ZED-128 version 0.77.00 [February 16, 1993]

Here is a text editor program for the Commodore 128 80-column screen that I am currently working on. IT IS NOT COMPLETE, but I think that it is complete enough to be quite useful. To use it, just LOAD and RUN the "ZED-128" binary file. Be sure to save it in PRG format.

==========

New features of version 0.77 over version 0.75:

Well, really not much. The bug with loading a file that is too large from a non-burst device causing the machine to crash has been fixed. A couple of other minor bugs have been fixed.

Also, CT-(Enter Commodore DOS command) has been implemented. This is very useful for CMD drive users out there.

==========

Here is a summary of Zed's important features:

- Works with *BIG* text files. It gives over 100K bytes free for the unexpanded 128 and almost 630K bytes free with a 512K RAM expander. It auto-detects whether you have a RAM expander and supports up to 8 Megs.

- 100% unadulterated machine code with high-speed VDC accessing.

- Uses Burst commands for reading 1571 and 1581 files. Reads about 3,500 bytes/sec from a 1571 and about 6,100 bytes/sec from a 1581. Works with non-burst devices as well.

- Uses a dynamically allocated data structure to hold the editor document and the "Kill Buffer", so there are no fixed limits on the size of each; they both can use all of the memory that is available.

- Will optionally expand TAB characters into spaces while loading a file and compress spaces into TABs while saving.

- Global search and replace.

- Range delete and recall.

- File translation to and from ASCII-CrLf, ASCII-Lf, ASCII-Cr, and SPEEDSCRIPT character codes. ASCII-CrLf is used by MS-DOS and ASCII-Lf is used by Unix.

And here is its major limitation:

- Has a maximum line length of 80 characters. It will split file lines longer than that. The complete version will support lines up to 240 characters and use horizontal scrolling, but that's another day.

And there is also a known bug:

- Don't let the number of bytes free get lower than around 150 or you run the chance of having the internal memory allocate function fail. Most of the routines do not check whether a memory allocate call succeeded, so they proceed as if it did, and all kinds of bad stuff can happen.

==========

Here is the action key summary (an * precedes the keys that have actually been implemented). For the keys preceded by a "CT-", hold the Control key while typing them (duh!), "SH" means Shift and "CO" means Commodore. The UP, DOWN, LEFT and RIGHT keys are the cursor arrow keys. When you have to hold down SH, CO, or CT with a arrow key, use the arrow keys on the top of the keyboard. For convenience, SH-UP and SH-DOWN do the same as CT-UP and CT-DOWN.

ZED-128 Command Key Summary:

Control Commands:

I CODE	KEY	ACTION
* $e0	CT-@	Exchange cursor position with mark position
* $e1	CT-A	Alter case of letter under cursor
* $e2	CT-B	Byte value input
$e3	CT-C	Copy range
* $e4	CT-D	Delete range
* $e5	CT-E	Exit with save
* $e6	CT-F	Find next occurrence of hunt string
$e7	CT-G	Goto given line number
* $e8	CT-H	Set Hunt string
$e9	CT-I	Insert new file into current one
* $ea	CT-J	Juggle range of lines for text formatting
* $eb	CT-K	Kill current line
* $ec	CT-L	Load file
* $ed	CT-M	Set Mark for range operations
* $ee	CT-N	Set Name of current file
* $ef	CT-O	Set Options: input/output translation/tab-expansion, etc.
* $f0	CT-P	Print current file

* $f1	CT-Q	Quit without save
* $f2	CT-R	Recall text from buffer
* $f3	CT-S	Save file
$f4	CT-T	Translation utils:WC,Rot13,Up/Lowcase,Indent,Justify,Prefix
* $f5	CT-U	Use new disk device number
$f6	CT-V	Verify file
$f7	CT-W	Write range with new name
* $f8	CT-X	Exchange cursor character with next character
* $f9	CT-Y	Replace (all the other letters were taken!)
$fa	CT-Z	Goto bottom of screen
* $fb	CT-[	Toggle insert mode
* $fc	CT-£	Toggle modified flag
* $fd	CT-]	Toggle indent mode (Indent / Noindent / WordWrap)
* $fe	CT-↑	Enter Commodore DOS command
* $ff	CT-←	<nothing>

Key Commands 1:

I CODE	KEY	ACTION
* $00	<none>	<nothing>
$01	CT-RETURN	Go up one paragraph
* $02	SH-TAB	Backtab
* $03	STOP	<stop some operations>
$04	SH-HELP	<same as HELP>
* $05	CT-2	Clear buffer
* $06	SH-LEFT	Word left
* $07	SH-LINEFEED	?
* $08	CO-DEL	Rubout
* $09	TAB	Tab
* $0a	LINEFEED	?
* $0b	SH-RIGHT	Word right
* $0c	CO-UP	Goto top of document
* $0d	RETURN	Split current line (indent not yet implemented)
* $0e	SH-ESCAPE	?
* $0f	CO-DOWN	Goto bottom of document
* $10	CO-LEFT	Goto beginning of line
* $11	DOWN	Cursor down
* $12	CT-9	Reverse screen on
* $13	HOME	<nothing>
* $14	DELETE	Delete character
* $15	CO-RIGHT	Goto end of line
* $16	CT-UP	Page up
* $17	CT-DOWN	Page down
$18	CT-TAB	?
$19	CT-LEFT	Page left
$1a	CT-RIGHT	Page right

* $1b	ESCAPE	<nothing>
$1c	CT-3	Directory with block counts
* $1d	RIGHT	Cursor right
* $1e	CT-6	?
* $1f	CT-7	?

Key Commands 2:

I CODE	KEY	ACTION
$80	CT-F1	Function key 9
* $81	CO-1	Set display to 25 lines
$82	CT-F3	Function key 10
$83	SH-STOP	?
$84	HELP	Display help message
$85	F1	Function key 1
$86	F3	Function key 3
$87	F5	Function key 5
$88	F7	Function key 7
$89	SH-F1	Function key 2
$8a	SH-F3	Function key 4
$8b	SH-F5	Function key 6
$8c	SH-F7	Function key 8
$8d	SH-RETURN	Go to next paragraph
$8e	CT-F5	Function key 11
$8f	CT-F7	Function key 12
* $90	CT-1	Clear document
* $91	UP	Cursor up
* $92	CT-0	Screen reverse off
* $93	SH-HOME	Cursor home
$94	SH-DELETE	Insert one space
* $95	CO-2	Set display to 27 lines
* $96	CO-3	Set display to 30 lines
* $97	CO-4	Set display to 45 lines
* $98	CO-5	Set display to 51 lines
* $99	CO-6	Set display to 29 lines
$9a	CO-7	?
$9b	CO-8	?
$9c	CT-5	Display code of current character
* $9d	LEFT	Cursor left
$9e	CT-8	?
* $9f	CT-4	Display directory with byte counts

==========

To delete a range, use CT-M to set the mark for one bound of the range and move the cursor to the other bound of the range. Then press CT-D to delete. The range includes both bounding lines. CT-K (kill current line) is the same as pressing CT-M and then CT-D on the same line.

CT-R recalls the text at the current cursor line. To recall after the end of the document, add a new blank line to the end, recall, and then delete the extra line you added. You can recall the kill buffer text as many times as you wish.

To search or replace, use CT-H to set the string to hunt for, and then use CT-F (find) or CT-Y (replace). Zed searches in a case INSENSITIVE manner. Thus, "STRing" will match with "sTrInG".

The status line on the top of the screen displays the current file line, the cursor column, a flag ("*") indicating whether the file has unsaved changes, Insert and Indent mode flags, the number of bytes the document uses, the number of bytes free, the current device number, and the document name.

The options setting feature (CT-O) provides a full-screen interface. You move the field cursor among the various fields on the screen with the cursor keys. When the cursor is on a field that you want to change, press the RETURN key. If the field is a numerated field (like Read Translation Mode), the value will change in a wrap-around fashion. If it is a numeric field, the character-cursor will flash and you are to type in the new value and press RETURN. DELETE is the only editing key. When you are finished setting (or viewing) the options, press ESCAPE, CT-O, or SPACE to exit back to editing mode. If you re-save the Zed program by exiting back to BASIC and using DSAVE, all of the options settings will be saved and will be set when you run the program in the future.

There are some fields (like the function keys) that are not used and do nothing. The color fields allow you to set the colors for the various items on the editing screen. Color changes take effect when you exit from the Options screen. Read and write translations take effect when you are loading (CT-L) or saving (CT-S) a file. The possible translation values are: None, ASC-CL (MS-DOS - lines end with Cr Lf), ASC-Lf (Unix - lines end with Lf), ASC-Cr, and SpdScr (Speedscript - which uses screen codes and back-arrow for return). Tab expansion will convert the TAB character into the equivalent number of spaces when reading, and TAB compression will replace a number of spaces with the TAB character whenever it can to make the file shorter. TAB compression is very effective on indented program files or other files that have a lot of spaces in them.

There is a field for disabling the use of burst mode when reading files. Zed will auto-detect whether a device is Fast or not, but this option is provided in case the auto-detection fails for some odd device that you have. When Zed detects that a device is Slow or if the burst option is disabled, only the standard Kernal routines are used for reading files. I haven't had the opportunity to check Zed out on the line of CMD products, but I'm sure I will be catching some flack if Zed does not work on them.

The text line length and tab spacing fields show a value but they are not implemented in the rest of the program. The cursor delay and repeat characteristics can be set with the fields of the same names. The time units are in jiffies (1/60th of a second, but you should know that!). Experiment with these to determine what you like the most. I wrote my own custom key-scanning routines, so I was able to easily provide these parameters. You may also notice that I fixed the problem of the kernal mistaking Port#1 joystick movements for keystrokes. In the future, I may implement a two-key rollover.

A field is also provided for setting the maximum amount of REU memory that Zed is allowed to use. The default is 127 Banks, so Zed will use up to 8 Megs of expansion memory (if you have it). If this field is set to 0 Banks, Zed will leave your expansion memory completely untouched. The value in this field only takes effect when Zed is started up, so you will have to exit back to BASIC and RUN Zed again after changing it.

Finally, fields are provided for the printer DeviceNumber, SecondaryAddress, and TranslationMode. The default is Dev=4, SA=7, Trans=None, which is the Commodore standard. Since I have a Panasonic IBM-PC compatible printer with a SuperGrafix J. interface, I use Dev=4, SA=5, Trans=ASC-CL. This way, I can print the £ ╬ ╟ ═ and characters.

To use the Juggle lines feature (CT-J), set the mark (CT-M) to the first line of the first paragraph to juggle and move the cursor to the last line of the last paragraph to juggle and press CT-J. After juggling, the display will always go to the last line of the last paragraph. There are three objects that Juggle concerns itself with: paragraphs, sentences, and words. Paragraphs are delimited by one or more blank lines, words by one or more space characters or by a new line, and sentences by a period, question mark, or exclamation mark. If the first word following a ".", "!" or "?" does not start with a capital letter, then the previous word was not the end of a sentence. This definition of sentence does not always work (eg. "Dr. Bruce"), but it works most of the time.

The reason that distinguishing sentences is important is that when juggling a paragraph, words can be ripped from the start of one line and put onto the end of a previous line (or spill forward) and most text files do not contain a correct number of spaces following the last word on a line. Juggle will put one space after each word that gets ripped and two at the end of a sentence. Words that don't get ripped from one line to another will retain their original spacing. Juggle eliminates spaces after the last word of a text line. The target line length is selected by the "TextLineLen" field on the options screen. Don't set this field any higher than 8 characters. Juggling speed is approximately 1350 words per second. These last two paragraphs were juggled.

The TAB key will move the cursor to the next tab stop and will extend the line with space characters if necessary. The number of characters between tab stops is selected by the "TabSpacing" field on the option screen. Note that this setting does not change the tab spacing for the file read/write tab expansion/tab compression features; they always use a tab spacing of 8.

==========

Please send any questions, comments, or suggestions to me at the below address, even if they are just an "Awesome Dude!" or a "Sucks Rocks!".

I am a Ph.D. student in Computer Science at the University of Waterloo in Canada. I have been a die-*HARD* Commodore enthusiast since I first got my VIC-20 ten years ago. Until I started Zed, I didn't have an acceptable editor for all the text files I deal with since I was exposed to the Internet, so the light bulb went off above my head.

My plans for Zed are to rip it apart and start it over (in assembler this time rather than raw machine language) and make it for the ACE programming environment (which is still under construction). ACE (when it is more complete) will run on either the 64 or the 128, so Zed will someday work on a 64 too.

Zed-128 is Public Domain Software.

==========

The CRC32 of "ZED-128.077" is 3571615286.

==========

-Craig Bruce
csbruce@neumann.uwaterloo.ca
"Shit will always happen, but shit will never be rushed."

==========