NOTES FOR CURSOR 4
October 1978
Ron Jeffries, Publisher
Glen Fisher, Editor

CURSOR™
Programs for PET™ Computers

"Ballet never becomes easy. It just becomes possible." - Agnes de Mille

THE CODE WORKS
Box 550
Goleta, CA 93116
805-683-1585

A CURSORY GLANCE

The road to hell, it is said, is paved with good intentions. Well, I guess I know where I'm headed, because I sure had good intentions about getting #4 out early! This one is so late that it is almost crazy to call it the "October" issue, but we will anyway.

We must have done something right with CURSOR #3, because we've gotten lots of very nice letters. Seems that lots of people tried the sound effect on QUIX!, and became addicted. Well, you ain't seen nothing yet! Glen Fisher has produced an outstanding musical cover this month. In fact, if you don't hook your PET up for sound for this issue, you will lose out on two programs: COVER! and CLOCK! See the Notes for CURSOR #3 for technical details on how to hook up sound (Pin M and 12 on the user port). Radio Shack sells a nice little 200mw Speaker/Amplifier, catalog number 277-1008, costs about $11, looks like a small radio. It comes with a short shielded cable - all you need is an edge connector for the PET, and two wires soldered to that connector. Hey, someone out there thought we meant for them to actually solder right to the connector that comes out the back. No way! You should buy the correct slip-on connector (female). It really should be "keyed" so it can only be put on in the correct direction; then you solder to the pins that come from this slip-on connector. Soldering stuff on the PET board has been known to cause the ghost of Chuck Peddle to cringe, even if he has defected to APPLE.

Thank you again for your support. We plan to start national advertising in the near future. Based on the best information now available, I expect that our subscription price will have to increase before too long. (Actually, my accountant is standing here with a .45 pointed at my head, forcing me to write these words.) We will try to maintain a very competitive price, but we need to allow for a bigger advertising budget; and we'd like a little more flexibility in what we can afford to provide to you. So, tell your friends that prices will be going up at the beginning of 1979.

SERIOUS USES OF SMALL MACHINES, Revisited

My comments about so-called "serious" uses of PETs and other creepy crawly computers kept the mailperson busy last month. I guess we must have struck a nerve: you go and spend about 800 perfectly good dollars for a nice looking little box. Your mother is impressed, your kids are impressed; but your spouse says "$800 would sure buy an awfully nice: (stereo, microwave, dishwasher, waterbed, used car, oven, etc)." But you say you were going to use your PET to keep your inventory or general ledger or payroll or whatever, right? Then you found out that it takes lots of storage, fast non-tape storage like double-density floppy disks, to store enough stuff for these applications to be practical. Then there is the little matter of "hard copy", and the sad story of the $595 PET printer that may never be built; or when it is available, it may cost $995.

OK, this all sounds kinda grim, and I think in some cases it is: people are being sold computers that just plain won't do what they need for a specific application. (My sources tell me that Radio Shack has this problem, and has put out the word to their dealers to "cool it" with over-enthusiastic but naive customers.) I'd still like to hear from those of you who are doing honest work with these machines. Let me know, and I'll be glad to share your experiences with our readers.

About the biggies: Texas Instruments will probably be ready about next summer. Please don't be surprised if they bring out a machine that has built-in reliable mass storage using "bubble memory". Also, expect to see both the Basic and Pascal as languages on the TI machine. Or look for an IBM personal computer (not to be confused with their 5110, which is small, but expensive). IBM might emphasize the APL language. It is not clear that anyone at IBM even knows how to spell Pascal, much less market it.

CURSOR #4 HAS THESE PROGRAMS: (Program names ending with '!' use CB2 sound)	
COVER04!	A delight to your eyes and your ears. Notes dance around the keyboard as the three musical pieces play. Songs are: Wiltshire reel, the fiddle tune "Devil Dream", and a Sicilian tarantella. By Glen Fisher
BOP	Chisanbop - the Korean technique of counting to 99 on your fingers. By Glen Fisher
CALC	Everything you ever wanted in an $800 calculator. Hex, octal, decimal. Integer math only. By Glen Fisher
CLOCK!	An $800 digital alarm clock that also chimes on each quarter hour. By John Fox
INP	A short but powerful subroutine for use in your programs. Designed to be easy to merge with your code. Solves your input problems in elegant fashion. By Glen Fisher
CED	The Cursor Editor. Allows you to enter, edit, save and load three screens of text. (Old ROMs only) By Glen Fisher

Distributed in Japan by:
SYSTEMS FORMULATE Corp.
Shin-Makicho Bldg., 1-8-17
Yaesu, Chuo-Ku, Tokyo 103

Distributed in England by:
AUDIOGENIC Ltd.
P.O. Box 88
Reading, Berkshire

MORE ABOUT THE PROGRAMS

BOP... We use some "flashy" PET graphics to teach the Chisanbop method of counting. In Chisanbop, you press your fingers against a table to "make a number". Your right fingers are each "worth" one; your right thumb is worth five. Your left fingers are each worth ten, and your left thumb is worth 50. There are four options; all are obvious except the one in which you "set" the fingers to produce a number. Use ">" to move the pointer to the right, and press RVS to change a finger. When you have all the fingers set, type [RETURN]to let the program know you are done.

CALC... This calculator has a display where you enter the number, and it has an "A" register where the results of each operation are stored. CALC also has a single "memory": you store numbers by pressing [S], and recall numbers by pressing [R]. There is a global number base, which starts out as base ten. You can change it by pressing the [#] key, followed by [H] for hex, [O] for octal, or [T] for base ten. To convert from one number base to another, set the global base as the base to which you want to convert, and then use our "local base" feature to enter the base and number from which you want to convert. You do this by simply prefixing the number with the appropriate base (e.g. to enter a hex number when the global base is set to decimal, type [H255] then an equal sign). You can use addition, multiplication, division and subtraction in the obvious fashion. Other operations:

[DEL]	Clear entry
[INST]	Clear all registers
[#H]	Set global base to hexadecimal. (#O for octal, #T for decimal).
[S]	Store contents of the A register in memory register
[R]	Recall memory
[HOME]	Change sign of entered number
[^]	Copy the A register down to the display
[<]	Show value of the high byte of the A register
[>]	Show value of the low byte of the A register
["]	Peek at the word at the location given by the A register
[']	Peek at the byte at the location given by the A register

CLOCK!... This one is pretty self-explanatory. Set the time in four digits and press [RETURN]. You can have music on command by pressing "!", and chimes by pressing "<". Alarm music is, appropriately, "Reveille".

INP... As many people have noticed, the INPUT statement has several undesirable features: it won't accept commas or colons as input (even for strings!); it prints nasty messages at you if you type too much; it can only accept 80 characters of input; and it will drop you out of the program if you type nothing but [RETURN]. INP is our answer to those problems. INP lives at line 60000, and can be called upon to do input for you at any time. It is explicitly public domain, so you can use it freely in all your programs. (We ask only that you include an REM saying, "INP routine used with permission of CURSOR, Box 550, Goleta, CA 93116".)

There are two parts to using INP: adding it to your program, and using it once it's there. As it must be part of your program before you can use it, we'll describe that first. The steps to follow to add INP to your program are these:

1. Save your program, if necessary. (We're about to erase it from memory!)
2. Load INP into your PET.
3. Position tape that has your program on it just before the program starts.
4. Clear the screen, and LIST the INP routine.
5. Move the cursor to right over the word "READY". Type "LOAD" (notice the two trailing blanks). Don't press [RETURN] yet. First, press [PLAY] on the recorder; then press [RETURN].
6. After your program has loaded, press [HOME] to get to the top of the screen. Now, press [RETURN] 12 times (once for each line of INP).
7. Now, save the results on another tape. (This all sounds hard—it isn't.)

INP Routine, continued: There are two more things that must be done for INP: Assign CHR$(13) to the variable CR$; INP prints out whatever is in CR$ when [RETURN] is pressed. Setting CR$ to CHR$(13) makes the cursor go to the next line when [RETURN] is pressed. (If you assign " " to CR$, the cursor won't move when [RETURN] is pressed.) Also, if you've got a PET that types upper case when in lower-case mode, set the variable FL to 1. This makes INP switch upper and lower case on input, so that you'll get upper case only when you press [SHIFT]. (Keep in mind that your lower case is a newer PET's upper case, and vice versa.) Finally, beware of using variables whose names start with the letter 'Z'. INP's private variables all have names starting with 'Z', in an attempt to avoid your using any of your variables accidentally. (If you don't care if INP mangles one of your 'Z' variables, feel free to use them. Just keep in mind that INP may use them, too.) Also remember that INP uses FL; if you change FL, INP may start acting a little oddly (it happened to us, it can happen to you).

Now that that's all done, we can do input! Whenever you want to input something, PRINT your prompt (with a semicolon at the end!), and GOSUB 60000 to do the input. When INP returns, the variable IN$ will hold the string that the user typed in. For example:

```
10 CR$=CHR$(13)
20 PRINT "WHO'S PLAYING AGAINST ME? ";
30 GOSUB 60000
40 PRINT "OKAY, ";IN$;", YOU'RE GONNA GET TROUNCED!"
50 BLACK$=IN$
```

Note that the PRINT printed a question mark. INP itself prints no prompts of any sort. It just sits there blinking a grey square at you. (It shows a grey square so you can tell it from an INPUT statement.) You'll find that INP won't accept anything that isn't visible (that is, [UP], [RIGHT], [RVS], [HOME], and so forth). However, it will let you correct mistakes: if you press [DEL], the last character on the line will be deleted. Pressing [SHIFT-RETURN] will throw away everything you've typed so far, so you can start over. To tell INP that you're done typing, press [RETURN]. (Those three keys act the same way as they do for the INPUT statement.) The usual troublemakers (quotes, commas, and colons) will cause you no problems at all. INP will calmly hand them back to you in IN$.

CED... (Old ROMs only!) With this simple little text editor you can enter and modify three PET pages of text at once (well, you actually work on a page at a time, but they are all available to you). When you first run the program, it will tell you that it is off doing some work. When it is ready for you, it will display the little "checkered flag" cursor. To enter the text, just type it in. The cursor control keys all work, except the [CLR] key which has been disabled to protect you from wiping out a bunch of typing by accident. ([CLR] does provide a home-to-left-lower-corner function, however.) [INST] and [DEL] insert and delete text. [DEL] works on the character under it, instead of the one to the left, as the PET does. These keys affect the entire page of text (i.e. lines will wrap around, etc). There is also a "local" or line-oriented insert key, the " " key, and the "backarrow" is the line delete function.

To give commands to CED, press the [pi] key " ". The program will store away the first line of text, and then give a number, followed by a colon (e.g. 2: means "You are working on page 2"). You can press [RETURN] and go on typing or you can give a command.

COMMAND SUMMARY FOR CED

[S] **SAVE** a file. It will ask you for a filename. BE CAREFUL ABOUT WHAT TAPE IS IN THE DRIVE!

[L] **LOAD** a file. If you can't remember the name, type [RETURN], and it will automatically load the next file on the tape.

[D] A shifted [D] puts you in graphics mode. Unshifted [D] gives you upper-lower case.

[PI] The pi key first gets you into command mode. Then, if you press it again, you can redefine this "escape to command level" character. The only reason to do so would be if you wanted to enter the pi character as part of your text.

[QUIT] Does just what it says. You are warned to save your file if you haven't. (You get one chance. After that we assume you are a big kid and know what you are doing!) It is quite important that you only quit via the [Q] command. Otherwise, turn your machine off, then turn it back on to clear some pointers that we diddle with. Or you could type: POKE 134,0: POKE 135,32. Also, if you want to re-run the CED program without reloading the tape, type [CONT] rather than [RUN] after you have quit. Otherwise you will get an "out of data" error.

[NUMBER] At the command mode, you can go to another page by just typing the number of that page. So, if you have typed all of page 1, and want to type on page 2, then you would type [" "], followed by [2].

[CLR] At the command mode only, [CLR] will clear your screen. (We made it hard on purpose — we sometimes need to be protected from ourselves.)

HACKER HINTS

You can "escape" from our fancy input routine INP by pressing the [RUN] key. [STOP] doesn't work, however.

As every PET programmer knows, Commodore didn't provide a DELETE command to get rid of a bunch of lines. You can use a little one-line program to help solve the problem. Say you want to get rid of lines 10 to 100. Type: FOR I = 10 to 100: STEP 10: ?I: NEXT. It will print a bunch of numbers on the screen. Just press [HOME], then hit [RETURN] about 24 times, and you won't have near as many lines to kick around.

After you read a tape, you can find out how much trouble the PET has reading your tape by telling it to print the number of "drop-out" errors it encountered. On "Old ROMs", type: PRINT PEEK(630). On Basic 3.0 or 4.0 ROMs type: PRINT PEEK(192).

If it says zero, you had a perfect load of the tape. Otherwise, divide the number reported by 2 to get the number of tape drop-outs. We find that anything over 8 will be a problem. (It doesn't have to be zero, however. The PET recording method puts two identical copies of your program on the tape. Since there are two copies, a certain number of errors can be corrected by the PET as it loads your program.)