UNIVERSITY OF EDINBURGH
Edinburgh Regional Computing Centre

User Note 86

March 1986

Title:

EDD - A Screen Editor for MS-DOS

Author: C. J. Adie

Contact: Advisory service

Software Support Category: n/a

SYNOPSIS

EDD is a screen editor for MS-DOS (version 2.0 or higher) systems. There are versions available for the IBM PC, the Sirius, and the Apricot (PC series) microcomputers.

KEYWORDS

Apricot, EDD, IBM PC, Screen editor, Sirius

Edinburgh Regional Computing Centre

James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ. Telephone 031-667 1081

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

EDD is a screen editor for MS-DOS (version 2.0 or higher) systems. There are currently versions available for the IBM PC, the Sirius, and the Apricot (PC series) microcomputers. This document describes all these implementations. EDD is intended to replace the awful EDLIN line-based editor supplied as part of MS-DOS, and is designed for preparing programs source files or .BAT batch files. It is not a word processor.

EDD was originally written in the C language by E.K.Ream for CPM 80 systems. A description was published in reference [1]. It was then modified for the IBM PC by A.D.Howard and M.Desmet, and distributed in source form through the US and UK IBM PC user groups. C.J.Adie subsequently further modified it, gave it its present name, and implemented it on the Apricot and Sirius machines. E.K.Ream has also progressed the editor, and distributes it in the USA under the name "RED". In fact, editors based on the original one in [1] appear in a number of places, including in the operating system OS9, where it is called "SCRED".

This version of the editor is in the public domain, is available free of charge, and may be freely copied, subject to the restriction that no charge (apart from reasonable distribution costs) be made for such copies.

To obtain a copy of EDD for your machine, please send a formatted blank floppy disc, indicating which machine it is for, and giving a return address, to:

Service Support Unit
ERCC
59 George Square
Edinburgh EH8 9JU.

1.1. IBM PC Version

The distribution disc contains the following files:

EDD.EXE	The editor program
EDD.DOC	A copy of this manual
EDD.HLP	The editor help file

1.2. Sirius Version

The distribution disc contains the following files:

EDDS.EXE	The editor program
EDD.DOC	A copy of this manual
EDD.HLP	The editor help file
EDD.BAT	Batch file to load keys and run editor
EDDS.KB	Keyboard for the editor

The editor is started by typing EDD (optionally followed by a filename), which loads the keyboard file EDDS.KB, saving the old keyboard in a temporary file. The MODCON utility is used to do this, and MODCON.EXE should therefore be in the search path when EDD is invoked. When the editor is exited, the old keyboard is restored.

Note that EDDS.KB must be in the current directory on the current disc when editing.

The Sirius does not support flashing characters. References to flashing in this documentation should therefore be ignored.

This editor requires MS DOS version 2.0 or higher, and therefore cannot be used on Sirius micros which have not been upgraded to run a suitable version of DOS.

1.3. Apricot Version

The distribution disc contains the following files:

EDD.EXE	The editor program
EDD.DOC	A copy of this manual
EDD.HLP	The editor help file

The Apricot does not support flashing characters. References to flashing in this documentation should therefore be ignored.

Note that the editor does not currently work on the Apricot F-series.

CHAPTER 2
IMPORTANT CONCEPTS

This chapter describes the essential concepts which you need to understand before using EDD.

2.1. The Edit Buffer

The edit buffer is an area of memory in your computer which is used to hold the text being edited. It is roughly 50 Kbytes long. If you are editing a file shorter than this, the entire file is usually read into the edit buffer, and it is the contents of the edit buffer that you will change. The original input file is unaltered until you write the contents of the edit buffer over it. (Naturally, you may write the contents of the edit buffer to another file if you wish.)

If the file is larger than the edit buffer, the usual procedure is to read the file in 'chunks' of (say) 1000 lines, edit the chunk, and then write it out to make space for a new chunk of the input file.

There are two other buffers used by the editor which are mentioned in this documentation. These are the "pick buffer" and the "record/replay buffer", and they are described in later chapters.

2.2. The Editor Mode

EDD has three 'Modes', and is always 'in' one of these modes, which are called "edit", "insert", and "command".

In command mode, you are prompted at the bottom left-hand corner of the screen with a flashing ">" symbol, and you may type in commands, perhaps with one or more parameters, terminated with return. Commands can do things like: read files into, or write files from, the edit buffer; move or delete groups of lines; or perhaps make specified changes through the buffer. Chapter 4 describes all the commands in detail.

From command mode, edit mode is entered by pressing the Escape key. (In fact, command mode may be returned to from either insert or edit modes by pressing Escape again.) In edit mode, the contents of the buffer are displayed on the screen, and you may use the arrow keys on the keyboard to move the cursor around. Changes can be made simply by overtyping with the correct text. In edit mode, several special keys may be used to do things like: move through the file a page at a time; move to the top or bottom of the file; mark a group of lines for further action; or delete characters or lines. These keys are different on different machines, and are described in detail in Chapter 5.

Insert mode is very similar to command mode, except that when a key corresponding to an ordinary character is pressed, the character is inserted into the buffer, instead of overwriting the character already there. All of the special keys which can be used in edit mode can also be used in insert mode. Changing between edit and insert modes is done with a special key, called "Insert Toggle" (which will be different depending on which computer you are using). See Chapter 5 for details of what key to use on your machine. There is a limit of 254 characters per line, and an attempt to insert a 255th will be 'belled'.

Note that there is no way to change from command mode to insert mode with a single keystroke.

2.3. The Status Line

At the top of the screen is the editor's status line. It usually carries five items of information: line number, column number, readfile name, writefile name, and editor mode.

2.3.1. Line Number

The line number indicates the number of the line on which the cursor is positioned (the current line). The first line in the buffer is line number 1, so if the file has been read or written in chunks, the line number displayed may not correspond to the line number in the file. However, if the entire file is in the buffer, the displayed line number does correspond to the line number in the file. The line number is especially useful when repairing compile-time errors, which are usually reported by the compiler in terms of the line number. Line numbers are also used when specifying a section of the buffer when in command mode.

2.3.2. Column Number

This is the current 'virtual' column, which is the position on the current line at which the cursor is displayed. The first character on the line is at virtual column 0. If a line is longer than 80 columns wide, the cursor may be moved along it until column 79 (the right-hand edge of the screen). If the cursor is driven beyond this position, the whole line moves left by about 40 column positions, allowing some more of the line to be displayed. The virtual column number displayed is always the position of the cursor with respect to the start of the line, not with respect to the edge of the screen.

For a very long line, this process may be repeated several times. Note that the maximum length of a line is 254 characters.

2.3.3. Readfile

The readfile is the file which is currently open for input. If there is no such file, a message *no rdfile* is displayed.

Consider the process of loading a file into the edit buffer. First, the file must be opened (the "open" command), then read (the "read" command), then closed (the "closeread" command). Normally, the "load" command performs all these actions, assuming the file is small enough to fit into the edit buffer. If this is not the case, or if the "open" command has been issued and the "read" command has not exhausted the file, the name of the file is displayed as the third item in the status line. When the entire file has been read, and the file closed, the name is replaced by the *no rdfile* message.

Note that the readfile name displayed is the last component only of the full pathname. Thus, if you issued a command such as:

```
>open c:\user\chris\myfile.pas
```

the status line would display only the `myfile.pas` bit. However, the editor will

correctly access the file using the full path specified in the open command.

2.3.4. Writefile

The writefile is the file which is currently open for output. If there is no such file, a message *no wrtfile* is displayed.

Very similar comments apply to the writefile as to the readfile. The commands which specify a writefile are "name", "rename" and "delname": these are equivalent to "open" for the readfile. The "write" command is used to output to the file in chunks, and the "closewrite" command terminates output to the file. The command which combines all three operations is "save", equivalent to "load".

2.3.5. Mode

The final item on the status line is the editor mode, which may be "insert", "edit", or "command", as previously described. However, several of the special keys will place some other information in the status line in this position. The best way to find out whether you are in edit or insert mode, if this is not clearly displayed in the status line, is to press the "Insert Toggle" key twice.

2.4. Tabs

The tab key is useful for laying out programs in an indented fashion. The tab stops are every 8 virtual columns, and inserting a tab into a file will ensure that the next character will appear at the next tab stop. There is a command "showtab" which displays tab characters in inverse video, and an equivalent "hidetab" to turn the display off.

Because a tab occupies only one character, but is usually several virtual columns wide, it is possible that a long line with several tab characters may have more than 254 virtual columns, but less than 254 characters, in it. The editor will cope correctly with this situation.

CHAPTER 3
USING EDD

3.1. Starting EDD

EDD is started from the MS-DOS command line by simply typing EDD followed by RETURN or ENTER. A filename may be specified following EDD, for instance:

```
A>edd myfile.prg
```

If `myfile.prg` exists, it will be loaded into the edit buffer, and the editor will start in edit mode. If the file does not exist, or if no filename is specified after the EDD command, the editor will enter command mode, ready for a load or open command. Alternatively, if a new file is to be created, EDD can be entered without a filename parameter, and the Escape key will put the editor into edit mode so that you may start to type in your program.

3.2. Editing a short file

Where a program is short enough to fit into the edit buffer completely, the usual sequence of operations is to "load" the file, edit it, and then "save" it, probably using the "backup" option of "save" (see the section on "save" in Chapter 4). This is the simplest way of using the editor.

3.3. Editing a long file

If the file is longer than will fit in the edit buffer, a "load" command will print out the message *caution: main buffer is nearly full*, and leave the readfile open. If this happens, there will usually be about 2 Kbytes left free in the buffer to let you do some editing on the chunk that has been read in. Open a writefile with "name", "delname" or "rename", and write out lines you have finished with using the "write" command. This will delete the lines from the buffer, freeing space for more lines to be read in with the "read" command. When the readfile has all been read in, it will be closed automatically. Having written all the lines to the writefile, you should close it with "closewrite".

3.4. Miscellaneous points

3.4.1. File format

The editor works with 'plain vanilla' text files. Lines may be up to 254 characters in length, and should be terminated with a Carriage Return (CR). Lines may have Line Feed (LF) characters following the CR, but these are ignored by the editor (if present) when it is reading a file in. A control-Z character in the input file will be treated as the end of file.

The output file will contain CR followed by LF at the end of each line. It will be terminated by a control-Z character (this ensures compatibility with the IBM Professional Editor).

3.4.2. Control characters

Control characters may be present in the input file. The following control characters are displayed as the corresponding characters in inverse video:

A B C D E F G K L N O P Q R S T U V W X Y [ ] ^ _

Top-bit-set characters can usually also be edited, but note that neither control nor top-bit-set characters can be inserted into the edit buffer from the keyboard.

CHAPTER 4
COMMANDS

4.1. General Comments

This chapter describes the commands which may be issued in response to the flashing ">" prompt in the "command" mode of EDD. The format which is used to describe the commands is as follows:

Command name: The name of the command in full.

Abbreviation: A one or two letter abbreviation which may be used instead of the name of the command.

Syntax: This gives the way in which the command may be used. If the command takes no parameters, it is just the command name. If there may be parameters, these are given names and enclosed in angled brackets <thus>. If some of the parameters are optional, they are enclosed in square brackets [<thus>]. Note that the square brackets, the angle brackets, and the parameter names are 'metacharacters', and should not be typed when actually using the commands. Another metacharacter which is occasionally used to describe the syntax is the vertical bar "|", which indicates that the two expressions on either side of it are alternatives.

Parameters: Each parameter named in the section on syntax is described in this section. If there are no parameters, or if the parameters are of the generic types described below, this section is omitted.

Description: This is a plain text description of the function of the command.

Example: One or more examples of the use of the command is given.

Errors: The errors which may be encountered when this command is executed are listed here. For a description of what the error messages mean, see chapter 6. Note that some "error messages" are informative, rather than indicative of an error condition. If a command cannot produce errors, this section is omitted.

The following parameter names are not described explicitly for each command which uses them, because there are many such commands.

<line range> — This is a parameter which may or may not be present. It specifies a range of lines over which the command will act. If it is not present, the line range will be the block-start line to the block-end line, inclusive. Chapter 5 describes how to set the block-start and block-end lines. It is an error if one or both of the block-start and block-end lines are not defined. If the <line range> parameter is present, then it may either be the asterisk character ("*"), indicating the entire edit buffer, or else two line numbers <start> and <end>, separated by one or more spaces. <start> must be less than or equal to <end>, and both of them must be greater than or equal to 1 and less than or equal to the last line in the buffer. The line range always includes the <start> and <end> lines.

<filename> — This is a file name specification which may include a device

name (e.g. `B:`) and a directory path (e.g. `\user\fred\`). There may, however, be no more than 64 characters in the complete file name specification.

4.2. Alphabetical list of commands

Command name: **change** Abbreviation: **ch**

Syntax: change [<line range>]

Description: This command changes all occurrences of a search pattern to a replacement pattern in a given line range. You are prompted first for the "search mask", then for the "change mask".

The search mask may contain one or more question marks "?". These match any character in the buffer, so that for instance a search mask "d?sk" will match both "desk" and "disk". The change mask may likewise contain question marks (so long as there are no more than in the search mask).

All lines where a match with the search mask is found have the matching text replaced by the change mask. Where there are question marks in the search mask, the characters they match go to replace any question marks in the change mask. Thus, if the search mask was "func(?)" and the change mask was "arr[?]" then lines with "func(i)" would be changed to have "arr[i]". (Note that the double quotes are just for emphasis in this documentation, and should not be typed in.)

Every line where a change has occurred is displayed on the screen with its line number. If several changes are made in a single line, the line is displayed after each change.

There is a special case where the first character in the search mask is a "". If this is so, the search mask is 'anchored' to the start of a line. For instance, "begin" would match "begin" at the start of a line, but not elsewhere. A "" in the change mask has no special significance.

The matching process is case sensitive. The process of change can be terminated at any time by hitting any key (except the space bar, which only pauses the process).

Example:
```
>change *
search mask ? reasonable
change mask ? unreasonable
```

Errors:
```
no line range
bad argument
invalid line range
new line too long
too many ?'s in change mask
```

Command name: **clear** Abbreviation: **cl**

Syntax: clear

Description: This command clears the edit buffer. If the edit buffer has been changed since it was last loaded from a readfile, you are prompted to see if you really want to clear the buffer. The prompt message is; *buffer not saved - proceed?* and the appropriate response is Y or y to clear the buffer - any other response will cancel the request.

Example: >clear

Errors: cancelled
buffer cleared

Command name: **closeread** Abbreviation: **cr**

Syntax: closeread

Description: This command closes the current read file and displays the *no rdfile* message on the status line.

Example: >cr

Command name: **closewrite** Abbreviation: **cw**

Syntax: closewrite

Description: This command closes the current writefile and displays the *no wrtfile* message on the status line. Note that any remaining lines in the edit buffer are NOT written to the writefile by this command. You should be sure that you have written all you wish to write using the "write" command before issuing closewrite.

Example: >closewrite

Command name:	**copy**	Abbreviation:	**cp**

Syntax: copy [<from> <to> <count>]

Parameters:

<from>	Line number of first line to be copied.
<to>	Line before which copied lines will be placed.
<count>	Number of lines to be copied.

Description: The <count> lines starting at line <from> are copied to before line <to>. If any of the three parameters are present, all must be. If none is, the lines copied are those delimited by the block-start and block-end lines, and the destination <to> is the current line.

Example: >copy 13 210 20

Errors: bad argument
interleaving not permitted

Command name:	**count**	Abbreviation:	**ct**

Syntax: count <n>

Parameters:

<n>	A decimal number.

Description: This command sets the "count" variable to the value of <n>. It controls the number of times the replay buffer is replayed. See Chapter 5 - section on Replay key - for further details. The default value of count is 1.

Example: >count 10

Errors: bad argument

Command name: **delete** Abbreviation: **dl**

Syntax: delete [<line range>]

Description: All lines in the indicated line range are deleted. If there is room, the lines are copied into the pick buffer, overwriting anything already there. (See Chapter 5 - the section on Copy to Pick Buffer - for details of this buffer.) The block-start and block-end lines become undefined.

Example: >dl 492 500

Errors: no line range
invalid line range
bad argument
deleted lines saved in pick buffer

Command name: **delname** Abbreviation: **dn**

Syntax: delname <filename>

Description: This command opens <filename> for output. If a file with this name already exists, it is deleted without further warning. <filename> becomes the writefile.

Example: >delname b:fred.txt

Errors: writefile open

Command name: **extract** Abbreviation: **ex**

Syntax: extract [<line range>]

Description: The indicated lines are written to the writefile. They are not deleted from the edit buffer.

Example: >ex 70 90

Errors: file not opened
no line range
invalid line range
bad argument

Command name:	**find**	Abbreviation:	**f**

Syntax: find [<mask>]

Parameters:

<mask> The search mask to use.

Description: If a <mask> is not given as an argument, it will be prompted for. The format of the search mask is the same as for the "change" command (qv). If a match is found, the editor will enter edit mode and position the cursor at the start of the matching text. The search starts on the line after the current line.

Example: >f d?sk

Errors: pattern not found

Command name:	**goto**	Abbreviation:	**g**

Syntax: goto [<n>]

Parameters:

<n> A decimal number.

Description: The editor enters edit mode and positions the cursor at the start of line <n>. If <n> is omitted, the cursor is positioned at the start of the current line.

Example: >goto 65

Errors: bad line number

Command name:	**help**	Abbreviation:	**?**

Syntax: help

Description: Help information to do with the available commands is displayed. If the file EDD.HLP can be found on the current drive in the current directory, its contents are displayed a page at a time. If the file cannot be found, a restricted amount of information (held in the editor program itself) is displayed.

Example: >help

Command name: **hidetab** Abbreviation: **ht**

Syntax: hidetab

Description: Tab characters can be represented on the screen as blank space, or as inverse-video blocks. This command selects the former representation, which is the default when the editor is started.

Example: >ht

Command name: **load** Abbreviation: **ld**

Syntax: load <filename>

Description: This command opens <filename> as the readfile, and attempts to read the entire file into the edit buffer. If the entire file is read successfully, it is then closed. However, if the file is too large, it is left open to allow further reading when there is room in the edit buffer. Note that the load command is the same as an "open" command followed by a "read *" command.

Example: load myprog.pas

Errors: readfile still open
no file argument
file not found
line truncated
caution: main buffer nearly full

Command name: **lowercase** Abbreviation: **lc**

Syntax: lowercase [<line range>]

Description: The alphabetic characters within the indicated line range are all converted to lower case.

Example: >lc *

Errors: bad argument
no line range
invalid line range

Command name: **merge** Abbreviation: **mg**

Syntax: merge <filename>

Description: Insert the named file into the edit buffer at the cursor. If the file is too big to fit into the edit buffer, an error message is issued and the file is closed.

Example: >merge mod3.pas

Errors: no file argument
file not found
line truncated
main buffer is full

Command name: **move** Abbreviation: **mv**

Syntax: move [<from> <to> <count>]

Parameters:

<from> Line number of first line to be moved.

<to> Line before which moved lines will be placed.

<count> Number of lines to be moved.

Description: The <count> lines starting at line <from> are moved to before line <to>. If any of the three parameters are present, all must be. If none are, the lines moved are those delimited by the block-start and block-end lines, and the destination <to> is the current line.

Example: >move 100 30 10

Errors: bad argument
interleaving not permitted

Command name: **name** Abbreviation: **n**

Syntax: name <filename>

Description: <filename> is opened as the writefile, if it does not already exist. If it does exist, an error message is issued, and the file is not opened.

Example: >name newfile.c

Errors: writefile open
disk file exists
no file name

Command name: **open** Abbreviation: **o**

Syntax: open <filename>

Description: <filename> is opened for input, and becomes the readfile. Nothing is read from it.

Example: >open oldfile.for

Errors: read file still open
no file argument
file not found

Command name: **print** Abbreviation: **pr**

Syntax: print [<line range>]

Description: List the indicated lines to the printer. Pressing the space bar during printing will pause the printout: pressing it again will restart printing. Pressing any other key will abort the operation and return to command mode.

Example: >p 100 200

Errors: bad argument
no line range
invalid line range

Command name: **quit** Abbreviation: **q**

Syntax: quit

Description: This command leaves the editor and returns to MS-DOS. If the edit buffer has been changed since it was last saved, you are prompted *buffer not saved - proceed ?*. A "Y" or "y" response will exit to MS-DOS, while any other key will cancel the command. If there are any open files, they are closed before the program terminates.

Example: >q

Errors: cancelled

Command name: **read** Abbreviation: **rd**

Syntax: read <n> | *

Parameters:

<n> A decimal number

Description: The next <n> lines are read from the readfile. If this exhausts the readfile, it is closed. If <n> is replaced by "*", as many lines as possible are read from the readfile.

Example: >rd 1000

Errors: no read file
bad argument
line truncated
caution: main buffer nearly full

Command name: **rename** Abbreviation: **rn**

Syntax: rename <filename>

Description: The current writefile is closed, and <filename> becomes the new writefile, unless it already exists. Very similar to the "name" command.

Example: >rename newprog.for

Errors: no write file
disk file exists
no file name

Command name: **save** Abbreviation: **sv**

Syntax: save [<filename>]

Description: The entire edit buffer is written to <filename>. Unlike the "load" command, writefile may be open when this command is executed. If <filename> is omitted, then the name used is the last readfile name, or the name used by the last save command, whichever was more recent. (The default name used may be found from the "savename" command.)

If <filename> already exists, you are prompted with *Cancel, Replace, Backup ?* Pressing C will cancel the save command (but the filename will still be recorded as having been used by save). Pressing R will prompt again to confirm you really want to overwrite the indicated file. Pressing B will rename the

existing file to have a .BAK extension, and then save the edit buffer as a new version of the file. (An existing .BAK file of the same name will be deleted.)

Although this description of the save command may sound a little complex, in fact it is the easiest way to store your edited text in a file.

Example: >sv myprog.pas

Errors:
cancelled
no file name
file already exists
cannot form backup name
cannot delete old backup
cannot rename old file as backup

Command name: **savename** Abbreviation: **sn**

Syntax: savename

Description: This command simply prints out the default name which would be used by the "save" command if it were to be issued.

Example: >sn

Errors: no default name

Command name: **search** Abbreviation: **sr**

Syntax: search [<line range>]

Description: You are prompted for a *search mask ?* which should conform to the convention described under the "change" command. All matching lines within the <line range> are displayed on the screen with their line numbers.

Example:
>search *
search mask ? ^begin

Errors:
no line range
bad argument
invalid line range

Command name: **showtab** Abbreviation: **st**

Syntax: showtab

Description: Tab characters may be represented on the screen as blank spaces, or as inverse video blocks. This command selects the latter representation. The "hidetab" command reverts to the former, default, condition.

Example: >st

Command name: **uppercase** Abbreviation: **uc**

Syntax: uppercase [<line range>]

Description: The alphabetic characters contained within the indicated line range are all converted to upper case.

Example: >uppercase 10 30

Errors: bad argument
invalid line range
no line range

Command name: **write** Abbreviation: **w**

Syntax: write <n> | *

Parameters:

<n> A decimal number.

Description: The <n> lines from the front of the buffer are written to the writefile and then deleted. If <n> is replaced by "*", the entire edit buffer is written to the writefile, and the buffer is then cleared.

Example: >write 1000

Errors: file not opened
bad argument

CHAPTER 5
KEYS

This chapter describes the special keys used for editing when in insert or edit mode.

5.1. General Comments

There are about 40 keys or key combinations which have special significance when in edit or insert mode. Most have the same effect in the two modes. Because the keyboards of the various MS-DOS machines which this editor runs on are different, the special keys may be different. For this reason, for the purposes of this documentation, the keys are usually named by their function, rather than by the actual keys pressed on the keyboard. Thus, we refer to the Split Line key rather than the F3 key (or whatever).

Many keys have an effect which is not limited to the current line (the line with the cursor on it). Where this is the case, the effect is normally *above* the current line. Thus, (for example) the Insert Line key will insert a new (blank) line *above* the current line.

The function key overlays are printed at the back of this User Note. The appropriate version for your machine may be photocopied, cut out, and placed beside the function keys to which it refers.

You may wish to go through the list of keys below, highlighting the appropriate key for your machine with a suitable felt pen.

5.1.1. IBM PC

The F2 key on the IBM PC may be used instead of the Escape key.

Note that use of the numeric keypad with the ALT key should be avoided, as it may produce unexpected results. The numeric keypad may be used unshifted, or with the shift or control keys, as detailed below.

5.1.2. Sirius

The Sirius ALT key has the same function as SHIFT when used in conjunction with the function keys. It may still be used with the arrow keys to adjust the display intensity and contrast, as normal. Apart from this, the Sirius keyboard is the ERCC standard layout.

5.1.3. Apricot

There are eight function keys at the right of the upper edge of the Apricot keyboard. These are engraved with legends which are not particularly relevant to the editor, and so for the purposes of this documentation, the keys will be referred to as F1 through to F8. The correspondence is:

F1	HELP	F3	REPEAT	F5	PRINT	F7	MENU
F2	UNDO	F4	CALC	F6	INTR	F8	FINISH

The Apricot Control key has the same function as the Shift key when used in conjunction with the function keys.

5.2. List of Keys

Key name:	**Backspace**
Description:	Delete the character immediately before the cursor, and move the rest of the line leftwards.

Key name:	**Return**
Description:	In edit mode, move to the start of the next line: if at the bottom of the buffer, move to the beginning of the last line. In insert mode, create a new (blank) line below the current line and move to the start of it.

Key name:	**Up Arrow**
Description:	Move up to the same screen column on the line above, if there is a line above.

Key name:	**Down Arrow**
Description:	Move down to the same screen column on the line below, if there is a line below.

Key name:	**Left Arrow**
Description:	Move one character position to the left on the current line. Tab characters are moved over as a single character. If the line has previously scrolled left, moving left beyond the left-hand edge of the screen will cause the line to be scrolled right again.

Key name:	**Right Arrow**
Description:	The cursor moves one character position to the right. Tab characters are treated as a single character. If the line extends beyond the right-most screen position, driving the cursor past the right-hand edge of the screen will cause the line to scroll left. This can be repeated until the end of the line is reached. Note that when the cursor is moved off the line, the line is scrolled back to its beginning.

Key name:	**Begin Line**
IBM PC:	Control left arrow
Sirius:	Shift left arrow
Apricot:	Shift left arrow
Description:	Move to the beginning of the current line, scrolling right if necessary.

Key name:	**End Line**
IBM PC:	Control right arrow
Sirius:	Shift right arrow
Apricot:	Shift left arrow
Description:	Move to the end of the current line, scrolling left as necessary.

Key name:	**Delete to EOL**
IBM PC:	End
Sirius:	Line del eol
Apricot:	Clear
Description:	Delete from the current cursor position to the end of the line.

Key name:	**Insert Toggle**
IBM PC:	Ins
Sirius:	Line ins mode
Apricot:	Line insert char
Description:	Toggle between insert and edit mode, reflecting the current mode in the status line.

Key name:	**Delete**
IBM PC:	Del
Sirius:	Del char
Apricot:	Line delete char
Description:	Delete the character at the cursor, moving the rest of the line left.

Key name:	**Home**
IBM PC:	Home
Sirius:	F7
Apricot:	Home or F8
Description:	Move the cursor alternately to the top and bottom of the screen, at column position 0.

Key name:	**Page Up**
IBM PC:	PgUp
Sirius:	Shift scrol
Apricot:	Shift scroll
Description:	Move the text display up by 20 lines, and position the cursor at the start of the top line on the screen.

Key name:	**Page Down**
IBM PC:	PgDn
Sirius:	Scrol
Apricot:	Scroll
Description:	Move the text display down by 20 lines, and position the cursor at the start of the top line on the screen.

Key name:	**Scroll Up**
IBM PC:	Control-PgUp
Sirius:	Shift up arrow
Apricot:	Shift up arrow
Description:	Start to scroll upwards through the buffer one line at a time. Pressing the space bar will pause the scrolling, and pressing it again will restart it. The scrolling process stops when the top of the buffer is reached, or when any key other than the space bar is pressed.

Key name:	**Scroll Down**
IBM PC:	Control-PgDn
Sirius:	Shift down arrow
Apricot:	Shift down arrow
Description:	Start to scroll downwards through the buffer one line at a time. Pressing the space bar will pause the scrolling, and pressing it again will restart it. The scrolling process stops when the end of the buffer is reached, or when any key other than the space bar is pressed.

Key name:	**Goto**
IBM PC:	F1
Sirius:	F1
Apricot:	F1
Description:	You are prompted at the top right-hand corner of the screen for a line number to go to. The editor will position the cursor at the start of the given line. If the Escape key is pressed instead of entering a number, the editor will return to edit or insert mode as appropriate.

Key name:	**Escape**
Description:	When in insert or edit mode, this key will cause the editor to enter command mode. When in command mode, it will cause the editor to enter edit mode. It is the only special key which is operational in command mode. (Note that the F2 key on the IBM PC serves the same function as the Escape key.)

Key name:	**Split**
IBM PC:	F3
Sirius:	F5
Apricot:	F5
Description:	The current line is split into two. The character the cursor is on becomes the first character of the lower line. The Join key reverses the effect of Split.

Key name:	**Join**
IBM PC:	F4
Sirius:	Shift F5
Apricot:	Shift F5
Description:	The current line is joined with the preceding line. If the resulting line would be too long, the join is not performed. The cursor moves to the start of the joined line.

Key name:	**Insert Line**
IBM PC:	F5
Sirius:	Shift line ins mode
Apricot:	Shift line insert char
Description:	A new line is inserted above the current line; the cursor is positioned at the start of it; and the editor enters insert mode (of course, it may already have been in insert mode).

Key name:	**Delete Line**
IBM PC:	F6
Sirius:	Shift line del eol
Apricot:	Shift line delete char
Description:	The current line is deleted from the edit buffer. The text below is moved up by one line. (The deleted line may be recovered by using the Undelete Line key.)

Key name:	**Mark Block Start**
IBM PC:	F7
Sirius:	F2
Apricot:	F2
Description:	The current line is marked as the start of a block. The block-start line, as it has been called earlier, is used by many of the commands. Note that after some of the commands, and when EDD is first started, the block-end line is undefined.

Key name:	**Mark Block End**
IBM PC:	F8
Sirius:	F6
Apricot:	F7
Description:	The current line is marked as the end of a block. The block-end line, as it has been called earlier, is used by many of the commands. Note that after some of the commands, and when EDD is first started, the block-end line is undefined.

Key name:	**Copy To Pick Buffer**
IBM PC:	F9
Sirius:	F3
Apricot:	F3
Description:	There is an area of memory called the "pick buffer" (about 2 Kbytes long) which is set aside for small copying operations. This key copies the text delimited by the Mark Block Start and Mark Block End commands to the pick buffer, if there is room for it. Anything in the pick buffer already is overwritten. If the marked block is too large to fit into the pick buffer, a message *not picked: no room* is displayed on the status line.

Key name: **Pick Buffer To Edit Buffer**

IBM PC: F10
Sirius: Shift F3
Apricot: Shift F3

Description: The contents of the pick buffer are copied to the edit buffer, such that they appear above the current line. Using this key in conjunction with the Copy To Pick Buffer key provides an alternative to the "copy" command (but unlike "copy", it is limited in the amount of text it can copy). The block-start and block-end lines become undefined after this operation.

Key name: **Buffer Top**

IBM PC: Alt F1
Sirius: Shift F1
Apricot: Shift F1

Description: This key places the cursor right at the top of the edit buffer (not just the top of the screen).

Key name: **Buffer End**

IBM PC: Alt F2
Sirius: Shift F7
Apricot: Shift F8

Description: This key places the cursor right at the end of the edit buffer.

Key name: **Goto Character**

IBM PC: Alt F3
Sirius: Calc
Apricot: F6

Description: An 'ordinary' key should be pressed following this key. The cursor will move rightwards along the current line until it finds an occurrence of that character, or until it comes to the end of the line.

Key name: **Delete To Character**

IBM PC: Alt F4
Sirius: % (keypad)
Apricot: Shift F6

Description: An 'ordinary' key should be pressed following this key. Characters are deleted from the line up to (but excluding) the next occurrence of that character on the line. If there is no occurrence of the character, the action is the same as Delete To EOL.

Key name: **Abort Changes**

IBM PC: Alt F5
Sirius: F4
Apricot: F4

Description: Any changes made to this line since it became the current line are abandoned. The cursor is positioned at the start of the line. Note that if the cursor has been moved off the current line, this key will not restore the previous state of the line.

Key name: **Undelete Line**

IBM PC: Alt F6
Sirius: Shift F4
Apricot: Shift F4

Description: The last line to have been deleted by the Delete Line key is replaced in the edit buffer immediately above the current line. This key provides a simple way to move or copy a line. Simply delete the line, undelete it into the same position if required, move to the line below where the copied/moved line is to go, then undelete it (again).

Key name: **Goto Block Start**

IBM PC: Alt F7
Sirius: Shift F2
Apricot: Shift F2

Description: Move the cursor to the block-start line. No action if the block-start line is undefined.

Key name: **Goto Block End**

IBM PC: Alt F8
Sirius: Shift F6
Apricot: Shift F7

Description: Move the cursor to the block-end line. No action if the block-end line is undefined.

Key name: **Virtual Column Left**

IBM PC: Alt F9
Sirius: Divide key on keypad
Apricot: Times key on keypad

Description: The cursor moves one virtual column to the left. Unlike Left Arrow, which treats tab characters as a single unit, this key allows the cursor to be positioned anywhere "within" a tab. If an "ordinary" key is typed when the cursor is in the middle of a tab, sufficient spaces are inserted before the character so that the character does appear at the position you expect. Experimenting with the "showtab" command in effect should clarify exactly what happens.

Key name: **Virtual Column Right**

IBM PC: Alt F10
Sirius: Times key on keypad
Apricot: Divide key on keypad

Description: The cursor moves one virtual column to the right. Similar comments apply to this key as to the Virtual Column Left key.

Key name: **Start/End Recording**

IBM PC: Minus key on keypad
Sirius: Minus key on keypad
Apricot: Minus key on keypad

Description: There is an area of memory called the "record/replay buffer" (about 100 bytes long), which is used to store keystrokes entered from the keyboard for later replay. The Start/End Recording key starts or terminates the process of storing the keystrokes in this buffer. It has a "toggle" effect, and the current record state is reflected in the status line.

Key name: **Replay**

IBM PC: Plus key on keypad
Sirius: Plus key on keypad
Apricot: Plus key on keypad

Description: This key tells the editor to take its input from the record/replay buffer instead of from the keyboard. The "count" command determines the number of times the record/replay buffer is scanned for this purpose. Replay is terminated if one of several conditions are met:

1. A key is pressed on the keyboard.
2. The end or the beginning of the edit buffer is reached.
3. Error conditions.
4. Attempts to replay file-oriented commands.

This feature of EDD is very powerful. It allows you to make an arbitrary sequence of changes, and then repeat them as many times as required throughout the edit buffer.

Key name: **"Ordinary Keys"**

Description: These are all the alphabetic, numeric and punctuation keys on the keyboard. In edit mode, the key pressed replaces the character in the edit buffer at the cursor position. In insert mode, the key struck is inserted before the cursor position.

CHAPTER 6
ERROR MESSAGES

Most of the following error messages are displayed in command mode. Those which are displayed in edit mode or insert mode appear in the status line. For these messages, it is necessary to press a key in order to return to normal edit or insert operation.

Note that some of these error messages are informative rather than indicative of an error condition.

bad argument - A command-mode argument which should be numeric contained a non-numeric character.

bad line number - As for **bad argument**.

buffer cleared - The edit buffer has been cleared.

cancelled - The requested operation has not been carried out.

cannot delete old backup - During a "save" command, the old .BAK file could not be deleted - perhaps it is write protected.

cannot form backup name - EDD could not parse the filename to construct the .BAK file name.

cannot rename old file as backup - The existing file could not be renamed to have a .BAK extension - perhaps it is write protected.

caution: main buffer nearly full - During a "load" or "read" operation, the edit buffer was nearly filled. There is usually about 2 Kbytes of space left in the buffer when this happens. See section 3.3 for further details.

deleted lines saved in pick buffer - A delete command was issued, and the deleted text was small enough to be copied to the pick buffer.

disk file exists - An attempt was made to "name" or "rename" as the writefile, a file which already exists.

file already exists - As for **disk file exists**.

file not found - A "merge" or "open" command was issued for a file which did not exist.

file not opened - A "write" command was issued when there was no writefile.

interleaving not permitted - A "move" or "copy" command was issued with the destination line within the source line range.

invalid line range - A <line range> has been specified where the start line is after the end line.

line truncated - While reading a file, a line in excess of the limit of 254 characters has been encountered, and the excess characters have been ignored.

main buffer is full - During a "merge" command, the edit buffer has filled up. The file is closed. Similar comments apply here as for the **caution: main buffer full** message. This error may also occur when inserting text into a file which as a result causes the buffer to become full. In this case, the last line of text entered is not stored in the buffer.

new line too long - A "change" operation would have resulted in a line which exceeded 254 characters.

no default name - A "save" command has been issued without a prior "load" or "open" command.

no file argument - A command which requires a <filename> argument has been issued without one.

no file name - As for **no file argument.**

no line range - No <line range> parameter has been given, and one or both of the block-start and block-end lines are undefined.

no read file - A "read" command has been issued without a readfile having been opened.

no write file - A "write" command has been issued without a writefile having been opened.

pattern not found - A "find" or a "search" command has been issued, but no match has been found.

readfile still open - an "open" command has been issued while there is an already open readfile.

too many ?'s in change mask - In a "change" command, there were more question marks in the change mask than in the search mask.

writefile open - A "name" command was issued when the writefile was still open.

I. References

1. E.K.Ream, Dr Dobb's Journal, January 1982 p18.

II. Possible Future Enhancements

In all software development, there is a tendency to go on adding bells and whistles, and perhaps detract from the usability of the end product. The author has resisted this tendency, and therefore the editor is basic and functional, rather than large and complex. However, there may be a case for adding some of the extra features described below:

- The Sirius and Apricot versions use a "virtual cursor" technique which prevents the cursor appearing except when the editor is actually waiting for keyboard input. This has the effect of preventing the cursor flashing about the screen when it is being updated. Perhaps this should be added to the IBM version.

- Some peculiar text files from certain environments use LF instead of CR to delimit lines of text. Should EDD recognize LF on its own as well as CR as indicating end of line?

- Some people may wish to insert control characters and top-bit-set characters into text files. A way to do this could be provided.

- Auto-indenting is very useful when developing programs. It could be included, being turned on/off by a command (like show/hide tabs). What should the default be?

- Detab and entab commands would be useful for some people.

- A tabs command, to change the tab width, might be feasible.

- The mode display in the status line could be changed to be more useful and more tidy.

- When "going to" a line, it would perhaps be more sensible to position the line in question at the centre of the screen rather than at the top.

If you have any strong views on the above possibilities, or on any other feature of the editor, please let the author know.

EDD:IBM PC	
GOTO LINE	MODE
GOTO TOP	GOTO END
SPLIT LINE	JOIN LINE
GOTO CHAR	DEL TO CHAR
INSERT LINE	DELETE LINE
ABORT LINE CHANGES	RESTORE LINE
MARK BLOCK START	MARK BLOCK END
GOTO BLOCK START	GOTO BLOCK END
BLOCK TO PICK BUFF	INSERT PICK BUFF
VIRTUAL LEFT	VIRTUAL RIGHT

GOTO TOP	GOTO BLOCK START	INSERT PICK BUFF	RESTORE LINE	JOIN LINE	GOTO BLOCK END	GOTO END	SHIFT ↓
GOTO LINE	MARK BLOCK START	BLOCK TO PICK BUFF	ABORT LINE CHANGES	SPLIT LINE	MARK BLOCK END	HOME	EDD: APRICOT

GOTO TOP	GOTO BLOCK START	INSERT PICK BUFF	RESTORE LINE	JOIN LINE	GOTO BLOCK END	GOTO END	SHIFT ↓
GOTO LINE	MARK BLOCK START	BLOCK TO PICK BUFF	ABORT LINE CHANGES	SPLIT LINE	MARK BLOCK END	HOME	EDD: SIRIUS