SuperPET Gazette

A WRAP-UP ON TELECOMMUNICATIONS
ISPUG RELEASES MASTER DISK FOR BOTH 6502 AND 6909 TC

We're proud of the package we announce this issue, on disk, in either 4040 or 8050 format: ISPUG's master telecom. We print a commented directory below (and managed to get everything on one 4040 disk).

9	"data.rs232:e"	SEQ	Location and connections on RS-232 port in SPET.
33	"telecom.larson:e"	SEQ	Jeff Larson: article on TC
35	"telecom.frost:e"	SEQ	John Frost : article on TC
3	"fasterm"	PRG	The FASTERM machine-language program. 6809.
17	"fasterm.inst:e"	SEQ	Larson's instructions for FASTERM.
31	"gen.instruct:e"	SEQ	General instructions for TC and for FASTERM.
1	"filetosend"	SEQ	A test program for FASTERM & NEWTERM.
20	"control_codes:e"	SEQ	An explanation of Control Codes and their use.
4	"index:e"	SEQ	This index.
4	"newterm"	PRG	The NEWTERM machine-language program. 6809.
46	"newterm.inst:e	SEQ	User instructions for NEWTERM
20	"note:e"	SEQ	Some general notes.
35	"telecom.6502:e	SEQ	John Frost on 6502 telecommunications.
43	"bbs.instr:e"	SEQ	Six pages, instructions, 6502 Bulletin Boards.
47	"bbs.instr.1:e	SEQ	Second installment, BBs.
43	"bbs.instr.2:e	SEQ	Third installment, BBs.
45	"bbs.instr.3:e	SEQ	Fourth installment, BBs.
4	"bbs.instr.4:e	SEQ	Fifth installment, BBs.
14	"SUPERCOM.BAS6"	PRG	The BASIC 4.0 telecom program, mod 6, for 6502.
3	"SUPERCOMX"	PRG	ML telecom program, loaded by SUPERCOM.BAS6.
7	"TERM.RS232"	PRG	The RS-232 ML program used by SUPERCOM.BAS6.
6	"SMART.DRIVER"	PRG	BASIC 4.0 loader and driver for SMARTERM.
3	"SMARTERM5.4"	PRG	A 6502 Smart Terminal Program, by Terry Peterson.
5	"smart.note:e"	SEQ	A short note on SMARTERM.

ON THE 6809 SIDE: For SuperPET to SuperPET communications, we recommend FASTTERM, written by Jeff Larson. You can upload or download as many files as you wish, calling them off disk by filename, and giving them names as they come in. Since there is only one version of FASTERM, there's no problem of someone else using a different, incompatible version.

OCTOBER-NOVEMBER MEMBERSHIPS Expire with This Issue

You won't get a bill or another notice. If the address label of this issue reads: e:10 or e:11 and the year is '82, this is your last issue if you don't send in yearly dues of $15.00 U.S. for memberships in North America or $25.00 U.S. elsewhere. And please send in your address label so we don't have to do a special search. Canadian members: Banks charge a total of $20.00 to cash personal checks in U.S. funds. So: please send Postal Money Orders, or have your bank remit with a check on its U.S. Correspondent bank. We're sorry, but--$20!!

Use NEWTERM to communicate with mainframes or computers other than SPET. NEWTERM lets you reconfigure the keyboard transmit tables so any non-character key can send any ASCII code. Likewise, signals which come in from the serial port may be translated into code which SuperPET understands. If, for example, a big computer insists on sending CONTROL Z to clear screen (ASCII 26), you can translate it to CONTROL L (ASCII 12), which clears screen on SuperPET.

John A. Toebes VIII, 145 C Jones Franklin, Raleigh, N.C. 27606, wrote NEWTERM to operate, initially, with a mainframe at NCS, and while he was at it, stuffed in the features shown below, left, on the programmed function (PF) keys of the numeric keypad. He also reassigned the OFF/RVS key as a CONTROL key; you can send all standard CONTROL characters from the keyboard. He also set up NEWTERM so that after it is configured to work with a specific system partner, you can save that configuration of NEWTERM to disk under a new filename with PF4. We tried NEWTERM on a coast-to-coast conference with John Frost, and initially had trouble because we were using one configuration of NEWTERM, whilst John was using another. After we sorted that one out, all went well. (Which is why we recommend FASTERM for SPET to SPET operations). John Toebes' instructions for NEWTERM are both clear and complete. He gives you great flexibility in dealing with different TC partners. As with FASTERM, you can operate at up to 1200 baud on Ma Bell's facilities.

PF0	Alpha Lock (Upper Case alphabetics)
PF.	Exit NEWTERM
PF4	Save configured copy of NEWTERM to disk
PF5	Invoke SETUP from NEWTERM
PF6	Transmit BREAK
PF7	Turn on/off Full/Half Duplex
PF8	Receive a file on disk and name it
PF9	Name and send a file from disk

ON THE 6502 SIDE We have two packages: the Commodore Public Domain program, SUPERCOM.BAS6 (as modified through version 6), accompanied by two supporting machine-language programs, SUPERCOMX and TERM.RS232, both of which are loaded by SUPERCOM.BAS6 (which runs in BASIC 4.0). The 6551 ACIA chip in SuperPET was not properly handled by previous SUPERCOMs, and jumpers were required on the RS-232/modem cabling to compensate. Associate Editor Terry Peterson revised the package to eliminate the problem. No jumpers are needed now.

The second 6502 package is Terry Peterson's SMARTERM, which makes a smart terminal of SuperPET in 6502. SMARTERM was designed to operate with a mainframe on a time-sharing basis, converts PET ASCII to ASCII both ways, sends CONTROLs from keyboard, and by report handles rates up to 2400 baud. It was published in MICRO in April, '83, and several users report well on it. Unluckily, Terry is in overload condition and had no time to write full instructions. For the intrepid.

The disk is available now, 4040 or 8050 format, for $25.00 U.S. Send checks only (no disk) to the Secretary (address on last page, this issue) for 4040, or for 8050 format, to the Editor, PO Box 411, Hatteras, N.C. 27943. Make checks out to ISPUG. If you want only the 6809 package, the price is $15.00 U.S.; the 6502 package is also available for $15.00 U.S. Those who got the 6809 package before we had the 6502 disk ready may update to the full package for $10.00. Send the old disk back with the 6809 programs on it; we'll send back the full 6809/6502 package. Protect the disk in a good mailer. State format! Note the change in price from last issue; we put in more material (and work) than expected. APLers: no way to send 6809 PRG files in this package. See COM-MASTER review, below.

* * *

COM-MASTER
A Full Terminal Emulator

We just finished an in-hand trial of a full terminal emulation package from Hawaii, from ISPUG member Dan Jeffers. It's called COM-MASTER; with it, SuperPET emulates the Lear-Siegler ADM-3A terminal, with full upload/download capability. This is a professional job, fully buffered and interrupt-driven. OFF/RVS becomes a CONTROL key; STOP, a BREAK key. It has alpha-lock, full control of baud rate from progran, lets you select handshake methods, optionally downloads also to a printer, handles APL or Waterloo Roman fonts, and lets you send or receive data word lengths of 5, 6, 7, or 8 bits. Yes, it handles PRG files (APLers, please take note!), and provides either half or full duplex from menu.

You get full parity control from program, can define end-of-record character, control linefeeds (out or in), the number of transmitted stop bits, can define file protocol, and can store 10 different 'canned' phrases (which include CONTROL characters) which you transmit with the PF (Shifted Keypad) keys.

COM-MASTER has two modes: CONTROL and ON-LINE. CONTROL mode gives you a menu on which you can specify all the characeristics listed above and the parameters you want. Make a mistake on menu commands, and the program patiently asks you to do it again. You list filenames of files you want to transmit/receive on menu; they are sent with up-cursor (prefaced with CONTROL), or received with down-cursor ('up' for upload; 'down' for download, a nice convention). Parameters you choose appear on menu in reverse field; other options in normal field. You even get a small ticking clock on line 25 (and can turn it off if it bothers you). The manual says the program works up to 19.2K baud, but we've not tested beyond 1200. Note: SuperPET's interrupt cycle lasts roughly 480 processor cycles, and 19.2 Kbaud is only a tetch over 520 cycles. We're skeptics on anything running well at 19.2K. Only glitch we could find in COM-MASTER: the REPEAT, DELETE, and CURSOR LEFT/RIGHT keys seemed to send different code to screen in on-line and command modes. Turns out they send the same CONTROL characters in both modes; Dan's added to the manual a table of transmit codes to show what's happening.

We needed about two hours to read and understand the manual, and another two to make up an alphabetical cross-index to commands and parameters, and we had the program up and running (Dan has now put a similar cross-index in the manual). We are impressed. The user controls all TC parameters, and should be able to talk to almost anybody, using almost any computer, almost anywhere. On a scale of 1 to 10, the manual rates an 8 (we're pretty tough on rating documentation).

Last, you can set up Command Files which configure COM-MASTER to your specifications very quickly. If you need five different configurations, you save five comand files, one for each configuration. Available for $95 from Quality Data Services, 2847 Waialae Avenue, Honolulu, Hawaii 96826, and worth it if you need a a professional-quality program of broad utility. Dan has kindly sent COM-MASTER to Steve Zeller for a try with mainframes running APL. We'll report next issue.

TELECOMMUNICATIONS : The Commodore Bulletin Boards

by John Frost, 7722 Fauntleroy Way, S.W., Seattle, Washington 98136

Software now available on the ISPUG communication disk supports full access to the family of Commodore Bulletin Boards (BBs) around the country and in Canada. These programs for the 6502 side of our machines allow full upload and download of sequential files, Word-Pro files, Basic 4.0 programs and machine language files to any of the many Commodore (Punter) boards. In addition, communication with other hosts or home computers is supported in both the terminal mode and in the ability to download to disk. You need programs SUPERCOM.BAS6, SUPERCOMX, and TERM.RS232, as found on ISPUG's master_telecom disk.

In this article, I'll highlight the basic features of the program and walk you thru a typical bulletin board session. In a previous article, (SuperPET Gazette June/July 1983) I alluded to the need for a special cable between the SPET and the modem with some jumpers at the SPET. With the availability of the program SUPERCOMX, contributed by (and our thanks to) Associate Editor Terry Peterson, this special cable is no longer needed. But if you have one, use it. It'll work.

In 6502, load and run SUPERCOM.BAS6 on drive 0. The two machine language programs, SUPERCOMX and TERM.RS232, will load automatically. I recommend you place SUPERCOM.BAS6 as the first program on your personal communication disk, so that the whole package will load and run from drive 0 with SHIFT/RUN. Soon after, you should see the telecommunication menu (left, below). Let me describe each menu selection; it's the easiest way to define the capabilities of the system.

1-Terminal Mode
2-Receive a Program
3-Transmit a Program
4-Open Disk File
5-Print Disk File
6-Change Operating Modes
7-Exit from Terminal Program

TERMINAL MODE. Characters you type at the keyboard are sent to the RS-232 port at 300 baud. Unlike 6809 'talk' mode, the characters are not sent to your screen immediately, but will appear there only after being echoed back by the bulletin board or host computer. This is a true full-duplex connection and it will not exhibit the screen echoes that we will see in 6809 mode when communicating with an 'echoing' host. If you use this software to communicate with a computer that doesn't return a character echo, as with another SPET or other home computer, you will probably need to switch your modem to 'half-duplex', so the modem will echo your output to your screen.

The ASCII CONTROL characters are available in the terminal mode directly from the keyboard. The OFF/REV key is reassigned as the CONTROL key. If you depress and release OFF/REV, and then key the desired character, you'll transmit a CONTROL character to the RS-232 port. For example, depress and release OFF/REV and then touch the 'c' key, and you'll send CONTROL C.

The HOME key is assigned an interrupt function. If you depress this key in terminal mode, you'll suspend modem operations and return to the communication menu while maintaining your telephone connection. You use HOME so you can select from menu the needed file-handling functions after you have attended to the log-in protocols of the host. (See instructions on disk for these details.)

OPEN DISK FILE. This feature allows you to download data both to screen and to a disk file. Saving to disk is a two-step operation. Step (1) opens and names the file on the proper drive, while Step (2) starts and stops the recording, and closes the file. When you select OPEN DISK FILE, you'll be asked to give a filename and to select a drive, using the BASIC 3.0 filenaming convention, in which you indicate drive and filename by: 0:filename for drive 0, or 1:filename if you want the file on drive 1. After you select your file, the software opens it and returns you to menu, where you select TERMINAL MODE.

You may now choose what incoming text you'll record on disk. CURSOR-DOWN starts recording to disk, while CURSOR-UP suspends recording. You can pick and choose which incoming material will be put to disk by toggling between CURSOR-DOWN and CURSOR-UP. When you're through recording, hit HOME to close the file. Don't be alarmed if the disk drive doesn't start immediately after CURSOR-DOWN; the disk buffer must fill first. Note: You can open a disk file at any time during a telecom session by first touching HOME to return to menu. The modem connection will be maintained while you open the file.

CHANGE OPERATING MODES. This menu option lets you choose various parity and line feed/carriage-return protocols. The default parameters in the software have been found satisfactory for the majority of telecom situations, but you can revise them (see detailed instructions on disk).

PRINT DISK FILE. This option lets you print sequential files previously recorded on disk. The program prompts you for filename and drive number. You respond in the same way you did for the OPEN DISK FILE option. For example, to print a file named 'oldfile' on drive 1, you enter drive and filename as: 1:oldfile. You are also prompted to select type of printer (CBM for Commodore dot-matrix printers, or ASCII for most others). After this, the program prints the file selected and returns you to menu.

EXIT FROM TERMINAL MODE. This option returns you to Commodore BASIC with a system reset (warm start).

RECEIVE/TRANSMIT. You use these menu options to upload or download from a SPET disk to a Commodore Bulletin Board. Files are transferred in this mode in a sophisticated way. Blocks of data are transmitted and echoed back to the sending terminal for a byte-by-byte error check before being committed to disk, and any blocks with errors are automatically repeated. Transmission is aborted if transfer is not achieved after five attempts.

Having covered the menu briefly, I'll now discuss how to receive a file or program from a bulletin board. I assume you have sent the LIST command to the bulletin board in terminal mode, and the board has responded with a list of programs or files available for download. Still in terminal mode, you send the LOAD command. The board will respond with some housekeeping questions and then ask for the name of the file wanted. Then the board displays the prompt at left, below. You give the START signal with HOME, and then select RECEIVE A PROGRAM from menu. You are then prompted for the name of the file as it will appear on your disk directory. You respond in BASIC 3.0 format. The type of file is established when you respond to prompts asking for the type of program (Program, Sequential, or WordPro). File transfer is slow, so the Board estimates and displays the time that'll be needed, before transfer begins.

```
Waiting for START signal
  (or 'A' for ABORT)
```

As the blocks of file data come in, the screen displays a row of '-'symbols for those transferred without error, and '+' symbols for those which must be repeated. You can monitor the transfer as the row of symbols progresses across your screen. (Be patient at <u>start</u> of file transfer; it's slow. You haven't crashed.)

You send a file or program to a bulletin board in much the same way you receive one, except that the command to the bulletin board to start the process is SAVE, and your response to the prompt 'Waiting for START signal' is HOME, followed by the TRANSMIT A FILE option from the menu. Good luck!

THE WATERLOO TERMINAL EMULATOR : A Review

by Steve Zeller, 6425 31st St., N.W.
Washington, D.C. 20015
(IPSA: SZ and CompuServe: 74425,1306)

As we learned in issue 8 of the <u>Gazette</u> , in an article by John Frost, the SuperPET contains a great deal of communications capability in its "native" form. You can, indeed, get into communications mode without using your disk drive by first selecting 'setup' from the main menu and then jumping into the passthrough mode from the monitor. However, if you plan to talk to mainframes frequently, you should consider obtaining a terminal emulation package. One such package was released by Waterloo last year.

The Waterloo Terminal Emulator (henceforth referred to as: WTE) operates with either the ASCII or APL character set. You choose this on its Setup menu, which you can enter at any time. The WTE comes with a "dongle" that must be placed in the cassete port. Users forgetting to put the dongle there will find subsequent treatment by the WTE to be extremely harsh (Query: Why?). Two very important key functions are supported: the OFF/RVS key becomes CNTL; the RUN/STOP key, BREAK. A number of hosts (CompuServe, for example) require you to use CNTL-char sequences to control events on the mainframe. And, if you have ever started listing a long file inadvertently without a BREAK key, you know that a BREAK key is an extremely useful feature. Waterloo did not implement the ESC key, however, and I wish a CAPS LOCK function had been added (perhaps from the setup menu).

For the most part, the setup menu provides the same features available elsewhere in the Waterloo software. Added are the ability to control echo (remote/-local), the wordlength (5-8 bits), and to set buffer size. At any time, you can "log" characters being received from the host onto a file. This is probably the most important contribution of the WTE. You may print output by using a filename such as 'ieee4' or 'printer.' You will not obtain a record of your portion of the terminal session unless the host echos characters back to you. The ability to set up a buffer ensures that incoming characters are not lost while internal processing is taking place. With my Hayes modem operating at 30 cps (300 baud), and an Epson printer running at 80 cps, a large buffer is not required. If you were to communicate at 1200 baud and print at less than 120 cps, however, the the buffer would begin to fill up during long print sessions. I'm not sure what the WTE does when the buffer suddenly fills up.

Disk filenames are, of course, as valid as those for printers. You may therefore download information from the host to disk. Since such files can be opened/closed at any time during the session, you can log output from the host selectively, or, in the case of the printer, you can take it "offline" when you don't want to save output. Only one device at a time can receive output: you cannot print and save to disk simultaneously. Since mainframes follow the standard convention of sending both a carriage return <CR> and a linefeed <LF> at the end of each line, while the Waterloo/Commodore convention is simply a <CR>, you'll find that logged output is double spaced when printed or when disk output is examined in the mED. In the case of the printer, look for a switch that will defeat the linefeed generated locally. Disk files have to be corrected later. [Ed. NEWTERM is configured to delete incoming linefeeds for this reason. You can change it. Neat.]

The WTE also recognizes particular character sequences for screen management from the host. From the host, the user can completely control SuperPET's screen. These features emulate those found in NABU (formerly Volker-Craig) terminals. While Canadian users may enjoy this feature, I have not found this type of terminal supported in the U.S. on any installation that I use. A better choice for me would have been eithe a DEC VT100 or IBM 3101 terminal. It is extremly unfortunate that Waterloo does not provide this alternative configuration.

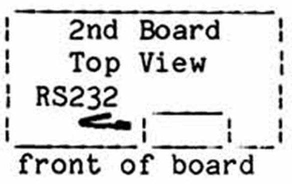
How does the WTE perform? I've been using the package regularly for almost a year and find that it performs as advertised. Furthermore, Waterloo does support the package. My dongle "crashed" once and Waterloo immediately sent a new one. (Jim Swift reports a similar experience). Perhaps the most common complaint is that rapid typing does produce keyboard "bounce." The problem seems to vary from system to system. On CompuServe, I find it extremely annoying. I've used the WTE on several other systems with much less difficulty.

One thing that drives me bananas during APL sessions concerns the extended character set found on most mainframes. Numerous characters can be underscored on a host. Thus, 'A' '<BS>' '_' is interpreted by the host as a new character, A̲ , which is distinct from 'A̅'. "Global or "system" variables in APL workspace are often underscored so that they are less likely to be overwritten. Since underscored characters are not supported by SuperPET's character generator, any such characters received from the host appear on SuperPET's screen as a series of unerscores--not very informative. (Actually, the three-character sequence appears on the screen, but my eyes don't function well at 300 baud.) Waterloo could have handled this by showing the character in REVERSE--an enormous improvement.

The WTE does not provide for uploading files to the host; the method of uploading depends importantly on what the host computer will do with the characters it receives. In some circumstances, you may upload with either mED or PIP (see review of PIP last issue). In other situations, uploading is best handled under program control. The best solution for SuperPET (memory allowing) would have been to merge the facilities of mED, PIP, and the WTE into one unified package. Indeed, all of these features should be available from each interpreter as well. No doubt future Waterloo implementations on larger micros will do just that.

Should you purchase a modem and terminal emulator? This is an important question, but beyond the scope of this review. In future issues, we hope to cover the features of systems such as CompuServe and the Source in order to help you make this decision. In the meantime, if you need a terminal package, the WTE will do the job. The package is $100 and is available from Watsoft Products, at 158 University Avenue West, Waterloo, Ontario, Canada N2L 3E9.

WHERE IS THAT *!!* RS-232 PORT?



In issue 8, we printed a note from Don Gilbreath about the use of the RS-232 port, and got some letters saying the *!* thing isn't where we said it was--and where is it? Well, on the old, three-board models, it's up front, as you view the open case, as we show at left. On two-board models, it's on the left of the top board, as we showed in issue 8. If you don't have microscopes for eyeballs, you probably can't see the pin numbers on any RS232 connector, so we print below a blowup of the thing, as you see it from the side where you solder the wires. Below find some useful notes on hooking up the RS-232 port in SuperPET. First, some general information:

```
\ 13  12  11  10  9   8   7   6   5   4   3   2  1 /
 \ 25  24  23  22  21  20  19  18  17  16  15  14 /
```

FASTERM, NEWTERM, and SUPERCOMX operate with direct connection of pins 1, 2, 3, 4, 5, 6, 7, 8, and 20. No jumpers are needed. If you intend to use the unmodified SUPERCOM.BAS TC program (Commodore Public Domain), you may need jumpers. See below a short summary of a technical note from Waterloo on SPET's RS-232 port.

SUPERPET RS232C WIRING DIAGRAM

The SuperPET uses the following pin connections on the RS-232 port. Some devices require a connection only to TxD, RxD, and to ground pins. The SuperPET requires, however, signals at all nine pin connections shown left. To satisfy this need, when only pins 1, 2, 3, and 7 are connected, pins in the SuperPET RS-port are connected as shown at left, below:

PIN		
1	Protective Ground	
2	Transmitted Data	TxD
3	Received Data	RxD
4	Request to Send	RTS
5	Clear to Send	CTS
6	Data Set Ready	DSR

7	Signal Ground	
8	Data Carrier Detect	DCD
20	Data Terminal Ready	DTR

```
SuperPET               Cable to
RS-232 Port            Device
      1 --connected-- 1
      2 --connected-- 2
      3 --connected-- 3
┌---< 4
|
└---> 5
┌---> 6
|     7 --connected-- 7
|---> 8
|
└---< 20
```

In the RS-232 port, the Clear to Send (CTS) must be at the right voltage for the transmitter of SuperPET to work. If there is no pin connection, you obtain the correct voltage by hooking Request to Send (RTS) directly to CTS (jumper 4 to 5, left, below). The signal going out to pin 4 goes directly back into pin 5, showing it is 'clear to send.'

Similarly, SuperPET expects to get a signal that the Data Set is ready (DSR), and that there is a Data Carrier (DCD); i.e., that a connecting cable exists. Again, if there are no pin connections from the device cable, a correct voltage is obtained when you connect DSR, DCD and DTR (pins 6, 8, and 20). The signal from 20 goes directly to pins 6 and 8, and back to SuperPET, so that it assumes that the 'data set is ready' and that 'data carrier has been detected.' This is merely an example of one hookup for one device, not a universal application. Ye ed summarized the material from Waterloo data. Any errors, blame him.

[Ed. If the very mention of machine language causes you to nn (nn = 'become ill' if genteel, or 'vomit' if not), then read on. No need to nn. This is painless.]

TAB : A MENU SET-RESET
by Dick Barnes

Here's a short, simple Assembler program which gives you a choice of three different tab sets from menu. You can tailor the tabsets to fit your needs. One of the sets is Waterloo default; the second, a 'document' set at tab increments of five; the third, a 'program' set with a first indentation of 7 spaces, followed by increments of two for structured programming. If you want more options, add more 'admit' statements. Change the tabsets by changing the 'fdb' values for the tabsets you want. The cursor will stop on the screen columns shown below in the 'fdb' options.

```
xref tabset_, getchar_, printf_
service_ equ $32 ; This is tab.asm

ldd    #prompt    ; load address of prompt
jsr    printf_    ; print it
jsr    getchar_   ; read keyboard
cmpb   #'p        ; want program tabstops?
guess
  quif ne
  ldd #pstops     ; so, load their address
admit
  cmpb #'d        ; want document tabstops?
  quif ne
  ldd #dstops
admit
  ldd #sstops     ; or use default stops
endguess
jsr    tabset_    ; do it
ldb    #0
stb    service_   ; return to menu
rts
```

01 stops the cursor on screen column 1; 80 at right margin. You must set all ten stops. If you want fewer than ten, set some of the stops at a value lower than that preceding, as in: 05, 15,25,35,01,01,01,01,70,80; here the four 01 entries will give no tabstops at all. Recommend you not enter 00 as a tabstop, because TabSet_ subtracts one from the values entered; 00 thus becomes 255, and that messes matters up more than somewhat.

You can leave any languages but COBOL and APL, reset to a different set of

(continued below)

```
prompt fcc "Enter 'p' for program tabs; 'd' for document tabs; "
       fcc "RETURN for default tabs."
       fcb 13,0                                   ; RETURN here gets new line

sstops       fdb     01,09,17,25,33,41,49,57,65,73  ; Standard Waterloo stops

pstops       fdb     07,09,11,13,15,17,19,41,61,80  ; Program-writing stops

dstops       fdb     01,05,10,15,20,25,30,41,61,80  ; Document stops
             end
```

*   *   *

```
"tab"  ; This is tab.cmd
Bank 15
include "disk/1.watlib.exp"
org $9f00
"tab.b09"
```

tabs, and return to your language with RESETO, published last issue. Don't use RETRIEVE, also published last issue. As you might expect, it returns you to...(guess what?)--tab, the last program you used. Note this program stores in Bank 15, which is not used in any languages or facilities but COBOL and APL. For those languages, relocate to high user memory. You can load 'tab' in Bank 15 with COBOL and APL if you load it before the languages, but don't use it after COBOL or APL are in memory. It will overwrite the languages in Bank 15 (APL, in particular, was stuffed in with a shoehorn). Since there are 4K bytes in bank 15, not used in m-BASIC, mFORTRAN, mPASCAL, mED, or DEVELOPMENT, why not take advantage of it?

*   *   *

FOR THOSE WHO NN Some readers are terrified by Assembler code (Aiii! Machine-Language! Flee!!!!). Inevitably, after we publish Assembler code, we get letters saying, "Leave that to the experts. Give me stuff I understand." Friends, you do not have to understand Assembler code to use it--easily. You must, however, be able to (1) breathe, (2) read, and (3) type. Load DEVELOPMENT from menu, and enter the microEDITOR. Type in the Assembler code above, exactly as given. File it to drive 0 as: tab.asm. Then type in tab.cmd, above, exactly as given, and file it to drive 0 as: tab.cmd. Then leave the mED with 'bye', and get the Assembler from DEVELOPMENT menu with: a <RETURN>. We show the Assembler screen below. Just follow the instructions--and be sure a language disk is in drive 1, to reference watlib.exp (the addresses of library routines).

```
---------------------------The Assembler Screen--------------------------------

ASM6809 V1.1 Copyright 1981 Waterloo Computing Systems Limited
Enter filename:
tab                 < Enter filename, hit RETURN. Assembler takes over. >
Pass one            < All the rest the Assembler does for you. This is hard? >
Pass two
Creating object file
File 'tab.asm' : 35 lines, 0 diagnostics
Assembly completed  < Hit RETURN to leave the Assembler >
-------------------------------------------------------------------------------
```

Now, get the linker from DEVELOPMENT menu with: l <RETURN>.

```
---------------------------The Linker Screen-----------------------------------

Waterloo 6809 microLinker Version 1.0
Copyright 1981 by Waterloo Computing Systems Limited
Enter filename:
tab                                < Again, enter the filename. RETURN >
Pass One (Export Collection Phase) < And the Linker does the rest. >
Including file: disk/1.watlib.exp
Current file : 'tab.b09', Bank 15
```

```
Pass Two (Ultimate Resolution Phase)
Current file : 'tab.b09', Bank 15
end of pass two
Creating file of Exports
Linker completed                    < Leave the Linker with RETURN >
```

That was painful? Now, leave DEVELOPMENT with: q <RETURN>, and enter, on menu: disk/0.tab.mod (which loads the machine-language module you just created). If it works okay, copy the module to the language disk as: tab, and load it from that disk with a plain: tab <RETURN> whenever you want new tabsets. Note the excruciating ease of doing all this. Shame on Waterloo for making it so easy. First we had instant orange juice, and now we have instant machine-language. You need to know nothing about Assembler code to use Gazette programs. Just copy the stuff!

[Ed. To help those who are learning Assembler: the directive FCC forms a character string by storing ASCII code (in 7 bits) in each byte; each byte represents one character. FCB is used to define 8-bit data (of whatever kind); FDB forms a double byte (or word) of 16 bits. Note how these are used by John Toebes and in 'tab.asm', this issue.]

USING MACROS IN DEVELOPMENT PROGRAMS

Part I : by John A. Toebes, VIII
145 C Jones Franklin
Raleigh, N.C. 27606

The macro capabilities of the SuperPET assembler are very extensive, and allow you to perform very complex functions as macros, which make your code much more readable. You will find that macros also make it easier to write machine-language code, and that with macros you can add some high-level constructs to the assembler. You now ask: "What is a macro?" It is shorthand which writes large chunks of code you'd normally write by hand, as the examples will show. [In them, I use CAPITALS for emphasis. Enter your code in lower case if you wish.] Essentially, there are four types of macros:

1) A straight-line macro that needs no parameters and always writes the same code when invoked. (You don't have to repeat the code).

2) A macro to which you assign parameters, which will produce the same set of machine-language statements, but with different operands, depending on whatever parameters you assign. (As in a string-writing macro, where the code to write a string is the same, but the string is not.)

3) A complex macro, which produces different code, depending on parameters.

4) A utility macro, which produces little or no code and is always called by another macro as a subroutine. These tend to be very specialized, so I will not talk about them.

Use straight-line macros when you want to perform a specific task and you also want readable code. For example, I use a series of macros to select the Commodore character fonts. This normally takes only four statements, but you'll find it far clearer to look at the code and see a COMMODORE statement, rather than some undecipherable loads and stores to unfamiliar addresses. So, take the code, put a MACR and an ENDM statement around them, and voila, readability--every time you see the label COMMODORE, you know you're changing a font. (See Appendix C, System Overview, for what is happening at left.) You will find you can use these straight-line macros for anything of a similar nature you find yourself doing constantly.

```
commodore macr
          lda   #$0c
          sta   $e880
```

```
        lda   #$10
        sta   $e881
        endm
```

For other things you do a lot, the straight-line macro isn't good enough, so you must add parameters. Example: I often define a string which ends with a null so I can pass it to a system routine which prints it. You can see this in Example 3 of the 6809 Assembler Manual (page 19), where Waterloo defines a string "hello" and terminates the string with a zero in an FCB (Form Constant Byte) statement. With a macro (FCS.MACRO) which passes a string parameter, we do the same thing, but in a way that is easier to read (and saves a lot of repetitive coding as you write string after string). First, file the macro at left to disk under the filename of 'fcs.macro'. It seems simple, but it is extremely powerful. You now use FCS the same way you would otherwise use FCC, with one exception, which you will understand when we look, later, at how a macro works.

```
fcs     macr
        fcc    \0
        fcb    0
        endm
```

Having FCS.MACRO on disk, you may reference it in your code and pass to the macro any string not longer than one code line, as shown below. I assign the label 'prompt' to the FCS macro, and compare the FCS statement to the FCC statement, so you can see the difference. Here, string "hello" is assigned to the psuedo-variable \0 by the macro processor. Any parameters beyond the zeroeth ("hello"), will be assigned to psuedo-variables \1, \2, etc., if our macro is set up for more than one parameter (FCS.MACRO, above, can accept only one screen line and one string because it has only one psuedo-variable, \0.) As we'll see later, we can add some more psuedo-variables to it--if we care to.

```
The FCS macro statement:     Normal FCC statement:

  prompt fcs "hello"         prompt  fcc "hello"
                                     fcb 0
```

Now, here is where a minor problem can creep in. Since the macro processor looks for commas (and hiccups on semicolons, which mark a comment!), how do we pass a parameter that contains commas or semicolons? Simple: enclose the parameter in parentheses, as we show left, below. If you must pass a parameter which includes parentheses, throw in an extra set, since only the outer set is stripped off by the macro processor. In the example which follows, we show how to use all the features discussed this issue (though the example is so short that it really needs no macros at all). Next issue, we'll get into more complex macros, which, while a bit harder to understand, have the lovely ability to clean up the stack for you. But for right now, here's a simple macro example, using 'tab' from this issue, which we've modified to use FCS.MACRO. I've underlined the change so you can see it quickly. Remember, you must put fcs.macro on disk/0 before assembly!

```
fcs   ("Enter 'd' for Daily entry; 'w' for Weekly.")
```

```
         xref tabset_, getchar_, printf_  ; tab.asm modified for fcs.macro
         service_ equ $32
        ;include <fcs.macro>  [Note: Use the semicolon, but no space after it!]
        ...
        ; The body of the code is unchanged until you reach the old 'fcc' and
        ; 'fcb' lines. There, substitute the single line of code below:
        ...
prompt  fcs ("Hit 'p' for program; 'd' for document; RETURN for default tabs.")
        ; The rest of the code for 'tab' is unchanged. The parentheses let us
        ; pass semicolons in the parameter. See end of article for how we <CR>.
```

You may now ask, "What about a second or third line for 'prompt'?" Here's how to handle a long string:

```
prompt  fcc "Please Enter 'p' for program tabs; 'd' for document tabs;"
        fcs "or a RETURN for Waterloo default tabs."
```

Here, our 'fcs' is the second line, while the first part of the string is passed as usual as an 'fcc' string. While you cannot use 'fcs' on two succeeding lines, you can always use it as the last and final line.

To conclude, let's revise FCS.MACRO to pass two parameters, as shown at left. We can pass two parms easily, with a comma between them:

```
fcs     macro
        fcc \0
        fcc \1
        fcb 0
        endm
```

```
fcs "We pass parm 1.", " And now pass parm 2."
```

Note how flexible the macro arrangement can be. In the example immediately above, we do not need to pass both parameters to the macro; if we pass only one, the second psuedo-variable (\1) is ignored. While the macro above is of no immediate utility, it does show how a macro, its parms, and the macro assembler are related.

In this article, I've put the macro on disk, since you then don't need to write the macro into your code each time. If you'd rather have the macro in your code, see Example 25, Development manual (p. 74). Also, note that the manual substitutes a **'%'** for the '\' which you must use for psuedo-varibles--apparently because the printer didn't have a '\' in his type font. This can be somewhat confusing, since the **'%'** is used in SuperPET's Assembler. Readers with sharp eyes probably noted that the original code for 'tab' used an: fcb 13,0, which stuffed a carriage return in at the end of the string, and that FCS.MACRO didn't do this. You can achieve that carriage return, in any string, very simply: add %n at the end of the string sent to PRINTF_ (p 175, Development manual), like this: "hello%n". This won't work with PUTCHAR_, but it always works with PRINTF_.

FROM FAMINE TO FEAST
Our Cup Runneth Over

Not long ago, you SuperPETters had one terminal package available, the Waterloo Emulator. Now we have a bunch, and more are coming. John Toebes, who wrote NEWTERM for us, is finishing an emulation package which covers the DEC VT 100, IBM 3101, and a bunch more (we have a 90% version in hand), and it even handles APL underlined characters by showing them in reverse field. As we went to press, Walt Kutz of Commodore called and said he'd seen a grand package from Nova Scotia (and would send details on where to get it), and we heard from Gerry Gold in Canada about a good package written in Toronto (where else?) for SuperPET. No details yet, so we'll report in later issues. From here on out, your problem isn't whether, it's what--and everything in Assembler, to make the feast sweeter. How times change!

A FIRST-RATE HOME ACCOUNTING PROGRAM FOR SUPERPET

Delton B. Richardson, of 4299 Old Bridge Lane, Norcross, Georgia 30092, an ISPUG member, sent in a professinally done Home Accounting (HA) disk a couple of months ago, beautifully organized and formatted. It is menu-driven and well documented. We're impressed with the program; it runs in mBASIC, employs relative files (transparent to the user), keeps double-entry books for those who, like ye ed, don't know a ledger from a cuckoo, does all the math for you, and outputs everything from summary statements to transaction reports at request. Delton's menus are neat and emphasized by reverse field; he traps obvious errors, screens input to prevent bad entries, and always provides a way to correct typos or bad data. He even 'beeps' at you whenever input is bad and entries are refused.

His program's a serious and complete one for those who want to handle their personal finances exactly and regularly (something most of us regret not doing on April 15). Delton provides a good tutorial on disk. The tutorial (as tutorials should be), is separate from the reference material, where you can find the details of the various sub-programs. You'll need a week-end to learn how to use the program and to set it up for yourself. Once learned and set up, it is easy to use and to keep current. Delton offers it, with instructions and tutorial, on 8050, for $15.00, at the address above. Send no disk. Sorry, not available in 4040. We add we've had a mess of stuff sent in which people wanted to sell, and much of it was.... This disk we mention because it is a first-rate job.

A BUG IN MICROBASIC RELATIVE FILES AND A COUPLE OF WAYS AROUND IT

While moving entries in relative files from one record to another, Delton B. Richardson learned that at least one of the entries is truncated as it comes off disk and before it is transferred to its new location. In the sample which came our way, set at OPTION BASE 0, the sixth entry always came off disk with two-thirds of the entry from record 6 missing. We found out that Delton's recommended cure (reading record=1 first in the input loop) indeed did work, but also found out that reading record=20 first in the loop also cured the bug. Since it seemed that the interpreter was confused on indexing, we tried stuffing an entry into record=0, and that indeed solved the problem. We print a program below which will demonstrate the bug and its several cures. From tests, we can recommend that you not use OPTION BASE 1 (where the bug persisted), but rather stick with the default OPTION BASE 0. The program, as listed below, will run perfectly so long as an entry is made in record=0 in line 155. Remove that entry, or stuff 'end' in record=20, and the bug comes to life. With line 155 deleted, you can still kill the bug with either of the optional entries in the input loop (line 227), by reading either record 1 or record 20 each pass through the loop--which is effective but slow. We've had no time to test for this same problem in the other languages. Such tests should be made. We thank Delton for identifying the problem and for sending in his solution.

```
100 ! test program for relative file bug: 'relbug.bd'
125
130 dim s$(20) : CS$=chr$(12) : U$=chr$(11) : print CS$;
135 data 'one-----','two-----','three---','four----','five----'
140 data 'six-----','seven---','eight---','nine----','ten-----'
145
147 scratch '(f)testjunk,rel'                ! Don't delete this line: we must
150 open #2,'(f:80)testjunk,rel', output     !    test with a new file.
155 print #2, rec = 0 ,'start'               ! ****The key line.**** Put an en-
160 for i=1 to 10          ! try in 155 for rec=20 if line 227 looks for one
165   read filler$
170   s$(i)=rpt$(filler$,9)  ! Here we set up the original relative file.
175   print #2, rec = i,s$(i)
180 next i
185 close #2
190                              ! And print that file.
195 call fileprint("Initial contents of file")
200
205 get z : if z<>255 then 205    ! Press OFF when through reading...
```

```
210 z=0 : print CS$; 'Moving strings. We print as they come off disk.'
215
220 open #2,'(f:80)testjunk,rel', inout
225 for i=1 to 9
227   ! linput #2, rec = 1, r$  OR linput #2, rec=20, r$
230   linput #2, rec = i+1,r$      ! The bug appears here on input from disk
235   print #2, rec = i,r$         ! if line 155 is deleted.
240   print r$ : if len(r$) >= 80 then print U$;
245 next i
250 close #2
255
260 get z : if z <> 255 then 260   ! Again, press OFF to continue
265
270 call fileprint('File contents after move')
275                                ! The bug also shows in the new disk
280 proc fileprint(hs$)            ! file if line 155 is gone.
285   open #4, 'terminal', output
290   print #4, CS$;hs$
295   open #12,'(f:80)testjunk,rel',input
300   for i=1 to 10
305     linput #12, rec = i, r$
310     print #4, r$ : if len(r$)>= 80 then print U$;
315   next i
320   close #12 : close #4
325 endproc
```

A REVISION OF PIRQ TO WORK WITH COMMODORE 8023 PRINTERS

```
     loop
       ldb    ,x+
       cmpb   #$a0       ; revision
       blo    norm       ; revision
       subb   #$80       ; revision
norm   pshs   y,x,d      ; revision
       ldd    outpt
       jsr    fputchar_
       leas   2,s
       puls   y,x
       leay   -1,y
     until eq
```

Delton B. Richardson couldn't use PIRQ, Gary Ratliff's screen dump (Vol. 1, pp 76 ff), to dump to his 8023P printer a screen which included reverse field characters (the reverse field prints as garbage). So he revised PIRQ and we print the revision at the left. On p. 78 there's a: loop...until eq. Revise that loop to read as shown at left, then reassemble and relink.

Delton sent us a sample of a reverse video screen (from the Home Accounting package we review elsewhere this issue). The printout was perfect. If any of you have changes to PIRQ for other printers, send 'em in.

BITS BYTES & BUGS : by Gary L. Ratliff, Sr.

PO Box 829, Sanatorium, Mississippi 39112

This month finds us again back in operation after having learned a great deal about the 64K memory board of the SuperPET and the 4116 dynamic RAM chip. Those who are good with a soldering iron may anxiously await my detailed report on how the memory board was repaired. Dick also tells me that Steve Zeller is back in operation [Ed. A new board, $85; labor, $35. Steve's pleased with AB Computers]. Those who have gone through the agony of being without a computer for weeks can can well understand my joy at again being able to operate. The first order of business is to answer the quiz presented in the last issue:

1) The 6809 carry flag will be set if an ADDITION requires a carry from either bit 7 or bit 15. After SUBTRACTION the carry flag is set if a borrow was needed and otherwise is cleared. During a MULTIPLY instruction, the carry flag represents the status of bit 7 of the result. In contrast, in the 6502, the meaning of the carry flag is inverted. For SUBTRACTION, it is clear if a borrow was required and set otherwise. Also, because the 6502 has no math operations which don't include the result of the carry flag, it always must be cleared prior to an ADDITION and set before SUBTRACTION. The action of the 6502 on subtraction is different from all other 8-bit processors.

2) The most direct method to clear the 6809 carry flag is to use the instruction: ANDCC #% 11111110. However, the carry flag also is cleared when you clear the A or B registers or a memory location. CLRA, CLRB or CLR loc thus will clear the carry flag. Third, the 6800 instruction for clearing the carry flag, CLC, will automatically generate the required code for the 6809 as it is a predefined macro, supplied to give compatibility with the earlier 6800 processor.

3) The difference between LEAX B,X and ABX: the contents of the B register are considered to be unsigned by ABX, and signed by LEAX. Hence, if the X register contained 1000 and the B register contained ff (-1): after ABX, the value in the X register would be 10ff (1000+ff); but after LEAX B,X the X register would contain 0fff (1000-1 = 0fff).

Those who read the Lewis text recommended last issue may be familiar with what follows. Dr. Cowan of the University of Waterloo has granted permission to quote the Length routine, which determines the length of a string addressed by the D register.

```
; set params
  ldd # text
  jsr length_  ;library routine
  swi
text fcc "This is a string."
     fcb  0

; The length subroutine
B045  JMP  $B7B1
B7B1  PSHS D
B7B3  LEAS -2,S
B7B5  CLRA
B7B6  CLRB
B7B7  STD  ,S        ; * loop
B7B9  LDB  [$02,S]
B7BC  LBEQ $B9C2
B7C0  LDD  $02,S
B7C2  ADDD #$0001
B7C5  STD  $02,S
B7C7  LDD  ,S
B7C9  ADDD #$0001
B7CC  BRA  $B7B7    ; * endloop
B9C2  LDD  ,S
B9C4  LEAS $04,S
B9C6  RTS
```

Here we see an almost text-book treatment of the 'Electrifying Streamlined Blueprint Speedcode Method', as covered by Lewis. The stack is the primary vehicle for the operations. After all parameters have been established, the stack appears as:

4,S	Return address
2,S	String address
0,S	Character count. (clra clrb set at 0).

The routine is a loop which examines the current character to determine if it is zero. (A string is a collection of characters terminated by a null byte.) Each character, therefore, is tested for zero by an LBEQ; if zero is not found the stack is adjusted by adding one to the count of characters, which is at 0,S, and incrementing the pointer to the string character, which is at $02,S. When the null byte (0) is found, the D register is loaded with the count and the stack is readjusted by increasing its count by four, so that it again points to the return address. An RTS instruction terminates the routine, with the D register now containing the count or length of the string.

```
[  2]   →(0=N)/L2
[  3]   →(0<N)/L3
[  4]   L1:'NEGATIVE'
[  5]    →0
[  6]   L2:'ZERO'
[  7]    →0
[  8]   L3:'POSITIVE'
      TEST1 ¯1
NEGATIVE
      TEST1 0
ZERO
      TEST1 1
POSITIVE
```

```
[  2]   L1:'NEGATIVE'
[  3]    →0
[  4]   L2:'ZERO'
[  5]    →0
[  6]   L3:'POSITIVE'
      TEST2 ¯1
NEGATIVE
      TEST2 0
ZERO
      TEST2 1
POSITIVE
```

```
 -------------------------
| HERE ARE EXAMPLES OF CON-
| DITIONAL BRANCHING. NOTE
| THE THREE BRANCHES IN
| ¨TEST1¨ HAVE BEEN COMBINED
| INTO ONE STATEMENT IN
| ¨TEST2¨.
 -------------------------
```

If the argument of the conditional branch is zero, the branch can be used to exit the function based on certain criteria. Alternatively, the condition might be false, in which case no argument would be selected and control would pass to the next statement. Branching of this sort is an important technique to master in APL. To develop an idea for branching, first try it in immediate mode and then move it into the body of a function. Experiment with building the boolean to perform the selection and then try the whole line. There are often several ways to accomplish the same task.

The second important element of program control is looping. An obvious example: reading or writing records to a file. An APL "purist" will not loop in virtually any other applications. With a small WS (such as SuperPET's), however, looping turns out to be quite important: using the full power of APL often results in the dreaded "WS FULL" message on the screen.

In the two examples below, however, looping is really not needed. For that reason, we show the solution with and without loops. The examples do represent the kind of looping done in the other languages, however, and will therefore be familiar. The example at left sums over the elements in X and calculates the mean. The APL "one-liner" shown below the loop is perhaps the most widely used in existence. It is much more elegant and it is a lot faster! The right-hand example searches through the rows of a table of entries to find a match. The APL alternative, shown below it, is much more compact. Note that failure to find a match results in the index being one larger than the size of the table (See p. 72 of the manual for an explanation of this operator.) Many APL programmers suggest that you always increment at the top of the loop, since you may then return to the top of the loop from more than one spot in a program without having to worry about the counter.

```
      ∇MEAN[⎕]∇
[  0]    R ← MEAN X ;I;N
[  1]   N←(ρX)+I←R←0
[  2]   S1:R←R+X[I←I+1]
[  3]     →(N>I)/S1
[  4]   R←R÷N
      X←10+⍳25
      MEAN X
23
      (+/X)÷(ρX)
23
```

```
      TABLE←10 3ρ'X1 X2 X3 X4 X5 X6 X7 X8 X9 X10'
      ∇FIND[⎕]∇
[  0]    R ← ENTRY FIND TABLE ;I;N
[  1]   N←(1↑ρTABLE)+I←0
[  2]   S1:→(∧/TABLE[I←I+1;]=ENTRY)/DONE
[  3]     →(N>I)/S1
[  4]   'ENTRY: ',ENTRY,' NOT FOUND'
[  5]   →R←0
[  6]   DONE:R←I
      'X2 ' FIND TABLE
```

```
2
      'X11' FIND TABLE
ENTRY: X11 NOT FOUND
0
      (TABLE∧.='X2 ')⍳1
2
      (TABLE∧.='X11')⍳1
11
```

Still another example of looping in APL was contributed recently by Mike Werner. He developed a function, VERTICAL, which vertically reorients data elements in an array. In his letter, Mike emphasizes that looping is very slow. Indeed, it takes 29 seconds with the sample array he provided. We show an alternative in ALTVERT; we do not loop; execution time is 4.5 seconds. While we may branch and loop with ease, it often pays, in speed and efficiency, to find a better way.

```
      ∇ALTVERT[⎕]∇
[  0]    R ← ALTVERT X1 ;X2;N                  |BY USING A NUMBER OF APL
[  1]   ⍝REORIENTS AN ARRAY FROM HOR. TO VERT. |PRIMITIVES, THIS FUNCTION
[  2]   X2←⍕((N←⍴X1),1)⍴X1                     |RUNS IN 4.5 SEC. VERSUS
[  3]   R←((2×N)⍴1 0)\⍉(¯1++/X2=' ')⌽X2        |29 FOR ONE USING LOOPS.
```

Of course, once programs begin to make decisions based on values being calculaed during execution, run-time errors become more and more likely. But the task of getting it all running is probably easier in APL than in the other languages on SuperPET. Because the APL syntax is so simple, most of these errors are out of the way from the beginning. And, since it is easy to build a system in logical modules, and then to test it, a good portion of the code is always debugged at any time. Finally, when errors do occur (and they do), all the variables involved in execution are available in the WS and can be examined. We may repair code "on the spot" and resume execution.

In the example following, we introduce an error in line 2 of the function MEAN by changing "N" to "M". Furthermore, a function, MAIN, calls COVER1 which calls MEAN. These are meant to represent a system of functions being built to accomplish some task. We list the functions on the left and show a sample session on the right. After entering "5" in response to the prompt from MAIN, APL bombs on line 3 of MEAN. At this point, a value for N is needed, and, because of the typo in line 2, N is not defined. The state indicator shows a suspension in line 3 of MEAN and tells us that MEAN was called from line 2 of COVER1, which was called from line 3 of MAIN. Note that the line counter, "quad LC", also contains these line numbers, and that a direct branch to the line counter will resume execution where we left off. Using the function editor, we could open and list MEAN in order to discover our problem, and then fix line 2. Yet our immediate problem is in line 3 of MEAN, and here we can temporarily fix the problem by giving N value before we resume execution. In a typical application, we would run into still other errors further down the chain of execution, but the basic approach is the same. First, list the function if need be. Next, examine values (including local ones) in the WS to see what the problem is. Finally, fix it!

```
      ∇MAIN[⎕]∇                     |      MAIN
[  0]    MAIN ;ARG                  | ENTER: NO. FROM 1 TO 10
[  1]   'ENTER: NO. FROM 1 TO 10'   | ⎕:
[  2]   ARG←⎕                       |       5
[  3]   COVER1 ARG
      ∇COVER1[⎕]∇
[  0]     COVER1 N ;SERIES
[  1]   SERIES←⍳N
[  2]   MEAN SERIES
      ∇MEAN[⎕]∇
[  0]     R ← MEAN X ;I;N
[  1]   M←(ρX)+I←R←0
[  2]   S1:R←R+X[I←I+1]
[  3]     →(N>I)/S1
[  4]   R←R÷N
```

```
| VALUE ERROR
| MEAN[3]  →(N>I)/S1
|              ^
|       )SI
| MEAN[3]         *
| COVER1[2]
| MAIN[3]
|       N←5
|       ⎕LC
| 3 2 3
|       →⎕LC
| 3
```

BOOKS Several APL books merit your attention. I recently came across a copy of APL is Easy, published by STSC, Inc.; it appears to be a good, quick introduction for beginners. Alan Rose is about to issue a revision of his classic text, APL: An Interactive Approach, so hold off buying Gilman and Rose for a few months. More advanced readers may try Adrian Smith's APL: A Design Handbook for Commercial Systems. His treatment of APL idioms is especially useful.

ENHANCED APL There are a number of features in our APL interpreter that make it ill-suited for serious applications. Version 1.2 of the micro languages has been released for the IBM PC, but not for SuperPET. Waterloo possibly may make available for SuperPET a subset of its more advanced APL, Version 2.0. This has a number of nice extensions, including component files. I'd guess the chance of it happening is less then fifty percent, and that the software would cost about $250 if it did.

6425 31*ST ST.*, *N.W.*, *WASHINGTON*, *D.C.* 20015 *U.S.A.*

HELP NEEDED William R. Ledger, 13782 Paseo Aldabra, San Diego, CA 92129, needs information on interfacing an OKIDATA ML82A printer to his SuperPET. Anyone who uses this printer with SPET: write Bill directly (and send the Gazette a copy).

RTC 4:A WORD PROCESSOR FOR SUPERPET

by Gerry Gold

Imagine a word processor that loads 40 screens of data into memory (from 1000 screens on disk), which you can then use by printing any combination of the 1000 available screens, in any order and by repeating screens if necessary. The RTC 4, SuperPET version, written in 6502, does all this and more by accessing the SuperPET memory to permit 756 lines in memory at any one time.

Documents can be loaded directly from the user-constructed index, which in turn can be sorted numerically or alphabetically. A document can be as long as a disk or can extend over several disks. Words, lines, paragraphs and screens can be relocated anywhere on the data disk or moved to other disks. If necessary, the entire disk can be reorganized with a utility program provided on the main program menu. RTC 4 can be used in two distinct ways: first, an unstructured 'word-stream' mode lets you enter data that you then format for printing. In a second, structured alternative you 'get what you see', with options such as word wrap, centering, and numeric mode. You have a keypad cursor and a redefine-character option in each mode.

You provide formatting instructions to a 'control map' which is called with the ESCAPE key and the letter 'c' (most commands are invoked with a single letter).

You may use one map for an entire session, set with default values. Another control map may be created for each screen to set pitch, spacing, line size, indentation and margins for part of a document or for envelopes. A powerful local or global search in RTC 4 gives you limited data base search capability by letting you add conditions to a search (search for - Super, if -PET).

You may call a dictionary/spelling checker from the menu and draw on a user-constructed dictionary that holds up to 40,000 words. You may print the dictionary and eliminate words. RTC also interacts with a mailing list/data base (RTC Scratchpad, sold separately). RTC 4 prints to both screen and printer, and can print spool to printer while you edit another file (and retain control over printer). RTC 4 files are compatible with an 8032 version which you can load by setting the R/W switch to 'read', and with 8096, C-64, and VIC versions. RTC 4 will read WordPro files if you replace carriage returns with '@' and delete blank lines.

The manual is informative and readable and includes a number of advanced features of the RTC 4. Order from Richvale Telecommunications, 10610 Bayview Ave., Richmond Hill, Ontario, Canada, L4C 3N8 (416 884 4165). Prices: $129 Canadian, or $105 U.S. plus $5 postage/handling in either currency. A French language character generator for the SuperPET is $50 Canadian or $41 U.S, and gives the user French on the screen. RTC 4 will send French to any appropriate printer. It's a powerful product. Personally, I do not know what I would do without it!

[Ed. We'd hoped to run a review of PAPERCLIP, 8096/SuperPET version, this issue, but had no room. If Professional Software gets off the dime and sends data, we will cover both PAPERCLIP and the rumored WordPro 6+ (for SuperPET) next issue; two guys we can't name say: yes, WordPro 6+ <u>definitely</u> will be released. By the Almighty, boys and girls, SuperPET is at last being acknowledged with SOFTWARE!]

Prices, back copies, Vol. 1 (Postpaid), $ U.S.

No. 1: not available	No. 4: $1.25	No. 7: $2.50	No. 10: $2.50
No. 2: $1.25	No. 5: $1.25	No. 8: $2.50	
No. 3: $1.25	No. 6: $3.75	No. 9: $2.75	

Send check to the Editor, PO Box 411, Hatteras, N.C. 27943. Add 30% to prices above to cover additional postage if outside North America. Make checks to ISPUG

AUTHORS If your deathless prose was disemboweled this issue, know that exactly 43 lines had to be ripped from this issue to make it fit. Sorry, but we don't yet have rubber paper. If it's any comfort, ye ed's gems suffered the same fate.

DUES IN U.S. $$ DOLLARS U.S. $$ U.S. $$ DOLLARS U.S. $$ U.S. DOLLARS $$

APPLICATION FOR MEMBERSHIP, INTERNATIONAL SUPERPET USERS' GROUP

(A non-profit organization of SuperPET Users)

Name:______________________________ Disk Drive: ________ Printer:__________

Address:____
Street, PO Box City or Town State/Province/Country Postal ID#

For Canada and the U.S.: Enclose Annual Dues of $15:00 (U.S.) by check payable to ISPUG. <u>DUES ELSEWHERE:</u> $25.00 U.S. Mail to: Paul V. Skipski, Secretary, ISPUG, 4782 Boston Post Road, Pelham, N.Y. 10803, USA.

<u>Canadians!</u> Please send Postal Money Orders or checks drawn on a U.S. Bank by your bank. Sorry, but we are charged $20.00 for processing other checks. !#!

Newsletter published by the International SuperPET Users Group (ISPUG); a non-profit association; purpose, interchange of useful data. Editorial offices at PO Box 411, Hatteras, N.C. 27943. Secretary, Paul V. Skipski, 4782 Boston Post Road, Pelham, N.Y. 10803. Membership applications, dues, and inquiries to Mr. Skipski; newsletter material to Hatteras, attn: Dick Barnes, Editor. SuperPET is a trademark of Commodore Business Machines, Inc.; WordPro a trademark of Professional Software, Inc. Contents of this issue copyrighted by ISPUG, 1983, except as otherwise shown; reprinting by permission only; SPUG members are authorized to use the material. Enclose a self-addressed, postpaid envelope with all material submitted and all inquiries requiring reply. Membership: $15.00 per yr. U.S. in North America, $25.00 overseas and elsewhere. See enclosed application.

For all outside the U.S.: All nations members of the Postal Union offer certificates good in the postage of any other country for a small charge. The Union includes most nations of the world. Canadian members: send Canadian dimes or quarters for postage, but no paper currency.

FIRST CLASS MAIL

FIRST CLASS MAIL

SuperPET Gazette
PO Box 411
Hatteras, N.C. 27943
U.S.A.

First-Class Mail
in U.S. and Canada