27
5-3-77

MASTER

UCID-17371

Lawrence Livermore Laboratory

PROGRAM MANUAL FOR THE DATA DIRECTOR EDITOR

Patrick R. McGoldrick

February 17, 1977

Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

DISCLAIMER

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $; Microfiche $3.00

Page Range	Domestic Price	Page Range	Domestic Price
001–025	$ 3.50	326–350	10.00
026–050	4.00	351–375	10.50
051–075	4.50	376–400	10.75
076–100	5.00	401–425	11.00
101–125	5.50	426–450	11.75
126–150	6.00	451–475	12.00
151–175	6.75	476–500	12.50
176–200	7.50	501–525	12.75
201–225	7.75	526–550	13.00
226–250	8.00	551–575	13.50
251–275	9.00	576–600	13.75
276–300	9.25	601–up	*
301–325	9.75		

*Add $2.50 for each additional 100 page increment from 601 to 1,000 pages; add $4.50 for each additional 100 page increment over 1,000 pages.

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

CONTENTS

PROGRAM MANUAL FOR THE DATA DIRECTOR EDITOR

ABSTRACT

The Data Director editor is a powerful, multiuser editor that will aid in the development and modification of APT part programs, assembly-language programs, and other text.

Some benefits of the editor are:

1) Most of the editor is reentrant, allowing several users to share it.
2) The user can use the editor as though the entire file being edited is in memory.
3) Editing takes place on a working file so that changes are not made to the original file until desired
4) The editor offers a powerful command set where most commands have the same syntax.

INTRODUCTION

This manual is written to aid those who maintain the Data Director Editor Program and assumes that the reader is familiar with the editor's operation. It further assumes that the reader has read and understood the Data Director Editor User's Manual (Ref. 1) and is moderately familiar with Interdata's assembly language and RTOS V (Interdata's real time operating system). This manual also can aid those who wish to use the editor on other systems and points out possible problem areas. If this manual is being used to implement the editor on systems other than LLL's Data Director system, then certain references to filenames and other items that exist only in the Data Director should be ignored.

EDITOR STRUCTURE

To allow several users to edit simultaneously without having one copy of the editor per user, most of the editor is reentrant and may be shared by many users. The editor is in two parts: the user task, where variables determining the user identity and editor state are stored, and a reentrant procedure, EDIT, that performs the editing function.

The User Task

In its simplest form, the user task consists of setting register 12 to a large expanse of free memory and calling the reentrant procedure, EDIT. EDIT uses the free memory to store the editor's state information. Since Working Files are named and can appear in the same disk partition as others, each user has to have a unique name for his Working File. This name is stored just before the large scratch area.

EDIT needs to know the input and output device number of the user console. The device numbers are stored in FILEID blocks [*] and the block's addresses are stored before the working filename.

Thus, we can write a simple user task:

```
R12      EQU 12
R15      EQU 15
         EXTRN EDIT
*
START    LHI R12,SCRATCH          R12 = A(SCRATCH)
         BAL R15,EDIT             CALL THE EDITOR
         SVC 3,0                  END OF JOB
*
INFID    DC 0,INDEVICE,X'4040' 4040 MEANS NOT A FILER FILE
OUTFID   DC 0,OUTDEV,X'4040'
         DC A(INFID)
         DC A(OUTFID)
         DC C'EDITUSER01'         WORKING FILE NAME
SCRATCH  DS 1742
         END
```

The size of the scratch area is dependent on some parameters in the reentrant portion of the editor that can be set at assembly time. A rule of thumb is that scratch size is equal to the working File's buffer size plus 718 bytes. The above example user task was written to run with an EDIT that has a buffer size of 1024 bytes. EDIT'S buffer size is set in file EDITCOMS by label BUFSIZE (see line 21, Appendix A).

---

[*] FILEID blocks are blocks of data that identify files to the file I/O routines. A user console is considered to be a file by the editor.

The user task is established via the task establisher task (TET) if running under RTOS. The file EDITTET contains a set of commands to TET to establish the editor task, and a copy of it is provided with the editor listings.

Logical Units

EDIT uses logical units 1 through 7, which are assigned as follows:

```
LU 1 - Input file device     {O FILE}
LU 2 - Output file device    {NF FILE}
LU 3 - not used
LU 4 - Working-file disk partition
LU 5 - Input commands
LU 6 - Output messages and lists
```

When creating a user task, only LU 4, 5, and 6 need to be assigned. EDIT will assign the other logical units to the directory used in an editor command.

Reentrant Edit Procedure

When called, the EDIT procedure handles all the editing functions (i.e., obtains commands from the user and performs them). In performing its task, EDIT calls on other reentrant procedures, which themselves may call even more reentrant procedures. The package of procedures that is called is labeled the reentrant section.

The procedures in the reentrant section are packed into modules that can be separately assembled. On a minicomputer, assembly time can be long. By having several independently assemblable modules, it takes little time to make a modification in one procedure. The modules can be added to the reentrant library or can be linked together with a link editor (TET). The object modules for the editor are kept in the Data Director in a file called EDITBxxxx, where xxxx is the month and day the last module was upgraded.

EDITOR DATA STRUCTURE

The editor data structure is stored in the user task and contains almost all of the editor state information for that user. Procedure INITIAL obtains the necessary memory from the scratch pointer, register 12, and sets register S to point to the memory for the structure. Data is referenced by an offset from register S; LH RX,L.INT(S) for example.

The offsets are defined via a STRUC assembler pseudo-opcode in the file EDITCOMS. Modules that use the data structure obtain EDITCOMS at assembly time via a COPY EDITCOMS pseudo-opcode. This permits the data structure to be changed without changing every module; reeassembly is all that is required.

Appendix A is an assembly listing of the Editor Data Structure.

THE WORKING FILE

When a file is opened for editing, the lines in the file are inserted into another disk file known as the working file. This file contains the lines, the line numbers and the pointers that place the lines in numerical order.

When lines are deleted, inserted, or modified, it is the contents of the working file that are changed. Only when an NF or END command is executed are the lines in the working file written to the opened file or to a new file.

Working-File Structure

The working file consists of one or more disk pages linked together in a doubly linked list. The disk partition that is to hold the working file must first be formatted in a special way. These specially formatted disk partitions are known as "filer" partitions, because they are compatible with Interdata's "filer" routines. Formatting of the disk partition may be done with task SF1024, which sets up its logical unit 4 to a 1024 byte record size. The disk partition for the working file need only be formatted once, as long as the partition is not aberrated by any nonfiler oriented tasks.

The sector numbers of the first and last working-file pages are called the head of the list of working-file pages and are stored in memory as FIRSTREC and LASTREC (see lines 105 and 106, Appendix A). When a page on the disk points to the head, its next pointer or last pointer is set to zero.

When any of the working file's pages are brought into memory, the page in memory is called the "edit buffer". The definition of the edit buffer, and hence any page on disk, is found on lines 135 through 144 in Appendix A. B.FIRST through B.TOP is a structure that contains lines of text with their line numbers and will be discussed in the next section, "The Edit Buffer."

Each working-file page contains lines of text, if the file is not empty. The pages are ordered in that all line numbers on a page must be less than all line numbers on the next page and greater than all the line numbers on the previous page. Lines are not allowed to cross page boundaries. (See Fig. 1 for an example of the working file.)

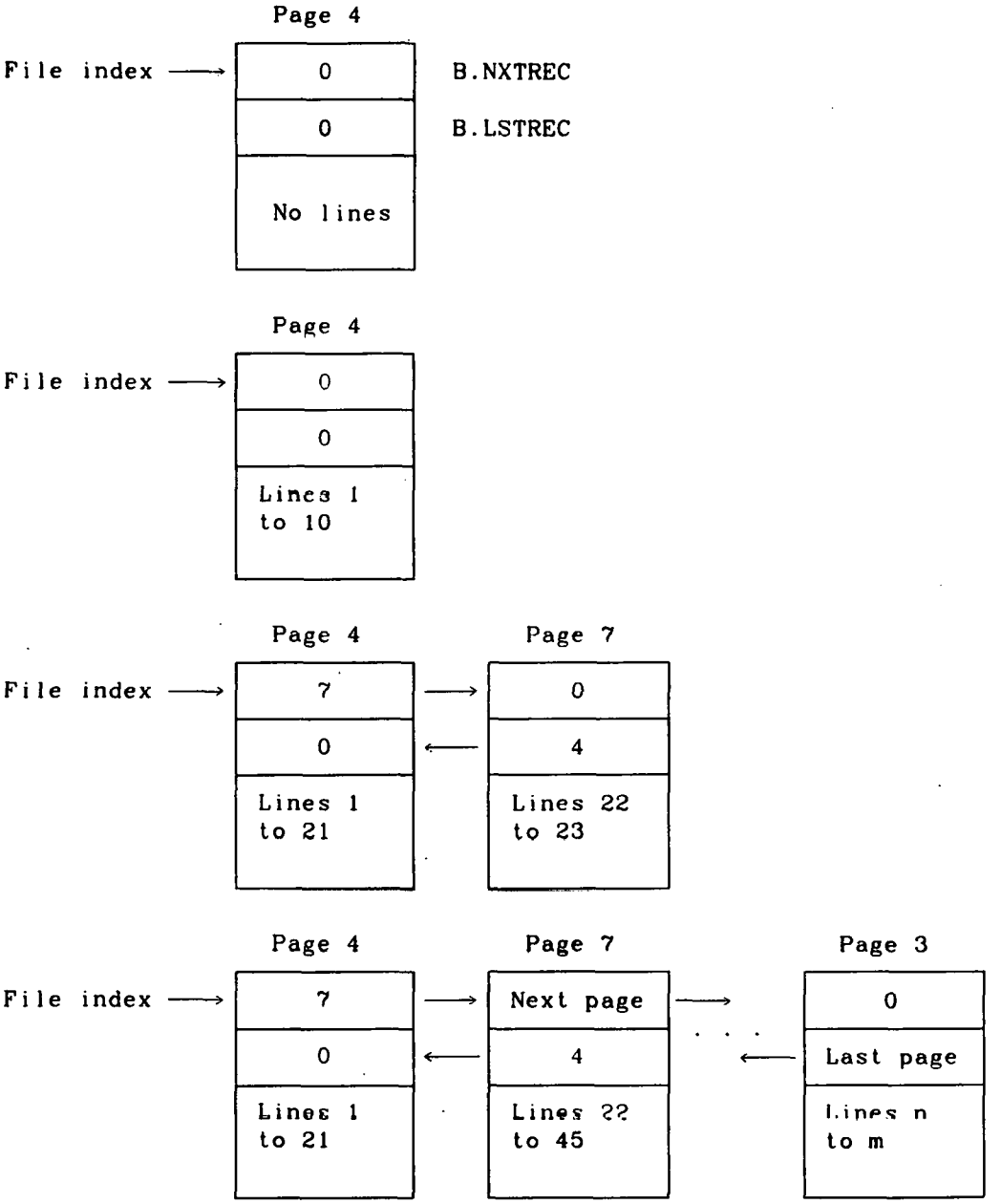


Fig. 1. Example of the working file. Line numbers n to m are greater than any other line numbers in the file. The pointers that form the linked list of pages are sector numbers, and can be listed in a random order as the free pages are obtained from the filing system. The head, {FIRSTREC, LASTREC}, in this example is {4,4}, {4,4}, {4,7}, and {4,3}, respectively.

The editor has a rule that unless the working file is empty, all pages must contain lines. As pages become empty from deletions, they are removed from the list, until only the page pointed to by the index remains. This page is allowed to be empty and signals an empty working file.

When the working file is first created, only one empty page exists. As lines are added, the page fills until it is full. When a new line is to be added and the current page is full, a new page must be added to accommodate the new line. This is done by splitting the current page, after the appropriate line, into two pages. Each page will be full sized, so at least one page will be able to hold the line to be inserted.

When the working file is forgotten (FF), it is merely destroyed and the pages are returned to the filing system. To manipulate the working file, the following procedures are used:

MODULE	PROCEDURE	PURPOSE
EDITF	CWORK	CREATE WORKING FILE & SETUP THE FIRST BUFFER
FINDL	SPLITPG	SPLIT PAGE AFTER LINE BP INTO 2 PAGES
FINDL	RELPG	RELEASE CURRENT PAGE IF EMPTY
FINDL	NEXTREC	READ IN NEXT RECORD UPDATING IF NECESSARY
FINDL	READER	READ RECORD F.CUREC
FINDL	WRITER	WRITE OUT F.CUREC
FINDL	GETREC	GET AN AVAILABLE RECORD
FINDL	GIVREC	GIVE UP A RECORD
FINDL	DWORK	DESTROY WORKING FILE AND RELEASE ALL FREE RECORDS
FINDL	LSTREC	READ IN RECORD BEFORE F.CUREC

The Edit Buffer

The edit buffer is a disk page of the working file in memory. The buffer is defined in lines 128 through 131, and again separately in lines 136 through 144 in Appendix A. Lines of text, with their line numbers and length, are stored in the edit buffer in numerical order in a doubly linked list. The head of the list is B.FIRST and B.LAST, pointing to the first and last lines in the buffer. All links in the edit buffer's list are offsets from B.FIRST.

B.AVAIL is a link to a list of available or free lines. This is used when spliting a page into two pages. For one page on disk, the lines after the line where the insertion is to go are made available by placing a link to it in B.AVAIL. The added page then has the lines from B.FIRST to the place of the insertion made available. This makes page splitting very fast, as CPU time is not used for freeing lines until a space is to be needed when an insertion is to be made.

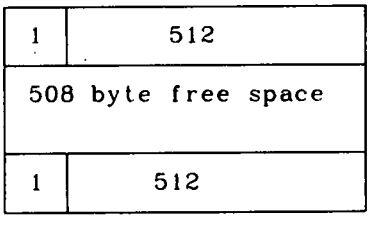
Each line in the edit buffer is stored in a line buffer defined by lines 147 through 154 in Appendix A. Each line buffer has two link fields, BP.NEXT and BP.LAST, pointing to the next line and to the last line in the list. BP.INT + BP.DEC/100 form the line number. BP.SIZE is the actual size of the line to follow, in bytes. BP.LINE is the start of the line that is BP.SIZE big.

If BP.SIZE is odd then the line ends on an even byte boundary. All line buffers are forced to end on an odd byte boundary due to technical limitations in the Inderdata computer (Halfword instructions only work on even byte boundaries). Also, in RTOS, all I/O must end on an odd byte boundary, so a carriage return is added to any line that ends on an even boundary. BP.SIZE remains unaffected by the above.

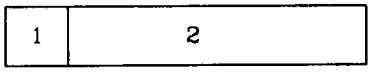
Adding or deleting an item to or from a doubly-linked list is straightforward and is explained in detail in Ref. 2. The insertions and deletions in EDIT are done by procedures INSERT, below label NOTEVEN, and DELETE, below label DELETE2.

Since line buffers are variable in length, the editor engages in dynamic storage allocation in the edit buffer from B.BOTTOM to B.TOP for storage. The dynamic storage algorithm is a modified form of Knuth's boundary tag method (Ref. 3). Our version of Knuth's algorithm requires no "available list" unless the page has been split. If the page has been split, all line buffers on the available list are released before the algorithm is evoked.

In the edit buffer, a free space is noted by a tag (most significant bit is a 1), followed by the size of the free space. Thus, a 512 byte free space looks like this:

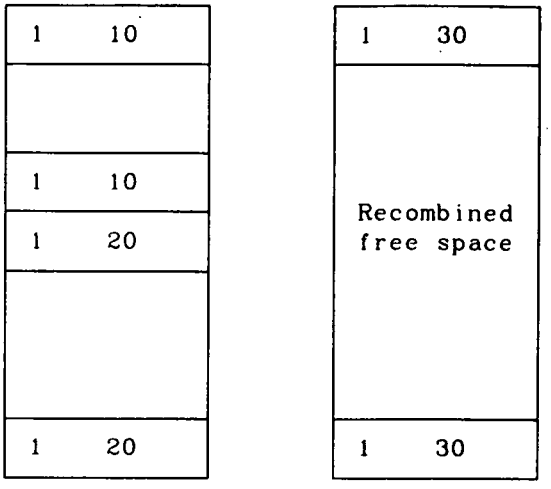


A 2 byte free space looks like:



Filled spaces (i.e., line buffers) do not need a size field before and after them, as their size is known and the first and last words in the buffer cannot have the tag set. This is assured by making a rule that the edit buffer must never be larger than 32,767 bytes. This ensures that BP.NEXT is less than 32,767, and the tag bit is off in the 16 bit field. Also, the editor only edits lines that are 7-bit ASCII. This ensures that the tag bit cannot be set on the end of the line buffer.

The memory allocation procedure, GETBUF, uses the first-fit method by scanning line buffers and free spaces until it finds a free space large enough for the insertion. If none is available in one piece, but there is enough free space in the edit buffer, GETBUF will call COMPRESS to compress all the free space to the top of the buffer. Memory is released by RELBUF, which makes tags and size fields for the released line buffer and recombines the buffer with any other adjacent free space:



The procedures below are used to manipulate the edit buffer:

MODULE	PROCEDURE	PURPOSE
FINDL	INSERB	INSERT LINE L. BEFORE LINE POINTED TO BY BP
FINDL	INSERT	INSERT LINE L. AFTER LINE POINTED TO BY BP
FINDL	DELETE	DELETE LINE POINTED TO BY BP
FINDL	RELBUF	RELEASE BUFFER OBTAINED BY GETBUF
FINDL	GETBUF	GET A BUFFER OF LENGTH IN R14
FINDL	SUBSPLIT	SPLIT A BLOCK OF LENGTH+N INTO 2 BLOCKS OF LENGTH AND N
FINDL	COMPRESS	COMPRESS AVAILABLE BUFFERS INTO ONE BUFFER

Finding a Line

When a line is to be operated on by the editor, the page that the line is on must first be brought into memory (if not in already), and a pointer (register BP) must be set to the address of the line's buffer. This operation is called finding a line and is performed by procedure FINDL. All references to lines at FINDL's level are by line number.

FINDL accomplishes the above by calling a procedure (FASTPG) to guess the page that contains the line number and bring it into memory if it's not in already. FINDL then starts at the first line in the page and tests for the desired line number, searching up or down, until the line is found or it has moved past where the line should be. Procedure MOVE allows FINDL and other procedures to easily move BP up or down a number of lines from its current position. MOVE brings in new pages as necessary.

Procedure FASTPG guesses what page the desired line number is on. FASTPG will either guess the page and bring it into memory, or will let the page currently in memory stay as being close enough. FASTPG uses a hashtable, RECORDS, where the sector numbers of pages are stored. The hashing method is such that numbers near each other tend to hash to the same number. Procedure GENKEY generates keys into the hashtable by the following formula.

KEY = FLOOR(MAXKEY * desired line number/MXNUM)

where:

FLOOR is the least integer function.
MAXKEY is the maximum number of entries in the table.
MXNUM is the maximum line number and is arbitrarily chosen.

When a page is split, the first line number in the new page is used to generate a key where the sector number of the page is stored (procedure SAVREC):

RECORDS(KEY) = sector number

When a line number is presented to SAVREC to be placed in the hashtable and the line number produces a key greater than that in the table, MXNUM is increased and the sector numbers are relocated down in the table. MXNUM is generally increased by 25% so that the relocation operation does not have to take place too often, and the table retains its effectiveness. When pages are released from the working file, procedure FRGREC removes the page from RECORDS if the sector number was in it.

The following routines permit finding lines and moving BP up and down lines in the edit buffer:

MODULE	PROCEDURE	PURPOSE
FINDL	FINDL	GIVEN THE LINE #, SET BP TO POINT TO THAT LINEIS LINE BUFFER
FINDL	MOVE	MOVE UP/DOWN R14 LINES, SET BP TO RESULTANT LINE
SAVREC	SAVREC	ASSOCIATE A RECORD # WITH A LINE # SO CAN FIND LINES FAST
SAVREC	FASTPG	GUESS PAGE NUMBER THAT LINE NUMBER IS ON AND GET PAGE.
SAVREC	GENKEY	CALCULATE KEY FOR LINE NUMBER
SAVREC	FRGREC	DISASSOCIATE RECORD NUMBER FROM LINE NUMBER

Copy and Move After

Copy and move after from a working file is accomplished by copying all lines in RANGE into a temporary disk file and then inserting into the working file all the lines in the temporary disk file.

The temporary disk file has no name and is in essence a FIFO (first-in, first-out queue) that is in memory. Lines are pushed into the FIFO as the lines in RANGE are found. When the FIFO overflows, the lines in it are put to disk giving more room in the FIFO for new entries. Thus when copying or moving small numbers of lines there probably won't be any disk activity concerning the temporary file.

Move after is accomplished in the same way as copy after, except that the lines in RANGE are deleted from the working file as they are pushed into the FIFO. Copy after from a FILEID does not need a temporary disk file. The FILEID is opened, and all lines in RANGE are inserted directly into the working file. Following are the routines for copy and move after:

MODULE	PROCEDURE	PURPOSE
EDITCA	DOMACA	DO MOVE AFTER OR COPY AFTER
EDITCA	CAFILE	COPY ALL LINES IN RANGE FROM FILEID AFTER CAL.
EDITCA	FNDCAL	SET BP TO LINE CAL. OR LINE BEFORE IF NONE
EDITCA	SETGETCA	SETUP FOR GETCA, FIND FIRST LINE IN RANGE IN FILE
EDITCA	GETCA	GET NEXT LINE IN RANGE FROM FILEID
EDITCA	CAWORK	MOVE OR COPY LINES IN RANGE IN WORKING FILE AFTER CAL.
EDITCA	SETEMP	SETUP FOR PUTEMP
EDITCA	PUTEMP	PUT LINE AWAY IN FIFO THAT CAN EXTEND TO DISK
EDITCA	ENDPUT	FINSH PUTEMP AND SETUP FOR GETEMP
EDITCA	GETEMP	GET NEXT LINE FROM FIFO
EDITCA	RELTEMP	RELEASE FIFO

Appendices B through E provide complete listings of the editor's modules and procedures.

EDITOR STATISTICS

The editor keeps statistics in itself. See lines 172 to 186 in Appendix A. These statistics can be of use in determining the effectiveness of certain algorithms. We can gather statistics from several users simultaneously and can accumulate them over a long period of time to give accurate results.

This method is much better than random testing, since EDIT users do not access lines randomly. Often, an EDIT user will start at the front of his file and move through it, editing as he goes. Or, the user may edit in only a small area of his file. Thus, these statistics give a true reflection of the efficiency of our algorithms.

There are three routines involved in gathering statistics:

MODULE	PROCEDURE	PURPOSE
STAT	STAT	SAVE STATISTICS IN TASK COMMON
STAT	STATIN	START STATISTICAL INCREMENT
STAT	STATIE	END STATISTICAL INCREMENT

Since there are only three statistics routines, they can be as simple or as complex as desired. We can modify them to make distributions or other statistical items of interest. When we no longer care about statistics, the routines can be removed or can be modified to do nothing.

Statistical Results

Table 1 gives the results of nine days worth of editing taken at different times. Files of differing lengths (300 to 2400 lines) were edited, but most of the editing was done on the larger files. The working-file page size for Table 1 was 512 bytes.

Table 1. Statistical results.

	Day								
	1	2	3	4	5	6	7	8	9
Number of calls to FINDL	212	138	180	165	29	45	49	252	572
Number of page faults during FINDL	1109	666	414	838	8	17	45	183	305
Total number of page faults	2278	1573	1766	1785	232	399	264	1161	3157
Number of disk I/O's	3453	2462	3441	2694	358	927	960	1515	4839
Number of insertions	9509	4807	13334	8045	1839	3593	9742	2825	10808
Number of deletions	1561	1784	2425	1719	57	1140	13	111	1796
Number of lines listed	–	–	–	–	378	483	405	2703	5569
Number of compresses	219	7	20	4	0	4	2	13	72
Number of released pages	34	101	134	92	0	61	0	2	101
Number of page splits	968	701	772	743	102	487	929	217	696
Number of times AVAILSTK overflowed	–	–	12	–	0	–	–	0	8
Number of times AVAILSTK underflowed	–	–	81	–	11	–	–	24	80

The first four columns of Table 1 represent statistics taken before FASTPG was implemented. Notice the higher percentage of page faults per call to FINDL before FASTPG was implemented. The average number of page faults per call to FINDL before FASTPG was implemented was 4.345, while the average number of page faults per call to FINDL after implementation was 0.551. This was an 87% improvement. It may be possible to obtain even better improvement, and the statistics will allow us to see the results of experiments with different algorithms. FASTPG was improved again in early October 1976, and the average number of page faults per call to FINDL decreased to 0.341.

THE EDITOR AND OTHER OPERATING SYSTEMS

The editor was written to operate on an advanced version of RTOS-5 and should run under standard interdata RTOS with no modifications (assemble EDITR with assembly option SVC11 EQU 0). With modification, EDIT should be able to run under any operating system. In this section, I will outline functions that the editor uses that may differ between our version of RTOS-5 and other systems. The editor or system may have to be modified to get the editor to operate at its full potential.

The editor can be assembled with assembly option NOFID EQU 1. With this option, the editor will allow no FILEID's on an open, NF, or END command, so the user can assign LU 1 to the source file and LU 2 to the output file with the user's operating system's supervisor. This option solves most of the problems listed below.

User Console

EDIT can operate with any ASCII input device as its command input and any ASCII output device for its comments. However, there is some extra power to be gained if the editor is run on a system with the FULL DUPLEX DRIVER controlling the user console (See Ref. 4.)

The editor prompts and reads via a single SVC 1. This causes no conflict on most systems, since the function code for this SVC is both the read and write bits set. On most systems this causes a read without the editor's prompt character.

Multiple commands can be input per line in the editor. These commands are separated with an X'A1' code. As this character cannot be typed on most terminals, a way must be found to implement it if one wants multiple commands per line. Our FULL DUPLEX DRIVER inserts X'A1' into the user buffer whenever linefeed is typed, and it echos "!" as a command separator.

Filer I/O and Dynamic Memory Allocation

If any filer files are opened or NFed, the editor needs 1K more memory. The editor uses dynamic memory allocation (SVC 2 option D) to get it. Therefore, if one wishes to open or NF a filer file on a system without dynamic memory allocation, the routines ALOMEM and RELMEM will have to be modified to do get-storage (SVC 2 option 2) and release-storage (SVC 2 option 3). The routines are between six to nine lines long, so their modification should be fairly simple. The above does not apply if the editor modules are assembled with assembly option NOFID EQU 1.

Logical Units

The editor makes logical to physical unit assignments dynamically while running. These are all made by procedure ASSILU in module OFILE. Since EDIT was designed to run under RTOS, this 12-line procedure fetches the UTCB pointer (SVC 2 option 5) and sets the physical unit by indexing into the logical unit table. ASSILU must be rewritten to operate the editor under operating systems that do not have the same UTCB format for I/O assignments. The rewrite should be fairly simple, since the current procedure, ASSILU, is not complex, being only 12 lines long. The above does not apply if the editor modules are assembled with assembly option NOFID EQU 1.

Files and Devices

The editor was designed to operate with the Data Director filing system, which is a set of procedures that are linked to the editor. A brief discussion of the filing system is included here so that a user can decide if changes should be made to it. If the editor modules are assembled with assembly option NOFID EQU 1, the following does not apply.

FILEID is the name or identification of a file. FILEID is what is typed after a C(create), O(open), NF(new file), END, or maybe CA(copy after) command. Procedure EVFILE evaluates FILEID and sets up a FILEID block (FID). All the I/O that the editor does is done by routines CRFILE, OFILE, OAFILE, READ, WRITE, and CFILE. (The editor does random reads and writes to the working file directly, but that does not concern us here and should be no problem to all but the most restrictive operating systems.) These procedures do I/O to the device in the FID block. All devices are treated as though they are files.

The FID block tells whether the file is a filer file or I/O device. The FID block also gives the file name, security level, and attributes of the file (see F. structure in EVFILE). Therefore, procedure EVFILE simply maps a symbolic FILEID into a FID block. To create custom mapping there are several alternatives.

1) Keep the Data Director formats. O .HSR means open a file from the high-speed reader. A list of device codes can be found in EVFILE in DEVTABLE for the mnemonic and DEVICE for the physical unit number. These two tables can be enlarged with custom devices if desired.

   O FILENAME means open a filer file, FILENAME, on the default filer library (label DFDEVICE in EVFILE).

   O .DOC:FILENAME means open a filer file, FILENAME, on the DOC filer library. A table of filer library mnemonics to physical disk partitions is contained in tables DIRTABLE and DIRDEV in EVFILE.

   If the system on which EDIT is to be implemented uses filer files, then DIRTABLE and DIRDEV could be modified to suit. Caution: the Data Director filer differs from the standard Interdata filer in several important respects:

a) Ten-character filenames.

b) File security, attributes, date, and time of creation are stored before the first data record in the file (transparent to most filer users).

c) Filer can be called with a reentrant call.

d) The filer block has been shortened from 66 to 24 bytes (see FILER label BUFFER).

e) The filer block and filer record buffer may be split in memory (see AREC in filer BUFFER).

f) Register 12 must point to a scratch area of 56 bytes that can be reused by other routines between filer calls.

In the Data Director operating system most user tasks do not call filer. They make FID blocks and call CRFILE, OFILE, OAFILE, READ, WRITE and, CFILE in order to remain device independent on a file level. The editor assumes through the above six procedures that the physical record size of the filer records is less than or equal to 512 bytes.

2) Rewrite EVFILE to provide one's own mapping between symbolic FILEID's and FID blocks. This could be done so that all FID blocks made are devices and not filer files. For example, O 4 could mean open a file from device 4, the card reader.

3) Rewite CRFILE, OFILE, OAFILE, READ, WRITE, and CFILE to handle filing as it exists in one's own operating system.

REFERENCES

1. P. R. McGoldrick, *Data Director Editor User's Manual*, Lawrence Livermore Laboratory, Rept. 17372 (1977).

2. D. E. Knuth, *Fundamental Algorithms* (Addison-Wesley, Reading, Mass., 1973), vol. 1, pg. 278.

3. *Ibid.*, pg. 441.

4. P. R. McGoldrick, *Full Duplex CRT and Teletypewriter Driver*, Lawrence Livermore Laboratory, Rept. UCID-16961 (1975).

APPENDIX A: LISTING OF THE EDITOR DATA STRUCTURE

```
PROG= *NONE*      ASSEMBLED BY CAL 03-066R04(16-BIT)

                                1  **EDITCOMS
                                2  * 9/28/76 PRM
                                3  *----------------
                                4  * CONSTANTS
          0000 0080             5  MINUSF   EQU    X'80'            MINUS INCREMENT FLAG
          0000 000D             6  CR       EQU    13
          0000 0020             7  BLANK    EQU    X'20'
          0000 00A1             8  LEOM     EQU    X'A1'            LOGICAL END OF MESSAGE
          0000 0008             9  LOVHEAD  EQU    8
          0000 000A            10  BOHEAD   EQU    10               BUFFER OVERHEAD
          0000 8000            11  EMPTY    EQU    X'8000'          EMPTY FLAG
          FFFF FFFF            12  FIRST    EQU    -1               CODE FOR FIRST LINE
          FFFF FFFE            13  LAST     EQU    -2               LAST LINE
                               14  *----------------
                               15  * SYSGEN PRAMETERS FOR EDIT BUFFER
                               16  * SIZE OF STACKS FOR AVAILABLE RECORDS
          0000 000A            17  AVAILSZ  EQU    10
          0000 0003            18  OVLSTKSZ EQU    3
                               19  * BUFFER SIZES
          0000 0082            20  LLENGTH  EQU    130              MAX LINE LENGTH < 255 & EVEN
          0000 0400            21  BUFSIZE  EQU    1024             EDIT BUFFER SIZE (MULTIPLE OF 256)
          0000 0028            22  MXKEY    EQU    40               # OF KEYS TO WORKING FILE
          0000 0190            23  MXNUMI   EQU    400              INITIAL # OF LINES FOR FASTPG
                               25  *
                               26  * EDIT DATA STRUCTURE DEFINITION
          FFFF FFF4            27  WORKNAME EQU    -12              NAME OF WORKING FILE (BY CALLER)
                               28  S.       STRUC
     0000                      29  EOJ      DS     1                END OF JOB IF TRUE
     0001                      30  CERF     DS     1                LOG COMMANDS IF TRUE
     0002                      31  TCF      DS     1                TYPE CHANGES IF TRUE
     0003                      32  BATCHF   DS     1                DOING BATCH IF TRUE
     0004                      33  ASMF     DS     1                ASSEMBLY FORMAT IF TRUE
     0005                      34  APTSYNF  DS     1                APT SYNONYMS ON IF TRUE
     0006                      35  APTCKF   DS     1                APT SYNTAX CHECK IF TRUE
     0007                      36  SYMF     DS     1                SYMBOL IF TRUE
                               37  *
                               38  * CURRENT FILE
     0008                      39  FC.OPN   DS     2                A(FILER BLOCK) IF OPEN,0 IF UNOPEN
     000A                      40  FC.DEV   DS     2                PHYSICAL DEVICE
     000C                      41  FC.IATTR DS     1                DESIRED ATTRIBUTES
     000D                      42  FC.FATTR DS     1                FILES ATTRIBUTES
     000E                      43  FC.NAME  DS     10               FILE NAME
                               44  *
                               45  * SUB STRUCTURE FOR TOKEN GETTER
     0018                      46  W.BUF    DS     1
     0019                      47  W.CODE   DS     1                TOKEN TYPE
     001A                      48  W.LENGTH DS     2                LENGTH OF TOKEN
     001C                      49  W.VALUE  DS     4                VALUE OF ASCII # IN BINARY
```

```
0020                50  W.LINEPT DS    2               A(INPUT LINE)
0022                51  W.STATE  DS    1
0023                52  W.STATUS DS    1
0024                53  W.LDELIM DS    1               LEFT DELIMITER
0025                54  W.RDELIM DS    1               RIGHT DELIMITER
0026                55  W.IPNTR  DS    2               POINTS TO END OF TOKEN+1
                    56  *
0028                57           DS    1               JUST TO EVEN UP HALFWORD BOUNDRY
0029                58  ESCHAR   DS    1               START COMMAND LINE WITH LEOM
002A                59  LINE     DS    LLENGTH-1       ROOM FOR A LINE
00AB                60  LINEND   DS    3               END OF LINE
00AE                61  SVC.FUNC DS    2               SVC PRAMETER BLOCK
00B0                62  SVC.STAT DS    2
00B2                63  SVC.B    DS    2
00B4                64  SVC.E    DS    2
00B6                65  SVC.PROM DS    2
                    66  *
                    67  * LINE NUMBER
00B8                68  L.INT    DS    2               INTEGER PART
00BA                69  L.DEC    DS    1               DECIMAL PART
00BB                70  L.INC    DS    1               INCREMENT OFF LINE #
                    71  *
                    72  * CURRENT LINE
00BC                73  LC.INT   DS    2
00BE                74  LC.DEC   DS    1
00BF                75  LC.INC   DS    1
                    76  *
                    77  * LINE # FOR START OF PAGE LISTING
00C0                78  P.INT    DS    2
00C2                79  P.DEC    DS    1
00C3                80  P.INC    DS    1
                    81  *
                    82  * START AND END LINE NUMBERS IN RANGE
00C4                83  LS.INT   DS    2
00C6                84  LS.DEC   DS    1
00C7                85  LS.INC   DS    1
00C8                86  LE.INT   DS    2
00CA                87  LE.DEC   DS    1
00CB                88  LE.INC   DS    1
                    89  *
                    90  * ASSOCIATE RANGE, PATTERN ADDRESSES
00CC                91  PATSTART DS    2               A(START OF PATTERN)
00CE                92  PATLEN   DS    2               PATTERN LENGTH
00D0                93  RPATS    DS    2               A(START OF REPLACEMENT PATTERN)
00D2                94  RPATLEN  DS    2               LENGTH OF REPLACEMENT PATTERN
                    95  *
                    96  * START AND END COLUMN NUMBERS
00D4                97  COL.S    DS    1
00D5                98  COL.E    DS    1
                    99  *
00D6               100  AVAILSTK DS    2*AVAILSZ+4     AVAILABLE RECORD STACK
00EE               101  OVLSTK   DS    2*OVLSTKSZ+4    OVERFLOW STACK
                   102  *
                   103  * SECTOR NUMBER OF RECORDS IN WORKING FILE (FOR FAST SEARCH)
00F8               104  MXNUM    DS    2               MAX LINE # INTEGER PART
00FA               105  LASTREC  DS    2               DISK ADDR OF LAST RECORD
```

```
00FC                    106  FIRSTREC DS     2
00FE                    107  RECORDS  DS     2*MXKEY          A(RECORDS ORDERED BY LINE #)
                        108  *
                        109  * BLOCK FOR EDIT BUFFER
      0000 014E         110  F.BLOCK  EQU    *
014E                    111  F.AREC   DS     2                A(FILER RECORD)
0150                    112  F.OFLG   DS     1                OPEN FLAG
0151                    113  F.LU     DS     1                LOGICAL UNIT #
0152                    114  F.PSIZE  DS     2                PHYSICAL SIZE OF FILER RECORDS
0154                    115  F.CUREC  DS     2                CURRENT OPEN RECORD #
0156                    116  F.ISEC   DS     2                INDEX SECTOR #
0158                    117  F.ILOC   DS     2                INDEX LOCATION
015A                    118  F.PREVS  DS     4
015E                    119  F.NEXTS  DS     4
0162                    120  F.EXP    DS     2
0164                    121  F.UFLG   DS     1                UPDATE FLAG
0165                    122  F.CODES  DS     1
                        123  *
0166                    124  ERCODE   DS     2                ERROR CODE
0168                    125  STATSAV  DS     4                ROOM TO SAVE DATA FOR STATISTICAL INCRE
                        126  *
                        127  * EDIT BUFFER
016C                    128  E.NXTREC DS     2                TOP OF EDIT PAGE
016E                    129  E.LSTREC DS     2                A(LAST RECORD) ON DISK
0170                    130  ED       DS     BUFSIZE-4        EDIT BUFFER
056C                    131           ENDS
      0000 01F6         132  MINSIZE  EQU    BUFSIZE/2-BOHEAD 1/2 OF AVAILABLE BUFFER
      0000 0173         133  ATTR     EQU    ED+3
                        134  *--------
                        135  * EDIT BUFFER DEFINITION
      FFFF FFFC         136  B.NXTREC EQU    -4               LINK TO NEXT BUFFER ON DISK
      FFFF FFFE         137  B.LSTREC EQU    -2               LINK TO LAST BUFFER ON DISK
                        138  B.       STRUC
0000                    139  B.FIRST  DS     2                HEAD OF DOUBLY LINKED LIST
0002                    140  B.LAST   DS     2                A(LAST LINE)
0004                    141  B.AVAIL  DS     2                A(AVAILABLE LINE LIST)
0006                    142  B.BOTTOM DS     BUFSIZE-10       BOTTOM OF AREA FOR LINES
03FC                    143  B.TOP    DS     2
03FE                    144           ENDS
                        145  *
                        146  * DEFINITION OF A LINE BUFFER IN THE EDIT BUFFER
                        147  BP.      STRUC                   LINKED LIST OF LINES IN EDIT BUFFER
0000                    148  BP.NEXT  DS     2                POINTS TO NEXT LINE
0002                    149  BP.LAST  DS     2                POINTS TO LAST LINE
0004                    150  BP.INT   DS     2                LINE # INTEGER PART
0006                    151  BP.DEC   DS     1                DECIMAL PART
0007                    152  BP.SIZE  DS     1                SIZE OF LINE
0008                    153  BP.LINE  DS     2                THE LINE
000A                    154           ENDS
                        155  *
                        156  * FILER BLOCK
                        157  FI.      STRUC
0000                    158  FI.AREC  DS     2                A(FILER BUF)
0002                    159  FI.OFLG  DS     1                OPEN FLAG
0003                    160  FI.LU    DS     1                LOGICAL UNIT #
0004                    161  FI.PSIZE DS     2                SIZE OF FILER BUF
```

```
0006                    162  F1.CUREC DS     2
0008                    163  F1.ISEC  DS     2
000A                    164  F1.ILOC  DS     2
000C                    165  F1.PREVS DS     4
0010                    166  F1.NEXTS DS     4
0014                    167  F1.EXP   DS     2
0016                    168  F1.UFLG  DS     1
0017                    169  F1.CODES DS     1
0018                    170          ENDS
                        171  *
                        172  * TASK COMMON DEFINITION OF STATISTICS
                        173  TSKCOM   STRUC
0000                    174  NFINDL   DS     1              # OF SEARCHES FOR A LINE #
0001                    175  NPGFAULT DS     1              # OF PAGE FAULTS DURING FINDL
0002                    176  NPGFTOT  DS     1              TOTAL # OF PAGE FAULTS
0003                    177  NDIO     DS     1              # OF DISK I/O'S
0004                    178  NINSERT  DS     1              # OF INSERTIONS
0005                    179  NDELETE  DS     1              # OF DELETIONS
0006                    180  NLST     DS     1              NUMBER OF LINES LISTED
0007                    181  NCOMP    DS     1              NUMBER OF COMPRESSES
0008                    182  NRELPG   DS     1              # OF RELEASED PAGES
0009                    183  NSPLIT   DS     1              # OF PAGE SPLITS
000A                    184  NGIVREC  DS     1              # OF TIMES AVAILSTK OVERFLOWED
000B                    185  NGETREC  DS     1              # OF TIMES AVAILSTK UNDERFLOWED
000C                    186          ENDS
                        187  *
      0000 0088         188  EOF      EQU    X'88'          EOF CODE
0000R                   189          END
```

APPENDIX B: EDITOR'S MODULES AND PROCEDURES IN ORDER OF APPEARANCE

MODULE	PROCEDURE	PURPOSE
EDIT	EDIT	PERFORM EDITING FUNCTION
EDIT	INITIAL	CREATES AND INITIALIZES THE DATA STRUCTURE OFF R12
EDIT	GETCMD	GET COMMAND FROM INPUT DEVICE, PROMPT = .
EDIT	GETINS	GET LINE FOR INSERT, PROMP = *
EDIT	DOCMD	DO COMMANDS IN INPUT BUFFER UNTIL ERROR OR END OF LINE
EDIT	LOGON	RESET OR SET CERF TO LOG ALL EDITOR INTERACTIONS
EDIT	OFF	TURN OFF ALL OPTIONS
EDIT	APTSYN	TURN ON/OFF OPTION APTSYNF FOR APT SYNOMYNS
EDIT	APTCK	TURN ON/OFF OPTION APTCKF FOR APT SYNTAX CHECKER
EDIT	ASM	TURN ON/OFF OPTION ASMF FOR ASSEMBLY LANGUAGE FORMAT
EDIT	TC	TURN ON/OFF OPTION TCF FOR TYPE CHANGES OPTION
EDIT	ONOFF	DECODE ON/OFF ON COMMAND LINE, SET R14 TO ZERO IF ON ELSE 1 IF OFF
EDIT	SETTOKEN	SET UP DATA STRUCTURE TO DECODE TOKENS ON COMMAND LINE
EDIT	GETTOKEN	GET NEXT TOKEN ON COMMAND LINE
EDIT	END	TERMINATE EDIT [DO NF, FF END EOJ]
EDIT	C	CREATE A WORKING FILE
EDIT	O	OPEN FILE TO A WORKING FILE
EDIT	SETUPOC	COMMON SETUP ROUTINE FOR O AND C
EDIT	NF	WRITE OUT WORKING FILE TO FILEID
EDIT	FF	FORGET FILE BY DESTROYING THE WORKING FILE
EDIT	NUMCMD	HANDLE NUMBER INPUT AS A COMMAND
EDIT	BL	INSERT NEXT LINE BEFORE FIRST LINE IN RANGE
EDIT	AL	INSERT NEXT LINES AFTER FIRST LINE IN RANGE
EDIT	LF	LIST THE FIRST LINE IN RANGE
EDIT	LA	LIST ALL LINES IN RANGE
EDIT	DFL	DELETE FIRST LINE IN RANGE
EDIT	DAL	DELETE ALL LINES IN RANGE
EDIT	P	LIST A PAGE OF LINE STARTING WITH FIRST LINE IN RANGE
EDIT	Q	RELIST A PAGE OF LINES STARTING WITH P.INT
EDIT	RFP	REPLACE PATTERNS IN FIRST LINE IN RANGE
EDIT	RAP	REPLACE PATTERNS IN ALL LINES IN RANGE
EDIT	CA	COPY AFTER
EDIT	MA	MOVE AFTER
EDIT	NXTNUM	PUT NEXT LINE NUMBER IN WORKING FILE TO LE.
EDIT	SETCMDUP	COMMON SETUPS FOR MOST COMMANDS USING RANGE
EDIT	FSTRNG	FIND FIRST LINE IN RANGE
EDIT	EXPRNG	EVALUATE EXPLICIT RANGE OF FORM LINE#,LINE#
EDIT	ASORNG	EVALUATE ASSOCIATIVE RANGE
EDIT	REPRANGE	EVALUATE REPLACEMENT RANGE
EDIT	PATTERN	EVALUATE PATTERN ON COMMAND LINE
EDIT	PATDELIM	DETERMINE IF A CHARACTER IS A PATTERN OR SYMBOL DELIMITER
EDIT	COLIMITS	EVALUATE COLUMN LIMITS
EDIT	COLUMN	EVALUATE CURRENT TOKEN FOR A COLUMN NUMBER
EDIT	ALRANG	DO FUNCTION FOR EACH LINE IN RANGE
EDIT	ALRANG1	DO FUNCTION FOR EACH LINE IN RANGE
EDIT	LINENUM	EVALUATE CURRENT TOKEN FOR LINE NUMBER
EDIT	RNUM	EVALUATE CURRENT TOKEN FOR LINE NUMBER, TOKEN CAN NOT BE A COMMAND
EDIT	DECNUM	SET L.DEC IN DATA STRUCTURE IF INTEGER BETWEEN 0 AND 99
EDIT	INCR	HANDLE INCREMENT LINE NUMBERS OF FORM +1, -1 ETC.

EDIT	TERR	LOG ERROR MESSAGE
EDIT	TERORS	WRITE OUT POINTER TO CURRENT TOKEN AND ERROR MESSAGE
EDIT	CLEANTOK	MAKE SURE WE'RE POINTING TO THE NEXT TOKEN
EDIT	CLEANCMD	MAKE SURE NO MORE DATA IN INPUT COMMAND LINE
EDITCA	DOMACA	DO MOVE AFTER OR COPY AFTER
EDITCA	CAFILE	COPY ALL LINES IN RANGE FROM FILEID AFTER CAL.
EDITCA	FNDCAL	SET BP TO LINE CAL. OR LINE BEFORE IF NONE
EDITCA	SETGETCA	SETUP FOR GETCA, FIND FIRST LINE IN RANGE IN FILE
EDITCA	GETCA	GET NEXT LINE IN RANGE FROM FILEID
EDITCA	CAWORK	MOVE OR COPY LINES IN RANGE IN WORKING FILE AFTER CAL.
EDITCA	SETEMP	SETUP FOR PUTEMP
EDITCA	PUTEMP	PUT LINE AWAY IN FIFO THAT CAN EXTEND TO DISK
EDITCA	ENDPUT	FINSH PUTEMP AND SETUP FOR GETEMP
EDITCA	GETEMP	GET NEXT LINE FROM FIFO
EDITCA	RELTEMP	RELEASE FIFO
EDITF	BATCH	HANDLE BATCH COMMAND
EDITF	FILWRK	FILL WORKING FILE WITH DATA FROM CURRENT FILE
EDITF	SAVRWD	SAVE STATE OF WORD AFTER R12
EDITF	RSTWRD	RESTORE WORD BLOCK TO STATE WHEN SAVRWD WAS CALLED
EDITF	GETC	GET A LINE FROM COPY AFTER FID
EDITF	GETF	GET NEXT LINE FROM CURRENT FILE
EDITF	CWORK	CREATE WORKING FILE & SETUP THE FIRST BUFFER
EDITF	ILC	INSERT LINES AFTER LINE POINTED TO BY BP
EDITF	RENUMBER	RENUMBER LINES IF NECESSARY
EDITF	GLINES	GET LINES FOR INSERTIONS WATCHING FOR LEOMS
CEDIT	NXTRNG	SET BP & CURRENT LINE TO NEXT LINE IN RANGE
CEDIT	RNG	SET BP & CURRENT LINE TO NEXT LINE IN RANGE
CEDIT	CHKEXRNG	CHECK TO SEE IF LINE POINTED TO BY BP IS IN EXPLICIT RANGE
CEDIT	CHKASRNG	CHECK LINE, BP, TO SEE IF IT'S IN ASSOCIATIVE RANGE
CEDIT	FNDPAT	FIND PATTERN OR SYMBOL IN A LINE
CEDIT	SYMBOL	DETERMINE IF A CHARACTER IS A SYMBOL
FINDL	FINDL	GIVEN THE LINE #, SET BP TO POINT TO THE LINE BUFFER OF THE LINE
FINDL	MOVE	MOVE UP/DOWN R14 LINES, SET BP TO RESULTANT LINE
FINDL	INSERB	INSERT LINE L. BEFORE LINE POINTED TO BY BP
FINDL	INSERT	INSERT LINE L. AFTER LINE POINTED TO BY BP
FINDL	DELETE	DELETE LINE POINTED TO BY BP
FINDL	REPAT	REPLACE PATTERN IN LINE BP
FINDL	FRAGCOPY	COPY A LINE FRAGMENT
FINDL	COPY	COPY A LINE FRAGMENT
FINDL	PATCOPY	COPY PATTERN INTO LINE NL
FINDL	RELBUF	RELEASE BUFFER OBTAINED BY GETBUF
FINDL	GETBUF	GET A BUFFER OF LENGTH IN R14
FINDL	SUBSPLIT	SPLIT A BLOCK OF LENGTH+N INTO 2 BLOCKS OF LENGTH AND N
FINDL	COMPRESS	COMPRESS AVAILABLE BUFFERS INTO ONE BUFFER
FINDL	SPLITPG	SPLIT PAGE AFTER LINE BP INTO 2 PAGES
FINDL	RELPG	RELEASE CURRENT PAGE IF EMPTY
FINDL	NEXTREC	READ IN NEXT RECORD UPDATING IF NECESSARY
FINDL	READER	READ RECORD F.CUREC
FINDL	WRITER	WRITE OUT F.CUREC
FINDL	GETREC	GET AN AVAILABLE RECORD
FINDL	GIVREC	GIVE UP A RECORD
FINDL	DWORK	DESTROY WORKING FILE AND RELEASE ALL FREE RECORDS
FINDL	LSTREC	READ IN RECORD BEFORE F.CUREC
FINDL	ERRHAN	IF I/O ERROR THEN LOGIT SETTING ERROR
EVFILE	EVFILE	EVALUATE A FILEID IN ASCII AND PUT RESULTS IN A FILEID BLOCK
EVFILE	DEFILE	SET DEFAULTS IN FILEID BLOCK

EVFILE	SATTR	SET CLASSIFICATION AND PROPERTY TO MAX IN FID
EVFILE	LATTR	LOG FILE SECURITY IF > UNCLASSIFIED
LIST	LIST	LIST LINE POINTED TO BY BP IN UNFORMATTED OR ASSEMBLY LANGUAGE FORMAT WITH LINE NUMBERS
LIST	LINENA	CONVERT LINE NUMBER TO ASCII
LIST	FORMAT	FORMAT A LINE IN ASSEMBLY LANGUAGE FORMAT
LIST	PUTCHAR	PUT A CHARACTER IN A BUFFER UNTIL END OF LINE
LOGIT	LOGIT	LOG MESSAGE TO LU 6
LOGIT	LSTIT	LOG MESSAGE TO LU 6
OFILE	OFILE	OPEN A FILER FILE
OFILE	CRFILE	CREATE A FILE
OFILE	OAFILE	OPEN OR ALOCATE A FILER FILE
OFILE	SETUPO	COMMON SETUP ROUTINE FOR OFILE, CRFILE AND OAFILE
OFILE	ASSILU	ASSIGN LU TO PHYSICAL DEVICE NUMBER
OFILE	CFILE	CLOSE FILER FILE
OFILE	RELEASE	RELEASE DEVICE
OFILE	RELAMEM	RELEASE ANY ALOCATED MEMORY FOR FILER
OFILE	WRITE	WRITE OUT A LINE TO FILER OR A DEVICE
OFILE	READ	READ A LINE FROM FILER OR A DEVICE
ALOMEM	ALOMEM	ALOCATE N CONTIGUOUS BLOCKS OF MEMORY
ALOMEM	RELMEM	RELEASE N CONTIGUOUS BLOCKS OF MEMORY
SAVREC	SAVREC	ASSOCIATE A RECORD # WITH A LINE # SO CAN FIND LINES FAST
SAVREC	FASTPG	GUESS PAGE NUMBER THAT LINE NUMBER IS ON AND GET PAGE.
SAVREC	GENKEY	CALCULATE KEY FOR LINE NUMBER
SAVREC	FRGREC	DISASSOCIATE RECORD NUMBER FROM LINE NUMBER
SEARCH	SEARCH	SEARCH A TABLE OF VARIABLE LENGTH TOKENS
STAT	STAT	SAVE STATISTICS IN TASK COMMON
STAT	STATIN	START OF STATISTICAL INCREMENT
STAT	STATIE	END OF STATISTICAL INCREMENT
STATD	STAT	DUMMY ROUTINE TO TURN OFF STATS
STATD	STATIN	DUMMY ROUTINE TO TURN OFF STATS
STATD	STATIE	DUMMY ROUTINE FO TURN OFF STATS
TERROR	TERROR	LOG ERROR MESSAGE N TO THE USER
TERROR	FILE	PLACE THE WORD 'FILE' AFTER R3
TERROR	FILL	INSERT MESSAGE N AFTER R3
TERROR	POINT	PLACE AN ↑ TO TOKEN DELIMITER WHERE WORD IS POINTING
TERROR	LOG	LOG MESSAGE TO LU 6
WORD	WORD	DECODE TOKENS OFF AN ASCII LINE

APPENDIX C: EDITOR'S MODULES AND PROCEDURES ALPHABETIZED BY PROCEDURE

MODULE	PROCEDURE	PURPOSE
EDIT	AL	INSERT NEXT LINES AFTER FIRST LINE IN RANGE
ALOMEM	ALOMEM	ALOCATE N CONTIGUOUS BLOCKS OF MEMORY
EDIT	ALRANG	DO FUNCTION FOR EACH LINE IN RANGE
EDIT	ALRANG1	DO FUNCTION FOR EACH LINE IN RANGE
EDIT	APTCK	TURN ON/OFF OPTION APTCKF FOR APT SYNTAX CHECKER
EDIT	APTSYN	TURN ON/OFF OPTION APTSYNF FOR APT SYNOMYNS
EDIT	ASM	TURN ON/OFF OPTION ASMF FOR ASSEMBLY LANGUAGE FORMAT
EDIT	ASORNG	EVALUATE ASSOCIATIVE RANGE
OFILE	ASSILU	ASSIGN LU TO PHYSICAL DEVICE NUMBER
EDITF	BATCH	HANDLE BATCH COMMAND
EDIT	BL	INSERT NEXT LINE BEFORE FIRST LINE IN RANGE
EDIT	C	CREATE A WORKING FILE
EDIT	CA	COPY AFTER
EDITCA	CAFILE	COPY ALL LINES IN RANGE FROM FILEID AFTER CAL.
EDITCA	CAWORK	MOVE OR COPY LINES IN RANGE IN WORKING FILE AFTER CAL.
OFILE	CFILE	CLOSE FILER FILE
CEDIT	CHKASRNG	CHECK LINE, BP, TO SEE IF IT'S IN ASSOCIATIVE RANGE
CEDIT	CHKEXRNG	CHECK TO SEE IF LINE POINTED TO BY BP IS IN EXPLICIT RANGE
EDIT	CLEANCMD	MAKE SURE NO MORE DATA IN INPUT COMMAND LINE
EDIT	CLEANTOK	MAKE SURE WE'RE POINTING TO THE NEXT TOKEN
EDIT	COLIMITS	EVALUATE COLUMN LIMITS
EDIT	COLUMN	EVALUATE CURRENT TOKEN FOR A COLUMN NUMBER
FINDL	COMPRESS	COMPRESS AVAILABLE BUFFERS INTO ONE BUFFER
FINDL	COPY	COPY A LINE FRAGMENT
OFILE	CRFILE	CREATE A FILE
EDITF	CWORK	CREATE WORKING FILE & SETUP THE FIRST BUFFER
EDIT	DAL	DELETE ALL LINES IN RANGE
EDIT	DECNUM	SET L.DEC IN DATA STRUCTURE IF INTEGER BETWEEN 0 AND 99
EVFILE	DEFILE	SET DEFAULTS IN FILEID BLOCK
FINDL	DELETE	DELETE LINE POINTED TO BY BP
EDIT	DFL	DELETE FIRST LINE IN RANGE
EDIT	DOCMD	DO COMMANDS IN INPUT BUFFER UNTIL ERROR OR END OF LINE
EDITCA	DOMACA	DO MOVE AFTER OR COPY AFTER
FINDL	DWORK	DESTROY WORKING FILE AND RELEASE ALL FREE RECORDS
EDIT	EDIT	PERFORM EDITING FUNCTION
EDIT	END	TERMINATE EDIT [DO NF, FF END EOJ]
EDITCA	ENDPUT	FINSH PUTEMP AND SETUP FOR GETEMP
FINDL	ERRHAN	IF I/O ERROR THEN LOGIT SETTING ERROR
EVFILE	EVFILE	EVALUATE A FILEID IN ASCII AND PUT RESULTS IN A FILEID BLOCK
EDIT	EXPRNG	EVALUATE EXPLICIT RANGE OF FORM LINE#,LINE#
SAVREC	FASTPG	GUESS PAGE NUMBER THAT LINE NUMBER IS ON AND GET PAGE.
EDITCA	FNDCAL	SET BP TO LINE CAL. OR LINE BEFORE IF NONE
EDIT	FSTRNG	FIND FIRST LINE IN RANGE
EDIT	FF	FORGET FILE BY DESTROYING THE WORKING FILE
TERROR	FILE	PLACE THE WORD 'FILE' AFTER R3
TERROR	FILL	INSERT MESSAGE N AFTER R3
EDITF	FILWRK	FILL WORKING FILE WITH DATA FROM CURRENT FILE
FINDL	FINDL	GIVEN THE LINE #, SET BP TO POINT TO THE LINE BUFFER OF THE LINE
CEDIT	FNDPAT	FIND PATTERN OR SYMBOL IN A LINE

LIST	FORMAT	FORMAT A LINE IN ASSEMBLY LANGUAGE FORMAT
FINDL	FRAGCOPY	COPY A LINE FRAGMENT
SAVREC	FRGREC	DISASSOCIATE RECORD NUMBER FROM LINE NUMBER
SAVREC	GENKEY	CALCULATE KEY FOR LINE NUMBER
FINDL	GETBUF	GET A BUFFER OF LENGTH IN R14
EDITF	GETC	GET A LINE FROM COPY AFTER FID
EDITCA	GETCA	GET NEXT LINE IN RANGE FROM FILEID
EDIT	GETCMD	GET COMMAND FROM INPUT DEVICE, PROMPT =
EDITCA	GETEMP	GET NEXT LINE FROM FIFO
EDITF	GETF	GET NEXT LINE FROM CURRENT FILE
EDIT	GETINS	GET LINE FOR INSERT, PROMP = *
FINDL	GETREC	GET AN AVAILABLE RECORD
EDIT	GETTOKEN	GET NEXT TOKEN ON COMMAND LINE
FINDL	GIVREC	GIVE UP A RECORD
EDITF	GLINES	GET LINES FOR INSERTIONS WATCHING FOR LEOMS
EDITF	ILC	INSERT LINES AFTER LINE POINTED TO BY BP
EDIT	INCR	HANDLE INCREMENT LINE NUMBERS OF FORM +1, -1 ETC.
EDIT	INITIAL	CREATES AND INITIALIZES THE DATA STRUCTURE OFF R12
FINDL	INSERB	INSERT LINE L. BEFORE LINE POINTED TO BY BP
FINDL	INSERT	INSERT LINE L. AFTER LINE POINTED TO BY BP
EDIT	LA	LIST ALL LINES IN RANGE
EVFILE	LATTR	LOG FILE SECURITY IF > UNCLASSIFIED
EDIT	LF	LIST THE FIRST LINE IN RANGE
LIST	LINENA	CONVERT LINE NUMBER TO ASCII
EDIT	LINENUM	EVALUATE CURRENT TOKEN FOR LINE NUMBER
LIST	LIST	LIST LINE POINTED TO BY BP IN UNFORMATTED OR ASSEMBLY LANGUAGE FORMAT WITH LINE NUMBERS
TERROR	LOG	LOG MESSAGE TO LU 6
LOGIT	LOGIT	LOG MESSAGE TO LU 6
EDIT	LOGON	RESET OR SET CERF TO LOG ALL EDITOR INTERACTIONS
LOGIT	LSTIT	LOG MESSAGE TO LU 6
FINDL	LSTREC	READ IN RECORD BEFORE F.CUREC
EDIT	MA	MOVE AFTER
EDIT	NXTNUM	PUT NEXT LINE NUMBER IN WORKING FILE TO LE.
FINDL	MOVE	MOVE UP/DOWN R14 LINES, SET BP TO RESULTANT LINE
FINDL	NEXTREC	READ IN NEXT RECORD UPDATING IF NECESSARY
EDIT	NF	WRITE OUT WORKING FILE TO FILEID
EDIT	NUMCMD	HANDLE NUMBER INPUT AS A COMMAND
CEDIT	NXTRNG	SET BP & CURRENT LINE TO NEXT LINE IN RANGE
EDIT	O	OPEN FILE TO A WORKING FILE
OFILE	OAFILE	OPEN OR ALOCATE A FILER FILE
EDIT	OFF	TURN OFF ALL OPTIONS
OFILE	OFILE	OPEN A FILER FILE
EDIT	ONOFF	DECODE ON/OFF ON COMMAND LINE, SET R14 TO ZERO IF ON ELSE 1 IF OFF
EDIT	P	LIST A PAGE OF LINE STARTING WITH FIRST LINE IN RANGE
FINDL	PATCOPY	COPY PATTERN INTO LINE NL
EDIT	PATDELIM	DETERMINE IF A CHARACTER IS A PATTERN OR SYMBOL DELIMITER
EDIT	PATTERN	EVALUATE PATTERN ON COMMAND LINE
TERROR	POINT	PLACE AN ↑ TO TOKEN DELIMITER WHERE WORD IS POINTING
LIST	PUTCHAR	PUT A CHARACTER IN A BUFFER UNTIL END OF LINE
EDITCA	PUTEMP	PUT LINE AWAY IN FIFO THAT CAN EXTEND TO DISK
EDIT	Q	RELIST A PAGE OF LINES STARTING WITH P.INT
EDIT	RAP	REPLACE PATTERNS IN ALL LINES IN RANGE
OFILE	READ	READ A LINE FROM FILER OR A DEVICE
FINDL	READER	READ RECORD F.CUREC
FINDL	RELBUF	RELEASE BUFFER OBTAINED BY GETBUF
OFILE	RELAMEM	RELEASE ANY ALOCATED MEMORY FOR FILER

OFILE	RELEASE	RELEASE DEVICE
ALOMEM	RELMEM	RELEASE N CONTIGUOUS BLOCKS OF MEMORY
FINDL	RELPG	RELEASE CURRENT PAGE IF EMPTY
EDITCA	RELTEMP	RELEASE FIFO
EDITF	RENUMBER	RENUMBER LINES IF NECESSARY
FINDL	REPAT	REPLACE PATTERN IN LINE BP
EDIT	REPRANGE	EVALUATE REPLACEMENT RANGE
EDITF	RSTWRD	RESTORE WORD BLOCK TO STATE WHEN SAVRWD WAS CALLED
EDIT	RFP	REPLACE PATTERNS IN FIRST LINE IN RANGE
CEDIT	RNG	SET BP & CURRENT LINE TO NEXT LINE IN RANGE
EDIT	RNUM	EVALUATE CURRENT TOKEN FOR LINE NUMBER, TOKEN CAN NOT BE A COMMAND
EVFILE	SATTR	SET CLASSIFICATION AND PROPERTY TO MAX IN FID
EDITF	SAVRWD	SAVE STATE OF WORD AFTER RIZ
SAVREC	SAVREC	ASSOCIATE A RECORD # WITH A LINE # SO CAN FIND LINES FAST
SEARCH	SEARCH	SEARCH A TABLE OF VARIABLE LENGTH TOKENS
EDIT	SETCMDUP	COMMON SETUPS FOR MOST COMMANDS USING RANGE
EDITCA	SETEMP	SETUP FOR PUTEMP
EDITCA	SETGETCA	SETUP FOR GETCA, FIND FIRST LINE IN RANGE IN FILE
EDIT	SETTOKEN	SET UP DATA STRUCTURE TO DECODE TOKENS ON COMMAND LINE
OFILE	SETUPO	COMMON SETUP ROUTINE FOR OFILE AND OAFILE
EDIT	SETUPOC	COMMON SETUP ROUTINE FOR O AND C
FINDL	SPLITPG	SPLIT PAGE AFTER LINE BP INTO 2 PAGES
STAT	STAT	SAVE STATISTICS IN TASK COMMON
STATD	STAT	DUMMY ROUTINE TO TURN OFF STATS
STAT	STATIE	END OF STATISTICAL INCREMENT
STATD	STATIE	DUMMY ROUTINE FO TURN OFF STATS
STAT	STATIN	START OF STATISTICAL INCREMENT
STATD	STATIN	DUMMY ROUTINE TO TURN OFF STATS
FINDL	SUBSPLIT	SPLIT A BLOCK OF LENGTH+N INTO 2 BLOCKS OF LENGTH AND N
CEDIT	SYMBOL	DETERMINE IF A CHARACTER IS A SYMBOL
EDIT	TC	TURN ON/OFF OPTION TCF FOR TYPE CHANGES OPTION
EDIT	TERORS	WRITE OUT POINTER TO CURRENT TOKEN AND ERROR MESSAGE
EDIT	TERR	LOG ERROR MESSAGE
TERROR	TERROR	LOG ERROR MESSAGE N TO THE USER
WORD	WORD	DECODE TOKENS OFF AN ASCII LINE
OFILE	WRITE	WRITE OUT A LINE TO FILER OR A DEVICE
FINDL	WRITER	WRITE OUT F.CUREC

APPENDIX D: EDITOR'S MODULES AND PROCEDURES ALPHABETIZED BY MODULE

MODULE	PROCEDURE	PURPOSE
ALOMEM	ALOMEM	ALOCATE N CONTIGUOUS BLOCKS OF MEMORY
ALOMEM	RELMEM	RELEASE N CONTIGUOUS BLOCKS OF MEMORY
CEDIT	NXTRNG	SET BP & CURRENT LINE TO NEXT LINE IN RANGE
CEDIT	RNG	SET BP & CURRENT LINE TO NEXT LINE IN RANGE
CEDIT	CHKEXRNG	CHECK TO SEE IF LINE POINTED TO BY BP IS IN EXPLICIT RANGE
CEDIT	CHKASRNG	CHECK LINE, BP, TO SEE IF IT'S IN ASSOCIATIVE RANGE
CEDIT	FNDPAT	FIND PATTERN OR SYMBOL IN A LINE
CEDIT	SYMBOL	DETERMINE IF A CHARACTER IS A SYMBOL
EDIT	EDIT	PERFORM EDITING FUNCTION
EDIT	INITIAL	CREATES AND INITIALIZES THE DATA STRUCTURE OFF R12
EDIT	GETCMD	CET COMMAND FROM INPUT DEVICE, PROMPT = .
EDIT	GETINS	GET LINE FOR INSERT, PROMP = *
EDIT	DOCMD	DO COMMANDS IN INPUT BUFFER UNTIL ERROR OR END OF LINE
EDIT	LOGON	RESET OR SET CERF TO LOG ALL EDITOR INTERACTIONS
EDIT	OFF	TURN OFF ALL OPTIONS
EDIT	APTSYN	TURN ON/OFF OPTION APTSYNF FOR APT SYNOMYNS
EDIT	APTCK	TURN ON/OFF OPTION APTCKF FOR APT SYNTAX CHECKER
EDIT	ASM	TURN ON/OFF OPTION ASMF FOR ASSEMBLY LANGUAGE FORMAT
EDIT	TC	TURN ON/OFF OPTION TCF FOR TYPE CHANGES OPTION
EDIT	ONOFF	DECODE ON/OFF ON COMMAND LINE, SET R14 TO ZERO IF ON ELSE 1 IF OFF
EDIT	SETTOKEN	SET UP DATA STRUCTURE TO DECODE TOKENS ON COMMAND LINE
EDIT	GETTOKEN	GET NEXT TOKEN ON COMMAND LINE
EDIT	END	TERMINATE EDIT [DO NF, FF END EOJ]
EDIT	C	CREATE A WORKING FILE
EDIT	O	OPEN FILE TO A WORKING FILE
EDIT	SETUPOC	COMMON SETUP ROUTINE FOR O AND C
EDIT	NF	WRITE OUT WORKING FILE TO FILEID
EDIT	FF	FORGET FILE BY DESTROYING THE WORKING FILE
EDIT	NUMCMD	HANDLE NUMBER INPUT AS A COMMAND
EDIT	BL	INSERT NEXT LINE BEFORE FIRST LINE IN RANGE
EDIT	AL	INSERT NEXT LINES AFTER FIRST LINE IN RANGE
EDIT	LF	LIST THE FIRST LINE IN RANGE
EDIT	LA	LIST ALL LINES IN RANGE
EDIT	DFL	DELETE FIRST LINE IN RANGE
EDIT	DAL	DELETE ALL LINES IN RANGE
EDIT	P	LIST A PAGE OF LINE STARTING WITH FIRST LINE IN RANGE
EDIT	Q	RELIST A PAGE OF LINES STARTING WITH P.INT
EDIT	RFP	REPLACE PATTERNS IN FIRST LINE IN RANGE
EDIT	RAP	REPLACE PATTERNS IN ALL LINES IN RANGE
EDIT	CA	COPY AFTER
EDIT	MA	MOVE AFTER
EDIT	NXTNUM	PUT NEXT LINE NUMBER IN WORKING FILE TO LE.
EDIT	SETCMDUP	COMMON SETUPS FOR MOST COMMANDS USING RANGE
EDIT	FSTRNG	FIND FIRST LINE IN RANGE
EDIT	EXPRNG	EVALUATE EXPLICIT RANGE OF FORM LINE#.LINE#
EDIT	ASORNG	EVALUATE ASSOCIATIVE RANGE
EDIT	REPRANGE	EVALUATE REPLACEMENT RANGE
EDIT	PATTERN	EVALUATE PATTERN ON COMMAND LINE
EDIT	PATDELIM	DETERMINE IF A CHARACTER IS A PATTERN OR SYMBOL DELIMITER

```
EDIT    COLIMITS  EVALUATE COLUMN LIMITS
EDIT    COLUMN    EVALUATE CURRENT TOKEN FOR A COLUMN NUMBER
EDIT    ALRANG    DO FUNCTION FOR EACH LINE IN RANGE
EDIT    ALRANG1   DO FUNCTION FOR EACH LINE IN RANGE
EDIT    LINENUM   EVALUATE CURRENT TOKEN FOR LINE NUMBER
EDIT    RNUM      EVALUATE CURRENT TOKEN FOR LINE NUMBER, TOKEN CAN NOT BE A COMMAND
EDIT    DECNUM    SET L.DEC IN DATA STRUCTURE IF INTEGER BETWEEN 0 AND 99
EDIT    INCR      HANDLE INCREMENT LINE NUMBERS OF FORM +1, -1 ETC.
EDIT    TERR      LOG ERROR MESSAGE
EDIT    TERORS    WRITE OUT POINTER TO CURRENT TOKEN AND ERROR MESSAGE
EDIT    CLEANTOK  MAKE SURE WE'RE POINTING TO THE NEXT TOKEN
EDIT    CLEANCMD  MAKE SURE NO MORE DATA IN INPUT COMMAND LINE
EDITCA  DOMACA    DO MOVE AFTER OR COPY AFTER
EDITCA  CAFILE    COPY ALL LINES IN RANGE FROM FILEID AFTER CAL.
EDITCA  FNDCAL    SET BP TO LINE CAL. OR LINE BEFORE IF NONE
EDITCA  SETGETCA  SETUP FOR GETCA, FIND FIRST LINE IN RANGE IN FILE
EDITCA  GETCA     GET NEXT LINE IN RANGE FROM FILEID
EDITCA  CAWORK    MOVE OR COPY LINES IN RANGE IN WORKING FILE AFTER CAL.
EDITCA  SETEMP    SETUP FOR PUTEMP
EDITCA  PUTEMP    PUT LINE AWAY IN FIFO THAT CAN EXTEND TO DISK
EDITCA  ENDPUT    FINSH PUTEMP AND SETUP FOR GETEMP
EDITCA  GETEMP    GET NEXT LINE FROM FIFO
EDITCA  RELTEMP   RELEASE FIFO
EDITF   BATCH     HANDLE BATCH COMMAND
EDITF   FILWRK    FILL WORKING FILE WITH DATA FROM CURRENT FILE
EDITF   SAVRWD    SAVE STATE OF WORD AFTER R12
EDITF   RSTWRD    RESTORE WORD BLOCK TO STATE WHEN SAVRWD WAS CALLED
EDITF   GETC      GET A LINE FROM COPY AFTER FID
EDITF   GETF      GET NEXT LINE FROM CURRENT FILE
EDITF   CWORK     CREATE WORKING FILE & SETUP THE FIRST BUFFER
EDITF   ILC       INSERT LINES AFTER LINE POINTED TO BY BP
EDITF   RENUMBER  RENUMBER LINES IF NECESSARY
EDITF   GLINES    GET LINES FOR INSERTIONS WATCHING FOR LEOMS
EVFILE  EVFILE    EVALUATE A FILEID IN ASCII AND PUT RESULTS IN A FILEID BLOCK
EVFILE  DEFILE    SET DEFAULTS IN FILEID BLOCK
EVFILE  SATTR     SET CLASSIFICATION AND PROPERTY TO MAX IN FID
EVFILE  LATTR     LOG FILE SECURITY IF > UNCLASSIFIED
FINDL   FINDL     GIVEN THE LINE #, SET BP TO POINT TO THE LINE BUFFER OF THE LINE
FINDL   MOVE      MOVE UP/DOWN R14 LINES, SET BP TO RESULTANT LINE
FINDL   INSERB    INSERT LINE L. BEFORE LINE POINTED TO BY BP
FINDL   INSERT    INSERT LINE L. AFTER LINE POINTED TO BY BP
FINDL   DELETE    DELETE LINE POINTED TO BY BP
FINDL   REPAT     REPLACE PATTERN IN LINE BP
FINDL   FRAGCOPY  COPY A LINE FRAGMENT
FINDL   COPY      COPY A LINE FRAGMENT
FINDL   PATCOPY   COPY PATTERN INTO LINE NL
FINDL   RELBUF    RELEASE BUFFER OBTAINED BY GETBUF
FINDL   GETBUF    GET A BUFFER OF LENGTH IN R14
FINDL   SUBSPLIT  SPLIT A BLOCK OF LENGTH+N INTO 2 BLOCKS OF LENGTH AND N
FINDL   COMPRESS  COMPRESS AVAILABLE BUFFERS INTO ONE BUFFER
FINDL   SPLITPG   SPLIT PAGE AFTER LINE BP INTO 2 PAGES
FINDL   RELPG     RELEASE CURRENT PAGE IF EMPTY
FINDL   NEXTREC   READ IN NEXT RECORD UPDATING IF NECESSARY
FINDL   READER    READ RECORD F.CUREC
FINDL   WRITER    WRITE OUT F.CUREC
FINDL   GETREC    GET AN AVAILABLE RECORD
```

FINDL	GIVREC	GIVE UP A RECORD
FINDL	DWORK	DESTROY WORKING FILE AND RELEASE ALL FREE RECORDS
FINDL	LSTREC	READ IN RECORD BEFORE F.CUREC
FINDL	ERRHAN	IF I/O ERROR THEN LOGIT SETTING ERROR
LIST	LIST	LIST LINE POINTED TO BY BP IN UNFORMATTED OR ASSEMBLY LANGUAGE FORMAT WITH LINE NUMBERS
LIST	LINENA	CONVERT LINE NUMBER TO ASCII
LIST	FORMAT	FORMAT A LINE IN ASSEMBLY LANGUAGE FORMAT
LIST	PUTCHAR	PUT A CHARACTER IN A BUFFER UNTIL END OF LINE
LOGIT	LOGIT	LOG MESSAGE TO LU 6
LOGIT	LSTIT	LOG MESSAGE TO LU 6
OFILE	OFILE	OPEN A FILER FILE
OFILE	CRFILE	CREATE A FILE
OFILE	OAFILE	OPEN OR ALOCATE A FILER FILE
OFILE	SETUPO	COMMON SETUP ROUTINE FOR OFILE, CRFILE AND OAFILE
OFILE	ASSILU	ASSIGN LU TO PHYSICAL DEVICE NUMBER
OFILE	CFILE	CLOSE FILER FILE
OFILE	RELEASE	RELEASE DEVICE
OFILE	RELAMEM	RELEASE ANY ALOCATED MEMORY FOR FILER
OFILE	WRITE	WRITE OUT A LINE TO FILER OR A DEVICE
OFILE	READ	READ A LINE FROM FILER OR A DEVICE
SAVREC	SAVREC	ASSOCIATE A RECORD # WITH A LINE # SO CAN FIND LINES FAST
SAVREC	FASTPG	GUESS PAGE NUMBER THAT LINE NUMBER IS ON AND GET PAGE.
SAVREC	GENKEY	CALCULATE KEY FOR LINE NUMBER
SAVREC	FRGREC	DISASSOCIATE RECORD NUMBER FROM LINE NUMBER
SEARCH	SEARCH	SEARCH A TABLE OF VARIABLE LENGTH TOKENS
STAT	STAT	SAVE STATISTICS IN TASK COMMON
STAT	STATIN	START OF STATISTICAL INCREMENT
STAT	STATIE	END OF STATISTICAL INCREMENT
STATD	STAT	DUMMY ROUTINE TO TURN OFF STATS
STATD	STATIN	DUMMY ROUTINE TO TURN OFF STATS
STATD	STATIE	DUMMY ROUTINE FO TURN OFF STATS
TERROR	TERROR	LOG ERROR MESSAGE N TO THE USER
TERROR	FILE	PLACE THE WORD 'FILE' AFTER R3
TERROR	FILL	INSERT MESSAGE N AFTER R3
TERROR	POINT	PLACE AN ↑ TO TOKEN DELIMITER WHERE WORD IS POINTING
TERROR	LOG	LOG MESSAGE TO LU 6
WORD	WORD	DECODE TOKENS OFF AN ASCII LINE

PL/crs

Technical Information Department
LAWRENCE LIVERMORE LABORATORY
University of California | Livermore, California | 94550