SuperPET Gazette

Our new logo, at left, again was drawn by Brother James Kane of Holy Cross High School in Waterbury, Connecticut, who comments: "Ever since the monastery ran out of squid ink in July, we've been trying to find more. You cannot imagine how hard it is to swim around looking for those wiggly little critters while wearing all these robes. Well, we finally caught one....The lettering is 8th Century Uncial as discussed in Carl Nordenfalk's beautiful book 'Celtic and Anglo-Saxon painting,' published by Braziller. Every now and then I slipped in a feature from 15th century Versal lettering--such as the large 'S'....I tried broad strokes that wouldn't look odd if [the printer] left a ragged edge here and there. We Celtic types feel right at home with ragged edges."

BEDIT ADAPTS TO AMIGA: Last issue we said (with great restraint) that editing on the Amiga was clumsy; that you couldn't cursor back and amend a command, or even recall your old one so you could execute it again. We concluded that the folks who wrote the editors were brung up on the wrong side of the tracks. Seems we were right. Our very own Joe Bostic just sent us V.11 of his new Amiga editor, which works as an editor should (read: what we are accustomed to...). Joe is making great progress; when the editor is finished old Commodore hands and those trained on BEDIT no longer will weep with rage and frustration. It's all in assembly and about five times as fast as ED.

HANDS ON THE ST!
See page 265

Before Joe's latest arrived, we bought a copy of Lattice's screen editor in hopes it would be better than ED; it certainly is, but obviously is a quick and dirty conversion of the Lattice editor for the IBM PC, and fairly clumsy, abiding by the conventions from the CP/M-MS DOS world, where nobody knows how to edit in a civilized manner (all strange ways are barbaric, of course, and if God is not an Englishman, he's not from Armonk, either.)

A GENERAL PURPOSE TEXT-PROCESSOR FOR TEN BUCKS
Bostic's BEDIT 2 Available

Gee, did you ever wish you could set the margins in a SuperPET editor to something beside 1 and 80? That you could center any caption or title? That you could reformat edited text to whatever new margins you want? That you could cursor forward by word, sentence, or paragraph? That you could assign any macro you wanted to any particular key? That you could underline or bold face text sent to your printer? That you could page your text with headers and footers and page numbers? Well, you now may have all these things and a great deal more. Joe Bostic has now finished debugging BEDIT, Version 2; the disk is now ready for issue.

All the features of original BEDIT are retained. The ISPUG disk contains both instructions and tutorials on that BEDIT, plus instructions and tutorials for the new features of BEDIT 2. Where else can you get a program editor and a what-you-see is what-you-get word processor for ten bucks?

TOEBES' FAST ASSEMBLER/LINKER: Also on the disk is John Toebes' speedy new assembler and linker ("Put a Tiger in your Tank," says John). You load and use them with DEVCALC 2, Joe Bostic's Version 2 editor for work in assembler. You simply load a new "DEVELOPMENT" from main menu, which gives you the new editor, new assembler, and linker, which work exactly the same way as the old Waterloo versions, but much, much faster--and with some handy, new capabilities set down in the instructions.

CONTROL Keys in BEDIT: The latest version of Bostic's editor includes CALC, which doesn't interfere with normal editing if you don't need it, but will do math for you whilst you edit. The new functions in BEDIT are implemented mostly by using OFF/RVS as a Control key, followed by some easy-to-remember keys. We show some examples at the left. In every case, the key assigned is either the first letter in the command, or graphically represents what is to be done [> says to 'split this line at the cursor'; < means 'join the next line to the one the cursor is on']. The JUSTIFY command does so with spaces (SPET's screen does not allow proportional spacing). If you don't like it, you may CRunch all the extra spaces out of any line with the 'crunch' command. Please note we use ^ (carat, or up-arrow) to mean CONTROL--it saves a lot of space.

```
^c  Centers a Line
^m  Margins set to default
^>  Splits a line at cursor
^<  Joins two lines
^f  Formats to new margins
^j  Justifies text
```

We demonstrate below the power of OVerlay, which lets you lay any set of lines on top of another set. We typed the columns at left first, then dropped down to the end of this article and typed the comments at right. Then we commanded an OVerlay, and joined the material.

```
...New Code,                    ; Suppose two different cases: 1) You were
...written to replace           ; smart enough to save your old comments, or
...some bum code--              ; 2) You want a new set, but don't want to
...of perhaps 20 lines          ; erase the old ones, line by line.
...or so. But the comments      ; Wherever the old or new comments may be,
...you wrote still apply.       ; from start of text to the last few lines,
...Don't retype 'em.            ; you can OVerlay them on the lines at left
...Overlay 'em on the right!    ; as though you typed 'em here.
```

If you work with tables or columnar material of any kind, OVerlay saves a lot of retyping. If you make a mistake, don't worry. The original material is not deleted by OVerlay, which makes a copy.

Paging Output to Printer: Is anyone out there not utterly tired of paging a long file to printer with: '1,60 p ieee4', followed by: '61,121 p ieee4', and so on into the night? BEDIT 2 will do this for you. You simply tell the editor how many lines you want per page (see left); if you want your printer to wait for a new piece of cut paper, you add the command to 'pause'. Then you say 'pprint' (to serial, ieee4, or printer), and BEDIT pages your file automatically. If you asked for a 'pause', the printer pauses until you hit <RETURN>. An ASCII 12, for top of form, is issued to printer at the end of every page.

```
page 60 pau
pprint ieee4
```

Still not satisfied? Want page numbers, headers and/or footers? Well, you can have them. In the command below, 'f' stands for 'footer', 'h' for header, and the starting page number is stated by '#'. The data may be stated in any order ('page #14 f pau h' works just as well, because Joe Bostic wrote a smart command parser). The command shown above prints the first line of your screen file as a header, the last line of that file as a footer, with 54 lines per page, and starts page numbering with page 14. And, of course, the first and last lines of the file--the header and footer--are not printed as text. Yes, you may have footers or headers only, or just page numbers--or none at all. And, if you cannot remember abbreviations, you can state commands in full: page 54 pause footer header #14.

```
page 54 pau f h #14
```

That Lovely FORMAT: Perhaps you want to indent this paragraph after it is written. It is quite easy to do. Steps: 1) reset your margins to 15 and 65 with the command: 'mar 15,65'. Then, 2) put the screen cursor on this paragraph and press ^f (the RVS key and then 'f'). See the results below:

> That Lovely FORMAT: Perhaps you want to indent this paragraph after it is written. It is quite easy to do. Steps: 1) reset your margins to 15 and 65 with the command: mar 15,65; 2) put the screen cursor on it, and press ^f (RVS and then 'f'). See the results below.

When the reformat is finished, you may continue to work with the new margins, or you can press ^m (to reset Margins to 1 and 80) and return to the old margins as we did here. No, the change in margins does not affect any previous text unless you cursor to it and FOrmat it with the new margins still in effect. Can you reformat again? Sure--as many times as you want.

Key Reassignment: You may reassign any key on the keyboard to execute any of command in BEDIT 2, for phrases you want inserted in text, for new commands you define yourself. You do it as we show at left, with a simple, straightforward assignment statement. You may over-ride any built-in key assignments, either manually or with a BATCH file which will automatically execute at bootup if you press ^a (for auto:exe, or 'auto execute').

```
key x = whatever you want
```

We could go on and on, but won't. You have the message. The disk with BEDIT 2 on it is available in 4040 or 8050 format for the usual $10, from ISPUG, at PO Box 411, Hatteras, N.C. 27943. Joe Bostic outdid himself; we are deeply obligated to him for this gem, and cannot thank him enough. Everything you do on any computer in the end depends on edited programs or text. The better your editor, the easier and faster your work is done--and this one makes it easy and fast.

ONCE OVER LIGHTLY
Miscellany

Disaster! Frustration! Rage! Strong men weep, and brave women cower... Anybody who attempts to edit text using the built-in editors in any SuperPET language might as well cast himself into a pit of vipers. We've warned about this before, but forget that we always have new members. All language editors are configured for the language they edit. The one in COBOL hates characters in column 1; Pascal's will change any capital letter to lower case; mBasic's will forcefully indent any text not preceded by a number...

If you want to edit text, load a standalone editor, of which we have three. The first is the plain EDIT on your language disk. It has none of the nasty habits of the language editors. The second is John Toebes'speedy V1.3 editor, which is on the ISPUG Utility Disk. The third is any form of BEDIT, issued on ISPUG Utility Disk II. All handle ASCII text with benevolence; all use the EDIT commands you have learned in whatever language you employ. None of these beauties will chew normal text to shreds.

THE KESLER PHENOMENON Gee, this important note slipped into never-never land (the crack behind our desk) last October and we just found it! Dan Kesler of Sunnyvale, CA writes "Shortly after receiving my copy of Version 1.1 Software from you, I tried to run it and promptly managed to erase it even with a write-

protect tab on it. Since Skyles Electric is not far from where I live...I went there; he found that the microswitch which feels the cutout in the disk had slipped. It couldn't sense whether or not a protect tab was in place! The switch is adjusted and held in place only by the pressure of two pop rivets, not the best of engineering ideas, but it works most of the time. Anyway, he fixed it simply by sliding it back where it should be, and it has been fine every since. The drive is an 8050 by Tandon (top-hinging drive door)."

We subsequently put a scratch disk in our Tandon drive with a write-protect tab, and then scratched it. And one drive wiped it out! We opened the lid, and sure enough, the microswitch had slipped. Check your Tandon drives every so often!

SPEEDY 68020 FOR AMIGA? Out in San Diego, Computer Systems Associates has developed an add-on board using a Motorola 68020 microprocessor which runs at 16 megahertz. We hear it doesn't work on Macintosh because of programming tricks Apple used to cram a lot of code into a small amount of ROM, and has problems on the Atari ST. But it does work, the report goes, on Amiga, with an immediate increase of 70% in speed of execution (a yetch-type generality unless we know what program was being run). We hear rumors Commodore knows about it and has approved Computer Systems Associates as a value-added reseller. Because the 68020 handles a 32-bit address bus, and runs at over twice the speed of the 68000 in Amiga, the potential for increased performance is certainly there. If any of you get further information, let us know! Thanks for the data, John Toebes.

OS9 FOR AMIGA? The CompuServe OS9 SIG insists we say this is printed with its permission (since when did publicity blurbs require permission?): "Attn: Amiga owners. You say you want a powerful, flexible multitasking multiuser operating system for your Amiga? Well, you're going to get it. A company...is now in the process of porting OS9 to Amiga." Multiuser??? We're willing to suffer the speed penalty of multitasking because of the time it saves us, but anybody who wants to borrow some of our 68000 CPU time we'll cram head-first into the nearest DisposeAll--after first boiling the madman in oil. As to the rest of OS9, we're willing to listen, because it may offer a leaner and meaner and faster way of using the 68000, minus all that code for windows, icons, gadgets and other simplicities, which often get in our way and slow us down. At worst, it will provide another Amiga option. Again, be watchful. Tell us what you hear.

POOBAH PRONOUNCEMENT AND HIGH WIND DEPT. Most of the computer publications we read become pompous now and then, but the champion of resounding platitudes is ComputerWorld, each isssue of which could be printed in half the space. From a recent issue: "Telecommunications in 1986 will continue to stabilize toward a period of vendor shakeout, but will be driven by three key factors: technology, the end user and principal participants."

We haven't yet figured out how telecommunications stabilize in a shakeout (your modem works better?). But we do learn that markets change with technology, supply, and demand. Golly.

Nomination for the 1986 Most Pompous Pronouncement Award: "software will have substantial costs associated with preparation and execution of defect removal activities." Gee; it will cost a lot to debug code. Who'd ever have guessed?

TO * OR NOT TO *: Those of you accustomed to getting selective directories in BEDIT with asterisks like this: di *.sam* (which will list all files contain-

ing the substring '.sam') are warned again, as you were warned in the instructions: never, ever scratch a file with a leading *, or you'll scratch a whole disk! The command: scr *.sam will scratch, we repeat, every file on a disk. You should never use a leading asterisk with any scratch command in any SuperPET editor--Toebes' V1.3, Waterloo's mED, or BEDIT. All will scratch a whole disk!! (Good thing you had a backup, Fred...)

MULTIPLE LINE STATEMENTS: Gee, we thought we covered this years ago, when V1.1 languages first arrived. You may continue any line in microBasic with two ampersands, one as the last character on a line, the second as the first on the next line after the line number. We've so far found no limit to the number of times you may continue a line. You may even have comments on the continued lines, as shown in the V1.1 manual on page 110, section 1.2.7. Okay, Fred?

```
which$ = "This string is continued" + &
 & "on a second line;" + &
 & "and we can do more..."
```

A BACKUP POWER SUPPLY THAT WORKS! A few issues back, we reported that no UPS (Uninterrupted Power Supply, the current poobah term for battery backup power) would keep SPET alive at brownout (low voltage), because none we tested switched over to battery power soon enough. We're glad to say that we found a unit that does, made by SAFT America, and sold by Quill Corporation, 100 S. Schelter Road, PO Box 4700, Lincolnshire, IL, (312) 634-4800. We've suffered both low voltage and blackouts with the 400-watt unit; it works. Unfortunately, the sale price is $649; a 200-watt unit is available, but may not be adequate if you eventually want a hard disk. Beware the units heavily advertised by Qubie Corporation at nice, low prices. We tested one and it wouldn't handle brownouts. Sigh. Reckon you get what you pay for, even when you shop hard and long. Yes, the SAFT unit works splendidly on Amiga, too.

MARILYN WANTS VISICALC! Can anybody tell us where to buy a Visicalc chip for SPET? Marilyn Post, Box 1665, Dillon, Colorado 80435 wants one (with a manual, of course). Please let us know who still handles them, and tell Marilyn, too.

SUPERPET for SALE: S.D. (Dee) Chapman, PO Box 92282, Warren MI 48092, has a new job which requires she use an IBM PC. Her SuperPET, with 8050 Tandon drive, Small Systems Engineering Softbox (CP/M - Z80 processor), WordPro 4 +, and IEEE interface for Epson printer, is available for $1500. Dee says all manuals come with the gear, which has been tended with love and care, and that she will include all past issues of the Gazette plus a set of schematics. Everything works. Call (313) 574-5750 at work; (313) 527-1974 at home.

THE SAD FATE OF STEVE ZELLER AND SUPERPETS: Our former APL Associate Editor, Steve Zeller, moved on to an IBM PC AT some time back, and kept his SuperPET. Recently, he went to work for a new firm where all the serious programming is done in Pascal. So Steve bought Turbo Pascal, and comments that it is quite well done and speedy, but that the verbosity of Pascal, after APL, takes a lot of getting used to. Steve decided to part with his SPET, and considered both sale and donation (for credit, come April 15, with the abominable IRS). He asked us for a list of schools in his area who used SuperPETs, with the idea of donating it to one. The response, says Steve, was astounding. All wanted his machine. If you want to move on to another computer, you may find that donation is a better route, after taxes, than outright sale.

THE SCROOGE AWARD: We are going to award the IRS the Scrooge Medallion for its consistent theft of Christmas (alongside the IRS, the Grinch was a piker).

For the past three years, the IRS has heartlessly cast its tax forms into our mail box between Christmas and New Years. Who but Scrooge would so humbug the Holidays with dust and gloom? Bill collectors, used-car salesmen, and even lawyers relent at Christmas, but not the flint-hearts in IRS.

TRANSPORTABILITY OF MFORTRAN CODE: SUPERPET and IBM PC
by Martin Goebel

At work, we have an IBM PC and the PC version of Waterloo's mFORTRAN. Because the language is one and the same as that for SuperPET, you'd expect that programs written on one computer would run on the other. Indeed this is so. There are, however, a few system dependencies that must be taken care of. For instance, disks on the IBM are labelled A: and B: rather than 0 or 1. A printer is called LPT1 and the serial port is COM1. These minor differences are quickly taken care of and with some foresight can be avoided. Filenames on IBM can only be eight characters long, so if I know that the program may be transported, I use eight-character filenames on SuperPET.

The microEDITOR provided with the IBM version is also virtually identical. It uses the extra function keys available on the PC rather than the nummeric keypad. The Editor also has a crude help screen but the serious user has probably converted to BEDIT, which is better yet. SETUP on the IBM works quite differently. Each 'setup' must be sent as a one-line command, which is awkward at best and usually sends me looking for the manual.

Recently, I patched my SuperPET mFORTRAN code to correct the problems reported in previous Gazettes. That led me to think about the IBM version of mFORTRAN. Sure enough, by running the programs which illustrate the 'arrays problem' and the 'negative integer problem', I discovered that the same problems exist in the IBM version. I do not know if Waterloo has issued patches.

In a sense, this is somewhat reassuring. After all, you might question how transportable a program really is if it didn't have the same bugs and if it did not reproduce the same errors!

HITHER AND YON WITH MA GRUNDY
Grist from the Rumor Mill

We heard this rumor in December, 1985; Rick White of Las Vegas, who just bought one, confirms it. When you load Atari ST Basic, you have 5K left for programs. The ST, unlike Amiga, crams its OS into 512K bytes of user RAM. That takes some 200K. ST Basic we reckon was cobbled up by Digital Research from its s-l-o-w Personal Basic (see the story on this one below).

A source we read once commented that Atari Basic couldn't handle arrays totalling more than 32,767 entries because the Basic was written in C, which can only manipulate 16-bit integers (16 bits with a sign bit reserved can count only to +32,767). Gee. Digital Research must have its very own C. All the others we know about handle integers up to 64 bits long. If Atari Basic has a limit on arrays, don't blame C; blame Digital Research.

Rick says you can increase the memory for Basic in the ST to about 37K by disabling buffer graphics, and by another 30K (to 67K) by disabling the GEM desktop accessories. Well, at the moment, that means you can't run any graphics out of ST Basic, nor handle the GEM windows, menus, and sech-like from it either. Rick reports you can run memory up to 1024K bytes for $50 (do-it-yourself) or buy a kit for $180. We hear rumors they will ROM the Atari OS right soon, which will

free up 200K. We hope that TOS (for Tramiel Operating System) is fully debugged and wisely planned for future expansion. Apple has learned, to its great regret on the Macintosh, what happens if you ROM an operating system too soon.

Kindly Uncle Jack ("Business is War") Tramiel, who heads Atari, seems intent on destroying Commodore for booting him out. He isn't resting on a mere ROM-ing of the OS for the ST. He's moving the 520 ST (the 512K model) into the stores of the mass merchandisers for $299, plus $199 for a single-sided drive, plus $199 for a black and white monitor--a minimum configuration for $697. Or you can add a color monitor for $299 and a double-sided drive for the same money--which brings the price of the color version (one drive) to $897. He obviously intends to slaughter the competition with low prices, just as he once did with the C64.

To keep his dealers from revolting, Kindly Uncle Jack is coming out with the 1040 ST, with 1 megabyte of RAM, a built-in double-sided drive, and an internal power supply (all 520 ST units require a separate power supply for the system and every added component--a messy warren of cables) It is supposed to sell, with a monochrome monitor, for $995, and for $1095 with color. This machine is, of course, aimed squarely at all of Radio Shack, at Amiga, Apple II, and Mac.

We've laid quick hands on an Atari ST but did not have time for a good, hard look. We were, however, singularly unhappy because we had no a command-line DOS. Everything you do must be done with $%&#! mouse and with icons, which we do not even load into our Amiga half the time. Mice and icons are very good for drawing pretty pictures, but useless in programming and for text entry. Nor will the ST multitask. And... Woops! As we were ready to print this issue, in came a hands-on report from our very own guru, Terry Peterson. We took the issue apart in a rush, substituted this page for one already done, and replaced an article by Marv Cox with Terry's piece. (Sorry, Marv; next issue!). See the back pages for Terry's report. We think it fair and unbiased.

HANDS ON THE ATARI ST!
by Terry Peterson
See back pages this issue!

INCESTUOUS BASIC: There's a language called BCPL used by our British cousins, which is, we hear, similar to C. Metacomco, the British firm which wrote Amiga DOS, originally did it in BCPL, and also wrote a Basic in that same language. It sold the U.S. rights (so the story goes) to Digital Research, which in turn distributed the language as 'Personal Basic'. We've never used it, but the reports we get say it is abysmally slow. Because Metacomco wrote ABasiC, the language initially distributed with Amiga, we suspect that ABasiC and Personal Basic are closer than kissin' cousins.

When we received Microsoft Basic for Amiga, we wondered if Commodore had abandoned ABasiC and Metacomco in favor of MS Basic. Then we read in BYTE that Metacomco is continuing development of ABasiC to add named procedures, passing of arguments to procedures, and a compiled version--so it may be that ABasiC will emerge from its present cocoon transfigured and improved. We do know that Metacomco had better speed it up and replace that 17th century line editor!

We also learned that the MS Basic for Amiga was arranged for by a deal between Commodore and Microsoft way back when Commodore first bought Amiga and whilst Microsoft was trying to get out a Basic for the Macintosh. We therefore suspect that MS Basic on Amiga is close kin to Microsoft Macintosh Basic--and that it is in assembler because everybody a year or so ago was complaining about how slow

programs were on Mac. Joe Bostic reports that he ran an Amiga Basic program on Mac with only one minor change, which tends to confirm that these two Basics are genetic twins.

A NEW, BIGGER MAC: Apple has emerged with a Mac with 1 Megabyte of memory and a revised ROM OS, which sells for the same price as the old Fat Mac (some $2500) whilst the Fat Mac has dropped to about $1900 with 512K. Gee, we do feel sorry for the Mac owners who paid $2500 for a 128K Mac a couple of years ago! The new Mac has double-sided 3.5-inch drives to ameliorate Mac's horrid disk-switching problem, the same small screen and no color. Apple did add cursor controls and a keypad to the keyboard--users who don't like mice now have their druthers. One ISPUGger has seen it running alongside an ST and an Amiga, and says it is slow, which you would expect because Apple at this point still burdens the 68000 with all the Mac's graphics. Mac has one thing going for it--a good software base. Otherwise, it costs too much, provides too little, and is too slow. The big question now is whether all that lovely software will become available for the ST and for the Amiga. If it does, you can kiss the black and white Mac goodbye.

68010 and 68020 Compatibility: As we write this, UPS dropped off a package of Version 1.1 Amiga DOS manuals from Commodore, which we eagerly scanned. Note this: "With the exception of the Calculator [Ed. A trivial on-screen utility which does calculator arithmetic with no tape], software in the 1.1 release is compatible with the 68010 and 68020 processors." Well, expect a downstream improvement in Amiga performance, because the 68020 can handle a full 32-bit bus and an instruction cache, and accesses memory in 3 cycles instead of 4. Those who want to unplug the 68000 and stuff in a 68010 or -20 be warned: some folks have; installing a 68010 is difficult; it breaches warranty and provides only some 5% improvement in speed if properly done. The 68020 won't just plug in; hardware changes are required. Commodore obviously plans on using the newer processors somewhere downstream--if Commodore's banks don't dry up that stream...

APL TO 4022 PRINTER DISK TESTED AND READY

In issue II, 7, we announced that Paul Rundle of Ajax, Ontario had developed a program which allowed the APL character set to be output on a 4022 Commodore printer. It was massaged and initially tested by our APL Editor, Reg Beck, but we like to have stuff tested by users less expert than Reg before we issue an ISPUG disk. Steve Johnson of Lakewood, Colorado volunteered, and from his experience wrote a short, clear set of instructions on how to use the disk.

You may send selected functions to printer or simply dump the screen. You may also direct program output to printer with one of the programs. Others allow you to combine APL functions, and there's one to send APL directories to the printer in four-column layout.

If you have a 4022 printer and want to print the APL character set, send $6 U.S. to ISPUG (see address, last page) and state your format: 4040 or 8050. We thank Paul Rundle, Reg Beck and Steve for getting this one into shape.

TRAPPED IN THE BLUEBERRY BUSHES
or,
How to Get Back Home to Mother

Breathes there a SuperPET programmer who has never said to himself, "This is my own, my native procedure, but how in heck do I get out of it before it's done???" All SuperPET languages let you exit most loops, but darn few allow you to abandon either a

procedure or function until it has ended. We show a typical trap at the left. Yet there are two ways to get back home to Mother whenever you care to...

```
proc trap
  ...statements
  if <something>
    ...go home to Mother
    ...but how?
  endif
  ...statements
endproc
```

The first will work everywhere but in Pascal, which won't let you abandon a FOR...NEXT loop until it has finished (Nicholas Wirth is a very inflexible guy). Even though you only want a procedure to execute once, you make a loop of it, and tell it to run just once, as in the example at left, below.

```
proc notrap
  for i = 1 to 1
    ...
    if <something> EXIT
    ...
  next i
endproc
```

The fact that it is a loop allows you to QUIT or to EXIT (or whatever your language uses to leave any loop) on any stated condition(s). You may employ as many different EXIT statements as you care to.

The second option uses LOOP...ENDLOOP; you may exit whenever you set a variable to an EXIT condition. See the last example at left.

```
proc leaveloop
  loop
    ...statements
    something = 1
    if something EXIT
  endloop
endproc
```

Sedulously avoid the solution we've seen come in from some programmers, who can't resist a call back to the calling procedure as an EXIT method. If you call back to the 'caller', which again calls your darling procedure, you are in a a recursive loop, and you'll never really get home to Mother, for you'll crash, sooner or later, from a stack jammed to overflowing with too many recursive calls.

USING A PORTABLE COMPUTER WITH SUPERPET

by Martin Goebel
79 Highland Drive, WedgeWood Park, Newfoundland, A1A 3C3

There seems to have been a recent proliferation of the portable, laptop TRS-80 Model 100 computers. In my opinion, Commodore completely missed the boat by not coming through with its LCD portable, as promised. According to the specs, it would have run circles around the competition. Anyway, sour grapes and all, I use a Model 100. The true portability of Model 100 makes it ideal for the use to which I have put it. It's often convenient or necessary to take work home so it can be done on my SuperPET. Transferring files to business computers via 'modem-phone-modem' connections is error prone and a source of endless frustration--which seems to increase with the importance and urgency of the task.

How often does it happen that you cannot get through or the host is down? You can't always be sure that the transferred files will really be there the next day, for a variety of reasons. Furthermore, despite the many excellent telecom programs now available, you often can't get the process to work properly over the phone and after hours. Just try, for instance, sending 7 bits to a computer which insists on receiving 8 bits...

With a Model 100, you can connect the machines directly and transfer files under your control, at the same time and on the same desk. It's simple to set up and check the communication protocols. When you finish, you know that files are securely stored on the other machine.

You then carry the portable computer home or to work; there you may unload all files. It's almost as simple as carrying a diskette back and forth--except, of course, that the Model 100 lets you transfer any files to and from non-Commodore machines.

I have used this system to transfer mFORTRAN programs, data, VisiCalc spreadsheets and simple text files. If for no other reason than to use the expensive letter quality printer at work, it's worthwhile to have access to a portable computer. It's certainly a good way to develop and test telecommunications programs and to access material from the non-Commodore world.

Connecting the Model 100 to SuperPET: The SuperPET and the Model 100 are connected using the RS-232 serial ports. Only 4 wires are needed. A null modem must be used between the two computers (see Vol II, No. 5, p 135 on how to make a null modem and the wiring therefor). The wiring is as shown a left, and is identical to that you need for a direct link between an IBM PC and a SuperPET. On SuperPET, be sure to jumper pins 4 and 5 to each other; similarly, jumper 6, 8, and 20 together. Don't miss the switch between pins 2 and 3 on the two machines!

```
SuperPET              Model 100
   1 -----connect------ 1
   2 -----connect------ 3
   3 -----connect------ 2
   7 -----connect------ 7
4 -> 5  and  6 -> 8 -> 20
```

Software: The best program for establishing communications between the two computers from the SuperPET is NEWTERM, a telecom program found on the ISPUG Master Telecom disk [Ed. $15 in 6809 in 4040 or 8050 format]. It allows file transfer, can be tailored to send appropriate characters for the Model 100, and can easily be reconfigured. The terminal program for the Model 100 is built into that computer. NEWTERM is simply loaded into the SuperPET on the 6809 side from main menu or from monitor. On the Model 100, select TELCOM from its menu.

Setting Telecom Parameters: From NEWTERM, press the setup key [PF5]. Only the baud rate should be changed, to 1200 baud. (A faster rate is OK for keyboarding but I get errors in file tranfers.) On the Model 100, press the status key [F3] and enter 57e1d for 1200 baud, 7 bit word, even parity, 1 stop bit and to disable XOFF. Both machines are set to echo. Then press the terminal mode key [F4] on the Model 100 to enter terminal mode. That's all, folks; the two machines can now type to each other's screen.

Upload and Download; from SuperPET to Model 100: First, prepare the Model 100 to receive a file. Press the DOWN key [F2] and give a file name as prompted. Then, on the SuperPET, press the send file key [PF9] and enter the filename as prompted. Press the RETURN key and watch the file transfer. After all data are transmitted, again press the DOWN key [F2] on the Model 100 to close the file.

From Model 100 to SuperPET: The procedure is virtually the reverse of that above. First, prepare a file for receiving the data on the SuperPET by pressing the get file key [PF8] and giving a filename as prompted. On the Model 100 press the UP key [F3] and give the file name. The next prompt asks for width. Simply press ENTER to send the file as-is (line length is determined by an end-of-line character). After the transfer, press [PF.] on SuperPET to close the file.

A few words to the wise. Try to avoid sending control characters and, of course, the SuperPET files should be ASCII and sequential (SEQ). The Model 100 files will be document (.doc) files . In neither case are the extensions (seq or .doc) ever included in any filename specifications.

Keyboard and Control Characters: The Model 100 and the SuperPET differ significantly in the cursor and screen control characters. Most annoying is the fact that the Model 100 cannot cursor up or right while in terminal mode. There is no INSERT; DELete works differently. Also, the Model 100 does not linefeed with a carriage return, which means that each screen line is constantly over-written. SuperPET has no bell. However, for those keys where there is a valid counterpart, I reconfigured NEWTERM to accomodate the Model 100 and vice versa, using the incoming and outgoing translation tables in NEWTERM (See the NEWTERM instructions by John A. Toebes for details. They're on the ISPUG disk.)

The remapped or functioning keys include HOME, CLR, CURSOR DOWN, CURSOR LEFT, ESC and TAB. RUB OUT (on SPET's repeat key) does the Model 100 DELETE. Pressing any other key will cause a harmless beep on the model 100. From the Model 100, control characters must be used to achieve some SuperPET functions. These are as shown at left, where ^ (as in ^G) indicates that you must press the CONTROL key and the character key shown.

```
^ G     Bell on Model 100, nothing on SuperPET
^ H     BACKSPACE or CURSOR left
^ I     TAB (deletes text passed over, Model 100)
^ J     CURSOR down
^ K     HOME
^ L     CLR (ASCII 12)
^ M     Enter ( RETURN with linefeed on SuperPET)
DEL     same as RUB OUT (REPEAT key on SuperPET9
```

Because the characters above aren't normally present in text files, file transfers aren't much affected. Problems occur only when you type between screens or when you set up for a transfer operation. Also the linefeed-with-CR problem does not occur in files saved by the Model 100. It seems the necessary linefeeds are generated for transferred document files! The relevant lines of the NEWTERM terminal translation tables after they are revised to work with the model 100, and as you see them in the monitor, are reproduced at left. With NEWTERM loaded, enter the monitor, match the table at left, RETURN each line, and save the reconfigured NEWTERM to disk, say as "newt.mod100".

```
;7114 00 00 00 00 00 00 00 00
;711c 08 09 0a 01 0c 0d 00 00
;7124 00 00 00 00 00 00 00 00
;712c 00 00 00 00 00 00 00 00
;7134 00 0b 07 07 07 07 1b 07
;713c 08 09 0a 07 0c 0d 00 00
```

THE APL EXPRESS

by REG BECK
Box 16, Glen Drive, Fox Mountain, RR#2, Williams Lake, B.C., Canada V2G 2P2

Whenever the topic of models and simulations comes up (and it often does in a computer science class) I find myself reaching for one of the best sources I have on the topic. I have mentioned "BASIC for Microcomputers," by Haigh and Radford, in an earlier column. It is a good reference because its examples are developed first in English and then in BASIC (the BASIC programs may be left unread without loss of understanding of the concepts). The development is clear and concise and is complemented by a set of exercises. The following definitions are found in Haigh and Radford:

model:	A mathematical representation of a system.
simulation:	An imitation of the behavior of a system through the manipulation of a model.
predictive model:	A model that predicts the time behavior of a system. Such a model may specify exact outcomes (a deterministic model) or probable outcomes (a probabilistic model).

The Leontief Model of an Economy (Vol.II, No. 2) is an example of a deterministic model. In this column, we develop a probabilistic model in APL, a model of a pasture ecosystem. It is a simple model but useful in demonstrating some useful techniques.

The model traces the movement of potassium through a pasture ecosystem which consists of soil, grass and sheep. The following matrix contains transition probabilities between parts of the ecosystem.

	soil	grass	sheep	outside
soil	0.5	0.4	0.0	0.1
grass	0.1	0.3	0.6	0.0
sheep	0.7	0.0	0.25	0.05
outside	0.0	0.0	0.0	1.0

The first row of this matrix represents the probabilities that potassium in the soil will remain there, move to the grass, to the sheep or will leave the ecosystem, in one time unit. Note that the probability of the potassium moving directly from the soil to the sheep is zero. Obviously, each row must add up to one. Also note that any potassium which moves out of the system remains out of the system.

Supposing that the time unit is one day, we might ask what are the probabilities after two or more days. We might also wish to ask the number of days the potassium stays in each part of the ecosystem, for each possible starting location. To obtain answers to the first question we take successive powers of the matrix. Labels are omitted from the matrices in the following examples but are the same as those above.

```
MAT←4 4⍴.5 .4 0 .1 .1 .3 .6 0 .7 0 .25 .05 0 0 0 1
      MAT
0.5  0.4  0     0.1
0.1  0.3  0.6   0                    ⍝THE TRANSITION MATRIX
0.7  0    0.25  0.05
0    0    0     1                    ⍝MAT+.×MAT IS THE SQUARE
      MAT+.×MAT                      ⍝(MATRIX MULTIPLICATION) OF THE
0.29   0.32   0.24   0.15            ⍝TRANSITION MATRIX. AS AN EXAMPLE OF
0.5    0.13   0.33   0.04            ⍝THE INTERPRETATION OF THIS MATRIX WE
0.525  0.28   0.0625 0.1325          ⍝WOULD SAY THAT POTASSIUM WHICH ORIGINATE
0      0      0      1               ⍝IN THE SOIL (FIRST ROW) WOULD HAVE A 15
      MAT+.×MAT+.×MAT                ⍝PERCENT CHANCE OF BEING OUT OF THE SYSTE
0.345    0.212    0.252    0.191     ⍝IN 2 DAYS.
0.494    0.239    0.1605   0.1065
0.33425  0.294    0.183625 0.188125  ⍝MAT+.×MAT+.×MAT IS THE CUBE OF THE
0        0        0        1         ⍝TRANSITION MATRIX. THE ENTRIES REPRESENT
                                     ⍝THE PROBABILITIES FOR 3 DAYS.
```

In order to answer the second question we must first remove any absorbing states from the matrix. An absorbing state is one from which potassium cannot escape. In this case the absorbing state is the outside. To remove the row and column representing an absorbing state we can remove any row and column which contains a one.

```
DEFINE
REMOVE:((+/((⍴Y)⍴0)=Y)≠¯1↑⍴Y)⌿Y←(~+⌿⌊⍵)/⍵
REMOVE
     MAT1←REMOVE MAT          ⍝'REMOVE' REMOVES ABSORBING STATES
     MAT1                     ⍝MAT1 THE TRANSITION MATRIX WITH ABSORBING STATES
0.5  0.4  0                   ⍝REMOVED. REMOVE IS IN DIRECT FUNCTION DEFINITION
0.1  0.3  0.6
0.7  0    0.25
```

The remaining 3 x 3 matrix represents non-absorbing states. If we subtract it from the identity matrix and invert the result we obtain a matrix in which the row sums answer the second question.

```
I←(⍳3)∘.=⍳3                   ⍝PRODUCES A 3 BY 3 IDENTITY MATRIX, I.
     I
1 0 0
0 1 0
0 0 1
     ⌹I-MAT1                  ⍝SUBTRACT THE MATRIX OF NON-ABSORBING
8.13953489 4.6511628  3.72093024 ⍝STATES FROM I AND INVERT THE RESULT.
7.67441862 5.8139535  4.6511628
7.59689923 4.34108527 4.80620156

     +/⌹I-MAT1                ⍝SUM THE ROWS OF THE RESULT OF THE ABOVE.
16.5116279 18.1395349 16.7441861

⍝SO THAT POTASSIUM WHICH ORIGINATED IN THE SOIL WILL TAKE 16.5 DAYS TO LEAVE
⍝THE ECOSYSTEM. POTASSIUM WHICH ORIGINATED IN THE GRASS WILL TAKE 18.1 DAYS
⍝AND WHICH ORIGINATED IN THE SHEEP WILL TAKE 16.7 DAYS TO LEAVE THE SYSTEM.

⍝THIS CAN ALL BE DONE IN 1 STEP (GIVEN THE TRANSITION MATRIX, MAT)
     +/⌹((⍳1↑⍴MAT1)∘.=⍳1↑⍴MAT1)-MAT1←REMOVE MAT
16.5116279 18.1395349 16.7441861
```

Justification for these procedures is given in "Discrete Mathematical Models with Applications to Social, Biological and Environmental Problems," Fred S. Roberts, (Englewood Cliffs: Prentice-Hall Inc. 1976) pp 263-270.

Next we pose some problems. APL solutions are given at the end of this column.

1. Find the longest side of a triangle with sides in the vector, X.
2. Determine if three angles given in the vector, X, (in degrees) are the angles of a triangle.
3. Determine if 3 positive numbers, in X, could form the sides of a triangle.
4. Find the area of a triangle with sides in X.
5. Find the volume of a rectangular solid with dimensions in X.
6. Find the surface area of a rectangular solid with dimensions in X.

When writing functions in direct definition, recursion and matrix manipulation are used to replace loops. APL programmers like to avoid loops when possible but they are often seen in functions defined with the del function editor. In direct definition the whole function is defined on one line and loops are impossible to employ. A recursive function is one which calls itself. It uses its own name in

the function definition. Conditional statements are used in direct function definition to terminate the recursion. Recursive definition is extremely useful but may require considerable effort to assimilate. It is helpful to study any recursive functions you find. In order to examine the details of the execution of a recursive function we may insert QUAD LEFT ARROW at appropriate points in the definition.

```
      DEFINE
BC:(Z,0)+0,Z←⎕←BC ⍵-1:⍵=0:1
BC
      BC 4      ⍝GENERATES BINOMIAL COEFFICIENTS, DISPLAYING THE RECURSIVE
1               ⍝STEPS. THIS EXMPLE IS FROM 'PROGRAMMING STYLE IN APL', K.E.
1 1             ⍝IVERSON IN 'A SOURCE BOOK IN APL' AVAILABLE FROM APL PRESS.
1 2 1
1 3 3 1
1 4 6 4 1
```

On first look it often seems as if a recursive definition can't work, but locks up in an infinite loop. This never happens with a correctly defined recursive function because the function is not actually defined in terms of itself but instead in terms of a simpler version of itself. We have presented several recursive functions in this column. The following table gives the function and the Gazette reference where an explanation can be found. Play around with these definitions if you feel you need some practice.

RECURSIVE FUNCTION	*REFERENCE*
`FAC:⍵×FAC ⍵-1:⍵=1:1`	*VOL. II, NO.* 3, *P*81
`MAT:(⍵↑V),[1]MAT ⍵:0=⍴V←⍞:(0,⍵)⍴''`	*VOL. II, NO.* 3, *P*83
`FMAT:(⍵↑((⍵-⍴V)⍴' '),V),[1]FMAT ⍵:0=⍴V←⍞:(0,⍵)⍴''`	*VOL. II, NO.* 3, *P*83
`FIB:(FIB ⍵-1)+FIB ⍵-2:(⍵-1)∧⍵=2:1`	*VOL. II, NO.* 6, *P*178

The excellent Pulitzer Prize-winning book "Godel, Escher, Bach: An Eternal Golden Braid," by Douglas R. Hofstadter, is devoted to recursion. If you get hooked on this book you may have found your reading for the next five years or so. It is definitely the book which would go with me on a desert island exile. Author Hofstadter defines a function, Q, which is related to the Fibonacci series. In the Fibonacci series each new value is the sum of the two previous values. In the Q-series the two previous numbers tell how far to count back in the series to find the two numbers which are added to make the new value. The first 17 of the Q-numbers are:

1, 1, 2, 3, 3, 4, 5, 5, 6, 6, 6, 8, 8, 8, 10, 9, 10

What is the next Q-number? Usually, if a function can be defined recursively there are other non-recursive definitions possible (and which may be faster and take up less memory when implemented on a computer). Hofstadter implies that no non-recursive definition of the Q-function is possible (what about the Fibonacci series itself?). A recursive APL definition of the Q-function follows:

```
      DEFINE
Q:(Q (⍵- Q ⍵-1))+Q (⍵- Q ⍵-2):⍵≤2:1
      Q 5
3
      ⍝THIS ONE IS REALLY  S L O W  ON SUPERPET!
```

Here are solutions to the problems posed earlier. They are written as functions in direct definition and will take a right argument, X.

ANSWERS TO THE QUESTIONS (AS FUNCTIONS IN DIRECT DEFINITION):
1. *LONGEST*:⌈/⍵
2. *TRIAN*:(+/⍵)=180
3. *SIDES*:(+/⍵×⍵≠⌈/⍵)>⌈/⍵:(+/⍵=⌈/⍵)>1:1
4. *AREA*:(×/(+/⍵÷2)-0,⍵)*.5
5. *VOL*:×/⍵
6. *SA*:2×+/(×/⍵)÷⍵

I would appreciate very much getting more comments and suggestions from readers!

MORE DATA ON USING THE SFD DRIVE WITH SUPERPET AND AN 8050

Rick White of 1605 E. Charleston Boulevard in Las Vegas, Nevada 89104 writes us that "Because of the article in the Gazette (II, 166), I purchased an SFD 1001. I paid $180, which was $14 over dealer's cost... After haggling on price, I haggled for several days with drive compatibility with my 8050. I came up with the following, which may be of use to you and other SuperPETters." We summarize Rick's advice below. Remember that the SFD 1001 is essentially half of an 8250, and writes on both sides of a disk. It must therefore have a different BAM (Block Availability Map) than an 8050.

Rick advises you read the two references at left, below. His recipe for making disks interchangeable between SFD and an 8050 without initialization: 1. ALWAYS FORMAT (NEW) all disks on the SFD!! This is essential.

Gazette, II, 166, p. 168
Commodore 'Disk System Reference Guide'
especially p. 45

2. If such an SFD-formatted disk is used on an 8050 to copy files from an 8050 formatted disk to SFD format, use the COPY command [in 6502, copy ds to dd, where 's' is source and 'd' is SFD destination; in 6809 mED, g ieee8-15.C1=0, where drive 1 is the SFD disk. In BEDIT, the command is similar: @ c1=0].

Never try to BACKUP or COPY a full SFD (or more-than-8050-sized) disk to another disk on the 8050, no matter where it was formatted. Rick say that's all there is to it, except that to maintain compatibility, you must:

1. NOT perform 'header', 'new', 'collect', or 'validate' on any SFD-formatted disk while it is on an 8050. 'Header' or 'new' will indeed wipe out one side of the SFD-formatted disk, but the other side will be inaccessible to the SFD. Any 'collect' or 'validate' wrecks the SFD BAM. Perform all such operations only on the SFD drive.

2. All other disk operations are okay, including a BACKUP on the 8050, provided that you have an SFD BAM on the disks (in 6809, you BACKUP with D0=1 [destination left of =] in either BEDIT or old mED; in 6502, use the BACKUP command). We questioned the use of BACKUP on the 8050, thinking it would wreck the BAM for the SFD 1001. Rick wrote back that "the command d0=1 in 6809 or BACKUP in 6502 initates the 'duplicate' DOS function, which copies a disk, block for block, without regard to contents. Thus, once a disk is formatted on the SFD, it will maintain the desired BAM format and pass it on to its duplicate, even though it is duplicated on the 8050. For the same reason, however, for the

INITIAL transfer from an 8050-formatted disk to an SFD-formatted disk, some version of COPY must be used. If you BACKUP from an 8050-formatted disk, the BAM of the 8050 will be duplicated on the SFD disk."

3. Always disable the SFD's relative file capability (see the referenced Gazette article on how) if you propose to use the disk on an 8050 to read any REL files whatsoever, or if you want to read 8050 REL files on the SFD.

Rick explains the rules above this way: The header, BAM, and directory on an 8050 and an SFD disk vary in only one significant way: there are four BAM blocks in the SFD format and only two on the 8050. Each drive can have only 224 files on a disk because each is limited to 28 of the 29 blocks on track 39. Each drive uses track 39, block 0 as the cosmetic header block. Similarly, each uses block 00 and block 03 on track 38. But--the SFD also uses block 06 and 09. Block 03 is different for SFD mode. The key to interchangeable use is to have a disk with all four BAM blocks in SFD format, which bothers the 8050 not at all--so long as you don't houseclean with 'collect' or 'validate' on the 8050, which knows absolutely nothing about those two extra blocks (06 and 09), and will change 03 at any 'collect'.

The SFD won't read or write relative files formatted on an 8050 unless you turn off its extended REL file capability. If you want total interchangeability with your 8050, always disable that capability on the SFD.

REVIEW OF NEW MICROSOFT BASIC FOR THE AMIGA

The buggy ABasiC for Amiga has been replaced by a new Amiga Basic from Microsoft. Commodore sent it (free) in mid-January, along with V1.1 of Amiga's operating software. We'll call the new language MS Basic for now. We weren't expecting much, and were considerably surprised to find that Microsoft has not only entered the 20th Century with with many structured features but has also maintained compatibility with older versions of the language--no mean feat. In a sense, we have a shiny new all-purpose truck which carries all of Amiga's features whilst providing harness and shafts up front if you care to hitch up Old Dobbin (existing Microsoft Basic programs). Unfortunately, a lot of antique baggage comes along with the horse.

The manual says that the interpreter is written in 80K bytes of assembler. That interpreter, the editor, stack, and 25K working space occupy 194K when loaded. In comparison, microBasic in SuperPET occupies about 85K, of which 29K is working space. (We use K alone to mean thousand, not K bytes in powers of 2).

A Flavor of the Language: At last you may indent code. WHILE...WEND is built into the language. You may construct any structure wanted. Labels are allowed--and must be followed with a colon. Line numbers are optional (so is free speech in the USSR, and on the same terms, as we later see). The language converts all keywords to capitals, whether you like it or not. Variable names may be up to 40 characters long. IF...ELSE IF...ELSE END IF is now allowed, in the form shown at left, with no limit on the number of ELSE IFs. The THEN statement is required as shown; ELSE is optional.

```
IF <condition> THEN
    ...do thus
  ELSE IF <another condition> THEN
    ...do this
  ELSE IF <a further condition> THEN
    ...do something else
  ELSE
    ...do the default
END IF
```

There is a price for upward compatibility. Variable names aren't sensitive to case; "alpha", "Alpha", and "ALPHA" are the same. You may use periods only in variable names (file_name fails; file.name is okay). There are no modern MAT statements; all work on arrays must still be done in loops.

Cursor handling also is primitive. While you may assign the cursor to any absolute row or column, you have no relative cursor commands whatsoever (any moves relative to the present location of the cursor)--up, down, right or left. Nor can you erase a line from cursor to end of line unless you primitively print a line of spaces. We tried the ANSI standard cursor commands for relative moves. They failed.

All Boolean statements (AND, OR, XOR, etc.) perform bitwise operations--not only on integers, but also on real (floating point) variables, which is quite mad. If, in the statements at left, 'x'= 1 and 'y' = 4.2, you print no message. Thus, you are denied "if x or y or z or..." as logical, conditional statements and are driven back on the longer "if x <> 0 or y <> 0 or z <> 0" form of comparison for all branching statements.

```
IF x AND y THEN
 PRINT "Both <> 0"
END IF
```

For compatibility, Microsoft hauled its archaic and clumsy method of handling REL files into Amiga. You have a bushel of separate intrinsic commands specific to random-access files, of which only two (record length and record number) are necessary for such work in a well-designed language. LINE INPUT, however, did see the light of day, although the simpler "linput" was lost along the way.

Exploring Structure: In Amiga Basic, the EXIT statement lets you leave any subroutine upon a stated condition, so we tried it from both WHILE...WEND and FOR...NEXT loops, only to be stopped by an error message forbidding it. Happily, you may, however, leave any loop with a GOTO a line number or a label (endwhile: at left is such a label).

```
WHILE a < 40
   a = a + 1 : b = a*z
   if b > 20 THEN endwhile
   print a;
WEND
endwhile:
```

We tried the approach at left in a FOR...NEXT loop and found it worked there also. You can leave such a loop at any time without mucking up the stack.

If you want any form of endless loop which you may quit on specific conditions, it must be in the form shown at left below, in which you GOTO named labels or to line numbers. The labels are obviously clearer. With indentation and a little care in programming, this language will let you write code that you can read and can maintain, but we intensely dislike the need to create ourselves, in each program, the optional structures that Microsoft should have included. Oh, well; Microsoft has gone from 1891 to 1920...

```
loop:
  ...statements
  IF <cond> THEN endloop
  ...statements
  GOTO loop
endloop:
```

Subroutines may be entered either with a GOSUB or with a CALL to a named label. You may pass arguments in a CALL, as shown below. You have two options: in the first, you pass arguments by reference. This changes values of A, B, and C in the main program by what is done to x, y, and z; see left.

```
...create A,B,C values
CALL sendscreen(A,B,C)
...
```

```
SUB sendscreen(x,y,z) STATIC
  ...manipulate x,y,z
  if <condition> THEN EXIT
  ...statements
END SUB
```

In the second option, you may enclose the arguments in parentheses, as in:

```
CALL sendscreen((A),(B),(C))
```

and the values of A, B, and C in the main program remain unaffected. In short, you have full control over whether subroutine variables are local or global. In addition, you may in subroutines define any variable in that sub as a SHARED variable, which may be used in any other subroutine or in a main program, and which passes its value there. So far, so good.

Unfortunately, recursive subroutines aren't allowed. Do you see the STATIC which follows the subroutine name, above? It's required. Nobody in his right mind demands that subroutines be designated STATIC if that's the only flavor you can define. And it is! STATIC (as other features of this Basic) is derived from C. Static variables retain their value in a subroutine between calls. The other C option of automatic variables, which do not retain value between calls, is not found in MS Basic, but the presence of STATIC indicates that Microsoft plans to use it in later versions. Either that or AUTO variables were taken out at the last minute. Anyway, because all subroutines are STATIC and all variables retain value, recursion becomes deadly; the manual warns users not to try. Too bad.

The MS Basic manual comes with an errata sheet which asks you to remove several references to RESET (which closes all open files). Seems they planned to have it and jerked it out at the last moment--along with AUTO variables. We'll later see why.

Screen Output and Control: One of our early test programs was supposed to print 'a' to screen fifty times. We printed 'a' ten times on screen line 1, but the last 40 went off into right field infinity. The language doesn't do a CR at right screen margin unless you program it in. Gee, back to counting characters and spaces again after four years of letting the computer handle the trivia.

Line Numbers Versus Labels: You are indeed free to use line numbers, which we prefer for reference when we LIST or edit. But there is utterly no way in MS Basic to generate those line numbers automatically, or to renumber them thereafter. The language doesn't even check to see if line numbers conflict or are in proper order. Microsoft obviously doesn't want you to use line numbers.

How do you LIST a part of a program? Well, er..., uh..., you'd better stuff in named labels every 20 lines or so, or add line numbers on the same basis--or you must LIST your program from beginning to end to find the part you want. When we tried to LIST the subroutine at left (without the label "test:"), MS Basic refused. You must add either a line number or a label (as with "test:") despite the fact that the subroutine is already named. And there is no search or search/replace in MS Basic's editor, either.

```
test:
SUB test STATIC
 ...
END SUB
```

Suppose you want to add a library routine from disk to a program on screen. You MERGE it at the end, and then COPY and PASTE into the program. Line numbers have no effect on a MERGE.

Directories: You're supposed to be able to get directories by saying, for example: files "df1:basic.programs". Do you get Amiga's typical two-column list-

ings? Nah! You get a list in one column, and it scrolls right off the screen unless you are fast enough to hit RIGHT AMIGA S to stop that listing. Golly, when will those folks at Belleview learn to pause listings at the end of each screen page? Sorry to say that half the files on our Workbench disk are never listed at all. It's a bug. Well, click over to DOS for directories...

Immediate Mode: You may not cursor up and amend. We're used to working out difficult algorithms in immediate mode, by defining our constants and variables and testing variations in often-complex code. You won't do that in this Basic; you must retype all the code each time you test, which is so time-consuming that you are far better off to make a program of the test. But--a RUN always clears the screen (you can't control that), which wipes out all previous results. Most of the utility of immediate mode is thus destroyed. We retreat to making pencil copies of data we've been handling on screen for years. Back to 1920.

You have two separate windows: one for program output, one for LISTings; you must interminably click your way back and forth between them. It seems this was done so that you can see your code execute in one window and trace the executing line in the other. We tried it for a couple of hours and conclude that the folks at Belleview are mad. The output window is always obscured by the LIST window; a little red border skips from one executing statement to another so swiftly that you can't follow it. If you try tracing one line at a time, you find that every comment is considered an executing line. You also find a bug. We had five lines of comments beyond the END statement in a program. Every single one was 'traced' as a part of the executing code! We'd much prefer a single window with the executing statement printed at top or bottom of the screen. Programmers often get carried away by enthusiasm for fancy and faddish new approaches. So it is with two windows in this Basic. They're a useless pain in the behind.

Error Messages: Despite a large error window ("Syntax Error!"); despite a red border around the line of error, Microsoft's error messages are no more useful than they were five years ago. You aren't told where on the line the error is, nor why it's an error. Example: What's wrong with the statements at left? You get an error message "ELSE/ELSEIF/ENDIF WITHOUT an IF". Is that the real problem? Those accustomed to Waterloo's intelligent error handling and marking will weep. The problem? Shucks, we left out a THEN.

```
IF x OR y
  PRINT "Whoa!
END IF
```

Editing: The editor in MS Basic is both primitive and clumsy. It is also very slow. On any long line, our normal touch typing gets ahead of the screen by ten characters or so. The screen scrolls very slowly when you page up or down by screen page. It takes one full second to move the cursor from the top line to the bottom of a 19-line page unless you shift the cursor with the mouse. The interpreter of MS Basic may be in assembler; the editor is done in molasses.

Editing within MS Basic otherwise is fairly handy. You delete and copy lines or sections of code by highlighting the material with the mouse. You then copy the material to a buffer (called the 'clipboard'). If you DELETE with AMIGA X, the code is erased but the space it occupied remains. You may COPY erased code to a new position, shown by the cursor, with AMIGA P (for PASTE). If you choose to COPY, the original code remains, but a copy is put into the clipboard; you may insert that copy anywhere else in text. This works well on small segments. We tried to delete the second half of a large program by highlighting with the mouse. By stopwatch it took over four minutes to highlight most of the material;

then we crashed (we suspect that the clipboard buffer overflowed). So you must delete in small sections. In sum, you'll probably use this editor to debug code; it is far too slow and clumsy for long editing sessions.

Fortunately, Amiga's multitasking allowed us to use ED, Amiga's screen editor, to write our programs. And then, with ED running concurrently, we crashed MS Basic six times in two hours and forty minutes, with nary a program in memory longer than 80 lines. We kept getting HEAP FULL errors...so we did a check on memory used, which is reported below.

Memory Used: MS Basic ran fine so long as we loaded only CLI (DOS), ED for good editing, and MS Basic itself. We used the default version of ED, which reserves 40,000 bytes for text. We haven't tried, but suspect that the size of ED can be cut back.

CLI, Workbench loaded:
System: 392,824 bytes free

CLI, Workbench & WB Menu:
System: 376,104 bytes free

CLI, Workbench, MS Basic:
System: 182,488 bytes free
MS Basic 25K free

CLI, Workbench, MS Basic, big file in ED:
System: 81,624 bytes free
MS Basic 25K, including program

The figures at left tell the story. Workbench and ED drop free memory too far; you do not have sufficient room for the Heap, which handles all the miscellaneous duties of the OS. As soon as free system memory drops below 90K, you start having problems. We were able to run ED, CLI, and MS Basic together with no problems, so long as we observed that limit. But we would not attempt to run ED concurrently if free memory were smaller than 90K.

Now we now why RESET is missing, and what probably happened to AUTO subroutines. MS Basic got too big; it was cut down in size--we'd guess at the last moment.

We were at first suprised by the small (25K) amount of memory allocated to an MS Basic program, data, and pointers. Later, we found out you can change the memory allocation with the CLEAR statement. The command at left will allocate 40,000 bytes (to the nearest lower 1024 bytes) to MS Basic, and 6000 bytes (on the same basis) to its stack. From all of this we conclude that you write your programs in ED, run MS Basic and ED from CLI without Workbench, and use the slow editor of MS Basic only for debugging. With only CLI and and MS Basic loaded, you do have room for fairly big programs, say around 150K or so, plus CHAIN if you need it.

```
CLEAR,40000,6000
```

Count Erich von Lundsdorff und Frankenstein: There's a text-to-speech demo on the MS Basic disk. We cranked it up and found it hilarious. The default settings for voice sound precisely like a Nazi villain, so we tried "Let me intodusse myzelf; I am Count Erich von Lundsssdorff, und diss is Frankensstine." The extra sibilants were necessary for proper effect. The range of voices possible is limited; we tried for hours to get rid of the Teutonic flavor, but couldn't. But those who want trolls, ogres, wizards and witches will have great fun. The great surprise is a built-in function, TRANSLATE$, which converts any English sentence into proper phonemes for the voice. If you want to bypass TRANSLATE$ and control the voice directly, you may do so--but with phonemes, glottal stops, and such.

Variations on A Theme: We were pleased by the quality of the stereo music on another demo, an organ doing variations on a theme. First class Bach it wasn't, but those who would rather make music on a computer than in an orchestra won't

be ashamed of its sound. One hard look at the program told us that it had been done by a pro. You won't knock out your own music in half an hour. We play three instruments; our knowledge of music was stretched to its limits to follow what the programmer did. Musicians, whether pro or amateur, now have a way to hear, alone, what it would otherwise require a combo to create.

Windows, Mice, Menus, and all that Jazz: MS Basic gives the programmer control over all the features of Amiga, from pull-down menus to windows and the use of the mouse and joysticks--if you care to use them. A very large part of the language is devoted to these things; they are powerful and easy to use.

Graphics: You can program just about anything graphically that you care to. The graphics functions are relatively simple to use (no PEEKS or POKES) and at a high conceptual level. Those who simulate or model will find the functions most useful. You may, of course, control both shapes and colors. Because Amiga hanles all text as graphics, there is no limit to the intermixture of text/graphics on any screen, as there is on the IBM PC and all other micros we've seen.

Machine Language Interfacing: MS Basic provides both VARPTR (the address of numeric variables) and SADD (the address of a string), so that you may access any variable by its address. You may therefore pass these named variables to ML routines. You may also call ML library routines which you write yourself or those already present in Amiga.

A Few Nice Touches: The index function to find a substring within a larger string (called INSTR) allows a secondary index, as in: INSTR(7,X$,"Brown"). In this case, you look for "Brown" within a larger X$, but you start searching at the 7th character in X$. The secondary index is optional. Very handy.

Last, and most important, you may identify variables either by defining their data type (as in Pascal or C), or by using suffixes, as in other Basics. We show the data types below. The identifiers shown (% for a short integer, & for a long integer, etc.), take precedence over the stated definitions, if given.

Definition	Means:	Identifiers
DEFINT var	(short integer; 16 bits)	var%
DEFLNG var	(long integer; 32 bits)	var&
DEFSNG var	(real, 7 digits, 4 bytes)	var!
DEFDBL var	(real, 16 digits, 8 bytes)	var#
DEFSTR var	(a string)	var$

In called subroutines, the definitions may be purely local, or they may be SHARED if so declared within the sub. If they are shared, they become global variables in the main program, and may be used in other subroutines if declared there as SHARED.

Tests on Variations: We ran a number of tests, and were surprised to find no difference in execution time with or without line numbers. Even more surprising: a program using defined variables (as above) runs faster than one where variables are suffixed with their type (DEFINT y is faster than y%). Last, and most astonishing, multiple statements per line are slower to execute than single statements per line! In this language, there is utterly no excuse for cramming code together in the fewest possible lines. The loop at left, for example, is a bit faster than the same loop on a single line. All these things are positive advantages; they reward simple, readable code.

```
FOR i = 1 to 1000
  ...do this
NEXT i
```

Summary: This Basic is much improved over ABasiC but just doesn't go far enough. The code is far easier to read and maintain than older, unindented, and unlabelled versions. It was hastily and drastically shortened at the last minute to let it fit in available memory. As time goes on and code is boiled down, or as more memory becomes available in Amiga, we wouldn't be at all suprised to see AUTO variables, recursion, RESET, additional optional data types, and some fully structured options. It is far superior to any Microsoft Basic we've seen before, and a large step in the right direction (as we said above, from 1891 to about 1920). We do wish it came all the way to 1986.

FURTHER BENCHMARKS ON AMIGA

We supplement the benchmarks published last issue with those we ran on Amiga using the new Microsoft Basic for Amiga, which was issued in early January 1986. It replaces the ABasiC developed by Metacomco and will be issued with Amiga, as part of the computer package. All owners of ABasiC should have received a letter allowing them to pick up the new Basic from dealers. Microsoft Basic is identified as MS Basic herein. See comments elsewhere in this issue on the language. We deeply appreciate the help rendered by guru Terry Peterson in reporting on some of the benchmarks and in catching an error in our Transcendental package (corrected).

Microsoft's Basic does substantially better than ABasiC, being faster. We suspect the results for True Basic as reported in BYTE. We had to abbreviate the names of microBASIC and Basic 09, under SPET, to cram them in.

Byte Calculation Benchmark (Vol. II, pp. 157, 159)

	Amiga		68000	6809 SPET		IBM PC	
	ABasiC	MS Basic	HALGOL	mBas.	Bas09	Turbo Pascal	True Basic
Time (sec) [Single]	22	16.02	Unk.	150	42	82.6	19.7
Error	-5.9605E-08	-5.9605E-08		0	0	1.3384..E-8	-4.483E-13
Time (sec) [Double]	30	21	2.46				
Error	0	0	0				

Transcendentals
Dr. Dobbs Benchmark (As Modified by Terry Peterson. Vol II, p.4, p. 157)

In double precision, MS Basic does very well on accuracy, but is not very fast. HALGOL results are in double precision. Results on SuperPET are in single precision. Remember that ABasiC's accuracy is no better in double than single. We made an error in reporting ABasiC accuracy last issue. Use data below.

	ABasiC		MS Basic		HALGOL	SuperPET	
	(Single)	(Double)	(Single)	(Double)	(Double)	mBASIC	Basic 09
Time (sec)	44	48.5	38.4	78.28	16.8	780	335
Error	8.08518E-5		8.1089E-5		4.1006E-12		3.902E-7
		8.08521E-5		1.290313E-13		6.1153E-7	

(Some errors rounded to save space)

BYTE Sieve of Eratosthenes (Set to 7000, not 8190; one iteration)

The results we show below depend on how you configure the Sieve. If you use the original BYTE GOTO version, you penalize any language in which you can use structure, which is far faster. For example, the GOTO sieve runs in MS Basic in 50.4 seconds. Structure it, and it runs in 32.6. The real buster here is the speed of Basic 09, running under OS9 in SuperPET. The language is, of course, compiled at entry of each line, but even so, a 1 MHz 6809 beats a 7 MHz 68000. Why is a good question; part is language design; part the respective operating systems. HALGOL shows what seems to be a Basic-like minimum, for it has neither multiuser nor multitasking features implemented on the 68000.

	On Amiga		68000	On SuperPET		IBM PC		
	ABasiC	MS Basic	HALGOL	MicroBasic	Basic 09	Better Basic	PC Bas.	Turbo Pascal
Time (sec) [integers]	53	32.6	1.79	168	30			
Time (sec) [reals]	64	76	Unknown	243	58	31.4	191	15.4

NORMALIZING COMPARISONS Because the microprocessors on various machines run at different speeds, we've 'normalized' the comparisons for HALGOL, Basic 09, and Amiga in the table below. Guru Terry Peterson suggested this approach; it puts the data above in perspective. Yes, OS9 is a compiled language and so is HALGOL (compiled as lines are entered); the rest are interpreted. We really don't care when the object is speed.

We normalize by multiplying the time to execute by the clock speed of the microprocessor in MHz. This reduces all results to comparative times, processor speed being factored out. (We figured Amiga at 7 MHz, SPET at 1, and HALGOL at 10. The 68000 in Hal Hardenbergh's rig indeed runs at 12.5 MHz, but has one wait state because the memory chips were slow, which makes the effective rate 10 MHz. For more money, you install faster memory, which raises the rate to 12.5.)

Normalized seconds:	On Amiga		HALGOL	On SuperPET	
	ABasic	MS Basic		MicroBasic	Basic 09
The Sieve:					
in integers	371	228.2	17.9	168	30
Transcendentals					
(Single Precision)	308	268.8		780	335
(Double Precision)	--	547.9	168		
Calc. Benchmark:					
(Single Precision)	154	112.1		150	
(Double Precision)	210	147	24.6		

We conclude that array handling (Sieve) on Amiga is far, far slower than we'd like to see it, and that Amiga handles transcendentals about 3 times more efficiently than SPET. Ordinary arithmetic routines are about 25% more efficient in Amiga. HALGOL and Basic 09 clearly show their compiled advantages and efficient assembly language code. It's most interesting to see that a multiuser and multitasking OS (OS9) causes the Sieve to run at 60% of the speed of an OS not so

encumbered, if we assume (and this isn't necessarily so) that no other differences exist between HALGOL and OS9. It's also clear that Basic 09's transcendental work is slow (although its accuracy is excellent).

COMPARISON, MATH BENCHMARKS IN ABasiC, MS Basic, HALGOL, AND C ON AMIGA

BYTE Calculation Benchmark:

In C, at any time a FLOAT variable (32 bits) is used in a calculation, there is a cast (a reformatting) to make it a DOUBLE (a 64-bit value). The accuracy, however, depends upon the initial definition. As you can see below, you might as well do all floating point work using DOUBLEs, because they are a bit faster and certainly more accurate. Please note that the bit-wise definition of FLOAT and DOUBLE in C is machine-dependent. The values above are for Amiga.

This benchmark assesses the speed with which normal arithmetic is executed. The Lattice C routines perform the calculations; the Amiga ROM/RAM library is not used (for reasons stated later). We contrast results with ABasic, MS Basic, and with HALGOL.

	Double Precision			Amiga C	
	ABasiC	MS Basic	HALGOL	Single Precision	Double Precision
Time (sec)	30	21	2.46	28.7	28.5
Error	0	0	0	0	-1.110223E-16

Transcendentals : Dr. Dobbs Benchmark, as Modified by T. Peterson

This benchmark assesses performance in handling infinitely repeating decimal numbers. We contrast performance with ABasic, MS Basic, and HALGOL. As you can see, accuracy is good, but the Lattice C IEEE FP routines are slow. Again, we did not use the RAM/ROM FP routines because of problems in access. These routines are in single precision; C converts all FLOATS (single-precision) into DOUBLES for all of its FP calculations, so unless you fool by C various kludges, you cannot tap the RAM/ROM FP routines from that language. John Toebes has sent us a sample of how to kludge around the problem, but we haven't yet got it up and running. It's supposed to cut time on single-precision by a factor of 10.

	ABasiC	HALGOL	MS Basic	Lattice C
Single Precision (sec)	44		37.56	Not done.
Error	8.08518E-5		8.1089E-5	
Double Precision (sec)	48.5	16.8 sec	78.28	240.0
Error	8.08520E-5	4.1006E-12	1.290313E-13	1.290356E-13

Note that the accuracy of the MS Basic package in double precision is for all practical purposes the same as that for the software routines in Lattice C, and that the accuracy of both is excellent.

HANDS ON ATARI 520ST: A PET Vet's View

by T. M. Peterson

Yes, I did it. I went out and squandered a bundle on an Atari 520 ST in addition to my Amiga "investment." (The fact is, the ST was acquired first, but I was committed to the Amiga purchase already). It was, and is, my hope to eventually recoup at least a major portion of these expenses via the enterprises of Peterson Software Products. Well, now we'll see if either

Atari or CBM will stay in business long enough for that strategy to pay! But, enough with the crystal ball--"What's this 520 ST beastie like?" you ask.

THE HARDWARE: Setting Up. The basic 520 ST system (the one in the ads for $799 mono, $999 color) is a complete computer system, less printer. (Any parallel interface printer may be connected via an "industry standard" [IBM-PC] parallel printer cable.) The system comes packed in three boxes, one each for the system/keyboard unit, the video monitor (either RGB or monochrome), and the disk drive unit. Included in each is a pamphlet which very briefly describes the piece(s) in that box and gives warranty information. I confess I didn't read the setup instructions, but just plugged the bits together "by inspection," as my math Profs. were wont to say.

The process took about 5 minutes exclusive of uncrating time. Although the standard system comprises some six pieces strung together with cables, the various connectors that are used and the well-marked sockets would seem to permit no mis-connections. [Some (very early?) systems apparently had one ambiguous connection: I read of a case where the system unit and disk drive power supplies were swapped. However, on my ST, these have DIN connectors with a different number of pins, making exchange physically impossible.] The whole ST setup--not counting mouse--takes up about the same amount of horizontal surface as SPET (including an 8050), but it is quite sprawling in comparison to the Amiga. Because the top of the system/keyboard unit won't support the monitor, one must raise and support it on some sort of specialized furniture.

Yes, Teeny Drives Too... Like its 68000 brethren, the ST uses the 3.5-inch microfloppy disks that seem to be taking over as a new industry standard. The ST system's drive is housed in a separate case that is even smaller than the Amiga's outboard drive (5.5 x 9.3 x 2 inches). Unlike the Amiga's outboard unit, it is separately powered by a cord plugged into the wall--said cord containing a "lump" (4.5 x 3.5 x 2 inches) housing its power supply. The drive is a single-sided unit that will only store 360K bytes. However, double-sided drives are now available that store twice that--nearly as much as Amiga's double-sided format.

I presume that in the future one will be able to opt for the double-sided instead of the single-sided drive when purchasing a system, but as far as I know, that is not possible yet. One advantage of the ST's disk format, somewhat offsetting its smaller-than-Amiga capacity, is that it is virtually identical to the format of IBM disks. This should make transfer to/from Big Blue's hardware easier--when, and if, the 3.5-inchers become more common in business computing. (By this I mean that the software--if, indeed, any is required--to allow STs to read IBM disks, and IBMs to read ST disks, will be less elaborate than that needed by Amiga). The disk read/write speed is similar to the Amiga's--i.e., enormously faster than 4040/8x50 rates. Also like the Amiga--and unlike the Mack--the disk drive has a button so you may eject a disk. The drive makes a peculiar (soft) grinding noise unlike any I've experienced before, but its performance seems first rate.

Although the standard system has only a single drive, the Trameil Operating System (TOS, the ST's DOS) "desktop" shows two disk icons, and either may refer to the one drive. TOS keeps track of which floppy was last inserted in the drive and prompts for a disk change when needed. This means that making backups is much simpler than, for example, using a PET with a single 2031. However, disk-swapping does get tedious--would you have guessed there wasn't enough free RAM in this .5-Mbyte machine to copy a 360K floppy in one buffer-load? Me either.

The Video Monitor(s): The color monitor I got with the ST equals the best I have ever seen. At 11.5" diagonal, its screen is about the same size as SPET's. Its colors are very clear with very crisp definition--for a color display. After all, one can see the individual phosphor dots/stripes on any color CRT, and that puts a limit on the resolution! I've seen the monochrome unit at the dealer's showroom; if memory serves, it is similar in size to the color unit. But, of course, it has vastly better resolution than any color display.

The Keyboard: The keyboard housing holds both the keyboard and the computer proper, so it's rather bulky. For that reason--plus having cables to all the other parts connected along its backside--one can't hold it conveniently on one's knees, as may be done with the keyboards of the IBM-PC, Amiga, etc. It has a "Selectric" layout. In fact, it is apparently a clone of the DEC VT-220 keyboard--something of a standard in its own right. The keyboard feel is a little closer to that of SPET than to either IBM-PC or Amiga (A Tramiel influence?). It also has a feedback "click" (audible on the monitor speaker) that one may turn off. I haven't used it for straight typing enough to have a definite opinion, but the key placements seem quite livable. Surely, anyone who has tolerated SPET's keyboard won't find the ST's too bad!

The Mouse: I hate mice. The only thing for which they're better than a keyboard is pointing at icons--and I hate icons. To me, pressing a key to select a menu option--a la Lotus 1-2-3 or Multiplan--is far better than sliding things around on my desk (and, yes, knocking them off the edge). As on the Amiga, one may manipulate the mouse pointer and buttons with keystroke combinations. Alas, also as on the Amiga, such manipulation is awkward. Missing is the Amiga's ability to change the scale of mouse movements, so one cannot reduce the amount of desktop required for the mouse.

The ST mouse is used with TOS (or, more properly, with GEM) in the same way as is the Amiga's with the Amiga Workbench. That is, "point and click" or "double-click" or "point and drag." However, I find "double-clicking" to be more difficult on the Atari--a fault, I'm sure, of TOS/GEM. One somewhat significant difference between the two is the mechanism for accessing menus. On the ST, one merely slides the mouse up to the menu bar on the desired item; and the submenu for that item "drops down" on the screen below. Then the desired submenu item is selected by pointing to it and pressing the left mouse button. On the Amiga, one presses the right mouse button in order to bring the main menu into view on the menu bar. From there on, the procedure is essentially indentical to the ST's.

OPERATING SYSTEM: The ST's operating system is much closer in concept to the Macintosh's than is AmigaDOS. (Bear in mind that this assertion is made by one who's never touched a Mack!) It is alleged to be a combination of products from Digital Research--GEM and a a modified version of CP/M-68K, called TOS (for Tramiel Operating System). I've always suspected that this construction of the acronym was a joke, but I've never heard an alternative mentioned. TOS/GEM is strictly single-tasking, and allows a maximum of only four screen windows at a time. Also, it was apparently not intended to provide the user the alternative of a "command line interface" such as the Amiga's CLI.

The ST wakes up showing a "desktop" just like the Macintosh and Amiga. All interaction with the computer is via the desktop metaphor. A small program is included with the "developers' kit" ($300 extra) that implements a CLI, but very few commands are available--mainly, DIR, COPY, and I/O redirection. Presumably,

one could port CP/M-68K .CMD files to the ST, but none are included in the kit. TOS/GEM also supports the now-usual nested sub-directories, (alias "folders"). A holdover from CP/M is the restriction to eight-character filenames with three-character extensions--a curious limitation in these days of megabyte systems. [Ed. You're too kind, Terry. It is more than curious; it is plumb insane.]

Although I've heard one may now get a preliminary ROMed version of TOS/GEM for $25 from Atari, the machine, as now marketed, comes with the 200K+ OS on disk. Because TOS/GEM is about the same size as AmigaDOS; and because the ST does not have the Amiga's extra "Writable Control Store" RAM for the OS to load into, the 520 ST has about the same user RAM as a 256K Amiga. This means, for example, that one cannot really make good use of a RAMdisk on the ST until its RAM is upgraded to a megabyte or one installs the OS ROMs. Ads have already appeared offering the memory upgrade for as little as $175(!). Detailed instructions for the do-it-yourselfer appear in the February '86 Byte (pp. 372 ff). Several RAMdisk drivers are available, commercial and public domain, but none comes with the system, unlike the Amiga, where it is built into DOS.

DOCUMENTATION: My ST came with very little documentation. There was an 81-page "Owner's Manual" and a 69-page unbound "Sourcebook for ATARI Logo." More recently, I've received at no extra cost from my dealer several additional software packages, including documentation--but only for the new software. Nothing added came for the machine or Logo. The supplemental software included ST BASIC, Atari Writer, 1ST Writer, DB Master, a color graphics editor named "Neochrome," and a game called "Megaroids."

As I alluded earlier, there is a "software developers' kit" available from Atari for $300. This package includes several diskettes containing some utility and "read.me"-type system information files, a C compiler, macro assembler, linker and editor, together with a pile--over 6 inches thick--of loose xerographed sheets of miscellaneous information. This package is (for the time being) a must if you are planning any programming on the ST. The included editor and language-software are not terrific. (But then, neither are those of the Amiga developers' package!) Generally, the available ST machine information cannot compare favorably with that on the Amiga. The situation is reminiscent of--but not as bad as --that during the first year of the PETs: If you know where to look, you can find it, but it ain't in the index. (Tramiel influence, again?)

LANGUAGES/APPLICATION SOFTWARE: As I said before, the only language my ST came with was Logo. However, based on what my dealer has subsequently given me free, I take it that the current machines include ST BASIC, DB Master, 1ST Writer (and maybe Atari Writer?), Neochrome, and Megaroids. Another nice touch included with TOS/GEM is a terminal program that emulates the DEC VT-52. I have not really used this yet, but it is reported to work.

I consider both Atari ST Logo and ST BASIC to be too slow to be useful. Beside sloth, ST BASIC has an over-abundance of windows for the user to contend with. There are four(!!) in all: (1) direct mode/command, (2) program listing (why?!), (3) program output, and (4) program edit. At least ST Logo, apparently a derivative of DRI's DR Logo, is contented with three windows--a "dialogue" window, an "edit" window, and one for graphics output. Note that ST Logo's environment is quite similar to that of SPET's mBASIC. Except that SPET has no graphics capability, the major difference between the two is the ability to split the screen between the edit and dialogue windows--if only it weren't so god-awful slow! Paging the edit screen takes at least a second! (Give me BEDIT!)

I've played with both of the word processors a tiny bit. I prefer Atari Writer to 1ST Writer, in spite--or perhaps because--of the fact that Atari Writer does not use of any of the GEM/mouse stuff; and 1ST Writer uses both extensively.

Neochrome comes with a series of demonstration pictures that are quite spectacular; including a "Mandelbrot Set fractal" that is beautiful, and an animated waterfall. The editor itself provides the best excuse I've seen to date for the mouse, allowing one to doodle on the screen in many colors, fill areas, spray paint (i.e., incompletely fill) areas, etc.--much fun for those so inclined. My apologies to you data-base fans, but I haven't even fired-up DB Master to see its sign-on screen.

Aside from the freebies that come with the system, new commercial offerings appear monthly. I've seen reports of several C compilers, a UCSD Pascal package, and a BASIC interpreter. All are alleged to exist now; numerous products are supposed to be coming "real soon now."

SUMMING UP: The inevitable question, of course, is "Which is better--Atari ST or Commodore Amiga?" The answer: "It depends." Both machines embody hardware one could only fantasize about a scant few years ago. The ST has a clear edge in price and is already a quite useful system with much potential. I'm confident that the quality of its language and application software support will improve rapidly. The Amiga has several nice touches lacking on the ST. (Many of these, however, could easily be added by appropriate new software.) Probably the significant difference in the long run will prove to be the ST's lack of memory expansion capability. Believe it or not, when you can see the tantalizing possibilities offered by the 680x0 microprocessor's gigantic memory space, one crummy megabyte doesn't seem enough! If you're leaning toward the ST, and expect to keep the same system for a time, I suggest that, before you settle on the 520 ST, you investigate the recently announced 1040 ST. The 1040 already has the one megabyte of RAM that is the 520's maximum, and it may be able to expand further.

DIRECT DISK ACCESS IN 6809
or,
The Delphic Mystery Explained
by Reg Beck

[Ed. Although many have tried, nobody until now has been able to handle direct disk access from the 6809 side of SuperPET. Because of a fruitful exchange of data between Stan Brockman, Joe Bostic, and Reg Beck, plus a lot of hard work by all, this barrier has been removed. Three cheers!]

I've been trying to get the U1, U2 and Block-Pointer commands to work from APL to take advantage of the power of APL in manipulating records as arrays. Apparently, the Waterloo ROMs lack routines to handle these commands. A letter to Waterloo yielded nothing. An SOS to Stan Brockman found him working on the same problem. He agreed to share information. I quote Stan, "Things to watch for: 1) how it seems to be necessary to force the disk to actually do the read, 2) how to deal with flag bits in the file control blocks (FCBs) to make up for deficiencies in Waterloo's ROM routines and 3) to force U2 to actually write the data back to the disk."

We need a map to work with FCBs. The following comes from Joe Bostic (of BEDIT fame). An FCB is actually made up of two parts. The first part (8 bytes) holds a pointer to the second part (13 bytes). Apparently the second part immediately follows the first part most of the time. To be safe, the pointer in the first part should be used to access the second part.

# of Bytes:	Description, Part One of FCB
[1]	0=FCB allocated, 1=FCB area is free to be reallocated. Put in brackets here because this byte is actually 1 less than the first byte of an FCB. A pointer to this byte is located in $73 just after the file is opened.
1	File type: F,T,V,P (FCB pointer returned after opening points here) [F=fixed, T=text, V=Variable, P=program??]
2	Record size.
1	Error flag: 0=OK, 1=EOR, 2=EOF, 3=Error.
2	Pointer to part 2 of FCB (it may or may not immediately follow part 1).
2	Pointer to next FCB in the FCB buffer area.

Description, Part Two of FCB

1	Error flag: (copy of one in Part 1).
1	Access mode byte: Bit 7 - Load program option. 6 - Replace write option. 5 - EOR or EOF (not sure about this one). 4 - Have read past last byte of record. 3 - Read is in middle of record. 2 - Append mode. 1 - Write mode. 0 - Read mode.
1	File type: 1=disk, 2=printer, 3=IEEE, 4=host, 5=terminal, 6=serial, 7=keyboard.
2	Current position in record count.
2	Size of record (sometimes). This location is somewhat mysterious.
2	Timeout constant to use with this file. Value is in 10ths of a second.
1	Device number. \ These two locations hold the pointer to
1	Secondary address number. / the ACIA chip if HOST file.
1	Drive #.
1	Status byte: [Each TRUE conditions sets the correspnding bit] Bit 7 - ? Bit 3 - Relative file format. 6 - Ready to start (maybe). 2 - Command channel read. 5 - Program file format. 1 - Catalog file. 4 - Sequential file format. 0 - File at EOF.

You must clear bit 0 of the status byte after each read so that the next read will be successful; the EOF bit is not automatically cleared by the DOS. From mBASIC and APL, we can PEEK and POKE the access and status bytes. Memory locations $73, $74 hold the location (-1) of the FCB right after a file is opened. By counting bytes in part 1 of the FCB we can PEEK the pointer to part 2 and then add 12 bytes to reach the status byte. In machine language, this is even easier, as the address to the first byte of the FCB is returned in the D accumulator. If this is transferred to temporary storage and then later moved to the X register, we have:

```
; X contains the pointer to the FCB
        ldx  4,x        ;X now points to part 2 of FCB.
        ldb  12,x       ;B now holds the status byte.

reset   EQU     *       ;We now pass to one of many possible subroutines.
        LDX     fcb2    ;address of FCB was stored in fcb2
        LDX     4,x     ;find second part of FCB
```

```
        LDB     12,x    ;find status byte
        ANDB    #$fe    ;mask zeroth bit to 0.
        STB     12,x    ;clear EOF bit
        LDB     1,x     ;find access byte
        ANDB    #$e7    ;mask 3rd and 4th bits to 0.
        STB     1,x     ;clear 3rd and 4th bits.
        RTS
```

The following mBASIC program demonstrates opening and using a disk buffer, use of the B-P command, and allows us to examine the workings of the Access Byte when a file has been opened for update (Read and Write).

```
170 ! Set up Variables and Masks            Program written 17 Dec 1985
180 !                                       by Stan Brockman
190 b_p$ = "B-P 2 0"
200 test_str$ = "This is a test"
210 mask1% = 231:       !($E7)
220 mask2% = 1:
230 mask3% = 0:
240 mask4% = 254:       !($FE)
250
260 ! OPEN  FILES (#8 = CHAN. 15, #9 = CHAN. 2 TO DISK BUFFER) and set up
270 ! pointers to the second part of the File Control Blocks.
280
290 f_ptr = 16*7 + 3:           ! addr. containing addr. (-1) of FCB
300 open #8, "disk8",inout
310 fcb8 = 256*peek(f_ptr) + peek(f_ptr+1) + 1
320 fcb8 = 256*peek(fcb8+4) + peek(fcb8+5)
330 print "Part 2 of Ch. 15 FCB at addr.";fcb8;", Access =";peek(fcb8+1)
340 open #9,"ieee8-2.#",inout
350 fcb9 = 256*peek(f_ptr) + peek(f_ptr+1) + 1
360 fcb9 = 256*peek(fcb9+4) + peek(fcb9+5)
370 print "Part 2 of Ch.  2 FCB at addr.";fcb9;", Access =";peek(fcb9+1)
380 var2% = peek(fcb9+12):poke fcb9+12,(var2% and mask4%): !Clr Ch. 2 EOF bit
390
400 ! Read the Disk Buffer (Just for Practice)
410
420 print #8, b_p$:                  ! Point at 0th byte in disk buffer
430 for i = 1 to 500
440   get #9, a$
450   var1% = peek(fcb9+1): var2% = peek(fcb9+12)
460   if ((var2$ and mask2%) = 1) then goto 480
470 next i
480 call info("first",i)
490
500 ! Reset Buffer Pointer and Access/Status Bytes, then write to the buffer
520
530 print #8, b_p$
540 poke fcb9+1, (var1% and mask1$):poke fcb9+12, (var2% and mask4%)
550 for j = 1 to len(test_str$)
560   print #9, test_str$(j:j);
570 next j
580 var1% = peek(fcb9+1):var2% = peek(fcb9+12)
590 call info("second", j): print
```

```
595
600 ! Reset buffer pointer & access/status bytes, then read from the buffer.
630 print #8, b_p$
640 poke fcb9+1, (var1% and mask1%): poke fcb9+12, (var2% and mask4%)
650 for i = 1 to j-1
660   get #9, a$
670   print a$;
680 next i
690 var1% = peek(fcb9+1): var2% = peek(fcb9+12)
700 print : print : call info("third",i)
710 close #8: close #9
720 stop

740 proc info(b$,ij)
750   print : print "Index of ";b$;" loop is"; ij
760   print "Access byte is";var1%
770   print "Status byte is";var2%: print
780 endproc  ! Screen dump follows:
       (Item 1)
Part 2 of Ch. 15 FCB at addr. 1312 , Access = 3
Part 2 of Ch.  2 FCB at addr. 1286 , Access = 3
       (Item 3)
Index of second loop is 15
Access byte is 19
Status byte is 80
       (Item 5)
Index of third loop is 15
Access byte is 11
Status byte is 80
```

[ED. We had to squeeze like mad to get the following screen dump in this issue. Read ACROSS the page for the sequence of events.

(Item 2)
Index of first loop is 256
Access byte is 19
Status bye is 80
(Item 4)
This is a test

[Ed. We'll have a follow-on article on this subject in the next issue.]

Prices, back copies, Vol. I (Postpaid), $ U.S. : Vol. I, No. 1 **not** available.

No. 2: $1.25	No. 5: $1.25	No. 8: $2.50	No. 11: $3.50	No. 14: $3.75
No. 3: $1.25	No. 6: $3.75	No. 9: $2.75	No. 12: $3.50	No. 15: $3.75
No. 4: $1.25	No. 7: $2.50	No. 10:$2.50	No. 13: $3.75	Set: $36.00

----------Volume II----------

Numbers 1 thru 9: $3.75 each.

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

==========

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)

[] Check if you're renewing; clip and mail this form with address label on the reverse side. If you send the label, don't fill in the form below.

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 in U.S. Dollars. DUES ELSEWHERE: $25 U.S. Mail to: ISPUG, PO Box 411, Hatteras, N.C. 27943, USA. **SCHOOLS!: send check with Purchase Order.** We do not voucher or send bills.

This journal is published by the **International SuperPET Users Group** (ISPUG), a non-profit association; purpose, interchange of useful data. Offices at PO Box 411, Hatteras, N.C. 27943. Please mail all inquiries, manuscripts, and applications for membership to Dick Barnes, Editor, PO Box 411, Hatteras, N.C. 27943.

Send appropriate postpaid reply envelopes with inquiries and submissions. Canadians: enclose Canadian dimes or quarters for postage. The Gazette comes with membership in ISPUG.

Table of Contents, Issue 9, Volume II

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

First Class Mail
in U.S. and Canada.
Air Mail Overseas.