

E8 Manual

Bill Silver
11 April 2020
Version 3.0

E8 is an Emacs-like text editor for the PDP-8/I. It runs under OS/8 and communicates with the user via character I/O on the console terminal. The console is expected to be, or behave like, a simple fixed-size character-oriented display terminal, able to process a few basic ANSI escape sequences. This document assumes some familiarity with the PDP-8, OS/8, and Emacs.

I Have to Say This

The source code described here is copyright (c) 2020 by Bill Silver and is distributed under the terms of the GNU General Public License, Version 3, which grants you certain rights to copy, modify, and redistribute. The license can be found at <http://www.gnu.org/licenses/>. There is no express or implied warranty, including merchantability or fitness for a particular purpose. You assume full liability for the use of this code.

Cautions

I've been using E8 to further its own development with no trouble, but it has not yet been tested extensively. Save often and make backups. Note that OS/8 provides almost no protection for its file system from errant application code. E8 has some fail-safes to prevent bugs from overwriting other files, and there are no known bugs, but still, this is software. Until community use is further along, I recommend editing files on some removable media that doesn't contain stuff you can't afford to lose.

“Hardware” Requirements

E8 needs at least an 8/I with EAE and at least three fields of memory (12K). The maximum file size (characters) that can be edited is simply the number of words of memory minus fields 0 and 1. So if you have all eight fields, you get a max size of 24K characters. E8 can display, but not properly edit, larger files.

Terminal Requirements and Processing

Display

The terminal must be able to process the following ANSI escape sequences:

ESC [<i>row;col</i> H	set cursor position
ESC [2 J	clear screen
ESC [K	clear to end of line

E8 sends the BELL code (007) if you try to do something that can't be done, like entering an unimplemented command character, moving the cursor past the ends of the buffer, or entering a bad filename character. If your terminal doesn't beep or flash the screen, you'll miss these.

The basic Linux screen (e.g. CTRL-ALT-F1), and MobaXterm, are fine.

Keyboard

Many E8 commands are intended for use with the ALT key. E8 recognizes the sequence

ESC char

to mean ALT-*char*. Many terminals and terminal emulators that have an ALT key will send that sequence when the Alt key is pressed. If yours doesn't, just type the ESC.

Many control characters that Emacs has used since the beginning are captured by Linux or SimH and not sent along to the terminal. ALT alternatives are provided in each case, but it takes a little getting used to if you have Emacs muscle memory. If you happen to be using an actual terminal, you can enjoy proper Emacs behavior. The captured characters that I've found so far are ^A, ^E, ^Q, ^S.

There is some ambiguity about whether the contemporary backspace key should send the backspace code (^H, 010) or the delete code (177). E8 considers them the same and converts 177 to 010.

Installation

Source Files

There are three equivalent configurations of source files, depending on whether you use PIP, or something that can handle larger files, to transfer to OS/8, and depending on whether you want files smaller than 24K characters so E8 can edit them.

Use PIP	E8 can edit	Files
yes	yes	E8.PA, EA.PA, EB.PA, EC.PA, ED.PA, EE.PA, EF.PA, EG.PA
no	yes	E8BASE.PA, E8FILE.PA, E8SRCH.PA
no	no	E8ALL.PA

Setting Screen and Buffer Size

Before moving anything to OS/8, edit E8.PA, E8BASE.PA, or E8ALL.PA, as follows. All of these are near the beginning of the file.

The symbols SCRWD and SCRHT define your screen size. These are decimal values. Make them whatever you like, as long as

$SCRWD * (SCRHT + 1) \leq 3968$

There is no limit on the size of lines that can be in files and edited, but you can only see the first SCRWD characters of each line. What you can't see is there and not lost.

The symbol ENDFLD specifies the size of memory. The default "100" is for all eight fields. If you actually have n fields (n > 1), make it "n0", i.e. for 4 fields it would be "40".

Getting the Code Onto OS/8

Choose an OS/8 device to hold the E8 source and the files you want to edit, and assign it to DSK:

```
.AS <physical device> DSK
```

My method is to copy/paste the source code into a MobaXterm session connected to OS/8, while PIP on the OS/8 receives it, like this:

```
.R PIP
*E8.PA<TTY:
```

After each file is copied this way, type ^Z to PIP to signify end of file, and then you're back at the PIP command prompt ready to do the next file. I have found that PIP can't handle files longer than 549 lines, so use the eight small files. Often my MobaXterm stops sending characters for a few seconds and then resumes, so make sure to wait until the last line is sent. This also can happen during an E8 screen update, so beware.

[This link](#) has other and probably better ways to do it. If your method can handle arbitrarily large files, you can use the other source configurations.

Build and Run

```
.R PAL8
*E8<E8,EA,EB,EC,ED,EE,EF,EG/L
*E8<E8BASE,E8FILE,E8SRCH/L
*E8<E8ALL/L
*^C
.SA SYS E8;200=1000
.R E8
```

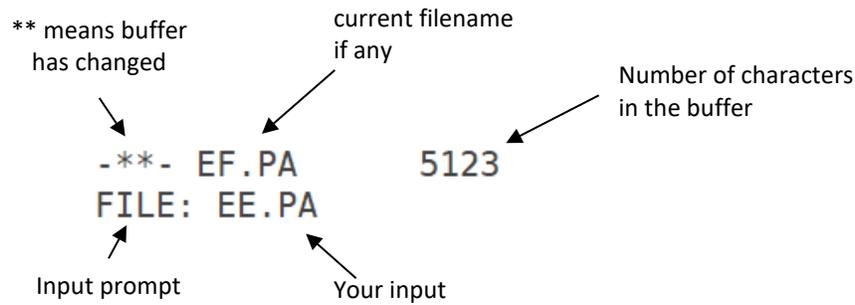
} one of these three

The screen will start cleared expect for the mode line showing an empty text buffer.

Files

- E8 can display and edit OS/8 text files, which contain 7-bit ASCII codes that include lowercase.
- The character parity bit is cleared on input. If your file has the parity bits set, E8 will clear them. Usually this is not a problem, but it could be fixed if it is.
- On input the CR code (^M, 015) is considered new line, and the LF code (^J, 012) is discarded. On output, CR is written as CR, LF.

Mode Line



Entering Filenames

In a single edit session you can create new files and view or edit as many files as you like. When prompted for a filename:

- Names must be alphanumeric, no more than six characters, with an optional extension of up to two characters.
- Lowercase letters are made uppercase.
- Any character that would not result in a legal filename will be rejected.
- The CR code (Enter on modern keyboards) terminates and accepts the entry.
- Backspace clears the filename so you can start over.
- ^G aborts the operation.
- You cannot enter a device. DSK: is assumed.

Query-Replace (ALT-%)

When entering strings at the REPLACE and WITH prompts:

CR	Accept string
BS	Delete last character entered
^G	Abort query-replace
^N	Put CR (newline) in string

You will be shown successive instances of the replace string, and you can:

SP	Replace and continue
n or N	Don't replace and continue
.	Replace and quit
CR	Quit
!	Replace all without asking

Change Protection

If there are unsaved changes in the buffer and you try to exit E8, create a new file, or read in an existing file, you will be offered the opportunity to save the changes. The responses are Y (yes), N (no), or ^G (abort). If you select Y and there is no filename, you will be asked for one. Only uppercase Y and N are accepted.

Editing

Like emacs, E8 is a character editor. All characters are traversed and edited the same way, including TAB and CR. The other control characters are displayed with the customary ^ prefix, but remember that they are just one character in the buffer.

Commands

The ALT commands are case-insensitive. The ^X commands consider control, uppercase, and lowercase letters to be all the same. For example, ^X ^S, ^X S, and ^X s are all the same. Highlighted **characters** are captured by Linux or SimH so use the ALT version.

^@ or ^SP	Set the mark to the current position (cursor)
^A or ALT-A	Beginning of line
^B	Back one character
^D	Delete forward one character
^E or ALT-E	End of line
^F	Forward one character
^H (BS)	Delete backward one character
^I (TAB)	Insert TAB
^J (LF)	Insert CR, TAB
^K	Kill (delete) to end of line; if at end, delete CR
^L	Erase and redraw screen with cursor at the middle line
^M (CR)	Insert CR
^N	Beginning of next line
^O	Open new line (CR, ^B)
^P	Beginning of previous line
^Q	Insert next typed char as is

^S or ALT-S	Incremental search (case sensitive): <ul style="list-style-type: none"> • CR terminates search with mark set to starting point • ^F finds the next occurrence of the search string • BS erases last search character and backs up • ^N matches CR (newline) in search text
^V	Forward one screen
^W	Write region (text between cursor and mark) to the file <code>clip.e8</code> and delete the text in the region.
^Y	Insert the file <code>clip.e8</code> at the cursor
^Z	Exit to OS/8
ALT-%	Query-replace
ALT-<	Beginning of buffer
ALT->	End of buffer
ALT-B	Back one word
ALT-D	Delete forward one word
ALT-F	Forward one word
ALT-H (BS)	Delete backward one word
ALT-N	Search for the last search or replace string
ALT-Q	Insert next typed character as a control character, e.g. ALT-Q A inserts ^A.
ALT-V	Back one screen
ALT-W	Write region to the file <code>clip.e8</code> ; do not delete the region.
^X F	Open existing file or create new one
^X I	Insert file at cursor
^X R	If the previous file read filled the buffer before the end of the file, clear the buffer and read more text from the file starting at some point up to 384 characters before the last one read.
^X S	Save buffer to current file, prompt for filename if none
^X W	Write buffer to new filename.
^X ^X	Exchange cursor and mark.