# open apple gazette

Fourth Edition Volume 1, Number 4 September-October 1982

## THE QUARK CATALYST FOR THE APPLE ///

by

ALLAN M. BLOOM, PhD CDP

### Introduction

The Quark Catalyst for the Apple /// ($149 from Quark Engineering, 1433 Williams, Suite 1102, Denver CO 80218, 303-399-1096) is billed as a utility allowing you to "boot from your hard disk." Quark's advertising goes on to say that there is no longer a need to swap disks and re-boot when changing programs and interpreters, that "practically all your programs can be put on the hard disk." The Catalyst becomes the boot diskette and other programs and interpreters are initiated from Catalyst's menus. Further, with a spooler -- such as Quark's Discourse , you can spool a file for printing and switch to another environment immediately, without losing the spool file. To conclude the series of Quark's advertising claims, "you will never need to shuffle floppy disks again. You can lock your copy of Word Juggler, Visicalc, etc. in a vault for safekeeping."

For those of us with ProFile, the Profile manual's instructions (Appendix B) on setting up the Pascal Language System on the hard disk are something of a boon -- requiring only a single PROFILEPASCAL boot diskette to initiate the system. However, it really doesn't take much of a SOS.DRIVER or SYSTEM.LIBRARY -- both of which must be on the boot diskette -- to get one cramped for workspace for the SYSTEM.WRK.TEXT and SYSTEM.WRK.CODE files. Both, again, must be on the boot diskette. In program development, relief from such a nuisance could be worth the purchase price all by itself. I would suspect that is the primary reason that Catalyst is selling faster than Quark can photocopy its User Manual. For some odd reason, Quark's advertisements do not show all the benefits that its package gives.

- Continued Inside -

**Original Apple ///rs**

**CLUB INFORMATION**

MEETINGS

Meetings are held at 7:30 PM on the third Wednesday of each month. The location is the Board Room of the California Bar Association offices at 555 Franklin St. San Francisco.

MEMBERSHIP

Annual membership dues are $25 from the date application received. Your check payable to the Original Apple ///rs may be mailed to the address below.

OPEN APPLE GAZETTE POLICY

All manuscripts, photographs, and other materials are submitted free and released for publication. They become the property of the Original Apple ///rs and the Open Apple Gazette. Authors should clearly mark all material submitted for publication so that credit may be given.
The publishers/editors do not necessarily agree with, nor stand responsible for, opinions expressed or implied by other than themselves in this publication.
The Original Apple ///rs is a non-profit organization comprised of, and supported by, Apple /// owners and users. The Original Apple ///rs is run by volunteer officers and committees, and the club endeavors to aid other Apple users through this educational publication - "OPEN APPLE GAZETTE". Address all inquiries to: Original Apple ///rs, P. O. Box 813, San Francisco, CA 94101.

REPRINT POLICY

All articles appearing in the Open Apple Gazette not copywrited by the author may be reprinted by another non-profit Apple user group so long as proper credit is given to both the Open Apple Gazette and the author. Proper credit is defined as article title, author, and the words "Printed from VOL X, NO Y of the Open Apple Gazette." Permission to reprint a copywrited article may be obtained by writing to the author c/o the Original Apple ///rs.

OFFICERS

| | | |
|---|---|---|
| PRESIDENT<br>673-7635 | Don Norris | (415) |
| VICE PRESIDENT<br>521-1771 | Kent Hockabout | (415) |
| TREASURER | Julia Amaral | |
| SECRETARY<br>386-8623 | Charles Coles | (415) |
| CONSULTANTS | Randy Fields<br>Ken Silverman | |

- /// -

ARTICLE SUBMISSION POLICY

The Open Apple Gazette welcomes any and all articles dealing with the Apple /// Computer and its associated hardware and software. Articles may be submitted doublespaced and typewritten, or on the APPLE WRITER /// word processor.
We will send your disk back to you as soon as we output the article on our printer.

**Public Domain Software for the ///**

Public domain software for the Apple ][ was undoubtedly one of the primary reasons for its success. This software enabled owners to learn more about their machines and how to use them profitably. Public domain software for the /// has been slow in coming but here are some of the first that are available. The Applecon program from Apple Computer Inc. will greatly add to the library of public domain software for the ///. You can help with this by sending us programs you have converted to or written for the ///.

Applecon from Apple Computer Inc.

Applecon is a new utility for the Apple /// which converts Applesoft BASIC programs to Apple /// Business BASIC programs to the extent that they can be machine converted. This program will not convert any copy protected programs or diskettes. This utility will take an Applesoft (Apple II) program and move it up to SOS and into Apple Business BASIC and then will make the proper changes. Those lines it cannot convert directly

into Business BASIC will be flagged into a REM statement for you to correct. The disk comes with several pages of documentation on the disk in a text file. The file can be read by Apple Writer ///, or you can output it via the Pascal System.

File Cabinet ///

This is a small general purpose data base management system written in Business BASIC. The use of File Cabinet /// is simple and most of it is self documenting. File Cabinet provides a means of interactively defining data files, entering data, sorting, retrieving records containing specific data, deleting records, and printing reports. Because all of the data in File Cabinet is memory resident the size of the data base is limited to a relatively small amount but the handling of this data is very fast.

DOS to SOS text File converter.

This program enables you to move DOS 3.3 text files to SOS. It is useful in moving VisiCalc Models from the ][ to the ///. If you own Apple Writer the Apple Writer Utility diskette already will do this for you.

These diskettes are available to members for $8.50 each. Non Members $10.00. Canadian Residents add $ 1.00 for postage, add $2.00 for other foreign postage. Make your checks payable to the:

Original Apple ///rs
P. O. Box 813
San Francisco, CA
94101

**Back Issues**

Open Apple Gazette
Volume 1 Number 1 $ 3.00
Volume 1 Number 2 $ 4.00
Volume 1 Number 3 $ 4.00
Mail requests for back issues to:

Open Apple Gazette
P.O. Box 813
San Francisco 94101

- /// -

**CATALYST**

Continued

Less surprisingly, the advertisements do not mention the drawbacks of the program product, and the product's documentation is less than perfect. This article presents one user's experience with implementing the Catalyst, both the good and the bad. Not to keep you, gentle reader, in suspense any longer than necessary, I believe that Catalyst is, on balance, a superior product. I recommend it to any serious Apple /// programmer.

This article begins with an examination of the faults that seem to exist in the product and in its advertising and documentation. There follows an entry about the idiosyncracities found in operating systems -- not a complete set, alas -- under Catalyst as opposed to in a stand-alone environment. While it may be unfashionable in this day of criticism's being viewed as negative, I close with a rather unusual number of hosannnahs. As I said, I like the product.

Pre-Purchase Caveats

The advertised list price of Catalyst is $149. If you've no truck with ordering from your friendly local retail dealer, Quark is quite willing to sell direct at the same price, no shipping or handling charge. In my case, however, the price actually came closer to $1,149. One thing that the advertisements don't tell you is that a 256K machine is necessary to get the full benefit of the product. With Catalyst, all SOS drivers that you may ever need must be in memory, in addition to Catalyst itself. With a 128K machine, user working storage is a bit cramped. For example, SYSTEM UTILITIES cannot run under Catalyst in a 128K machine and must be separately booted. Since the ///'s are now shipped with 256K, that is certainly no problem to new purchasers. I had planned to upgrade my memory anyway, though not immediately, so my only real loss was a few months of savings account interest. Others in different circumstances could feel a bit sandbagged.

If you have a 128K machine, and the interest/ability to upgrade to 256K, I

suggest ordering the memory at least coincidental to ordering Catalyst. My friendly local dealer's promise of two-days delivery of the memory board was off by an order of magnitude. If you don't wish to upgrade immediately, add $40 to the real purchase price -- the cost of replacing your Catalyst diskette and its backup. Once configured, there is no going back. If you don't mind losing the available user memory, you might best re-consider your purchase decision. There could well be sufficient nuisance value to make $149 less than cost effective.

A second caveat arises from Catalyst's ability to handle even copy-protected disks. The advertisements make no mention of the fact that protected packages -- Visicalc, Word Juggler, Applewriter ///, etc -- are useless for anything else after Catalyst has taken them unto its bosom. To quote page 1-1 of the Catalyst User's Manual, "Catalyst will permanently lock these disks to itself. The original disks will no longer be bootable. They will also not be copyable by any other catalyst." Be prepared to spend the money to replace any copy-protected diskette that you wish to run under Catalyst.

As a last caveat, try not to be any more surprised than absolutely necessary that your particular application does not fall under the umbrella of "practically all your programs can be put on the hard disk." Your 1981 version of Applewriter /// does not fall into the "almost all" category. Your hard disk that is not a ProFile isn't covered, either. The former is an insurmountable obstacle. The latter requires at least consultation with Quark.

## Surviving the User Manual

The installation instructions END, of all the dumb things, with a short section titled "BACKING UP YOUR HARD DISK." If I might make a suggestion, keyed to dolts like me, the Catalyst installation instructions should BEGIN with

BACK UP ANYTHING ON PROFILE THAT YOU'RE FOND OF!

I recommend that bit of advice to anyone installing Catalyst, if for no other reason than a deep respect for Murphy's Law. I don't know what I did wrong, but my ProFile directory was damaged after the first installation pass. It was not badly enough damaged to prevent my copying of everything that I absolutely had to onto diskette, but I was in abject terror for a while. Much like a collision at sea, breaking a Profile directory can ruin your whole day.

My next-to-last quibble relates to step 16 of Catalyst installation procedures on page 2-2 of the manual. Instruction 16 states "If you have a serial printer, skip to step 29," There follow a set of instructions for installing the driver for a parallel printer. One should "skip to step 29" only if his or her serial printer is attached to the built-in RS232C interface and uses .PRINTER or .QUME as the SOS driver. My printer is serial, and it is a Qume, but it is attached to an Apple ][ Super Serial Card and driven by SERIAL.X.DRIVER. I can't imagine how I missed tripping myself up on that one. I certainly stubbed my toe on everything else. Others may be sandbagged.

My last quibble is related to the real or apparent typos in the manual that I thought you might like to know about. They are not many, but they are significant when the manual states (page 2-1) "even the sophisticated user should follow these procedures to the letter."

1. Page 2-5: Setup instructions 5-6 are duplicated in instructions 9-10.
2. Page 2-22: Instruction 17 is "Change the name of the SYSTEM.STAR.LIB file to SYSTEM.STAR.LIB."
3. Page 3-6: Instruction 6 ends with "Not all interpreters this prefix setting."

I remain convinced that glitch 1 was responsible for my blowing the ProFile directory. Follow steps 9-10 at your peril.

## Installation Pitfall

On to a more substantive problem with product installation. Catalyst has two menus -- the Catalyst menu itself and one for application programs that you might wish to run under either Basic or Pascal. Both sets of menu entries (named INTERPS

and x.MENU, respectively) can be added to and edited. It would be awfully friendly if Quark could add some sort of write protection flag to the entries in the INTERPS and MENU files. The Catalyst Edit, Sys Reboot, and Menu Editor subfiles are necessary to the system. They can be as freely deleted as any user-added subfile. Being perhaps the perfect system tester, and an almost perfect dummy, I deleted Catalyst Edit. Bless them for telling us the format of INTERPS file entries (Appendix C) and for designing INTERPS as an ASCII file that Pascal can edit. Rather than re-initializing INTERPS from the Catalyst backup diskette and regenerating the user entries, I just noted the Catalyst Edit entries on the backup diskette and used Pascal's editor to replace the deleted subfile in the on-line INTERPS file.

It is obscene to not write-protect necessary files. Under Catalyst 1.0, however, you must pay attention to what subfiles you are editing and to where in the editing process you are. Beware.

## Wierd Things With Mail List Manager

The naming of section subtitles is an art form. The above is dissatisfying, but I'm at a loss to come up with anything that better expresses the particular situation. Very wierd things happen with Mail List Manager (Version 1.0) under Catalyst. Upon initiation from the catalyst menu, the program announces that it "can't find MLMSET." This is a baldfaced lie. When I booted a copy of MLM without MLMSET, the same message appeared, for real. The screen sure looked funny, too. I wonder why running under Catalyst causes the phoney message?

A pleasant surprise, once I figured out what was going on, is that MLM 1.0 renumbers the diskette drives. The internal drive becomes .D2, the first external drive becomes .D3, etc. Once one knows about it, the situation is delightful -- effectively having an extra external drive for MLM data files. Unless, of course, one has the full complement of drvies already. Rather interestingly, MLM 1.0 reads from and writes to my "phantom" .D3 without that drive being configured into the system, but it will only initialize an MLM volume on the drive if I lie to the System Configuration Program and tell it that I have 3 Disk /// drives.

## Access /// Rumor Debunked

A colleague indicated that he'd heard stories of insurmountable difficulties with installing Access /// under Calalyst. I had no problems at all, even in translating my one-step kluge (a buggered SETPREFIX not only setting the prefix to .PROFILE, but also automatically initiating ACCESS3 via chaining) to run under Catalyst. The rumor appears baseless.

## Hosannahs

I've discussed the drawbacks and pitfalls of Catalyst to this point. It is, perhaps, past time to discuss the product's strengths. It is no small praise to say that Catalyst does what it claims to do. Every advertised feature is implemented.

I'd like to end with a few words of praise for some features of Catalyst that the advertising didn't mention and that make life just a little more pleasant. I very much like the boot program knowing enough to wait for the ProFile to warm up rather than making me re-boot. I am also fond of the automatic date/time checking. Until the built-in clock/calendar becomes a reality, it is very nice to have that reminder. Using Pascal under Catalyst is also something of a system-saver. Under PROFILEPASCAL, the interpreter always looks for a needed file on .D1, then on .D2, before finding it on ProFile. That is not only a nuisance. It also causes unnecessary diskette and head wear. With Catalyst, the situation goes away. Nice.

## Summary

In summary, I'm delighted with Catalyst. The pleasant surprises far outnumber the unpleasant ones. The package is, however, far from idiot-proof. If you decide to go the $149, or $1,149, be prepared to be a slavish follower of each and every imnplementation rule (except maybe steps 9-10 of page 2-5). Not being so can result in many hours and dollars wasted. After set-up, however, the beast is an absolute dream. To date I've installed System Utilities, Pascal, Access ///, and Applewriter /// -- plus some of my

commonly used Pascal programs. Catalyst is just as quick and easy to use as the advertisements claim.

Editors Note: Catalyst and Discouse once you get them operational on your Profile are two fantastic pieces of software. The ability to change between programs so easily is a real plus with the ///. Next issue we will have even more about Catalyst and Discourse.

- /// -

## Softcard Patches

### Wordstar

-Install Wordstar normally for "any terminal"; accept other defaults as shipped from MicroPro.

-Enter DDT program

```
>DDT wsu.com

Fill $240-$2AF with 0

Substitute starting at $190:  41 50 50 40
45 20 2f 2f 2f 20 2e 43 4f 4e 53 4f 4c 45

Fill $1a2-$1af with $20

Substitute at $246 18 50 18 50 01 1a
              $25d ff
              $26d 01 1f
              $284 01 12
              $28b 01 11
              $292 04 10 03 01 1c
              $29b 01 1c

Exit ddt
```

Save the TPA (transient program area) to a new file, e.g. WSTEST.COM, test the program and fix if necessary.

You should read the manual on DDT to learn how to load a file, fill, substitute, and save the TPA.

I believe that Wordstar is 42 hex pages long ($4200) so you should specify 42 pages when using the SAVE command of DDT.

You can check by using the DDT command to show length of the program just loaded, following the LOAD command. See manual for an example.

### Dbase II -

Run the install program provided with the Dbase package, selecting the Soroc option for video setup.
Accept the basic defaults.
Save the file.
Run the install program again.
Select the user-defined video option.

There will be eight steps in all. You can learn the meaning of the screen control codes from an appendix in the back of the Standard Sos Drivers Manual. Knowing these codes will help keep you in synch with the installation steps, just in case you get lost.

```
Select Decimal data inputs
(1)  21,0,0 - clear screen and home
     sequence (3 bytes of data)
(2)  2,0,0
(3)  26,0,0 - select skeleton
(4)  28,0,0
(5)  17
(6)  18
(7)  0
(8)  17,26,0,22 - 4 bytes of data
(9)  17
```

If you revise the inputs at any step, be sure to re-enter the data when prompted. If you don't, the values will be set to zero. I.e.

ZIP needs to be installed as well, using the ZIPIN.COM program. Use the same screen control codes as above when prompted, (read the Device Drivers manual for help understanding the screen control codes).

- /// -

**VisiCalc Template Disk no. 1**

Original Apple ///rs
VisiCalc /// version of

San Francisco Apple Core
VisiCalc Templates

Converted by Don Norris & Ann Riley

The disk contains a collection of VisiCalc templates and demos which will prove useful to home and business users. They were produced by the VisiCalc users group of the San Francisco Apple Core for use on Apple ][, they have been converted to Apple /// VisiCalc format.

This was done in several steps. First using the Apple Writer Utilities diskette the Apple ][ VisiCalc templates, which are DOS 3.3 text files, were converted to SOS text files. Since all labels on Apple ][ VisiCalc files are UPPER CASE only, the next step of the conversion used an Apple Writer /// Word Processing Language Program, to convert all of the UPPER CASE labels to lower case. The final part involved loading the models in to VisiCalc ///, changing to UPPER CASE any lables which required it, such as at the beginning of a sentence.

Catalog of VisiCalc template disk no. 1

| Title | Use |
|---|---|
| Catalog | utility |
| Net Worth Worksheet | household mgt |
| Linear Regression | statistics |
| Exponential Regression | statistics |
| Income Tax Demo | household mgt |
| Stock Analysis | financial |
| Spearman Rank | statistics |
| Pearson Product | statistics |
| Budget Template | household mgt |
| Small Business Start Up | business |
| Net Worth Demo | demo |
| Small Bus Start Demo | demo |
| Comparative Shopper | household mgt |
| Comp Shopper Demo | demo |
| Avon Records Demo | demo |
| Bank Reconciliation | business |

The disk also has the WPL routine that was used to convert the upper case Apple ][ VisiCalc characters to lower case and instructions for its use.

A brief explanation of each template follows below:

Catalog -- This template lists and briefly describes each template on the disk.

Net Worth Worksheet -- This template works down the length of a few columns. It builds from a series of worksheets to a net worth statement by bringing the subtotals of each asset or liability sheet to the appropriate line in the statement.

Linear Regression -- This template computes the slope, y intercept, & correlation factor of 15 points listed as x and y values on the sheet. It also computes the predicted y value for a new x value. You may substitute x and y values in the sheet as you desire, insert rows, and replicate formulas as required to expand the sheet. The least squares smoothing method is used in this template.

Exponential Regression -- The template is used to smooth a series of values to provide a curve fit by using exponential smoothing methods. Once a series of x and y values has been input into rows a and b, the y value for a new x value may be computed by inputting the new x value.

Income Tax Demo -- This template uses annual amounts which are entered into each category listed for 1980 and 1981 to compute the total tax burden for each year. The window used on the bottom of the screen is used to highlight the tax refund and burden for each year. The federal tax burden is computed by using a lookup table for the 1980 tax. Another lookup table is used to compute the 1981 tax level.

Stock Analysis -- This template allows the user to compute the long term and short term gains or losses of his portfolio. Current price must be entered from the daily paper. Number of shares, stock name, purchase price, and commission should be entered as stock is bought. Computation date and date of purchase are entered as four digit entry (year number followed by month number - i.e. 8111) in the same cell to allow the boolean statement in row i to operate properly.

Spearman Rank Analysis -- This template first computes the square of the difference between two ranked scores. Rank to be precomputed by the user. It then

computes the correlation of the differences between the ranks listed.

Pearson Product -- This template computes the Pearson Product moment coefficient from x and y data input by the user.

Budget Template -- This template allows the insertion of monthly income and expense data for a wide variety of income and expense classifications. The user may wish to replace or delete categories to reflect his needs. Monthly summaries are provided at the bottom of the sheet.

Small Business Start Up -- This template is a straightforward template which may be used to plan the first year's cash flow for a small business. It may also be used to develop budgets or examine alternate courses of action. Cost of sales may either be input for each month or may be computed by inserting a formula.

Demo Net Worth -- Demo of first template on diskette.

Small Bus Demo -- Demo of small business start up template presented above.

Comparative Shopper -- This template allows the user to select the minimum price for each item offered by three different stores. He may have to travel from store to store to obtain the absolute lowest price or he may just select the store with lowest store price. The @min command is used in this template.

Comparative Shopper Demo -- This template demos the comparative shopper presented above.

Avon Records Demo -- This template is a complex analysis of the profit and business expense associated with a series of Avon sales campaigns.

Bank Reconciliation -- This template computes whether or not a bank statement for a business agrees with the financial records. It spans a number of months and may be printed each month by the user.

Your favorite template(s) are needed for inclusion in future VisiCalc template disks. Please send it to the Original Apple ///rs. It will be added to our library as well as converted to Apple ][ version and forwarded to the San Francisco Apple Core. If your template is used on a future template diskette, you will receive a free template diskette.

- /// -

**Original Apple ///rs**
**VC Templates 83-1**

Order your VC Templates Diskette for only $ 12.50, which includes and handling. Foreign add $ 2.00, Canadian $ 1.00. Foreign orders should be in US Funds drawn on a US Bank or an International Postal Money Order payable in US Funds.

Save formatting, copying and labeling and order a backup at the same time for only $ 7.50.

Order from:
Original Apple ///rs
P.O. Box 813
San Francisco 94111

- /// -

**IDS Micro Prism Printer**

User Comments

by
Barney Simonsen

I have installed with no problems an IDS Model 480 Micro Prism Printer. It is hooked up using an RS-232 plug and Apple Modem Eliminator. I am using the 8-bit no parity setting on the Apple /// configuration. I haven't extensively explored all the features, but have successfully used the control/escape commands from the Apple Writer and Quick File to adjust the print density (10 CPI, 12 CPI, 16.8 CPI) and use enhanced printing. I have not yet tried the graphics mode.

I look forward to reading the next Newsletter and will try to get communications capability to try your new network.

- /// -

## The Third Basic

By: Taylor Pohlman
Reprinted from Softalk Magazine

Exploring Business Basic,
Part 3

Lots has happened to the Apple /// since my last article, and I appreciate all your comments about the articles in this series. We'll have a chance to pick up on some of your suggestions next month, along with more news about the Apple ///. For now we'll continue our exploration of the Business Basic file system as promised last time.

After reading this article and working with the examples, you should have a good knowledge of the differences between the text and data file types as well as more information about string handling functions and techniques.

We are going to stick with relatively simple indexing techniques for now, but next month we'll also cover some advanced indexing and file access methods to give you an idea of some of the ways that the popular data base programs retrieve data so rapidly.

Looking Back. The last article concluded with an example program that showed how the SOS file system could be used to store and rapidly find records in a file. We did that by using the random file access method that SOS and Business Basic have built in.

That technique allows file records to be numbered from 0 to 32767 and read directly without having to read all records from the beginning of the file.

The example at the end of last month's article also demonstrated that SOS uses a special storage and indexing method that wastes very little space in storing records on the disk, even if they have widely varying record numbers.

To go into further depth on this subject, and to compare the text file type we were working with last time to the more mysterious data file type, it's going to be necessary to create a more general version of last month's program. That example program allowed us to create a file that contained four pieces of information about a hypothetical parts distribution company. The data were the part number, description of the part, location in which the part was stored, and quantity on hand.

Unfortunately, the program was just designed to make some clever points about files, not to be really useful to parts companies. For example, we could only perform two functions, creating files and adding records. Most parts companies would want to look up parts, delete parts, get lists of parts, and so on. This month's version gets closer to that ideal, without denying you some of the fun of making your own changes. In addition, some of the functions that the program performs are generalized into subroutines so that we can make changes later without wholesale rewriting.

The New Parts Program. Well, now that your're breathless with excitement, here's the new version of the program:

```
5     HOME
7     PRINT
10    PRINT"Parts File Create and
      Modify Program"
20    PRINT:PRINT"Type:"
30    PRINT"      1 to Create a parts
      file":PRINT
40    PRINT"      2 to Use an existing
      parts file":PRINT
49    PRINT"      9 TO Quit":PRINT
50    PRINT:INPUT"Your selection:";a$
60    IF a$=""THEN 1000
70    a=ABS(VAL(a$))
80    ON a GOSUB 100,400
90    IF a =9 THEN 1000:ELSE 5
100   PRINT:INPUT"Name of new parts
      file:";a$
110   IF a$="" THEN RETURN
120   CREATE a$, TEXT, 64
130   PRINT"Parts file";a$;"created."
140   RETURN
400   HOME
405   PRINT:INPUT"Name of existing
      parts file:";a$
410   IF a$="" THEN RETURN
420   OPEN#1,a$
425   file$=a$
430   HOME
435   PRINT:PRINT"Modify Parts
      File";CHR$(34);file$;CHR$(34):PR
      INT
437   PRINT"Type:"
440   PRINT"      1 to add to your
      parts file":PRINT
445   PRINT"      2 to delete a part
      from your parts file":PRINT
```

```
450   PRINT"     3 to find a part in
      your parts file":PRINT
460   PRINT"     9 to quit the modify
      mode":PRINT
465   PRINT:INPUT"Your selection:";a$
467   a=ABS(VAL(a$))
470   IF a=9 OR a$=""THEN RETURN
475   ON a GOSUB 500,700,800
480   GOTO 430
500   HOME
505   PRINT:INPUT"Part number to
      add:";a$
510   IF a$="" THEN RETURN
520   a=VAL(a$)
530   IFa<1 OR a>32767 OR INT(a)<>a
      THEN 500
535   rec=a
540   partnum$=a$
545   PRINT:INPUT"Description:";a$
550   IF LEN(a$)>35 THEN
      a$=MID$(a$,1,35)
560   desc$=a$
570   PRINT:INPUT"Location:";a$
580   IF LEN(a$)>15 THEN
      a$=MID$(a$,1,15)
590   location$=a$
600   PRINT:INPUT"Quantity on
      hand:";a$
610   a=0:a=VAL(a$):IF INT (a)<>a OR
      a>99999 THEN 600
620   quantity$=a$
630   PRINT:PRINT"Record
      is:";partnum$;"/"desc$;"/"locati
      on$;"/"
640   INPUT" OK? ";a$
650   a$=MID$(a$,1,1):IF a$<>"y" AND
      a$<>"Y" THEN 505
660   GOSUB 2000
665   IF erorcode=0 THEN
      PRINT:PRINT"Record
      added.":GOSUB 995:GOTO 500
670   PRINT:INVERSE:PRINT"Record not
      added, ERROR=";:NORMAL:PRINT
      error code:GOSUB 995:GOTO 505
700   HOME
705   PRINT:INPUT "Part number to
      Delete:";a$
710   IF a$="" THEN RETURN
715   a=VAL(a$)
720   IF a<1 OR a>32767 THEN 700
725   rec=a
730   GOSUB 1800
735   If errorcode=1 THEN PRINT:PRINT
      CHR$(7);"No such part
      number":GOSUB 995:GOTO 700
740   PRINT"Delete
      ";partnum$;"/";desc$;"/";locatio
      n$;"/";quantity$;"?";
745   INPUT"";a$a$=MID$(a$,1,1)
750   IF a$<>"y" AND a$<>"Y" THEN
      PRINT"Not deleted":GOSUB
      995:GOTO 700
755   GOSUB 1900
760   PRINT:PRINT
      CHR$(7);CHR$(7);"Record
      deleted":GOSUB 995:GOTO 700
800   HOME:PRINT
805   INPUT"Part number to find:";a$
810   IF a$="" THEN RETURN
815   a=VAL(a$)
820   IF a<1 OR a>32767 OR INT(a)<>a
      THEN 800
825   rec=a
830   GOSUB 1800
840   If errorcode=1 THEN
      PRINT:PRINT"No such part
      number":GOSUB 995:GOTO 800
850   PRINT:PRINT"Part number:
      ";partnum$
855   PRINT:PRINT"Description:
      ";desc$
860   PRINT:PRINT"Location:
      ";location$
865   PRINT:PRINT"Quantity on hand:
      ";quantity$
870   PRINT
890   PRINT:INPUT"Press RETURN to
      continue:", a$;GOTO 800
899   REM
900   REM delay subroutine
901   REM
995   FOR i=1 TO 1000:NEXT i:RETURN
996   REM
1000  PRINT:PRINT"End of parts file
      program."
1010  CLOSE
1020  END
1799  REM
1800  REM retrieve a record with
      record number = "rec"
1801  REM
1805  errorcode=1
1810  ON EOF#1 RETURN
1820  DEF FN
      scan(start)=INSTR(rec$,"/",start
      )-start
1830  INPUT#1,rec;rec$
1835  IF rec$="" THEN RETURN
1840  pointer=1:length= FN
      scan(pointer)
1850
      partnum$=MID$(rec$,pointer,lengt
      h)
1855
      pointer=pointer+length+1:length=
      FN scan(pointer)
1857  Desc$=MID$(rec$,pointer,length)
1860
      pointer=pointer+length+1:length=
      FN scan(pointer)
1870
      Location$=MID$(rec$,pointer,leng
      th)
1875
```

```
      pointer=pointer+length+1:length=
      FN scan(pointer)
1885
      Quantity$=MID$(rec$,pointer,leng
      th)
1890  errorcode=0:RETURN
1899  REM
1900  REM delete a record with record
      number = "rec"
1901  REM
1905  PRINT#1,rec;""
1910  RETURN
1999  REM
2000  REM add a record with record
      number = "rec"
2001  REM
2005  errorcode=0
2010
      rec$=partnum$+"/"+desc$+"/"+loca
      tion$+"/"+quantity$+"/"
2015  ON ERR GOTO 2040
2020  PRINT#1,rec;rec$
2030  OFF ERR:RETURN
2040  errorcode= ERR:OFF ERR:RETURN
```

Well, nobody said that this series wouldn't get more interesting as we went along!

Let's take a quick look at the changes in this version of the program, as well as its major features. First, as to structure, the program looks something like this:

```
5-90          Initialization and first
      menu
100-140       Create a new parts file
400-480       Open an existing file
      and set up second menu
500-670       Add a record
700-760       Delete a record
800-890       Find and display a
      record
900-995       Subroutine to create a
      delay
1000-1020     Terminate the program
      and close files
1800-1890     Subroutine to find a
      record and load data values
1900-1910     Subroutine to delete a
      record physically
2000-2040     Subroutine to add a
      record with given data values
```

Note that for simplicity we have assumed a fixed file record structure. That is, we have hard-coded into the program the fact that the data items in each record are part number, description, location, and quantity on hand.

We have also coded into the program some restrictions as to the length of each item (lines 550, 560, and 610).

A real data-base program would have all this information stored in tables for more flexibility.

For example, there is no practical way, short of rewriting parts of the program, to add an extra data item to the records or change the meaning of the existing items.

Obviously, the more such generalizations we put into the program, the larger and more complex it will be.

Our purpose is to learn something about files first and then write the world's greatest data-base program.

To help understand the program and check out a few new features that make Business Basic really handy, let's look at the subroutines in the program.

First, examine the record retrieval routine at line 1800, which is used by the "Find" section and the "Delete" section. We will communicate any problems encountered in a subroutine by using the errorcode variable, with 0 indicating no error found.

The ON EOF statement in line 1810 will return with "errorcode" set to 1 in the event that the INPUT statement in line 1830 reads past the current end of file.

Line 1820 sets up a function definition that comes in pretty handy.

The function scan uses the Basic INSTR function to determine how many characters there are to the next occurrence of the "/"character.

Remember that we used the "/"character to delimit the fields within the string record we stored in the file. The INSTR function returns the character position of the string being searched for, starting with the position given by "start."

Subtracting the starting value from the position gives the total length of the field. More about INSTR can be found in the Business Basic manual. Give that section a look, because INSTR is one of the most useful functions you'll find. Some other Basics may use a different name for this function; POS is one example.

Line 1830 inputs the record according to the record number "rec." After checking for a "null" string (line 1835), lines 1840 through 1885 are responsible for breaking up the record into its separate fields. This is done by setting the variable pointer to the beginning of the field and then setting length to the number of characters in the field, using the scan function defined previously. The MID$ function is then used to make the assignment to the appropriate variable.

Study this section carefully to be sure you see how this works. One technique to understand routines like this is to make a diagram of the data and work through the statements while playing computer.

Now that the individual fields are assigned to the proper variable, they can be used in the calling routines (at lines 730 and 830) to display the values as desired. Later on we are going to change the structure of this file considerably, and it will be handy to be able to handle that by changing the routine at 1800 rather than making changes throughout the program.

The delete routine at line 1900 is really simple, just consisting of printing a null record over a previously existing record. As we change the file structure, this may become much more interesting.

The add routine is also simple, consisting (at line 2010) of packing the various field values together using the String Concatenation Operator (the world's longest way to refer to "+"). There is one thing of interest, however. Note that the ON ERR statement is used to trap any errors which may occur in writing to the file. We again use errorcode to communicate that an error has occurred and are careful to turn error trapping off before returning to the main routine.

It would have been possible, and even desirable, to use the ON ERR statement to check for all errors in the program, but the routine required to make the program that bullet-proof would have made this program unnecessarily long. It's probably a good subject for a future article.

Well, now that we've been through the major features of the program, we suggest you enter the program and start fooling around. As we mentioned last time, this program was never meant to be the ultimate in user friendliness or elegance of coding style. As you add records, find, and delete them, try to imagine ways you could improve the way the program works or asks for information.

Business Basic DATA files. While it's certainly true that most files contain data, Business Basic uses the term DATA files in a special way. You may remember that a TEXT file consists of strings of characters with the carriage return character as the terminator between strings.

If you print a numeric variable into a text file, it will automatically be converted to a string value, just as is done when printing numbers to the screen. This sounds pretty nice, but it can cause some real problems and inconvenience. For example, you know that an integer variable (which ends with the % character) occupies two bytes of storage in memory. However, representing the value in string format can take up to six bytes (-32000, for example). Add a RETURN character to delimit it and you have up to seven bytes to store an integer in a file.

Furthermore, it's impossible to tell beforehand just how much space a given set of numeric variables will take without checking each one beforehand. This can cause design problems for programmers. As you can imagine, these problems are even more acute for the long integer data type, which can be up to nineteen digits long but only requires 8 bytes of internal storage.

There's another problem with using text files that shows up only when you are using real numbers. Reals are represented in Business BAsic as 32-bit floating-point quantities requiring four bytes of internal storage. Normally, they are displayed with six digits of precision, and the format itself may vary greatly-especially if the magnitude of the number is very large or very small. In those circumstances, Basic will display the number in scientific notation. This means that the output format of a real can vary from something simple like 3.45 to something like -1.36723E-06.

Interestingly enough, it's not so much the space that this notation takes up that causes the trouble but that the printed representation of a real may not correspond exactly to the value stored in

memory. If a number's representation is not exact or requires more decimal places than can be displayed, the number is rounded before printing.

By contrast, this does not occur with integers. Since rounding occurs during printing, and text files are storage of the printed format, values of real numbers may be different in the text file than they were in memory. A short example will illustrate:

```
10    OPEN#1,"numberfile"
20    INPUT"type two numbers: ";x,y
30    z=x*y
40    PRINT#1,1;z
50    INPUT#1,1;z1
60    IF z=z1 THEN PRINT"they
      compare":GOTO 20
70    PRINT"they don't compare:
      ";z,z1
80    GOTO 20
```

Note that by printing the value to the file with the random access method in line 40, we are able to read it back directly in line 50. This lets us check to see if any value change has occurred as a result of the file operation. Try this with values like 500 and 4.25. Everything should go normally.

Now try a value like 3.033 and .031. Still okay. Now try 3.031 and .031. The result should print out appearing exactly the same, yet the comparison in line 60 fails. If you wish, you can insert a statement at line 75 to print out the difference. It will be small but obviously significant. For the real reason the product of this number pair fails to work, we commend you to your local math professor or textbook on numerical analysis.

Suffice it to say that certain real numbers cannot be stored exactly as binary numbers, nor can certain binary numbers be displayed exactly in a finite number of digits. As soon as these situations occur, the quantities stored in the text file will not exactly match what was calculated in memory. Play around with this program further. There's almost an infinite number of combinations that will also fail the test but appear to be equal.

You've just seen two reasons for the need, from time to time, to store numbers in a file in the exact form they have in memory. Can you think of a circumstance where you might want to do that with a string?

Among others, if you have a string that contains (or could contain) a return character, the text file input statement will terminate wherever the return occurs, thereby losing the rest of the characters in the string.

The key is that with DATA file format, you can store any numeric or string quantity without worrying about what might happen to the information. In addition, Business Basic adds an identifier to the front of each item of data you store in a data file, to indicate what kind of data it is. This is called the data Type, and allows you to intermix integers, reals, and strings in any order and still read them back correctly.

The information about the type of a particular data item is retrieved, astoundingly enough, by the TYPE function. This allows a simple program to read the contents of any data file, without having any information about it beforehand. Much more information about data files can be found in your manual, and I suggest you spend some time reviewing it.

In the meantime, let's look at what using data files will do to the parts program I listed at the beginning of the article.

First, we'll need to change the file type specification on the CREATE statement at line 120. The new line will look like this:

```
120    CREATE a$, DATA,64
```

Since the program was fairly modular, with the file access done in subroutines, the other changes are minimal as well. The idea is to store each item we used before (part number, description, and so on) as a separate data item in the file. Since the part number is always a four-digit number, we can use an integer to store that data. Description and location are string quantities, and quantity on hand will fit nicely into a real value, since it's a maximum of five digits (line 610 checks for that).

The first subroutine to change is the one at line 2000, which writes a record. The new statements look like this:

```
2010
```

```
      partnum%=VAl(partnum$):quantity=
      VAL(quantity$)
2020
      WRITE#1,rec;partnum%,desc$,locat
      ion$,quantity
```

There, that was easy.

Note that WRITE was substituted for PRINT since this is a data file, and instead of packing all the strings together as we did in the old line 2010, we simply converted the string values to the appropriate numeric ones.

If we had designed the program to use data files from the beginning, we probably would have handled that in the program's data entry section.

Next come the changes to the subroutine that reads a record back. Now things are very simple. We can replace all the lines between 1820 and 1885 with these:

```
1815     READ#1,rec:IF TYP(1) = 5
      THEN RETURN
1820
      READ#1,rec;partnum%,desc$,locati
      on$, quantity
1825     IF partnum%<0 THEN RETURN
1830     partnum$=STR$(partnum%)
1840     quantity$=STR$(quantity)
```

That's it! Since all the items are stored separately, there's no need to go through the process of splitting them out of the string record.

We must confess, however, that we really wanted to discuss the INSTR function, and that previous technique seemed the most logical way to show its features. Oh, well, it's always more fun to find an easier way!

Two more things are of interest here. Note that we have checked in line 1815 for TYPE 5, which indicates end of file. This takes care of checking for empty records. In line 1825, we introduce a new concept.

Previously, when we wanted to delete a record, we simply printed a null string over the existing information. There are times when it's useful simply to flag that a record is deleted, not actually wipe the information out. This allows deleted information to be retrieved in the event of mistakes.

Periodically another routine can be used to go through the file and physically delete the flagged records. Here, and below in the actual delete routine, we make the part number negative to indicate that it's no longer an active record.

The "delete" routine now will look like this:

```
1905     partnum%=-partnum%
1907
      WRITE#1,rec;rec;partnum%,desc$,l
      ocation$,quantity
```

That will write the record out with a negative part number, which will flag it as logically deleted.

Well, that should just about do it. In addition to giving you several things to try out before next time, the above changes illustrate an important programming fact of life. (You've been waiting for this article to get juicy, right?) This fact is that the more modular your program design, the more painless it is to make inevitable changes. I know that isn't your favorite fact of life, but there's nothing worse than to stare at several thousand lines of Basic, knowing that it has to be completely rewritten.

Next time we'll cover some new but related topics that will require completely rewriting this month's program. (Just a joke!) Actually, we'll talk about different ways to store and retrieve records on disk that give more flexibility than the simple record number scheme used so far. That should complete the effort to make you a file expert. In addition, Business Basic has an incredible output formatting capability, and now that you have learned the techniques for storing data, it should be fun to go through some tips on how to make your printouts look like professional reports.

Until then, have fun practicing the facts of life-programming facts, of course.

-///-

## Beginning Business BASIC

Part 1

By Stan Guidero

This is the first in a series of articles covering Business BASIC from the ground up. It is for the nonprogrammer that I write these articles. The best way to follow these lessons is to have the Apple /// computer up and running with Business BASIC.

For those who don't have BB (Business BASIC) you may purchase it from your local dealer as it does not come with the Apple /// and must be booted from disk. When you get the BB package you will find lots of goodies inside the box.

The program disk has a lot of Utilities and example programs on it. You also get one blank diskette, Apple Business BASIC Reference Manual Volumes 1 and 2 and a Software License Agreement. The first thing you should do is to boot your SOS Utility disk and back up the only copy of Business BASIC you have.

If you're not sure how to do this, I will give you a short lesson. First Boot your SOS Utility by inserting the disk into the onboard drive then hold down the CONTROL key and press the RESET button just behind the top of the keyboard. After the utility program is loaded you should first set the date. Press "D" to enter the device handling command level, then "T" to set the date. Press ESCAPE to return to the menu. Now press "C" to enter the copy mode. Place the blank that came with your BB package into the onboard drive. If you have a second drive, you should now place the Apple Business BASIC disk in it, if not you will be required to swap disks back and forth in the onboard drive. For now I will assume you have two drives. Near the bottom of the screen you will see the cursor set just before the .D2 under the prompt "Copy the volume:". If you have two drives just press RETURN. If you have only one drive, you will have to type in ".D1" in place of the ".D2". Then press RETURN. Two drive owners again only press RETURN, single drive owners will again have to type in ".D1". Press RETURN. Press RETURN again when promted "With the new volume name:". You will be prompted to reinsert the Utility disk back into the inboard drive. This occurs because the blank diskette must be formatted. Just follow instructions of the utility program and the computer will do the work. After you back up your BB disk, place it in the inboard drive and reboot the disk with a CONTROL- RESET.

Now we're ready for our first lesson. There are two modes that Business BASIC can be used in. The Immediate and the Deferred mode. First we will try some things in the Immediate mode. I want you to type into the Apple /// the command NEW followed by RETURN. This will clear the memory of the computer so we can start with a clean sheet. Now to clear the screen and place the cursor in the upper left hand corner simply type in the word HOME then RETURN. Now type in the command PRINT followed by 23+32. Press RETURN. The number or answer will appear on the next line which should be 55. We can use just about any mathematical function such as subtraction, multiplication, division and some trig functions. You may use "+" for addition, "-" for subtraction, "/" for division and "*" for multiplication. You may use multiple statements on a single line.

Try this: PRINT 23+32*2-12/2 Press RETURN.

The total will be 81. However, this may not be what you wanted to do. To solve the problem you may use parenthesis around the functions you want done first like this:

PRINT (23+32)*2-(12/2) Press RETURN.

Now the total is 104. Parenthesis are handled left to right first, then the rest of the statement. Power can be used using the caret "©". As in 23©2 .

There is a lazy way to type the PRINT statement, and that's to use a "?" in place of the PRINT statement. Like ? 23+32. This can also be used in program statements in the deferred mode. We can also print characters to the screen by using the PRINT statement. Try this:

PRINT "HI THERE!"

Now we are going to try some things in the deferred mode. You will be first typing a number. you can use sequential numbers like 1,2,3,4 , but this will leave you no room to add statements between numbers so you should use numbers like 10,20,30,40.

You will notice as we type in commands that nothing will happen. Before typing our first program we should clear the memory with the command NEW. Now type in this program:

```
10 PRINT "THE ANSWER IS:"
20 PRINT 23+32
30 END
```

To make sure the program is in memory correctly type LIST and RETURN. Now to activate the commands simply type RUN and press RETURN. Your screen should look like this:

```
THE ANSWER IS:
55
```

There are a few tricks available to you through Business BASIC to help you format the screen. Normally each line or statement has a RETURN built into it. That's why the 55 appeared on the next line instead of following the statement. In order to allow the answer to appear at the end of the line we can place a semicolon at the end of our PRINT statement as follows.

```
10 PRINT "THE ANSWER IS:";
```

Now LIST.

As you can see we only had to retype the line to change it. Now RUN the program. It now should look like this:

```
THE ANSWER IS:55
```

We also have available to us a method of using the built in TAB setting inherent in the APPLE /// computer. Placing comas in print statements allows us use of them. Try this:

First type NEW.

```
10 PRINT "POS#1","2","3","4","5"
20 PRINT "HI";
30 PRINT "THERE"
40 PRINT "HI"
50 PRINT "THERE"
60 PRINT "BY","THERE"
```

LIST and RUN.

Your output should look like this:

```
POS#1           2       3       4       5
HITHERE
HI
THERE
BUY     THERE
```

I used several examples to show print output to the screen. The first showed TAB positions, the next how to concatenate (to add on to the first statement) and the final three lines for variations. If you take a look at line two you will notice that HI and THERE is bunched together. To correct this you will have to add a space just after HI and before the close quote. Change line 20 to read:

```
20 PRINT "HI ";
```

Another way to use the PRINT statement is with a variable name. A variable name is a location in memory, we provide the name. We can use the letter "A" to store a number. First type NEW to remove our previous program. Now type in this small program.

```
10 LET A = 23
20 LET B = 32
30 PRINT A + B

RUN
```

We will cover variables later. To continue. You can mix strings of characters and numbers together in a print statement.Try this.

```
30 PRINT "THE ANSWER IS "; A + B
```

We can get really fancy.Type:

```
30 PRINT "The answer to ";A;" + ";B;" is:
";A+B
```

PRINT is an output statement. To input information, we can use the statement INPUT. Change our program this way.

```
10 INPUT A
20 INPUT B
```

Our program should look like this when you type LIST.

```
10 INPUT A
20 INPUT B
30 PRINT "The answer to ",A," + ",B," is:
", A + B
```

When you run the program you will be prompted with a "?". The computer is waiting for an input number. Type a number and press RETURN. Type in a second number and press RETURN for the second

question mark. The answer is then displayed.

Our program is rather hard to understand. What the heck does "?" mean any way. We can improve our program by including a character string in our INPUT statement. Change lines 10 and 20 as follows.

```
10 INPUT "Type the first number to be
added: ";A
20 INPUT "Type the second number to be
added: ";B
```

Now our program is readable.

A variable may be used to store a string but we must identify the variable name with a type. Incidentally, you can use up to 63 characters in a variable name. Variable names must start with a letter. Here are some examples of different names.

```
A  or ANSWER  : Real numbers. ( Floating
                 point i.e. 12.435 )

N$ or NAME$   : String

A% or ANSWER% : Integer numbers. ( Whole
                 numbers )
                Numbers between -32768 and
                 32767

A& or ANSWER& : Long integer number (
                 Whole number )

                Numbers from
          -9223372036854775808
                    to
          9223372036854775807
```

This ends our first session. In future articles we will cover all of the commands available to you in Business BASIC. Some commands are very powerful. Next issue will explain PRINT statements and variables in greater detail. There after we will learn the finer things of programming. So for now happy programming.

## Apple Writer: A BASIC Editor

by Paul Wilson

Have you ever been frustrated when composing a Business BASIC program because all you have as an editing tool is the escape-mode? Well hearken your ears to this good news: If you have Apple Writer, or some other means of editing a text-file, you can use the editing features of the program to either edit existing BASIC programs, or create BASIC programs. The following instructions apply to Apple Writer, but the principles can be applied to basically any editor which outputs a text-file.

Before we begin, a brief note on file types for the uninitiated is in order. Apple /// SOS has a number of different file types. What this means is that information is stored on disk in several different formats. For example, plain old text, such as an Apple Writer file, is stored on disk character for character, whereas BASIC programs are stored as a special compressed code which appears unreadable (sometime try using the UTILITIES SYSTEM to 'copy' a BASIC program from disk to .console). Apple Writer files are known as ASCII or TEXT files, depending on whether you're in Pascal, Basic, or something else. This information is displayed as part of the directory, or catalog, information on the diskette. For this exercise it is important to be able to tell the difference between file types, and so included below is an equivalence chart for file types in different systems. A disk containing only TEXT (ASCII) files and BASIC programs shows the following when "cataloged" with:

| | File Type |
|---|---|
| Apple Writer | text |
| | BASprg |
| Pascal filer | Asciifile |
| | Basicprog |
| System Utilities | Asciifile |
| | Basicprog |
| BASIC | text |
| | BASIC |

The trick to what I will be describing basically involves the changing of BASIC

program files to Apple Writer files and back again. The process is not complicated, but if you feel that you can handle a bit more complexity you can make your programming much easier. For example you can, using Apple Writer, write programs completely disregarding line numbers, then insert line numbers after, using Apple Writer's WPL capabilities.

As writing a program in Apple Writer is the easiest part, I will start with it. After booting Apple Writer type in your program as you would if you were in BASIC, but you can, if you want, disregard line numbers. That is, type all your lines without numbers at the beginning. After a while you may need a GOTO or GOSUB statement which references a line number which does not yet exist. Instead of a number, put in something easily recognizable like ***, and we will deal with this later. Mark sections of your program clearly with rem statements if you want to interrupt the regular numbering pattern and restart at numbers like 3000 or 10000. Another help is to limit individual rem statements to about 70 characters in length as anything over this will, when numbered and entered into BASIC, end up over 80 characters in length and unwanted screen wrap-around will result.

After a preliminary or final program is ready to test, replace all the <returns> with "xxx" with this command: "[F]<><>xxx<a" (see page 42 of the Apple Writer manual). You may have to insert "xxx" in the first line of the program. Save this program twice with names something like .d2/PROG.1 and .d2/PROG.A (the dual saving is to provide an unnumbered copy of your program for future use). Then clear the screen and enter and save the following WPL program (the ";" are comments which refer to the previous line):

```
  ny

  ;clear the screen

  1 .d2/prog.1

  ;load in .d2/PROG.1

  b

  ;go to the beginning of the file

  psX 1000

  ;set X to a starting number

loop f<xxx<(X)<

  ;find "xxx" and replace it with the
value X

  y.

  ;replace and the "." stops the next
replacement

  psx +10

  ;increment X by 10

  pgo loop

  ;go back to the beginning of the loop
```

Save this WPL program as .d2/WPL and type "[P] do .d2/wpl". The screen will go blank, drives will whirr, and your program will appear on the screen. You will notice that, at a rate of about 2-3 per second, the xxx's are being replaced by numbers which keep incrementing by 10. You can speed up this routine by inserting the following line:

```
  ppr [V][N][V]

  ; This is the same as pressing CONTROL 5
(5 on the numeric keypad)
```

Your screen will go blank while the WPL routine is executing.

You may stop the line numbering at any time by pressing the <escape> key. Then you can save the partially numbered program as /PROG.1 again, load the WPL program after clearing the screen, and reset the starting number. This can generate very neat numbering and, with practice, the whole process for a 100 line program with several starts and stops takes only about 5 minutes.

Now you must deal with all those GOTO *** and GOSUB *** statements which you made previously. As you now have actual line numbers to reference, use [F]/***/ to find the statements and the split screen function ([Y] see page 49 of the Apple Writer manual) to find what the line references really should be.

With the completely numbered program saved on disk, quit Apple Writer and boot BASIC. Once in BASIC, type the command EXEC

.d2/prog.1 (see Apple Business BASIC - Volume 1 page 28). Almost immediately a whole column of )'s will appear as the disk drive is accessed. At this point BASIC is loading in your file as a program. After all this stops you can LIST your program (to make sure it's all there) and SAVE it to disk. You will either have to save it to a different filename, such as /PROG.BASIC, or you will have to DELETE .d2/PROG.1 and then SAVE it under that name. If you don't a ?TYPE MISMATCH ERROR will occur.

This error occurs because the Apple Writer file is a TEXT file, and a BASIC program stored on disk is not text. It is trying to store compressed BASIC commands as text and it can't. Therefore you must either use a different name, or change the file type by deleting it and re-saving it.

The process of editing existing BASIC programs with Apple Writer is very similar. First you need to get a TEXT file version of your program. This is done in BASIC by loading in your program and typing the following BASIC commands:

```
)create ".d2/prog.listing", text

   ;"text" specifies the file type

)open #1, ".d2/prog.listing"

)output #1

)list

)close #1
```

Now you have an Apple Writer editable version of your program.

Boot Apple Writer and edit as outlined above, keeping in mind that you need to keep consistent line numbers, although the actual order of the lines need not be sequential. That is, if your program had lines 100-1000 and 5000-6000, if you added lines 3000-4000 it would not matter whether they were inserted at the beginning, middle or end of the file, Basic would rearrange them. Another alternative is to remove the line numbers through the use of the Apple Writer command "[F]<>=  <><a", and then renumber the program after the editing process. This command searches for a <return> ( > ), followed by a number of unknown characters (=) and ending up with three spaces, and replaces them all with a <return>. The three spaces are something BASIC inserts as formatting after each line number.

This process, although it may sound confusing at first, is quite workable. It is unfortunate that the inbuilt editing capabilities of Business BASIC are inadequate, but this is a step up from any inbuilt editing capabilities which I have ever seen on any system.

Happy programming!

- /// -

## FIG FACTORY

### Unclear Advertising

Dear Sirs:

A brief update on a Software program called FIG FACTORY sold by SUN SOFTWARE, P.O. Box 189, Tustin, CA 92680 for use with an APPLE /// (advertisement is attached).

To print hard copy from this product, the company has informed me that you must have either a Silentype Printer or a PKASO card for your Apple ///. The company is unwilling or unable to develop drivers for other configurations (interface cards and/or printers) and will not refund your purchase price if you order the program and subsequently discover this information. Please advise your readers as the advertising is not clear.

Sincerely,

Jack R. Scholl

Editor's Note: Advertisement referred to made 0 reference to printer requirements.

## DATA MANAGER ///

Evaluation by: Lloyd J. Brammer, CPA

Micro Lab
2310 Skokie Valley Rd.
Highland Park, Ill. 60035
(312) 433-7550

GENERAL

The following comments and observations come from having worked with Data Manager /// for only about a month.

- Program is now out in "preliminary release" form
- Written in Apple Business Basic
- Program is on two diskettes, and the user must build a third diskette to boot the system by copying the SOS files from a Business Basic or other diskette containing the drivers and OS.
- Will run on a 128K Apple with floppy drives, but Micro Lab recommends the use of a 256K machine and a hard disk to enjoy the program's full power (this writer is using 128K and floppies). Disk swapping is required on a 128K machine because the program is too large to fit on one disk.

DATA FILE CONSTRAINTS

- Can have 254 characters in each field
- Can have up to 200 fields per record
- Can have up to 32767 records per file on a hard disk
- Can have multiple files on a hard disk, depending on the storage capacity

VENDOR SUPPORT

Back up disks are not provided, but the price of the package at $500 includes Micro Lab's full one year warranty on the software. They will replace blown disks at no charge and will provide the user during the warranty period any updates or changes that are developed. This writer has called Micro Lab with questions and has written a couple of times. Response from Micro Lab has been marginal.

FEATURES

- Can design multiple data entry screens and save them to disk for any datafile. If a record has 50 fields, for example, and normal data update and maintenance is needed on only 3 fields, a separate screen can be designed to accommodate data entry on only the 3 fields most frequently needing change. The operator would not need to look through all 50 fields to find the fields needing change. Screens can be modified even after saving to disk.
- Fields using real numbers or integers can be protected so that alpha characters cannot be entered where a number is required.
- Output options include a "custom output format." There are over 40 math and manipulations commands available on the custom output format, allowing an almost unlimited combination of calculations between fields and on printout (but see further comments on this under the PROBLEMS section).
- There is a simple printer output option that allows quick printer formatting, without math functions.
- The "reconfigure" routine allows the transfer of data from one data file to another data file, to an Applewriter file, or to a Word Juggler file. All records on the database may be transferred, or it may be limited to only those records meeting a specific "search" criteria. Or, only certain fields of selected records may be transferred (such as a user might want for transferring name and address fields to an Applewriter file for mass mailings).
- The program has multiple level sorting capability.
- With the "replace" routine, you can replace the information in selected records or act mathematically with the record, all in one pass through the file. For example, if a field contained price information for 10,000 inventory items, a 10% price increase could be entered by using the "replace" routine to multiply the contents of the field by 1.1 and then replace the old

field contents with the new price.

- Seems that every conceivable search criteria has been included. Lower than, greater than, equal to, not equal to, character string included, character string not included, and more. The search criteria can be grouped so that multiple conditions must be met before a record is selected. The search also can be done by comparing fields, so that, for example, a record is selected only if one field meets the criteria established for comparison with another field. The search criteria can be saved to disk for repeated usage, rather than having to reconstruct the search specifications.

- At any time from the main menu, a user can check the "data file" info to see how much room is left on the disk; or to see what the field name, number, and type is for the record format used in the database.

- Fields can be designated as "read only" fields so that information contained in the fields cannot be altered when making modifications to the records (at least without going through a separate menu and routine to consciously make a change in the format of the screen).

- The user has total control of the printed output. Each location to be printed can be defined, so that the printout can be in any form - columns, rows, or on any format that may be required with preprinted forms, such as invoices, etc. Special titles and headings (text constants) can be included on the printout anywhere the user wishes.

PROBLEM AREAS

- The program has bugs. One bug which I discovered early pertained to the Reconfigure routines wherein a data file can be transferred to another data base or to Applewriter or Word Juggler files. A record may have 200 fields, but the Reconfigure option allowed only the first 9 fields of any record to be transferred. I called Micro Lab and they gave me a programming change on the phone to correct the problem so that all fields could be transferred. Other problems exist in the search routines which result in "bad subscript error" messages. These bugs have been pointed out to Micro Lab, but as yet they have not responded.

- A major deficiency in the custom output routine sharply restricts the usefulness of DATA MANAGER ///'s math and manipulations programming. In designing a check register with a view toward getting a printout of checks written with subtotals by account number, I found that the results were printed out in scientific notation. On talking with Micro Lab on the problem, they said that you can't get subtotals, totals, or perform any of the math routines in the custom output programs when using LONG INTEGERS. The math routines will only work with real numbers. That means that a user can only get 6 places of precision on printout. A user could not have individual checks, subtotals, or totals which would exceed $9,999.99. For this reason, DATA MANAGER /// will have very limited usefulness in a business application and will even be limited in home and hobby use. (There are many less expensive programs on the market that will handle search and sort routines.) Micro Lab said there is presently no intention of reprogramming the custom output routines to solve the problem. The manuals did not discuss this limitation or warn the user about it.

- The manual is very brief and in some places contradictory. For example, in one section we are told that only one database can be set up on a floppy diskette, and in another place we are told that we can set up more than one database on a floppy. The tutorial in the manual is not comprehensive enough and gave no guidance in how to use the subtotals and totals function and was extremely brief in describing the math and manipulations routines. Learning the system for me took many hours of experimentation (with much frustration). On two occasions, I lost about one third of the records entered in the database, which, according to Micro Lab, probably occurred because I did not properly exit the system when shutting down. The manuals contained NO COMMENT on the importance of exiting the system using the menu when shutting down. Re-keying the lost records took several hours.

- Printer format setup is very time consuming, because every print location is determined by the user. Also, the printing is very slow. It took approximately 2 1/4 hours to print out a floppy disk that was 80% full, without using any of the math routines on the output.

SUMMARY

DATA MANAGER /// has some powerful features and it has the potential to be a very useful package. However, the program bugs, such as the inability to use subtotals, totals, and math functions with long integers, and the relatively poor manuals prompts me to warn "buyer beware."

- /// -

**Problems with PFS ///: File**

There is also a problem with PFS: File for the Apple /// in version B:01. With very large files, like those you have to put on a Profile, file linkages can get damaged. This will be evidenced by not being able to find a record on a search on the first field, normally a very fast operation, but that record will be found on a sequential search. Note this is a linkage within the PFS file structure, not the SOS file structure.

This is corrected in version B:02 which is being released with SOS 1.3 in the near future, probably by the time you read this.

Note that you can check the version of your PFS: FILE or REPORT by entering a V for the selection number on the main menu and then pressing Enter (not Return).

Reprinted from HAAUG Apple Barrel.

- /// -

**Updates for Access /// and Quickfile**

Access /// has been updated to version 1.1. Quickfile is now at revision B. If you don't have these revisions see your Apple dealer to have your disks updated.

Reprinted from HAAUG Apple Barrel.

- /// -

**NEW CLOCK-ON-A-CHIP KIT FOR APPLE ///**

**Reviewed by: Fred Winkenpaw**

When the Apple /// was first introduced it was heralded as having many options built-in to the computer that would normally require the use of extra peripheral slots in the Apple II. These built-in features included a disk drive and controller for up to 4 drives, the capability of installing up to 256k RAM on the main board (with theoretical expansion to 512k), an RS-232C asynchronous serial communications port, two analog I/O ports (for gamepaddles, joysticks, etc.) with port A doubling as a printer port for the Silentype, and an on-board real-time clock.

Well, as some of you may know, Apple had a bit of trouble when they first released the Apple ///; after some trials and tribulations the problems were all worked out except for the promised clock. Apple had continuing problems with their clock chip and finally stopped all production and advertising for the clock capabilities of the Apple ///.

Apple however, has continued to provide for the possibility of a clock in the Apple ///. SOS (the Apple /// operating system) has built-in provisions for accessing a clock, as well as Business Basic which uses time$ and date$ to access timekeeping functions of a clock. So, how can you take care of time-keeping functions in your Apple ///? The 6502 CPU has an ability to keep track of time, but it must be set every time you turn on the computer. Some people have adapted real-time clock cards designed for the Apple ][ (such as the Thunderclock) and retrofitted them for use with the Apple ///. This method creates the problem of needing special S.O.S. "drivers" (machine code instructions) to access the peripheral card since it was not designed for the Apple ///. The card also uses up one of the valuable 4 slots in the Apple /// and the cards are quite expensive costing several hundred dollars.
Apple provided an empty socket on the board of the /// for a clock chip, but so far Apple has not come out with a chip of their own.

However, an innovative company in San Francisco called APEX has introduced an inexpensive clock-chip kit for the Apple /// that plugs into the main board, has battery back-up, and is totally software compatible with S.O.S., Business Basic, Pascal, and other languages. All that is necessary is the simple installation of the chip and connection of the battery unit and your Apple /// will be able to tell time, and remember the date. The chip provides full clock and calendar functions, and works without any further modification. From then on, all your S.O.S. files will be dated and timed; when using the system utilities (and other programs that provide this feature) the time and date will be displayed in the upper right-hand corner of the screen. The kit comes with complete installation instructions with diagrams, and does not violate the warranty of the computer since it does not require any soldering on the main board. The installation with these detailed intructions is greatly simplified.

The product is called the "Clock /// Kit" and is available for a special price of $60.00 plus $2.00 shipping charges from:

APEX Information Systems, Inc.
PO Box 11109
San Francisco, CA 94101
(415) 885-1633

- /// -

**Making a Turnkey CP/M Program Disk**

When you first receive a CP/M product such as the Palantir Word Processor, the disk likely does not contain the CP/M system and will not boot up running the application program. Three steps are required to make the program disk bootable. First, you should duplicate the disk for safety's sake. With the CP/M system disk in Drive A: (the built-in one) type (don't type the "A>" CP/M prompt):

A>COPY B:=A:

When prompted, insert the program disk in Drive A:, insert the blank disk in Drive B:, and press return. This will cause the blank disk to be formatted and the contents of the program disk to be duplicated on Drive B:.

Next place the CP/M system disk in Drive A: and transfer CP/M to the new program disk by typing the following command:

A>COPY B:=A:/S

Next transfer the file TURNKEY.COM from the CP/M system disk to the program disk by typing:

A>PIP B:TURNKEY.COM=A:TURNKEY.COM

Finally, set up the disk to automatically boot the desired program by typing the following:

A>B:TURNKEY filename

Where filename is the name of the COM file containing the program to be run on bootup.

Reprinted from HAAUG Apple Barrel.

- /// -

**PFS to the RESCUE**

A Rescue program for PFS files is being released soon to dealers. This program will copy every block on a disk that appears to have valid data in it. It will run on an Apple /// but will work on PFS files for both the Apple ][ and the Apple ///.

Reprinted from HAAUG Apple Barrel.

- /// -

# open apple gazette

P O BOX 813 SAN FRANCISCO 94101