HDE, INCORPORATED

USER'S MANUAL

HDE TEXT EDITOR

[TED]

AUGUST 1980

HUDSON DIGITAL ELECTRONICS, INC.
P.O. BOX 120
ALLAMUCHY, N.J. 07820

REVISION RECORD

REVISION	DATE	PAGES	COMMENTS
REV 0	AUGUST 1980	ALL	INITIAL ISSUE

CONTENTS

INTRODUCTION

The HDE Text Editor is an advanced line oriented editor particularly suited for source program development and general purpose applications.

TED files are constructed of lines of data which may be up to 128 characters long (72 characters for cassette versions). Each line is preceded with a line number (1 to 9999), thereby facilitating individual line retrieval and modification as necessary.

The HDE Text Editor (TED) features:

* Line-numbered text entry and editing
* A powerful command structure
* Capability for user defined commands
* Complete compatibility with KIM input/output routines
* Multiple resident files
* Address independence

LIMITED WARRANTY

The HDE Text Editor (TED) has been carefully constructed to assure correct operation when applied as described within the limitations identified in this manual. The nature of computer program development prohibits any claim that the TED is error free. Therefore, this program is distributed with a "Limited Warranty".

* Any errors discovered will be corrected in the first 90 days.
* Catastrophic errors will be corrected by automatic mailing to all registered owners.
* Minor errors will be corrected by request.

In any case, this warranty is limited to replacement of the disk or cassette and/or manual. No liability for consequential damages can be assumed or is implied.

If you believe you have discovered an error, document the conditions concerning the error, what you think is wrong, the version number and serial number of your copy and mail to:

HDE, inc.
P.O. Box 120
Allamuchy, N.J. 07820

Phone: (201) 362-6574

SYSTEM REQUIREMENTS

As a minimum, the HDE TED requires a 6502 based microcomputer with 4K of memory, a cassette and/or HDE disk system and a suitable terminal. 'Working memory' should be at least 4K with 12K recommended.

LOADING AND INITIALIZATION

A correct load and initialization is indicated by the issue of a prompt (:) which signifies that the editor is ready for a command. Only a limited subset of the total commands available may be used at this point; all other commands will result in a 'FILE?' error denoting that a file must be established (see FIL).

FILE CONCEPT

A TED file is comprised of one or more records (lines). Each record is defined as a line of text, preceded with a line number and terminated with an ASCII carriage return (hex 0D) character. Files are address independent and may be placed anywhere in working memory. Several files may be resident concurrently (provided that they do not overlap).

Files are opened through use of the 'FIL' command, either to create a new file or to transfer between files resident in memory. A trap has been included to inhibit the use of commands until a file has been opened.

A file may be of any length up to 32K characters, limited by contiguous memory. The prompt 'FILE?' is a warning that indicates that a file has exceeded available memory. If you are entering a file which may approach the memory capacity of your system, the '?' command should be used to monitor the file end address.

EDITOR OPERATIONS

Text is entered into a current file [the file you are working with] by typing in a line number followed by a text string. Line numbers may range from 1 to 9999. TED will enter leading zeros automatically. Any character string entered without a preceding line number is assumed to be a command.

Lines may be added, inserted, deleted or modified according to the needs of the user. TED does not use a fixed line length. The storage required for each line is the data length plus three characters (two are required for the line number which is represented in Binary Coded Decimal and the third marks the end of the data string]. The end of the file is continually adjusted for lines entered or deleted.

Errors made in the current line [the line you are entering] may be corrected by using the backspace function available on your input keyboard. Two ASCII hex codes are recognized as a backspace command and either of these may be used with the same result. These codes are '5F' which is generated by depressing the 'back-arrow' (<-) or 'shift O' and the backspace, hex '08'. Each use of the backspace function will delete one character by 'backing up' the line, and may be used as many times as required to delete characters in that line. If desired, the current line entry may be cancelled by using the cancel [or 'Control X') function on your keyboard [hex 18]. This causes TED to ignore the entry, issue a carriage return - line feed and display a prompt (:). 'Control A' may also be used with the same result.

To delete a line in the file, enter the line number of the line you wish to delete immediately followed by the return. To delete a contiguous block [from 1000 to 1510, for example] enter the first line number of the block, a comma [,), and the ending line number. Upon entry of the return, the entire block bounded by the line numbers entered [inclusive] will be deleted.

Every line of text entered into a file must have a decimal line number between 1 and 9999. If the same number is used twice, the current line will replace the previous line with the same line number. To add a line between two existing lines use a line number between the two existing numbers. If a line is to be inserted between two lines with adjacent line numbers, the file numbers must be resequenced with the 'RES' command. As an alternative, one of the existing lines may be moved [assigned a new line number] using the 'MOV' command. Each line typed will be inserted into the file in line number sequence. If your input device is so equipped, a warning sounds when five or less characters can be entered. This warning [bell or tone] will continue for each character until the line length is exceeded [five characters]. At this point three functions are available: return, which enters the truncated line to the file; cancel, which deletes the line entry; and [on terminals so equipped] backspace. The backspace may be used to 'back up' the line until 1] the warning sounds again, in-

dicating the legal limits of the line have been reentered and 2) a convenient end point for the line is reached. At this time, the line may be entered into the file in a normal manner.

There are two restrictions which must be considered in the construction of a file. First, if the line number maximum of 9999 is exceeded, certain command routines will not operate properly and subsequent line entry will overwrite existing lines. Second, a file is restricted by the amount of memory available for its' storage. If a file is permitted to exceed available memory, the data in that portion of the file is lost.

There are two commands available for listing a file, 'LIS' and 'PRI'. Both commands have the same options. 'LIS' displays the file contents with each line number. 'PRI' does the same but without line numbers. To display the entire file, the appropriate command (LIS or PRI) is entered immediatly following the prompt. The TED will now start displaying the contents of the file starting with the first line. To stop the listing, depress the break key on your keyboard at any point. Other options include display of a single line, display between two lines, display of a preset number of lines with the option to continue to the next page, etc. In summary, a wide variety of 'LIS' and 'PRI' options exist to meet a number of display needs.

EDITOR EXPANSION (CASSETTE VERSIONS ONLY)

If 'X' is entered as the first character of a line, TED will jump to an exit location. This location, hex 2FF6, is set for a default entry back to the TED at location hex 2100. This feature may be used for a controlled exit from TED to any user program or routine by changing the default address found at hex 2FF7 and 2FF8, to whatever other address is desired.

TED ENTRY

In cassette versions, TED is entered at the starting address of the object program (hex 2100, E100 or as otherwise ordered from HDE). A Warm start reentry to TED from the monitor may be accomplished by setting the program counter to hex 0000 and entering 'G' on your keyboard. In disk versions, TED is entered automatically when it is loaded by the operating system.

In all versions, a return from a user installed function or program should be made via a BREAK instruction (BRK - hex 00) through the IRQ vector. This will insure proper reentry and the initialization of key flags and locations.

Entry to TED at any other location is not recommended.

ZERO PAGE USAGE

TED uses all zero page locations from hex 00E0 to 00EF (00A0 to 00AF in the SYM). Also, in cassette versions, TED sets a jump to the warm start in locations 0000 to 0002.

EXTERNAL JUMP TABLE

The HDE TED uses an external jump table to allow user changes to certain key routines. This table is as follows:

	ADDRESS							
	CASSETTE	DISK SYSTEMS						
		KIM		TIM		SYM		KIMSI	
		5"	8"	5"	8"	5"	8"	5"	8"
C-Rtn/Line feed	2FE8	FB9F	FB9A	F39C	F397	7B95	7B90	DB9F	DB9A
Monitor entry	2FF1	FBA6	FBA1	F3A3	F39E	7B9C	7B97	DBA6	DBA1
Get character	2FE2	FB99	FB94	F396	F391	7B8F*	7B8A*	DB99	DB94
Output character	2FE5	FB9C	FB97	F399	F394	7B92*	7B8D*	DB9C	DB97
Cassette load	2FDC	Cassette versions only							
Cassette save	2FDF	Cassette versions only							
User Command	2FF7	Cassette versions only							
KIM Pack	2FEB	KIM cassette version only							
KIM Pointer	2FEE	KIM cassette version only							

* In SYM versions, the internal GET CHARACTER and OUTPUT CHARACTER routines are vectored through INVEC ($A660) and OUTVEC ($A663). The user may choose to modify these vector locations rather than use the external jump.

MODIFICATIONS TO TED

The user may modify TED using the jump table identified above. The following paragraphs describe the operation of each routine that affects, or is affected by, the function jumps identified in the table.

Cassette load - After entry of the 'LOD' command, the operand arguments are determined and the tape file ID number and starting address, if appropriate, are set for the monitor cassette load routine. The program then jumps to "cassette load" in the jump table. This address is set to routines in TED which accomplish the tape load using the monitor. To use a different tape load routine, enter the starting address of the routine at the jump table location. Examine the monitor cassette parameter locations to obtain the arguments which were defined in the 'LOD' command operand. Return to TED via the jump entry at location hex 0000, after placing the starting and ending address of the program just loaded to memory into the corresponding locations used by the monitor cassette load routine.

Cassette save - The TED routine interprets the 'SAV' argument list, placing the appropriate values into the monitor cassette parameter table. A jump is then executed to the 'Cassette save' location in the jump table. Return to TED is made via the jump located at hex 0000.

Get character, Output character, CRLF - TED accesses the monitor routines both internally and through the jump table. This technique permits the addition of peripheral devices, such as a printer, without the need for internal changes and allows program control of device selection. User access to the table I/O jumps is controlled by a flag which is set to hex 00 upon initial entry. This 00 is a default to the internal I/O accesses. The values of this flag are as follows

```
BIT   7 - 0
      6 - 0
      5 - 0
      4 - 0 = internal Get character
        - 1 = external Get character
      3 - 0
      2 - 0
      1 - 0 = internal Out char/CRLF
        - 1 = internal and external Out char/CRLF
      0 - 0 = internal Out char/CRLF
        - 1 = external Out char/CRLF
```

For example, to set output to a printer through the external table,

1) Set the addresses for the output character and CRLF routines supporting the printer into the appropriate external table addresses.

2) Set the input/output flag to hex 01. In cassette systems, the location is hex 20FA; in disk systems, KIM is hex F272, SYM is hex 7272, KIMSI is hex D272 and TIM is hex F272.

3) Direct the action (i.e. 'LIS'). All output is now through the external jump table. (The 'Break' function is inoperative with the external table.)

Effective use of the external I/O jumps is best accomplished under program control. TED supports this by allowing user selection of an output device by allowing a '#' sign to preceed any command. For example, to list a portion of a file to a printer using the external table, the following command would be used:

```
#LIS 1000,2000
```

The '#' sign signals an exit to the location labeled 'Printer init.' in the jump table. At this location, the user should enter the address of the initialization routine for the printer in use. The initialization should include steps to set the Input/Output flag to the code desired, as well as the appropriate addresses in the jump table (if these had not been previously set). Return to TED with an 'RTS' (return from subroutine). When the command is entered, all output will be to the addresses set in the external table. Upon completion of the command function, the I/O flag is is reset to 00 returning I/O to the internal TED routines.

KIM pack, KIM pointer - These are jumps to the KIM Pack and KIM pointer monitor routines.

Monitor entry - A jump to the monitor entry point.

User command - TED jumps to this location if an 'X' is entered as the first character of a command.

COMMAND STRUCTURE

Each TED command is based upon a three character mnemonic which attempts to describe, to some extent, the function involved. Commands must be entered as shown in each command description. Errors in the command name or format are answered with 'ERROR'. Commands may be entered in either upper or lower case characters.

There are several exceptions to the command format. Except as noted, the following directives must be entered as the first character following the prompt. If entered anywhere else on a line they are ignored as TED directives.

? Used to determine the size of a file. The information is presented as follows:

STARTING ADDRESS (HEX)	ENDING ADDRESS (HEX)	CHARACTERS IN FILE (HEX)	LINES IN (NOT IN FILE AIM-65) (DECIMAL)

< Used to change the TED prompt character (:). Any ASCII character is acceptable. The TED prompt will be restored on reentry via the starting address or disk load.

^A 'Control A' is used to initiate the automatic line numbering feature. The default increment is 5. This may be altered with the 'RES' command. Automatic line numbering is terminated by entering a 'Control A'. 'Control A' is also used to cancel a line entry at any point in the same manner as 'Control X'.

. A period (.) is used as a semiautomatic method for calling the next line number. Both 'Control A' and '.' track the last number in the file and generate the next number according to the default value of five or the increment established with the 'RES' command.

^C Some versions of TED include the ability to recall the last command or data line entered through use of the 'Control C' character.

When entered as the first character of a command, this causes a jump to subroutine to location "Printer init." in the external jump table. This location may be used, at user option, to initialize a printer under program control.

ERROR NOTIFICATIONS

ERROR	The command interpreter does not recognize the entry as a legal command.
FILE?	A file has not been opened and an operation has been attempted with a command other than LOD or FIL.
	A file load or work in a file has resulted in an overflow from available memory.

COMMAND SUMMARY

ADR <n>	ADDRESS OF LINE n
APP <n>	APPEND STARTING WITH LINE n
BLM <n,m/x>	BLOCK MOVE LINES n THRU m TO x
DLY <c>	SET SCROLL DELAY TO c
EDT <n>	EDIT LINE n
END <n>	END FILE AFTER LINE n
ENT <name>	Same as SAV - SYM versions only
ENT <$hhhh$iiii=name>	Same as SAV - SYM versions only
FIL N<hhhh>	OPEN A FILE AT HEX ADDRESS hhhh.
FIL A<hhhh>	ENTER THE ACTIVE FILE AT HEX ADDRESS hhhh
JMP <hhhh>	JUMP TO HEX ADDRESS hhhh
JMP.<hhhh>	EXECUTE PROGRAM AT HEX ADDRESS hhhh
JOB#	EXECUTE THE FOLLOWING INSTRUCTIONS
JOB <name>	EXECUTE JOB FILE (name) - (DISK)
JOB <hhhh>	EXECUTE JOB FILE AT HEX hhhh
LIN <n>	SET LINE WIDTH TO n CHARACTERS PER LINE
LIS	LIST FILE STARTING WITH FIRST LINE
LIS <argument>,	LIST PER argument
LOC <string>	LOCATE ALL OCCURANCES OF (string)
LOD <name>	LOAD FILE (name)
LOD <$hhhh=name>	LOAD FILE (name) TO HEX ADDRESS hhhh
MOV <n/m>	MOVE LINE n TO LINE m
NOP	SCAN FOR NON PRINTING CHARACTERS
PAG <n>	SET LINES PER PAGE TO n
PRI	PRINT FILE STARTING WITH FIRST LINE
PRI <argument>	PRINT PER argument
RES <n>	RESEQUENCE LINE NUMBERS WITH n (1 TO 10)
SAV <name>	SAVE FILE (name)
SAV <$hhhh$iiii=name>	SAVE FROM hhhh TO iiii WITH (name)
SCR <string>	DELETE ALL LINES CONTAINING (string)
SET <string1><LF><string2>	SET (string1) TO BE (string2)

****
ADR <n> [ADDRESS OF LINE n]
****

This command displays the hex address of the first byte of the line number specified.

RESTRICTIONS: None

EXAMPLE: ADR 1450

DISCUSSION:

This command is used to identify the starting hex address of the specified line number. The address displayed is the memory location of the two high order line number digits in their packed decimal representation.

****
APP <n> [APPEND STARTING WITH LINE n]
****

This command facilitates the addition of comments to a source program listing. When used, the specified (or next) line is moved from the file to the text buffer area. Entries may then be made to the line. Use of the return key reenters the line to the file and moves the next numbered line to the buffer. The function is cancelled by use of the cancel function (control X).

The function is cancelled and a prompt is issued when the last line of the file is reentered to the file.

RESTRICTIONS: NONE

EXAMPLE: APP 1200

DISCUSSION:

This command allows the addition of comments to source program statements. The routines used for the original entry of a line are used for this function. Therefore, backspace and cancel are the same for this command as for the entry of a line. If the maximum line length is exceeded only the return, cancel and backspace keys will operate.

****
BLM <n,m/x> MOVE BLOCK BETWEEN LINES n,m TO x
****

The record block from line 'n' through line 'm' (inclusive) is moved to a new location defined by line number 'x'. The block will be inserted in the file at a location following the line defined by 'x'. Consequently, the move is non-destructive and no lines will be overwritten or lost. The line numbers of the file are automatically resequenced by 5 as a function of the move. Any number of lines may be moved either up or down in the file. A comprehensive series of checks is made on the command to insure illogical moves are not made. See below.

RESTRICTIONS:

Checks are made on the command argument to insure against moves which may disrupt or destroy the integrity of the file. No block move is permitted which will allow the block to wrap within itself. For example, 'BLM 1000,2000/1500' is not permitted since the target line number '1500' is inside the block.

The definition of the block line numbers must be from low to high. For example, 'BLM 2000,1000/500' will result in an 'ERROR' since the value '2000' is greater than '1000'.

EXAMPLE: BLM 1000,2000/3500

DISCUSSION:

The 'BLM' command works entirely within the boundries of the file to avoid errors which could occur with moves within files that are close to using all of available memory. Moves of large blocks may take up to several minutes.

****
DLY <c> SET SCROLL DELAY TO c.
****

DLY is used to set the scroll speed for the AIM-65 on-board display which is set to the medium rate when FODS is entered.

RESTRICTIONS:

The DLY command may only be invoked from TED.

EXAMPLES:

DLY S (set slow speed)
DLY M (set medium speed)
DLY F (set fast speed)

DISCUSSION: None

****
EDT <n> [EDIT LINE n]
****

This command is used to move a line from the file to the text buffer for editing purposes and implements the complete line editing capability. Execution of the command is indicated by display of the line specified. At this point the cursor, type head or carriage is positioned at the first character of the line. Character insertion, substitution and deletion actions may then be accomplished on the line. At the completion of the edit the line is echoed for operator check prior to entry into the file.

Character Substitution. The line is called to the text buffer area using the command EDT nnnn (where nnnn is the line number). The 'commercial at' (@ - 40 hex) is used to move the type head or cursor to a position under the character(s) to be substituted. When the head or cursor is directly below the character to be changed, the new character is typed. This substitution may be continued through the end of the line. If the substitution has been within the confines of the original line the edit function may be closed out with entry of a line feed character. This will cause the line, as edited, to be echoed for operator review. If the original line length is exceeded, the line must be closed out with a carriage return. If a return is used in lieu of a line feed, as previously discussed, the portion of the line following the carriage return will be dropped. In other words, it is not necessary to 'space' to the end of the line being edited with the '@' character once the desired edit has been accomplished. The edit may be closed out and the remaining portion of the line retained by using the line feed key. If it is desired to drop any portion of the line, the return key should be used.

Character Deletion. Actions for deletion are the same as for substitution. The 'back arrow' (<-), hex 5F, is used to delete a character and the line closes up for each character deleted.

Character Insertion. Actions for insertion are similiar to substitution and deletion except that the ^ character (5E hex) is used to open the line for insertion and to close the line once the insertion has been completed. The line is opened (moved to the right) for each character inserted. The warning bell, or tone, will sound for each character entered that will cause the line length to exceed LINE LENGTH-5 characters. All characters entered which exceed the defined LINE LENGTH will be ignored. If an error is made, it may be corrected by using the 'backspace' key and then retyped. The 'back arrow' key may also be used.

Edit Closeout:

LF - Line Feed, will close out the edit function and echo the line. The line feed is used to terminate an edit when it is desired to retain that portion of the line following the edit.

CR - Carriage Return, will close out the edit function, echo the line and drop that portion of the line following the CR (if any). A CR must be used if the edit function exceeded the original line length.

Special Actions: The line is echoed for operator review after the use of either an LF or CR. A 'question mark' character (?) will immediatly follow the line indicating a special action requirement. At this point three actions are possible; repeat edit, cancel edit or line entry.

1) Repeat edit - if the operator enters an 'R' immediatly following the ? character, the typing element will be repositioned to the beginning of the line for another edit action.

2) Cancel edit - the edit may be canceled by using the 'cancel' key (18 hex). The routine will abort and return to the command input point. No changes will be made to the file.

3) Line entry - the line is entered into the file by using the return key.

4) No other character entries are permitted and will be ignored.

RESTRICTIONS:

Because of the actions that can be performed it is suggested the user gain confidence in use of the command by practicing the the various features on a trial file.

Since the @ character is used to move the typing element with out altering the line, it is not possible to substitute this character in an edit action. To make a substitution with the @ character, first delete the character to be substituted and insert the @ character in a subsequent edit (character insertion).

EXAMPLE: EDT 1250

DISCUSSION:

The EDT command facilitates the correction of errors in a line by permiting changes to be made without retyping the entire line. It may also be used to repeat lines in a file. This is done by calling the line to the text buffer and then changing the line number to that desired. The original line will be retained in the file and the new line added.

****
EDT <n> (EDIT LINE n)
****

** FOR AIM-65 KEYBOARD AND DISPLAY **

The AIM-65 single line, 20 character on board display and keyboard offer an interesting challenge for convenient line editing. The TED program supplied for the AIM-65 includes a line editing capability similiar to the EDT routine for more conventional terminals. However, several modifications have been made to accomodate the limitations of the display.

When the EDT command is invoked, the line is moved from the file to a text buffer area. The first 20 characters will be displayed. If there are more than 20 characters in the line, the remaining portion will be scrolled to the end of the line. Once this is done, the first 20 characters will be displayed with the cursor at the first character position. The cursor is moved toward the right by entering the F1 key. Each depression of the key moves the cursor one position to the right. At the 10th character position, the cursor stops moving to the right and any subsequent entry of F1 will cause the entire line to move one position to the left under the cursor. In this manner, the operator has a view of the characters in the line both left and right of the cursor to limits of the display. If the cursor is at the mid point of the display and the backspace (control H) is used the line will move one character to the right until the first 10 characters of the line are displayed. Further use of the backspace will move the cursor to the left until it reaches the first character position. Note that both the CONTROL and H keys must be released before the display will return.

Character substitution is accomplished by typing the characters required when the cursor is at the desired position in the line similiar to the conventional EDT function.

Character deletion is performed by entering the DEL key when the cursor is at the desired position. Note that the cursor will remain at the current position and the right hand portion of the line moves to the left to close up the deleted character position.

Character insertion is accomplished by moving the cursor to the point in the line where the insertion is to take place. At this point, entry of F3 will start the insertion process and all subsequent characters typed will be inserted into the line starting at that point. The portion of the line to the right of the cursor will remain stationary and the left side will scroll left as each character is entered. When the insertion is completed, entry of F3 terminates this edit mode.

Edit close out is the same as for the conventional EDT except that F2 is use in lieu of the line feed key.

```
**************************
  END <n>                    (END FILE AFTER LINE n)
**************************
```

Terminates a file by placing an end of file character at the end of the line specified.

RESTRICTIONS: None

EXAMPLE: END 455

DISCUSSION:

The 'END' command facilitates the segmentation of file blocks by providing a capability to end a file at any point, regardless of original file length.

```
**************************
  ENT <name>                 (Same as SAV - SYM versions only)
  ENT!<name>                 (Same as SAV - SYM versions only)
  ENT <$hhhh$iiii=name>      (Same as SAV - SYM versions only)
**************************
```

```
**************************
  FIL N<hhhh>                (OPEN A FILE AT HEX ADDRESS hhhh)
  FIL A<hhhh>                (ENTER ACTIVE FILE AT HEX hhhh)
**************************
```

Used to open or enter files at the hex address specified. The 'FIL' command is included among those that are acceptable immediatly upon load and initialization of TED.

A 'FIL Nhhhh' (hhhh signifies a hex address) entry will open a new file at the address indicated.

A 'FIL Ahhhh' command permits entry into the existing file at the address specified and allow transactions with that file.

RESTRICTIONS:

Leading zeros must be entered.

EXAMPLE: FIL N4000
FIL A0500

DISCUSSION: None.

****

JMP <hhhh>	[JUMP TO HEX ADDRESS hhhh]
JMP.<hhhh>	[EXECUTE PROGRAM AT HEX ADDRESS hhhh]

****

This command has two variations. In the first case, the command will cause an exit to the monitor, to the location specified and display the content of that location. If a period [.] is included as shown, the command will cause an exit to the location specified and execute the instruction at that location.

RESTRICTIONS: None

EXAMPLE: JMP 200
JMP.2300

DISCUSSION: NONE

```
**************************
  JOB#                      (EXECUTE THE FOLLOWING INSTRUCTIONS)
  JOB <name>                (EXECUTE JOB FILE name)
  JOB <$hhhh>               (EXECUTE JOB FILE AT HEX ADDRESS hhhh)
**************************
```

The 'JOB' command permits the sequential execution of a number of instructions immediatly following the '#' symbol or in the job file called in the operand. The command permits the development of complex job files which are executed by the entry of a single command. Any TED command may be included in a job string except directives such as (.), (<), etc. Each command in the command string must be followed by a colon (:), including the last one.

The command 'JOB#' permits the entry of as many commands as can be entered on a single line. (Disk and cassette versions).

The command 'JOB <name>' allows development of job files of any length. Each line in a job file is constructed of a line number followed by a space and the commands to be executed, with each command followed by a colon (:). When building a command file ensure that the file will not be overwritten by subsequent files read into memory as a function of the job. For example, if the job is to print a number of files currently saved on disk, the job file must not be in a location of memory that will be overwritten by any of the files that will be loaded into memory during the job. Any command may overflow to the next line, but care must be taken to follow the command structure as well as the file structure, i.e., a space must follow the line number. (Disk versions only).

The command 'JOB $hhhh' is similiar in operation to the command 'JOB name' described above. It is included to permit the construction and use of temporary JOB files and in cassette versions as a method of exploiting the full JOB capability. When invoked, the file whose address is specified in the operand is assumed to be a JOB file. All restrictions of the 'JOB name' command apply. (Disk and cassette versions).

RESTRICTIONS:

The JOB command is NOT included in AIM-65 systems.

A job file must not be loaded in memory to a location which will be overwritten by any of the subsequent file transactions of the job.

```
EXAMPLE: JOB#LOD FILEA:ASM:LOD FILEB:ASM:
         JOB UPDATE (Disk versions)
         JOB $4500
```

DISCUSSION:

A JOB file is saved to disk or cassette as is any other file. In disk systems, JOB files cannot be executed directly from the scratchpads.

****
LIN <n> SET LINE WIDTH TO n CHARACTERS/LINE
****

In disk versions, the width of the line may be set from a minimum of 10 to a maximum of 128. All line oriented commands (such as 'EDT') are adjusted accordingly. The default value, set upon bootstrap or subsequent entry from the monitor into the disk operating system, is 72.

RESTRICTIONS:

Disk versions only.

The AIM-65 display is limited to 40 characters. Users of the on-board keyboard/display should set the line length to 40 characters since no audible warning device exists to alert the operator that the limits of the display have been exceeded.

EXAMPLE: LIN 80

DISCUSSION: NONE

****

LIS	(LIST)
LIS <argument>	(LIST PER argument)

****

Used to display the contents of the file as specified in the command operand. 'LIS' will display the contents of the entire file starting with the first line regardless of the line number. 'LIS <argument>' will display the file contents as specified in the argument (see below).

The command displays the lines with line numbers. Another command 'PRI' is used to display lines without line numbers.

If a period (.) is entered immediatly following the command (e.g., LIS.,400) the file (or portion) is displayed double spaced.

RESTRICTIONS:

The command form 'LIS <file name> may be used only if a file has been opened. (Disk only).

EXAMPLE:	LIS	List the entire file
	LIS FILEAB	Load & list file FILEAB)Disk only)
	LIS 2300	List line 2300
	LIS,452	List from start of file to line 452
	LIS 450,	List from line 450 to end of file
	LIS 10,50	List from line 10 to line 50
	LIS ,P	List from start of file for one page
	LIS 1000,P	List from line 1000 for one page
		(Use 'L' to list one line)

DISCUSSION:

The examples demonstrate the range of arguments that may be included as the operand of the 'LIS' command. When using either the 'P' or 'L' option the list will pause after one page (default set to 22) or one line, as specified. The listing is continued by entering a space (or non-numerical character). The listing is terminated with the 'RETURN' key or 'BREAK'. At the pause, the list may be directed to start at a different point by entering a valid line number followed by a 'RETURN'.

The display of file contents is halted using the 'BREAK' key or function on the input device. If routines other than the KIM programs are being used for output, a break test must be included as a part of the output routines.

****
LOC <string> (LOCATE ALL OCCURANCES OF string)
****

Used to locate all precise occurances of the character string specified. The entire file is scanned and all matches are displayed. If no match is found, the prompt will be displayed.

RESTRICTIONS:

'LOC' may not be used to find a line number, since line numbers are ignored in the scan. The editor is line oriented. thus, a string containing data from two or more lines will not be found.

EXAMPLE: LOC 10 DOWNING STREET

DISCUSSION: NONE

****

LOD <name>	(LOAD FILE name)
LOD <$hhhh=name>	(LOAD FILE name TO HEX hhhh - DISK)
LOD <$hhhh>	(LOAD FILE TO HEX hhhh - CASSETTE)

****

This command causes the current disk index to be read into memory. The index will be searched for the specified file name and, if found, the corresponding file will be read into memory. The load location for the file is that location from which the file was originally stored to disk unless the <$hhhh=name> option is used, in which case the file loads at the address specified. If the name is not found, the system will respond with 'UNDEFINED' indicating that the file name cannot be located. If the disk index has not been initiated, the system will respond with 'INDEX UNDEFINED'. If a name is less than six characters it does not need to be padded with spaces to make it equal to six.

In cassette based systems, the file name is the KIM format i.e., 01, A3, etc. This file name is loaded to location $17F9 for a sequential tape search for the file specified. The command 'LOD $hhhh' sets location $17F9 to $FF and loads the next file encountered on the tape to the location designated. If a source file is LODed, insure that the "?" directive is used to establish the file parameters before any lines are added, deleted or modified. Otherwise, such transactions may be treated as if they are with a new file, and all old file data could be lost.

RESTRICTIONS:

No attempt is made to distinguish between an object or data file when using the form 'LOD <hhhh=name>. Consequently, the use of 'PRI' or 'LIS' immediatly following the load of object files will result in an attempt to list the object deck. Insure that the proper file parameters have been set by using the 'FIL' command, if appropriate.

EXAMPLE: LOD FILEAB
LOD $5000=FILEAB (Disk only)
LOD $4500 (Cassette only)

DISCUSSION:

Up to three tries are made on load if the disk controller detects a CRC error. After the third unsuccessful try, the CRC status error will be output. See the FODS MANUAL section on Disk Status Errors.

****
MOV <n/m> (MOVE LINE n TO LINE m)
****

The line specified by the first line number is moved to the location identified by the second line number. If the second line number is of an existing line, that line will be overwritten and the data will be lost.

RESTRICTIONS: None

EXAMPLE: MOV 250/375

DISCUSSION: None

****
NOP (SCAN FOR NON PRINTING CHARACTERS)
****

This routine will display each line containing a character with a hex value that falls outside of the printing limits. A second line provides the absolute address of the line, a pointer to the approximate location and the hex value of the character.

RESTRICTIONS: None

EXAMPLE: NOP

DISCUSSION: None

****
PAG <n> (SET LINES PER PAGE TO n)
****

The 'PAG' command is used to set the number of lines to be output with the 'LIS' or 'PRI' command 'P' option. The default value set upon bootstrap (disk systems) or on entry into the TED (cassette versions) is 22. The command may be used at any time to change the number of lines output.

RESTRICTIONS: NONE

EXAMPLE: PAG 10

DISCUSSION:

The 'P' option of the 'LIS' and 'PRI' commands is intended for use with video monitor or CRT based terminals to limit output to a single screen.

****

PRI	(PRINT)
PRI <argument>	(PRINT PER argument)

****

The 'PRI' command lists the file without line numbers. All functions and options are the same as for 'LIS'.

RESTRICTIONS: SEE 'LIS'

EXAMPLE: SEE 'LIS'

DISCUSSION:

See the discussion section for the 'LIS' command.

****

RES <n>	(RESEQUENCE LINE NUMBERS WITH n)

****

Resequences the file line numbers by any number from 1 to 10.

RESTRICTIONS: None

EXAMPLE: RES 4

DISCUSSION:

This command may change the interval for subsequent automatic or semi-automatic line number generation.

```
**************************
   SAV <name>                (SAVE FILE name)
   SAV!<name>                (SAVE FILE name)
   ENT <name>                (Same as SAV - SYM versions only)
   ENT!<name>                (Same as SAV! - SYM versions only)
**************************
```

This command is used to name a file and store it on disk or cassette (as appropriate for the version). In disk versions, the file name and parameters are recorded in the disk index.

RESTRICTIONS:

DISK VERSIONS

If a 'SAV' is attempted with a file named the same as a file already on disk, the FODS will respond with "ERASE FILE?". If the operator responds with N, the save is aborted. A Y will cause the save to continue and the old file parameters will be deleted from the index.

If a number (0-9) is used as the first character of the file name, the command form 'LIS <name>' will not function as a call to that file.

A dollar sign ($) may not be used as the first character of the file name.

A percent sign (%) as the first character identifies the file as an executable program.

A 'commercial at' sign (@) as the first character of a file identifies the file as a 'BASIC', 'FOCAL' or similiar file.

CASSETTE VERSIONS

The file name must be in the form acceptable to the KIM cassette save routines, i.e., 2A, 01, etc. When the command is used, the file parameters are set into locations $17F5 - 17F8. The file end address is properly incremented by 1. The file name is set to location $17F9.

```
EXAMPLE: SAV FILEAB (DISK)
         SAV!NEWFIL (DISK)
         SAV 2A     (CASSETTE)
```

DISCUSSION:

DISK VERSIONS

A file name may be any character string, except as limited above, up to a total of six characters. If more than six, the name will be truncated to six. If less than six, the FODS will pad the name with spaces to equal six.

The command form 'SAV!<FILE NAME>' is used when it is desired to inhibit the comment "ERASE FILE?". In this case, the old file will be deleted and the new file entered without a notification by the system, if a duplicate file is found in the disk index.

```
**************************
   SAV <$hhhh$iiii=name>      (SAVE hhhh TO iiii WITH name)
   ENT <$hhhh$iiii=name>      (Same as SAV - SYM versions only)
**************************
```

This command defines a file by starting address, ending address and name. It allows the save of otherwise undefined files or programs. In disk versions, if the file saved is a program the first character of the name must be percent (%) for it to be called by the 'RUN' command.

RESTRICTIONS:

DISK AND CASSETTE VERSIONS

The first address must be the low order address of the file for proper operation.

DISK VERSIONS

Files located in zero page and page two (0000 to 00FF and 0200 to 02FF) cannot be saved with this command.

```
EXAMPLE: SAV $2000$3EFE=%UPDAT (DISK)
         SAV $4050$4FFF=B3     (CASSETTE)
         ENT $200$450=FILE1    (SYM DISK VERSIONS)
```

DISCUSSION:

Note that a dollar sign ($) must precede each address specification. Leading zeros need not be entered. The delimiters $,$ and = must not be bounded by spaces.

In cassette versions, the ending address is automatically incremented by 1 for the KIM cassette save routine.

```
**************************
 SCR <string>               (DELETE ALL LINES CONTAINING string)
**************************
```

All lines containing 'string' will be deleted from the file.

RESTRICTIONS: NONE

EXAMPLE: SCR TICK BITE

DISCUSSION:

In the example, all lines containing the string "TICK BITE" will be deleted from the file. The command is particularly useful in data file editing functions to remove all references to a particular item. Note that the entire line is deleted.

****
SET <string1<LF>string2> (SET string 1 TO BE string2)
****

The second character string will be substituted for each occurance of the first character string in the file. The strings do not need to be of equal length.

RESTRICTIONS:

Because the line feed (LF) is used as a delimiter, it may not be included in either string.

The on-board AIM-65 keyboard/display uses F2 in lieu of LF.

EXAMPLE: SET ABC
DEFG

DISCUSSION:

The 'SET' command will cause the display of each line where a substitution will result in the maximum line length being exceeded. In this case, the substitution will not be performed.

Prior to using this command, the operator should first use the 'LOC' command (with the old string, e.g., LOC ABC) to determine if the string will properly identify the string to be substituted. For example, the command 'SET ABC(LF)DEFGH' will result in substitutions as follows:

ABC changed to DEFGH
XABCD changed to XDEFGHD

If only the first change is intended, the string ABC should be examined to establish a unique feature. Assume that ABC was in the string ABC DEF. The 'SET' command could be

SET ABC D
DEFGH D

And the result would be

DEFGH DEF

Thereby accomplishing the substitution without unintended alterations to the file, since the string XABCD would not have matched the search string ABC D.

APPENDIX I

```
*******************************
*                             *
* USING THE HDE TEXT EDITOR *
*                             *
*******************************
```

APPENDIX I

The HDE TEXT EDITOR (TED) employs a powerful command set capable of performing a number of functions not immediatly apparent to the beginning user. The purpose of this section is to provide several examples of TED command usage which will demonstrate some of the inherent capabilities of the program.

FILE LINKING

A situation which often occurs in text editing is the need to join two files into one package. This is an easy task with TED.

1) Load the first file into memory.
2) Use the '?' directive to determine the starting and ending addresses of the file.
3) Load the second file to the ending address of the first file with the command 'LOD $hhhh=name' (in this case 'hhhh' will be the ending address of the first file). At the completion of this second load operation there are two files in memory, the file loaded in step one and the file loaded in this step. The active file is the one just loaded.
4) Use the 'FIL Ahhhh' command to reopen the first file, in this case 'hhhh' is the starting address found in step two. This action has now opened up the new, merged file.
5) Resequence the line numbers using the 'RES' command.
6) Save the new file.

FILE PARTITIONING

At some point it may become necessary to subdivide a file into smaller, more manageable pieces or excerpt a portion of a file for use in another program.

A) DIVIDE A FILE INTO TWO FILES

1) Load the file into memory.
2) Determine the point at which the file will be divided and use the 'ADR n' command to find the address of the line number that will be the first line of the second file.
3) Open a new file at the address found in step two by using the 'FIL Ahhhh' command.
4) Resequence the line numbers, if desired.
5) Save the file.
6) Now use the 'FIL Ahhhh' command to reopen the original, still complete, file.
7) Determine the line number of the line which will be the last line in this new file and use the 'END n' command to terminate the file at this point.
8) Save the file.
9) There are now three files. The original, and the two that were just created.

APPENDIX I

B] REMOVE A PORTION OF A FILE

1] Load the old file
2] Determine the starting point of the new file and use the 'ADR n' command to find the address of the line number that will be the first line of this file.
3] Open a new file at this point with the 'FIL Ahhhh' command
4] Use the 'END n' command to terminate the new file at the desired line.
5] Resequence the line numbers.
6] Save the new file. There are now two files, the original file and the new file just created.

ADDING A NEW BLOCK OF LINES TO A FILE

There are several methods for adding a new block of lines to a file. If the block of new lines is small, simply entering new lines by either using 'MOV' or 'RES' to create the necessary room is usually sufficient. With large blocks, the new block may be added to the end of the file and 'rolled' into the proper location with 'BLM'. A third method is to divide the file, add the block to the first file and link the second file to the file that has had the new lines added.

REPEATING A LINE

Repeating a line can be tedious and error prone, particularly if a special characteristic of the line, such as word spacing, must be preserved. To repeat a line, use the 'EDT' command to move the line to the text buffer area. Change the line number of the line to the number desired and save the new line. This may be repeated as often as necessary. If your version of the editor has the 'Control C' command recall function, the task is even easier. Simply use 'Control C' to recall the 'EDT' command.

REPEATING STRINGS

Some data files require that certain words or number strings be repeated for clarity or completeness. For example, a mailing label may require 'MR and MRS', or a number of labels may be prepared for the same zip code. In such cases, it is considerably more efficient to enter a 'token' character for the string and use the 'SET' command after all the data has been entered. In the 'MR and MRS' example above, a '$' could be entered in the line at the point the string would appear. When all of the data had been entered, the command 'SET $[LF]MR and MRS' would complete the data entry. Note that each line in which the appropriate substitution will be made must have sufficient room for the string. The 'token' selected must be unique in order that unwanted changes are avoided.

APPENDIX I

EDITING THE RIGHT HAND PORTION OF A LINE

Often, in assembler source programs or in data files, the information in the right hand portion of a sequential series of lines must be changed. Although this may be accomplished with the 'EDT' command another approach is to use 'APP'. This command calls a line to the text buffer for additions to the end of the line. The backspace is then used to 'back up' the line to the desired point and the modified text then added. When the RETURN key is typed, the modified line is returned to the file and the next line is moved to the buffer area. This is continued until all of the changes have been made. Lines may be skipped by using the RETURN immediatly after the line is echoed to the terminal.

SOME ADDITIONAL CONSIDERATIONS

TED files are address independent. That is they may be moved or placed anywhere in the working memory area without effect to the integrity of the file. A TED file is also independent of the editor itself. If TED is exited for any reason and subsequently reentered, the only action needed is to reopen the file with the 'FIL Ahhhh' command. Of course, if the file has been overwritten this may not be possible. The important point to remember is that a TED file cannot be 'lost' if for some reason it is necessary to exit the editor. The file remains intact. Only the starting address of the file needs to be redefined for TED to operate with the file.

If, because of an error, the 'FIL Nhhhh' command was used instead of 'FIL Ahhhh' it is still possible to recover the file. TED does not erase the old file if a new file is opened in its stead. What it does do is place the end of file marker (hex 1F) at the starting point of the new file. To correct the error, exit TED using the 'JMP hhhh' command with 'hhhh' being the address used in the 'FIL Nhhhh' command. The character displayed will be a hex '1F'. Change the '1F' to a hex '00' and reenter TED. Using 'LIS' will verify the existence of the desired file.

TED constantly checks for the end of contiguous memory and will provide the warning 'FILE?' if a line entry causes the file to 'fall out of' valid memory. If this occurs, the last line in the file may be incomplete. To recover, use the 'END n' command to terminate the file at the line just prior to the last line. Save the file and open a new one to continue the data entry.