away (it cannot be 'CLOSE'd).

K.3.3.2 CLOSE – The 'CLOSE' command is used to close an output file previously 'OPEN'ed. It has the form:

CLOSE

and must be on a line by itself. If given with no file open, it is simply ignored.

K.3.4 Batch Operation

Operation of FUTIL under BATCH allows repeated operations to be done without re-entry. All of the operations provided under interactive operation are provided except that the "RUBOUT" character is simply ignored, input is taken directly from the BATCH stream and console output goes to the log output device.

Four commands have been added specifically to support use of FUTIL under BATCH: 'IF', 'END', 'COMMENT' and 'EXIT'. These commands are also available for interactive use, but are not as important in that mode.

K.3.4.1 IF – The 'IF' command was implemented specifically to allow FUTIL, when operating under BATCH to be sure that the correct operations are proceeding before modifying something incorrectly. It has the form:

IF <expression> IF C-3575 .

where <expression> is a general expression of the same form as used by the 'EVAL' command. If the expression evaluates to exactly zero (as a 24-bit integer), command execution will continue as though the command had not been seen. If the result is not exactly zero, command skipping will begin and will continue until a line containing the single word 'END' is found. Command execution will then resume.

This command was set up to test only for zero under the assumption that a test is to be made for some exact quantity. However, the capabilities of the expression evaluator can be used to generate sufficiently complex expressions for other tests. For example:

IF 40000000&(. . . . .)	will test for positive
IF -(40000000&(. . .))-1	will test for negative
IF 10000&(-(77770000!(. . .)))	will test for 12-bit non-zero

K.3.4.2 END – The 'END' command is used to re-enable command execution following an unsatisfied 'IF' command. It has the form:

END

and must be on a single line by itself. When encountered during command execution, it is ignored. Note that the 'IF'/'END' commands cannot be nested because the first 'END' found will re-enable command execution for any number of previous 'IF' commands. For example:

IF . . .
IF . . .
IF . . .
END will terminate all three!

K.3.4.3 COMMENT – The 'COMMENT' command allows optional comments in command input which will simply be ignored during execution. It has the forms:

COMMENT [<comment>]	COMMENT THIS IS ONE
C [<comment>]	C

where [<comment>] is an optional comment. Note that blank lines may also be used for formatting of the output log but that they will also close any open location.

K.3.4.4 EXIT – The 'EXIT' command provides a method of return to OS/8 other than "CTRL/C". It has the form:

EXIT

and the rest of the line is ignored. Exit does not write out the last block modified. Use 'WRITE' to make changes permanent.

K.3.4.5 EVAL – The 'EVAL' command is used to evaluate a parenthesized expression of signed double-precision integers. It has the forms:

EVAL <expression>	EV S*^D4096+D
E <expression>	E B*400+L

where the <expression> follows the normal rules for arithmetic expressions. Legal operators, in their order of precedence are:

(	evaluate inner expression
/	signed division
*	signed multiplication
-	subtraction
+	addition
&	logical product ("and")
!	logical sum ("or")
)	expression end

Besides 24 bit numeric input (which can be octal, decimal or mixed octal and decimal under the control of the CTRL/D and CTRL/K switches and ASCII and packed ASCII using " and ', the following "variables" may be used:

C	curent contents (of location "L").
L	current location (15 bit, same value as is output by the 'SHOW' 'RELATIVE' command).
B	current block number (as for "L").
F	contents of 'FILLER' (12 bits).
T	contents of 'TEMP' (24 bits).
S	contents of the console switch register.
R	the remainder of the last division or the high product of the last multiplication. 24 bits, the sign may not be correct.
D	contents of OS/8 Monitor "date" word.

Overflow on addition, subtraction and multiplication are ignored, but trying to divide by 0 will cause an error.

If no errors occur, the program evaluates the expression and types out the results in the form:

= oooooooo (sddddddd)

where "oooooooo" is the double precision result in octal and "sddddddd" is the signed double precision result in decimal (the sign is either "-" or " ").

K.4 EXAMPLES

These examples are to help provide an overview of the use of the program and to stimulate the thoughts of the user. Example 3 and those that follow are not as well commented as the first two examples since it is intended to show concepts of what can be done with the program rather than the mechanics of the operations. Should questions arise on the mechanics, it is suggested that the first two examples and the discussions of the commands in question be reviewed.

EXAMPLE 1:

Assume that you would like to know what CCL remembers of your last ".UA" command. The remembrances are stored on block 65 (octal) of the system device. As described in the source of CCL, each of the remembrances is allocated 40 (octal), or 32 (decimal) words in this block, the first four of which contain binary information and the last 34 of which contain the last input command, stored as packed ASCII characters. The lines contain the inputs for the commands as follows: TECO and MAKE (line 0), EDIT and CREATE (line 1), COMPILE and EXECUTE and PAL (line 2), UA (line 3), UB (line 4), and UC (line 5). Thus, the saved ".UA" command can be listed by outputting the contents of the 4th through 37th words of area 3 in block 65 as packed ASCII characters as follows:

```
.R FUTIL <cr>                    – call FUTIL from OS/8

EVA 3*40+4<cr>                   – calculate start displacement
= 00000144 ( 0000100)            – of the 3rd "line" (=144[8])
```

Now list the words of this line with the LIST command, specifying the output format to be PACKED ASCII characters and the words to list to be block 65 locations 144 (from above) through 144+33 (the expression for the location of the last word of this "line"). FUTIL responds with the start location and a line of characters, and the next location with a multiple of 10[8] as an address and a line of characters.

```
LIST PACKED 65.144-(144+33)<cr>  – list the words wanted

0065.00144:  DIR R:FUT???.*/E/R=3
0065.00160:                      – that's it!
```

NOTE

For the examples above and below, the symbol "<cr>" is used to show that you need to terminate your command lines with a "carriage return". All other lines above are output by the program.

EXAMPLE 2:

Now assume that you would like to make the simple patch for OS/8 FORTRAN IV users with an FPP-8/A to use the lockout feature of the FPP-8/A, as given in the August 1976 DIGITAL Software News. This requires changing the contents of location 15776 of FRTS (the Fortran Run Time System) from 400 to 410 (which adds the lockout bit). You also want to update the date word of the directory entry for FRTS (the 4th word beyond the start of the entry) to show that the file has been updated. This is done as follows:

```
.R FUTIL<cr>                     – call it
```

```
SET MODE SAVE <cr>                    – set FUTIL to a mapped mode
FILE FRTS<cr>                         – look up the file to map
FRTS.SV 0671-0722 0032 (0026) 1.327 31-DEC-75
                                      – "1.327" is start of entry!
```

Now use "ODT" command "/" to open and change one word.

```
15776/ 0400 410<cr>                   – add LOCKOUT bit

SET MODE NORMAL<cr>                   – switch to unmapped
```

Now use "ODT" command "/" with an expression to open the date word, command "@" to output it in "date" format and then put today's date (as an octal value) in its place.

```
1.(327+4)/6375
@31-DEC-75 (D)<cr>                    – change file date to today's date

WRITE<cr>                             – send out this change
```

NOTE

First the file FRTS.SV is changed, and then the OS/8 directory is updated to the current date. Changing the address desired from FRTS to the directory automatically writes out the modified block of FRTS before reading in the directory segment that contains the file name. However, the changed directory segment must be written out explicitly because there are no other blocks to examine for this example.

EXAMPLE 3:

While doing a "/S" transfer with PIP, PIP gives a read error in your file "SOURCE.PA". Attempting to read it with EDIT causes EDIT to type "?0^C" and return to the Monitor. Find out what is wrong as follows:

```
.R FUTIL

FI SOURCE.PA                          – look up the file
SOURCE.PA 0243-0351 0107 (0071) 2.005 30-AUG-74

SE MASK 0 LO 243.0 UP 351.377         – set up mask & limits

W UNE 0                               – search the file

?ee AT 08 FATAL READ ERROR            – here is the problem
[Note: "ee" may change with version, so is left out.]
SH ABS                                – find out where it is
ABS. LOC = 0271.00000

WR                                    – attempt to clear error

DU OS (B+L/400)                       – it worked, now dump it

0271.00000: . . . .^P                 – change your mind

W UN FR 272.0 0                       – check the rest of the file

^C                                    – ok, now go fix the source
```

This sequence can also be carried out using the SCAN command as follows:

```
.R FUTIL
F1 SOURCE.PA                          – use CCL to call & lookup

SOURCE.PA 0243-0351 0107 (0071) 2.005 30-AUG-74
SCAN 243-351                          – scan the area

0271 BAD BLOCK                        – here is the problem!

271.0/ ?ee AT 07 FATAL READ ERROR     – get block with trouble

WR                                    – attempt to clear error

DU OS (B+L/400)                       – it worked, now dump it

0271.00000: . . . .^P                 – change your mind

^C                                    – ok, now go fix the source
```

If the error had been of some type other than a clearable error, the 'WR' command might also have failed.

EXAMPLE 4:

After using BUILD to change your system, find out the device number for "DTA1":

```
.R FUTIL

SE DEV DTA1                           – fetch the device handler
SHOW DEV
DEVICE = DTA1 (06)                    – number is decimal
```

EXAMPLE 5:

By accident you zero a DECtape directory which contains the only copy of a file you need. You have the PIP "/E" listing of the directory but only want to re-build it enough to get the wanted file. The name of the file is "LOST.FI":

```
.R FUTIL

SE DEV DTA1                           – it was here
EV ^D5+14+11+10+16+13+8+5             – lengths of all preceding
= 00000122 ( 0000082)                 –  files
EV ^D730-^K61-^D82-25                 – rest of DECtape room
= 00001076 ( 0000574)

1.0/ 7777 (-3)                        – now 3 files
4/ 7777                               – 1 extra word per entry
0001.00005\ 0000 'DU                  – set up a "DUMMY" file
0001.00006\ 7556 'MM                  –  over the old <EMPTY>
0001.00007\ 1752 'Y@
0001.00010\ 3451 0                    – a null extension
0001.00011\ 6234 (D)                  – put in today's date
0001.00012\ 4235 (-^D82)              – length
0001.00013\ 5761 'LO                  – the desired file
0001.00014\ 3341 'ST
0001.00015\ 2371 0
```

```
0001.00016\ 1107 'FI          – the extension
0001.00017\ 1366 (D)
0001.00020\ 3015 (-^D25)      – its length
0001.00021\ 3415 0            – an <EMPTY> to end it
0001.00022\ 2713 (-^D574)     – the rest of the tape

WRITE                         – now write it out
^C                            –  & exit to use it
```

The "LINE-FEED" key was used to advance through the words.

The above example is exactly the same as hand calculating the required length of the "DUMMY" file and then doing the following sequence using PIP:

```
.R PIP
*DTA1:DUMMY</I=122            – enter the DUMMY file
*DTA1:LOST.FI</I=31           – enter the LOST.FI
*^C
```

Note that the lengths of the files are specified for PIP in octal.

EXAMPLE 6:

Search for the end of each page of text in the file "WRITE.UP". Since the file is an OS/8 ASCII file, which has two characters packed in the low 8 bits of two words and a third character packed in the high 4 bits of both of the two words, the form-feed character (^L) may be packed as the third character in some cases. So it is necessary to search both through the low 8 bits of each word and through the high 4 bits of each pair of words. Do it as follows:

```
.R FUTIL

FI WRITE.UP
WRITE.UP 0301-0437 S^P        – typeout stopped
SE MA 377
SE LO 301.0 UP 437.377        – char mask & limits set

W A "^L                       – search for form-feed

. . . . . typeout occurs here

SMASK 7400,7400               – set up string mask

ST M A ("^L*20),("^L*400)     – search for 3rd char f-f

. . . . . more typeout here   – only even addresses are real
                              –  parts of form-feed pair!
```

In the string search, both the string and the data searched are "masked" by the string mask.

EXAMPLE 7:

You just assembled and saved PROG.SV but forgot to use the "/P" switch to ABSLDR. Fix the CCB (core control block) as follows:

```
.R FUTIL
```