LET'S DEFINE SUPERPET AND WHAT IT CAN DO

SuperPET (in these columns hereafter, SPET), isn't exactly well defined by the manuals which came with it. How, for example, do you clear the screen from program? How do you print a directory to your printer from the upper 64 (6809 mode)? What is the precise coding (with examples) for setting up relative files? How do you control the cursor? What do the various codes do (we can call them ASC II Codes, but they aren't), in immediate mode or within a program, to the screen and to the printer? And how to these problems and solutions vary with the language used? The questions are endless.

What is our objective? To define the machine. We'll need all the help we can get from SPUG (SuperPET User's Group) members, from Commodore, and from anyone else willing to help, and we must secure the help of Associate Editors, with the skill and knowledge to review, edit, and submit articles in the various languages: Assembly, FORTRAN, PASCAL, COBOL (yes, the new language disk gives us COBOL), and APL. Volunteers are wanted: please be not bashful.

The present editor can speak MicroBASIC and BASIC, and has a long background in word processing (at which, by the way, SuperPET, without any software at all, other than the language disk sent with machine, is remarkably good). An assistant editor in MicroBASIC is welcome; the editor is overwhelmed by the simple problems of production and distribution.

Volunteers: please send name, address, area of specialty, and any key comments on your talents to: Editor, SuperPET Gazette, PO Box 411, Hatteras, N.C. 27943. Your active help is solicited; you will receive full credit for all material submitted or edited.

Meanwhile, we submit useful data for reference.

MicroBASIC: The Keyboard and its Codes

The keyboard can be addressed as "keyboard" in an open statement, as in some programs which follow. Later, in listing 1, is a program called "get keyboard", and it does just that: printing to both screen and to printer the name of the key and the SPET code which that key generates. The little procedure which prints the data should work with any printer to give a two-column printout.

Type the program in and run it. You'll be surprised. The ESCAPE key doesn't print chr$(27), which is ESCAPE, but chr$(6), which erases from cursor position to end of screen line. Instead of the graphics symbols you're accustomed to using in BASIC 4.0 for cursor control, note the codes you can use. (You have much more powerful cursor control available, however).

When you print the keypad, you learn that the shifted keypad prints different code than the unshifted pad, and that a whole set of graphics symbols is available from the shifted pad.

Summarized below are the some of the handiest of the chr$ codes you will often use: (buffer effect is what happens if you stuff code in the keyboard buffer and use a get# statement to fetch it).

chr$	Screen effect	Printer effect	Buffer effect
1	Homes the cursor	?	Same
2			Runs program
3			Stops program
4	Deletes rightward	?	Same
5	Inserts a space (from keyboard only)	? None	None
6	Erases to end line		Same
7	Cursor right	?	Same
8	Cursor left	?	Same
9	Tab	?	Same
10	Cursor Down	Extra linefeed	Same
11	Cursor Up	No linefeed	Same
12	CLEARS SCREEN, HOMES CURSOR	Feeds paper to top of form	Same
13	Carriage Return	Carriage return & linefeed	Strange! (See comments)
127	Backspaces, deletes	?	Same

Some short demo programs are enclosed, to show what happens when the codes above are used in program, both alone and with the keyboard buffer. Printer effects probably depend on the printer used. Those above apply to the Diablo 630, (that used is the same as the Commodore 8300). If you use any Commodore printer but the 8300, change the "ieee4" to "printer" in open and close statements. When testing cursor up and down effects on the printer, suppress the carriage return at the end of a print line with a semicolon, or the carriage return will mask the effect of the cursor command.

One important difference between SPET and older CBM machines is that you can get the keyboard input either as a string or as a numeric variable. If you use "get i", for example, a carriage return will return a 13; if you use "get a$", the carriage return key will return a null string. The program in listing 1 can be run both ways, with a few changes. If you decide to write a program to automatically print out characters and codes, be advised that the printable characters start at chr$(32) (which prints a space), and run through chr$(126); then start again, in reverse video, at chr$(160), and finish at chr$(254).

The graphics codes (also on the shifted keypad) start at chr$(129) and end at chr$(139). Chr$(140) prints a hollow tooth, and chr$(160) a reverse video space. Chr$(0) and some other codes print a small square.

Except for the graphics codes above, some undefined codes lie between 127 and 160, and there has been no time to test their effect. Could we get a report?

Note that the reverse video characters lie 128 above their normal counter-

parts (no suprise); you can print in reverse video with a simple procedure which gets i from the keyboard, and prints chr$(i + 128). Be sure it rejects any i below 32 or above 126, or a mellofahess results. If you take this precaution, your printer will never know the difference (or at least Diablo 630 doesn't).

We should not call the screen and keyboard codes we use ASC codes; they aren't strictly; Waterloo calls the codes the "ordinates", and has a substitute name for them in MicroBASIC--ord(s$). Instead of asking (as we used to) ? ASC("s"), you now ask ? ord("s").

When you stuff material into the keyboard buffer and call for it, you must allow for spacing between numbers. If you stuff in a the number 1, you find on printout of "i" (the chr$ number of each character in the buffer) that a space (chr$(32)) has been inserted between each chr$(49). Yet, if you stuff the buffer with chr$(49), the code for #1, and print a$, you will print out 39 "1's" in a row, single-spaced. This is no surprise, but it must be allowed for. Buffer capacity apparently is 40 characters (though I've printed only 39 no matter how many are stuffed in). Anyway, in program listings is a screen dump to printer for SPET which uses the keyboard buffer to store RETURNS. You must stuff two in to make it work; take one out and the program will not dump. (The procedure was invented to print directories from the upper 64; the updated language disk now gives us this capability, but is no substitute for dump as a utility procedure.)

Using Procedures in Immediate Mode

Proc dump is a procedure designed to dump the screen, or any part of it, for any purpose--program or text. I've used it extensively in this text to dump a program from the screen to the printer and then to comment on it. It's a lovely thing to be able to write a program, run it, stop it anywhere, dump what's on the screen, make your comments on it, and then type "cont <RETURN>", be back in program, and able to continue your run. For debugging, it can't be beaten.

Proc dump, like any procedure, can be tucked into RAM and just sits there until you call it. If you give it high line numbers, you can write a program under it without interference. Dump, like any procedure on disk, can be tacked onto the end of a program in RAM by using the MicroEditor. Just put the screen cursor on the last line of your program, and type: get "dump" <RETURN> on the command cursor at the bottom of the screen. When it's appended, call it when you need it. You can delete it (or any other utility procedure) either with the standard del XXXXXX- command in the regular editor, or the YYYYY,$ d command of the MicroEditor (XXXXX standing for the starting line number in the regular editor, and YYYYY for the RELATIVE LINE NUMBER in the MicroEditor, $ being the code for end of program, and d code for delete).

Once you are used to having utility procedures in RAM while working, you'll find them invaluable in immediate mode as well as while running programs. Dump, for example, when called, lets you clear the screen and type any text, and without line numbers. Just don't press RETURN. Use the Tab or cursor keys to get back to the left margin.

You can dump up to 24 lines. If you want the whole screen, use cursor to put the first line to be printed on screen line 1. Go to line 25. Type: call dump <RETURN>, and the whole page will dump to the printer (minus Ready).

Erase whatever you want before printing. Edit to heart's content. Just don't press RETURN.

If you want to stop printing on a certain line, clear the next line and type: quit (no quotes, no RETURN).

Then type: call dump <RETURN> on any lower clear line, and the procedure will dump down to quit, and stop (without printing quit or Ready).

A word of warning on ALL utility procedures like dump: be sure the open#'s you use are high enough NOT to interfere with any open#'s you use in a regular program, or +!#$%&! I reserve the following "logical file numbers", as Waterloo calls them (hell, they're just telephone numbers!) for utility procedures:

#9	disk error channel	#10	terminal
#11	disk input	#12	printer
#13	keyboard	#14	disk output

What numbers you reserve doesn't matter, but reserve some which ARE NOT USED in your regular programs. Yes, you can have several channels open to the printer, disk drive, terminal, etc. at the same time, just as my rich Uncle George can have three telephone lines open, and be talking on them, all at the same time.

There are two other precautions: (1) watch out for the variables in your utility procedures. They can modify variables in your main program. Here, habit helps again. All my utility procedures use identical double letter variables (ii, jj, aa$, etc.), and such variables are NEVER used in main programs; (2) calling a procedure, unlike a RUN, does not zero out or null variables. This you MUST do in the procedure, or the next time you call it #$%&/ breaks out.

Program listings, comment, and news are on the following pages.

****

Newsletter published by the SuperPET User's Group (SPUG): editorial offices at PO Box 411, Hatteras, N.C. 27943. Secretary, Paul V. Skipski, 4782 Boston Post Road, Pelham, N.Y. 10803. Membership applications and inquiries to Mr. Skipski. All newsletter material to Hatteras, attn: Dick Barnes, Editor.
 Enclose a self-addressed, postpaid envelope with all material submitted and all inquiries requiring reply. Membership: $10.00 per year. See enclosed application form.

Program Listings, News, Comments

```
62000 ! Screen dump for SuperPET: title: "dump"
62010 proc dump
62020 open#12, "ieee4", output
62030 open#13, "keyboard", inout
62040 aa = 1
62050 for ii = 1 to 24
62060   print#13, chr$(13);chr$(13) ! stuff keyboard buffer with two CR's
62070   if cursor (aa) then get#13,cr$
62080   linput "", aa$ ! quotes suppress question mark
62090   if aa$ = "quit" then 62130
62100   print#12, aa$
62110   aa = aa + 80
62120 next ii
62130 aa=0 : ii=0 : aa$="" ! clear variables; calls do not clear them.
62140 close#12 : close#13
62150 endproc
```

The procedure was printed by dumping it from the screen. These comments are likewise dumped. Please note that the quotes in line 62080, following the linput statement, are essential to keep SPET from printing a question mark.

It may be bad practice to jump out of the middle of a for-next loop with a goto, leaving some poor pointer somewhere wondering what happened. So far, no trouble from line 62090. If pointer troubles should crop out, you can always use a separate for-next loop to stuff the CR's into the buffer (as was done original ly), but you'll have to call it twice--you need 48 RETURNS to print 24 lines; the keyboard buffer won't take more than 40.

If somebody can tell ye ed why the procedure requires two RETURNS for each line, the answer will be welcomed.

The editor apologizes if some material is basic, but the experts can skip what they already know; beginners can't.

Program Listings, News, Comments

SOFTWARE AND MANUAL UPDATE: Two new language disks, plus manuals and changes to old manuals, version 1.1, for SPET have been announced by Waterloo Computing Systems, Ltd., and by Commodore. All new SPETs are to be supplied with the new material, which is reported to include:

1. Enhancements to microAPL, including the)PCOPY command, changes to permit sorting of the rows of a character matrix, better IO error control, and others.

2. The addition of a MicroCOBOL Interpreter, a tutorial and reference manual. One of the disks apparently is a COBOL tutorial.

3. Improvements to MicroBASIC, in particular one to allow statements to continue for several lines: for example, a long print statement with many long variable names. In addition, improvements to the CHAINing arrangement, to pass variables and matrix values, and TYPE command to allow files to be displayed.

4. MicroPASCAL has received three new functions to improve ability to call assembly language subroutines.

5. Last, but far from least, the microEDITOR now allows the user to print a directory to the printer, and the SETUP facility is reported improved.

Commodore reports that the two diskettes, replacement pages for the microEDITOR, MicroBASIC, MicroPASCAL and MicroAPL manuals, and the MicroCOBOL data are available as a package, from dealers, at $29.95. Apparently part of the packet is available for $9.95, but it is not clear at this writing exactly what. We'd welcome more information, and user reports.

* * *

TROUBLE FIGURING OUT ANY OF THE FOLLOWING? You can get copies of a technical letter on the subject by writing to:

Program Listings, News, Comments

Following is a MicroBASIC program to define the "ordinates" of the keyboard. You may type the name of any key (or combination, such as Shift Clear), press the combination of keys, and the program will print to screen and to printer the name or names of the keys, and the chr$ number (ordinate) for the key or combination of keys.

```
100 !                    Listing 1
105 ! A program to define the keyboard : title: get keyboard
110 ! To get out, press Home (chr$(1))
115 open#2,"keyboard", input
120 open#3,"ieee4", output
125 loop
130   print chr$(12) ! clears screen
135   input "Type name of key you will press; no quotes: ",a$
140   print :  print "Press it."
145   get#2,i
150   if i = 0 then 145
155   call define
160   if i = 1 then quit
165 endloop
170 print #3
175 close #2 : close #3
180 stop
185 !
190 proc define
195   g = g + 1
200   g$ = " "
205   n = 35 - len(a$) - len(value$(i)) ! Counts spaces left from text
210   print a$;" =";i
215   for j = 1 to 500      ! Screen delay loop
220   next j
225   print#3, a$;" =";i;rpt$(g$,n); ! puts in extra spaces with rpt$
230   if g = 2 then print#3  ! return printer to left margin
235   if g = 2 then g = 0
240   a$=""
245 endproc
```

On the page following is a second program which demonstrates very simply the employment of the ordinates which were listed on page 2 of this issue, as well as showing how to stuff into and use the codes stuffed into the keyboard buffer...

Program Listings, News, Comments

```
 10 ! a program to demonstrate chr$ effects : "try codes"
 20 print chr$(12) ! clears screen; homes cursor
 30 if cursor(1720) then print "1720"; chr$(1); "Home?"
 40 if cursor(81) then print chr$(6) ! erases screen line 2
 50 open#2, "keyboard", inout
 60 print#2, chr$(3)   ! stuff stop into keyboard buffer
 70 print "If you wish to continue after program stops, type cont RETURN"
 80 get#2, st$   ! fetch stop from buffer
 90 print chr$(12);"Program continues after you restart it."
100 print : print "Let's try a rightward delete, chr$(4)"
105 print : input "When ready to delete, press RETURN ", o$
110 for i = 1 to 40
120   if cursor (1) then print chr$(4)
125   if cursor (161) then print chr$(4)
130 next i
140 print : input "When ready to try a leftward delete, press RETURN ", o$
150 print chr$(12);"Delete trial mit chr$(127) and semicolons";
153 for j = 1 to 300 ! screen delay loop
155 next j
160 for i = 1 to 42
170   print chr$(127); ! semicolon here and on line 150 suppress CR
180 next i
190 print "Now we test tab, chr$(9) and cursor commands." : print
200 for i = 1 to 8
210   print chr$(9);"Tab";
220 next i
230 print
240 for j = 1 to 10
250   print chr$(7);j; ! cursor right
260   print chr$(10);j;  ! cursor down
270 next j
280 close#2 : stop
```

PLANNED FOR FORTHCOMING ISSUES

--Variation on the Theme: SPET's Powerful Cursor Control

--SPET's Stand-Alone Capacity as a Word Processor

--PROC PRINTALL: A utility procedure which prints anything printable from any SPET sequential disk file, at any page length, any line spacing, and to your desired printer margin. It provides a screen proof before you print (if you want one), and lets you print to the printer any line, lines, or any page of that proof as desired.

--An Evaluation of the Updated Software

WANTED: Articles defining SPET--in all its languages.

ARTICLES AND SUGGESTIONS WANTED FROM READERS

All articles must be liberally illustrated by example: what may be clear to the writer often is a dismal disaster to the reader if shorn of examples.

1. How to set up and use relative files (ILLUSTRATED!).

2. A clear exposition on the metacharacters in the microEDITOR; we have complaints that reader's cannot get many of them to work.

3. Using SPET graphics (we can draw useful graphics, but have found utterly no way to save what is drawn, except in the form of a program).

4. Useful utility procedures, with instructions for their use (similar to Dump in this issue).

5. Announcements of available LONG programs in any SPET language. Tell us briefly what they do; we'll publish the summary, your name, address, and terms. We do not have space to print long programs.

6. SUGGESTIONS ON WHAT YOU WOULD LIKE TO READ: We'll ask for articles on the subjects most sought.

7. QUESTIONS: If we cannot find the answer, we'll publish your name, address, and the question in ADVICE NEEDED! and ask whoever answers you to send us a copy of that answer. If it's important, we'll publish the answer.

8. Summaries, in any SPET language, which answer concisely the questions left unanswered by the manuals. Confine yourself to one area, and write the bible on it.

All articles and programs submitted for publication, and all letters requiring reply, must be accompanied by a self-addressed, postpaid return envelope, or we cannot promise either return or reply.

We will accept copyrighted material only if permission is granted to publish, in writing, by the person holding copyright, at no charge; and will properly protect and give notice of that copyright in publication, but accept no further responsibility for the subsequent use of the material after printing and mailing.

NOTICE TO APL USERS: A suitable APL wheel is available for the Diablo 630 printer (Commodore 8300P). We prefer to receive APL material pre-printed. The plastic wheel is Diablo PN 38150-01, and the metal PN 311951-01. See COMMODORE magazine, p. 14, August/September issue.

Any material submitted on disk, on any matter, must be in 8050 format. The editor is equipped with WordPro 4+, and can also use text prepared by the microEDITOR of SPET, (in no language); page width should be 80 characters (which just fits the screen, and the 12-pitch wheel we print with).

SOME SUGGESTIONS ON HOW TO MAKE SPUG GROW

1. Get copies of the SPET Gazette to your Dealer; get from the dealer the names and addresses of SPET owners in your area. Send them to the editor, PO Box 411, Hatteras, N.C., 27943. We'll send a complimentary copy of the Gazette.

2. Get help from the nearest PET club (see the listing in the August-September issue of COMMODORE magazine). Send us names and addresses of SPET owners.

3. We're in touch with COMMODORE magazine and with a group of SPET owners in England, have associate memberships in the Toronto Pet User's Group (TPUG); and will do our part to spread the word. Get local publicity!

4. Please join SPUG; the work and cost of this issue were borne by a a few of us. Give us a hand! We need your skills, articles and help, not just money for paper, printing and postage.

The SuperPET Gazette
PO Box 411
Hatteras, N.C. 27943