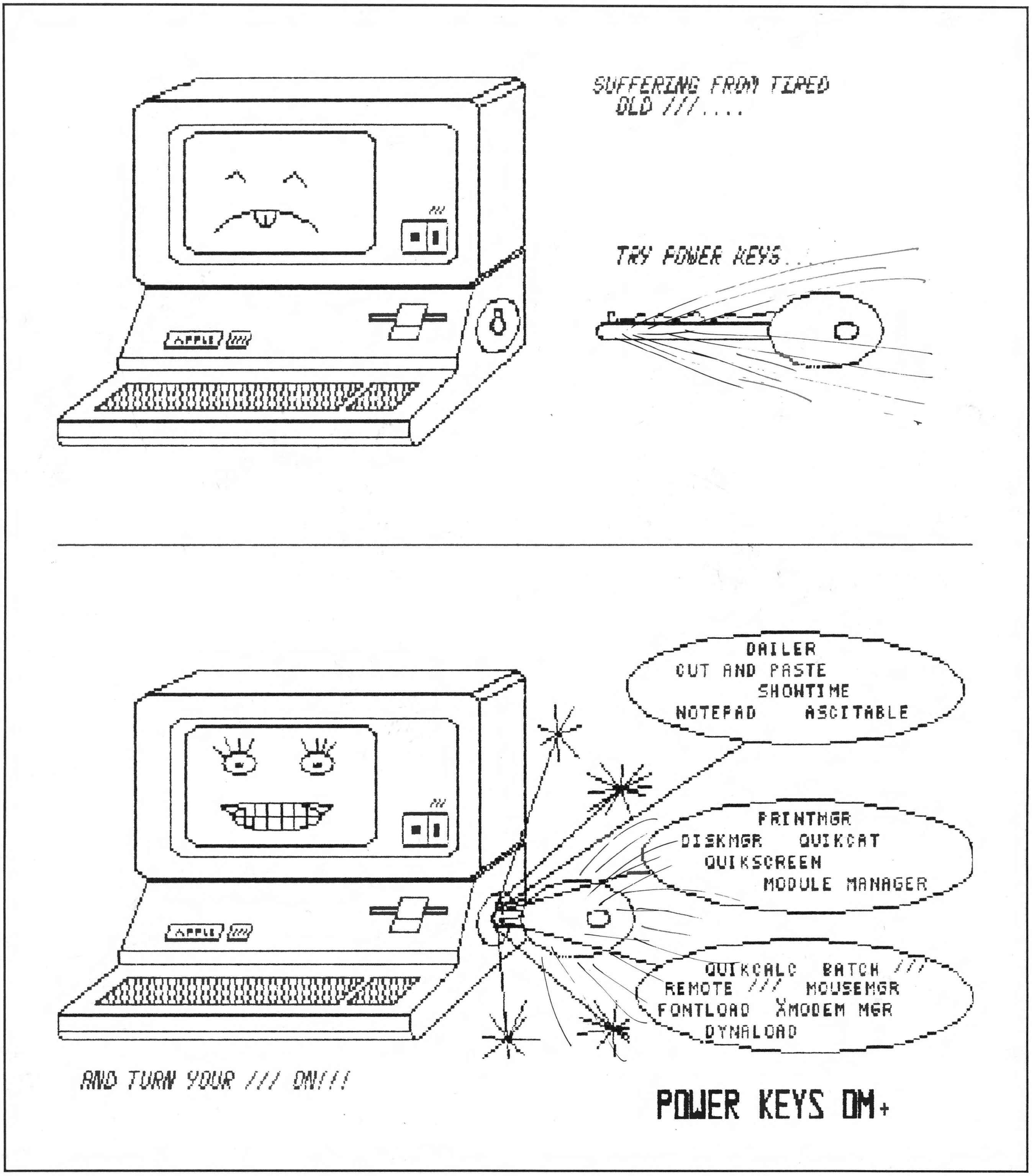
# THE /// MAGAZINE

MAY 1986



# THE / / / MAGAZINE is THE source for APPLE / / / owners !

## AN INDEPENDENT JOURNAL FOR:

/ / / INFORMATION
TECHNICAL DATA
SOFTWARE REVIEWS
BUSINESS TIPS
FUN

## OUR 2ND YEAR

Our magazine, successor to THE / / / NEWSLETTER and founded January 1,1985, is an international monthly review of Apple /// tips, articles, products, rumors and news. We have published each and every month ON TIME AND WITHOUT FAIL.

All / / / MAGAZINE subscribers begin their subscription as of the first of each year. You will receive copies of ALL of the back issues of the magazine for 1986 (copies of the 1985 issues are available). You won't miss a thing.

---

Return to: THE / / / MAGAZINE
3201 Murchison Way
Attn: New subscriptions
Carmichael, CA 95608 (916) 485-6525

I want to subscribe to THE / / / MAGAZINE. Enclosed is my check for $40.00 (2nd Class, USA only; Canada / Mexico / USA First Class, $50.00; outside USA, Canada and Mexico, $60.00) for 12 issues (one year).

Company Name :______________________________________________

Attention :______________________________________________

Address :______________________________________________

City :____________________State :__________Zip :__________

Telephone # :____________________

I am subscribing as a: Individual □ Business □ Vendor □

Signed: ______________________________

# THE /// MAGAZINE

May 1986
Volume 2, Number 4

"The machine that would not die"

# CONTENTS

NEXT MONTH:

Catalyst vs. Selector ///, Auto-Line Numbering in Basic, Omnis 3, and much more.

---

THE /// MAGAZINE is published monthly by MOORE ENTERPRISES, 3201 Murchison Way, Carmichael, CA 95608. (916) 485-6525. COMPUSERVE 74666,3660. Library of Congress ISSN #0885-436X. Subscriptions (12 issues, 1986) are $40.00 a year Second Class Postage in the U.S.A., $50.00 Canada, Mexico or U.S.A First Class Postage, $60.00 outside U.S.A., Canada and Mexico. All funds must be paid in U.S. dollars. All subscriptions include all back issues for a given year. Second Class postage paid at Carmichael, CA. POSTMASTER: Send change of addresses to: THE /// MAGAZINE, Attn: Subscription Corrections, 3201 Murchison Way, Carmichael, CA 95608-3165.

For a copy of our Author Guidelines, please mail your request to the above address. Return postage must accompany all manuscripts, drawings, and diskettes submitted if they are to be returned, and no responsibility can be assumed for unsolicited materials. All letters sent to THE /// MAGAZINE will be treated as unconditionally assigned for publication and are subject to THE /// MAGAZINE's right to edit and comment editorially. No guarantees are made for any of the contents herewithin.

# ON TOP OF THE APPLE TREE by Frank W. Moore

## MAGAZINE CONTEST

We come to you a bit late this month, since our original plan to have the MAY issue in the mail by May 1st. The reason for the delay was a business decision on my part to spend our resources and employee time, after the double March/April issue, filling Pair Software orders developing and preparing for a new mass mailing to all Apple /// owners and vendors. As you may have noted during the last month, THE /// MAGAZINE has been mentioned by a variety of computer publications, including NIBBLE MAGAZINE, ON-LINE TODAY (Compuserve) and AIM. It is our plan to increase our advertising budget even further.

Why? Because I have been very frustrated personally at the average size of THE /// MAGAZINE, which has generally been about 24 pages long. (this issue is 28). As with any business, the size of the magazine is dictated by its income. Each month we have to provide for present production expenses, author payments and reserve for future costs of creating THE /// MAGAZINE. To put it simply, a larger magazine requires more subscribers and more advertisers. We now have a salesperson (Luann Sanders) to work with developing advertisers for THE /// MAGAZINE, and our planned mass mailing will help lay the foundation for even greater expansion. Getting the 'word' out, and providing the /// with a large impartial forum for its users, is the key to its survivial.

To encourage further growth, this month we are starting a new subscriber campaign. If you are a PRESENT subscriber to THE /// MAGAZINE (you cannot reward yourself for signing up!) and encourage a new subscriber to sign up, you may choose one of our $10 or $15 Pair Software packages (see Page 27) AT NO CHARGE for each new subscriber. Just have the new subscriber include your request and subscription number (see the top of your mailing label) along with his or her subscription order, and we will mail your order ASAP. How's that for a deal!

## SLOT EXPANSION BOX

In ON TOP OF THE APPLE TREE in the March/April issue of THE /// MAGAZINE we noted that ON THREE was considering the development of a slot expansion box for the ///. This box, which I imagine could also be used in the Apple ][ Computer, would give the /// user up to 8 high speed expansion available in his or her system instead of the limited four slots we have now, extra slots for TITAN ///+//e cards, printer cards, CP/M cards, Legend Memory cards, etc. Unfortunately, ON THREE has only been talking about the expansion box. They need some idea of the number they can sell BEFORE developing it.

As a solution, in the article we offered to set up a mass purchase of expansion boxes with THE /// MAGAZINE magazine acting as a broker for the purchase. We would develop the buyer list, collect the purchase funds and hold them in an interest bearing trust account while ON THREE completes the expansion box design. Once the device is complete, they will be paid. If not, the potential buyers will be refunded with interest. We estimated the cost of the box between $300 and $500, with $400 a likely compromise. AT THIS TIME WE ARE LOOKING FOR REQUESTS, NOT MONEY.

To date, we have had about 40 requests for the expansion box. I feel we should have at least 100 requests before ON THREE can rightfully consider investing the time and money in the creation of the box. If you are interested in participating in such a purchase, we need to hear of your INTEREST. There is no need to commit or promise money NOW. Once we have developed a list, we will take it to ON THREE and get specs and cost from them, and THEN ask for a money commitment.

I know this is a heck of a way to run a market, but servicing the /// community is one of the most unusual markets I've ever seen. We ///r's need to be a clannish bunch if we are going to see continued development for our machines. WRITE TO US OF YOUR INTEREST!

## 512K MEMORY BOARDS

Another group purchase I would like to propose is a mass purchase of ON THREE's excellant 512 Kilobyte (1/2 megabyte) memory upgrade memory boards. These boards solve a variety of problems for /// owners who have found that even 256K is too small a memory space for their needs. Getting system crashes under CATALYST with memory hogs like DRAW ON ///, LAZARUS or DISK MAKER? Lack enough room for POWER KEYS or the DESKTOP MANAGER. The 512K board is the answer!

Again, we are looking for names of interested parties, not money, at this time. ON THREE has offered reductions on their $399/exchange price for the board, and I'll bet we can get at least 15% off. WRITE US OF YOUR INTEREST!

# LETTERS TO THE EDITOR

{Ed.Note: The following letter is printed by request of the editor of ON THREE, unedited and without comment}

The February issue of THE /// MAGAZINE contained an advertisement by D.A. DataSystems that was highly uncomplimentry to ON THREE and insulting to the intelligence of Apple /// users in general. The entire staff of ON THREE and our attorney feels that a number of statements in that advertisement were in error. The following is a statement of true factsw that were not brought out in the referenced ad.

1) We resent being called the 'Off & On Company'. We have worked harder than any other company to bring new and timely products to the market for the Apple ///.

2) The Desktop Manager by ON THREE does not, as stated in the advertisement require '40K minimum plus more for planned modules'. It in fact used a maximum of 40K of memory and in some circumstances takes away only 32K from the available system memory.

3) The Desktop Manager does indeed have many modules available for it (the NotePad, Calculator, Appointment Calendar, Desktop Controller, cut and paste, mouse control, and built-in help screens are included). Optional Disk Utilities, an Ascii Table, are among those avaiable now, not the 'many planned, call next week' as stated in the ad.

4) The Desktop Manager automatically installs itself on any 256K or greater Apple /// system and does not require running the System Utilities to add a driver. It is not, as stated in the ad, 'another Pre-Boot (sigh...).'

5) In regards to ON THREE's reputation, we take exception to the line that ON THREE will have a reputation 'real soon now'. With the dozen plus products that we have introduced over the last year and with new products that we have planned, our reputation speaks for itself.

6) The statement in the ad describing ON THREE's support as 'file a report with missing persons', is ridiculous and defamatory. ON THREE prides itself on having the best customer support in the industry. On each ON THREE product shipped a sticker is affixed bearing the statement: 'WE GUARANTEE SATISFACTION AND FULL PRODUCT SUPPORT. NEED HELP? PHONE (805) 644-3514'. Just last evening, two of our people spent almost an hour assisting a caller to get Access /// to run on his Space Coast System hard disk using Catalyst. You'll notice that none of the products mentioned are our own. If this isn't full Apple /// product support, what is?

7) The statement describing people who buy ON THREE products as suckers, or there is, 'apparently one born every minute', may not sit too well with the thousands of Apple /// users who rely on ON THREE for all of their needs.

8) The price of the 512K Memory Expansion is not, as stated in the ad, '..conveniently available from The Off & On Company (T.O.O.C.) for $499.' The price of the 512K Memory Expansion remains $399, the same as it has been since December of last year when a price reduction took effect.

We would appreciate it if anyone who has read the D.A. DataSystems ad would write to us at ON THREE with their comments. If there are any questions, please do contact us at:

ON THREE, Inc.
4478 Market Street, Suites 701-702
Ventura, CA 93003

Dear Frank:

I am wondering if you can help me understand the difference between the system of expansion slots in the /// in comparison to the slots in the Apple //e. Aside from the physical difference (the //e slots accomodating cards that are longer with a lower profile to fit in the //e housing).. could I plug a card designed for the //e into the /// and not run any risk of damaging the ///???

The reason for my interest is that I have just inherited a Rana 8086/2 MS-DOS 2.11 system that was designed to be operated through an Apple //e (or Apple II+) using a supplied 8086 controller card configured for the //e. As you may know, Rana Systems went bankrupt about a year ago, just before potential plans to configure their 8086/2 to an Apple ///... I suppose they might have called it the 8086/3.

The Rana 8086/2 has two disk drives... they both read/write either the Apple disk format or the IBM-PC disk format.

At any rate, if I could extend one of the Apple /// slots outside of the /// housing so the longer 8086 controller card could be accomodated, and be assured that I would not damage the ///. I would like to try to communicate to the Rana system.

I would be very much interested in your opinion as to the practicality of such an effort.

Sincerely

Gilbert E. Keas
Los Oso, California

*Dear Gilbert,*

*ON THREE's proposed slot expander would be a high speed expansion box that would accomodate ANY of the long Apple JC or the tall Apple /// cards (See On Top of the Apple Tree, this issue), so IF it is built your problem of placement would be solved.*

*A greater problem exists in making it work. I will assume that a SOS.DRIVER could be written that would operate the card, but the problem now would be the writing. Since Rana is a Chapter 7 bankrupt, your chances of obtaining the necessary technical data on the card to write the driver are very slim.*

*If you need to have an MS-DOS system (yeech!) you have several* *alternatives. The first is simply to buy a Apple //e or //e clone with ProDOS to run your Rana card, and then port needed files to and fro with that system. CMC Systems, a big Apple /// supporter, is selling the highly rated LASER 128 //e clone for (I believe) under $400.00 now. Call (714) 835-2462 for information.*

*If it is just Ascii (DIF) or MS-DOS data files that you need to transfer, another alternative is to buy D.A. DataSystem's PC-DISK. PC-DISK is a disk drive system that allows a /// to read and write 5 1/4" diskettes in SOS format on 400K/800K MS-DOS/PC-DOS disk drives. With a supplied program called PC-COPY, you can also read and write Ascii and data files in the MS-DOS format. And, you get the advantage of the extra drive for other uses (Backup ///, files, data disks, etc.) that will work with your standard double density 5 1/4" floppies!*

*One last note: The Engineering Department of Campbell, California, will soon release a FULL IBM compatable MS-DOS board for the //e. This board will not only turn a //e into an MS-DOS compatable machine, but will also have the ability to port DOS 3.3, ProDOS and MS-DOS ascii and data files between systems.*

*At this time the board, unfortunately, is a long //e board and will not fit in a standard ///. However, The Engineering Department is headed by Wendell Sander, who you may remember as the creator of the Apple ///. After talking with several of their engineers (former ///r's all!) I found that there was some sentiment to make the board /// compatable for 'old times sake'. I suggested it that it would be good business, too, since the vast majority of /// users (more than 85% of our readers) are business people.*

*I strongly suggest that you, and all interested ///r's, call or write The Engineering Department, Attn: Wendell Sander, 271 East Hacienda, Campbell, CA 95008, (408) 446-1444, and STATE YOUR OPINION. I have already told then I will buy the first one they put out for the ///.*

Dear Frank,

Three Easy Pieces has a heart stopping bug. I discovered it while working on a lengthy word processing project. The project involved cutting and pasting many ASCII files (for use with Super Applewriter and Micro Terminal), all carefully assembled from various Apple /// newsletters.

The disk that I wanted to save my work to was nearly full. I had carelessly not yet saved my hours of work (shame on me!). When I tried to print my first ASCII file to disk, Three Easy Pieces gave me a "Can't finish writing to this disk" error. The prompt said to "Press the Space Bar to Continue". Escape claimed that it would return me to "REVIEW/ADD/CHANGE". Neither key did as it suggested it would. I got what looked like a software lockup! Nothing would get me out of the error message! I knew that the system wasn't hung because the flashing cursor was still doing its thing. The program simply didn't know where to go next. Three hours of work were about to go into RAM heaven, never to be seen again!

In desperation, I tried several other 'open-apple' keys. At first I got a lot of "beeps". But, when I pushed the 'open apple S' key (Save to disk), the program took off and started to save my file as a regular Easy Pieces file. I pushed escape to cancel the save and, lo and behold, the program then came back and displayed my file on the screen. All was normal again.

Sincerely

Jeff Fritz
Williamson, West Virginia

*Dear Jeff,*

*Thanks for the valuable tip. I'll bet it saves a lot of lost work. But you did state the most important practice to learn: FREQUENT SAVING OF YOUR DATA TO DISK IN /// EASY PIECES. Too bad the program does not have a timer to remind you to save every 10 minutes or so!*

Dear Frank,

I am trying to teach myself Assembly language. It seems I have either a weak teacher or a slow student. I have been reading the Right to Assemble series but I am a bit frustrated. Could you ask Mr. Street to include a paragraph or two in each article about using those modules with PASCAL? It might also be helpful to say something about writing aprogram in Assembly that stands alone, i.e., that can be run directly from boot and does not need to be called from either BASIC or PASCAL.

Chris Acreman
Austin, Texas

# ON LEARNING PASCAL by Harry Hanson

For a old BASIC programmer, Pascal can be out and out intimidating. Three or four disks with a bewildering array of files. At least 4 Manuals and 3 Errata booklets. Program in Pascal? It seems hard enough to figure out how to get the system going.

That is what this article is all about - getting started and maintaining control. The actual programming will come. I will not be repeating the Pascal Manuals so do a little reading first. You will need to have read the INTRODUCTION, FILER AND EDITOR volume in particular. A text on the Pascal language is also useful. When I started I used PASCAL PRIMER by Fox and Waite, H.W. Sams & Co., Indianapolis. Also Jim Merritt had a monthly series in SOFTALK beginning in Feb., 1981

First of all, I urge you to get your Pascal upgraded to Version 1.2 with the 2.0 Compiler. This Compiler makes life so much easier for the programmer, and it really needs the 1.2 run system. As you get to writing larger programs, "Super" Applewriter or any text handler with the capability of handling large (larger than 32K) text files will be very handy. While the setup procedures are not dependent on the amount of RAM you have, once you start programming you may rapidly get yourself in trouble with demands on memory, particularly under Catalyst or Selector. Apple /// Pascal has the nasty habit of responding to excessive memory demands with a "Stack Overflow" (There is no "Out of Memory" error.), which is a fatal error that will require rebooting.

HARD DISK SYSTEMS - Now, about all those Pascal files. If you have a hard disk, you can avoid the whole problem by installing Pascal on it. Pascal 1.0 can be installed with the Jeppson Patch available on the Compuserve's Maug and from various Public Domain Libraries. Pascal 1.1 and 1.2 come with the PMOVE function which performs installation on a hard disk. All versions can be installed under Catalyst or Selector. If you have a Micro Sci A143, you can install Pascal 1.1 or 1.2 on it with PMOVE. Alternatively with an A143, you can load it up with most of the Pascal files without PMOVE, treating it like a couple of 140K drives.

TWO DRIVE SYSTEMS - If you only have 1 Disk /// or don't want to clutter your large mass memory device with all these files, that's fine too. [If you only have your one internal drive, forget the whole project. Pascal requires at least 1 external drive.] We start with old reliable - System Utilities. What we will be building first is a "bare bones", 2 drive system minus bells and whistles such as graphics, fancy sound, and higher math functions, but with provision to use any version of the Pascal files. They are all there, but they take up space and can be added later.

If possible, get a print out of the Directories of all your Pascal disks. The file locations vary somewhat among the 3 versions and I'll be identifying the various files by name. The print out will tell you which disk has which file. Be sure you have a good supply of formatted disks since, without a special program, you can't format a disk while in Pascal.

First copy the files SOS.KERNEL and SOS.INTERP from Pascal to a new disk. Then copy your standard SOS.DRIVER to the disk. You could also use the SOS.DRIVER file(s) that come with Pascal but the PRINTER driver may or may not be compatible with your printer. A third choice is to build a new SOS.DRIVER. If you do, you will need .CONSOLE; .PRINTER or .RS232 or whatever your printer driver is and, if it is to be used, the driver for your large mass memory device. Call this disk "PAS.BOOT".

To another disk, copy SYSTEM.EDITOR, SYSTEM.SYNTAX and SYSTEM.LIBRARY. If your Pascal has two versions of SYSTEM.LIBRARY, use the one on the LOWER numbered Pascal disk. Call this "PAS.SYS" but label it "PAS.SYS - Editor".

Make a copy of PAS.SYS and then add SYSTEM.PASCAL, SYSTEM.MISCINFO and SYSTEM.FILER. This will have the same volume name as PAS.SYS, but label it "PAS.SYS - System".

If you are using the 2.0 Compiler, just make a copy of the disk it came on. This will be "P.COMPILER.2.0". Otherwise add SYSTEM.COMPILER to "PAS.SYS - Editor"

You now have all you need to start writing Pascal programs. If using 2.0, put the compiler disk in .D2, otherwise put in a blank formatted disk. If you have a 3 drive system, put a blank formatted disk in .D3 if you are using 2.0. Boot with the PAS.BOOT, when the message comes up at the bottom of the screen to insert a Pascal disk, insert "PAS.SYS - System" and press <Return>. The Pascal command line will come up. You now have the opportunity to call the Filer and set a prefix. If you have a blank disk available, set the prefix to that drive, e.g. .D3. If you are using a hard disk, it would be a good idea to M(ake a subdirectory for hold your work and then set the prefix to point to this subdirectory. For a 2 drive, 2.0 system, you will usually want to set it to .D1 where you will have the most room. If you do use the Filer, Q(uit to the Main Command Line before proceeding.

Replace "PAS.SYS - System" with "PAS.SYS - Editor". Yes, we are misleading the system. However, even with the largest versions of Compiler, etc., you will have 54 blocks (27K) available on .D2 and 174 blocks (87K) on .D1 for your own work. If you don't have 2.0 then you have a complete disk available. Also note that, in most [but not all] cases, it makes no difference to Pascal which device has a given file. Thus the Pascal responses and prompts will be the same whether you have your "bare bones" system or a hard disk system.

NOTE: Apple Pascal offers the "Workfile" as a beginner's option. I suggest the beginner, particularly in a 2 drive system, avoid it completely. You have no control over disk utilization and must look at the text file to find out what is in it. In addition you need SYSTEM.FILER on line to use it conveniently.

THE EDITOR - E(dit will take you into the Editor. When you are done you must W(rite to save a new Text file. The W(rite command is also the means of getting a program listing by specifying your printer (e.g. .PRINTER) as the destination. The Pascal Editor is just a specialized word processor with a strange quirk. When you W(rite or S(ave to file, it adds .TEXT to your file name unless you tell it not to. You can tell it not to by putting a period at the end. Thus .D2/PROG1 will be saved as ".D2/PROG1.TEXT" while .D2/PROG1. will be saved as ".D2/PROB1" (the period you put at the end is not part of the name). A file saved by the Editor with .TEXT is stored in a different way from one without it. Files with .TEXT are stored as Pascal Text, the form required for compiling. Those without the suffix are stored as simple text or ASCII files. The Editor can read and make either type and convert one to the other (p. 104 of INTRODUCTION). Applewriter can only handle ASCII files.

THE COMPILER - This is the most tedious part of Pascal programming. For an old BASIC programmer, it is the "new" step. First of all, relax. The Compiler is just another program, one to convert your text or source code into p-code which is needed to run your program. The worst that can happen is that the Compiler will suffer its own little "nervous breakdown" and generate a Stack Overflow, forcing you to reboot.

NOTE: Pascal being run under Catalyst can be sneaky. When an error occurs, Pascal will come up with an error message which includes instructions to press space to continue. When you press space, Pascal attempts to recover from the error and will, most of the time, appear to do so. However, when you exit back to Catalyst, you may get SYSTEM FAILURE. This is just another expression of Stack Overflow. Reboot and fix the error.

If you restricted your reading to INTRODUCTION, it is time to break out PROGRAMMER'S MANUAL Vol. 2. and read pp. 88 - 92. You will find many references to the Assembler and the Linker which are extraneous to the beginner. You do not need either to write good, useful Pascal programs. Pp. 93 - 111 deal with Compiler Options. These will probably become useful to you long before the Linker or Assembler but they aren't necessry to begin either.

Compiling takes time. One error ruins the compile though you can continue with a tentative compile. If you use Compiler 1.0, 1.1, or 1.2, you will need to write down the point of each error. This is where 2.0 triumphs. The 2.0 Compiler allows you to create a Error File in which the Compiler will list all the errors, what they were and where they are. Thus you can just print them out, go back to the TEXT file and make the corrections - the Needed Corrections.

"Needed Corrections"? ....... One Pascal compile error can cause another. Indeed one error can cause a cascade of other, actually spurious, errors. For example, if you declare "PROCEDURE DOUBLE(X : INTEGER;" rather than the correct form "PROCEDURE DOUBLE(X : INTEGER);" every following correct call of the Procedure will be flagged as an error. Hence, in beginning, I suggest you go very carefully in allowing compilation to continue beyond the first error until you get some familiarity with Pascal.

One practice which can reduce compile time significantly is to turn off the screen. While this can be done with the Compiler ($Q) option, it is easier to just hit Control-5 (Keypad 5 that is) when you have supplied the Compiler with its required information. If you get an error, the screen will come back on by itself just as it will at the end of a successful compilation. You will reduce the time by roughly 20%!

If you have read the Compiler Options pages you are aware that you can generate a List File. Generating such a file takes extra time (ca. +10%) however and the file is useful only in debugging an already compiled program. If you wait until you have a successfully compiled program and then recompile with the List Option you will probably be well ahead in the long run. List files are also very large. For my typical programs, which use multiple library units, I figure that the List file will be twice as long as the Text file which will be twice the size of the Code file. If you don't have a hard disk, List files can easily overflow your disk space and ruin the compilation. They can also easily grow beyond the capacity of the Pascal Editor and when you try to look at it, the last part will not read into the Editor. A way around this is to generate your List file as an ASCII rather than a TEXT file. This is done the same way as in the Editor, by putting a period at the end of the pathname, e.g. ".D2/MYPROG.LIST.". Then you can read out with AppleWriter in a series of partial loads. If you have "Super" AppleWriter you can write it in one piece.

RAM DISK - These are fine devices for speeding up some operations but their usefulness for reducing compile time is small when compared to having everything on a .Profile. The best result is a ca. 10% reduction in compile time and that requires that the Compiler, the .TEXT file and the .CODE file all reside on RAM disk. Putting the .TEXT and .CODE files only on RAM disk yields a ca. 6% reduction. The situation changes though if your standard is an all Disk /// system. The all RAM disk compile was 30% shorter than an all .D2 compile. Thus if you cannot afford a hard disk, a large RAM disk could save you a great deal of time.

# POWER KEYS 2.0 by Frank W. Moore

In the August 30, 1986 issue of THE /// NEWSLETTER and in the October issue of THE /// MAGAZINE, we reviewed some of the powers and uses of D.A. DataSystems' POWER KEYS (Version 1.0) 'keyboard macro programmmer with CUT and PASTE'. A SOS.DRIVER module of the same school as the MS-DOS world's SIDEKICK, POWER KEYS could greatly increase the productivity of the typical /// user. Now, D.A. DataSystems has released their POWER KEYS DM+ (Version 2.0) system, with new 'powers' that make Version 1.0 seem downright primitive. SIDEKICK should have all of these features!

As a short description, POWER KEYS is run as a 'background' program which is accessable from any /// program, language or application. The POWER KEYS base SOS.DRIVER module is a new .CONSOLE driver (regular and CATALYST versions available) that replaces the present .CONSOLE driver on your system boot diskette. Once installed, the new .CONSOLE driver allows a variety of new functions to be routed through it as needed and without effecting the 'foreground' program (VISICALC, /// EZ PIECES, APPLEWRITER, etc.). To enter the POWER KEYS program itself, you simply press the user defined 'power key' (either the closed apple or open apple key) and the TAB key.

The POWER KEYS DM+ base module comes with the keyboard macro facilities. The macro menu is accessed by entering the POWER KEYS program, then using command 'M'. With keyboard macro's you can designate a single keystroke to type up to 128 keyboard characters simply by pressing a single key (pre-defined in the macro menu with the POWER KEYS 'K' option) along with the 'power key'.

Type multiple keystrokes. So what, you say? Well, how often do you repeat a sequence of keystrokes in your various applications? For example, I have a macro set defined for my word processor that includes 'CLOSED APPLE T' as the keystroke for printing the string, 'THE /// MAGAZINE'. I just touch the power keys and continue typing the rest of my text, knowing that my /// is going to quickly catch up. Over the course of a long job that can add up to a lot of keystrokes and time saved.

How about a VISICALC or /// EZ PIECES sheetsheet? Tired of mistyping cells? Set up a power key correctly and you'll never mistype that cell again! Are you a programmer? How often do you mistype program commands? With POWER KEYS you can predefine your PROCEDURE, GOSUB, WRITE and ABS( commands so they are never mistyped again! This all adds up to a lot of time saved not only in entry but in debugging (ed. Note: We will have an article next month on how to use POWER KEYS to do Auto-line numbering for you in BUSINESS BASIC)!

Work on a modem alot? I store logon codes and passwords in a macro file. Change to a new password, something you should do often? No problem with remembering passwords now- your /// does the remembering for you!

Lastly, do you have to type up form designs on your word processor? Hate typing those '!', '_' and TABs? With POWER KEYS you can predefine a cell like this with just a couple of keystrokes. 30 seconds work in 3 seconds!:

```
 ______________________________________________________
!             !                        !               !
!_____________!________________________!_______________!
```

Best of all, you can save and re-load macro menus as disk files with the S and L commands. This means you can define a macro menu for every application you own!

The macro functions were a feature of the original POWER KEYS, along with the 'CUT and PASTE' module that allowed a user to send a copy (a screen 'dump') of his or her CONSOLE screen to any printer or disk file. But, owners of the program wanted even more features available via POWER KEYS, so D.A. DataSystems came up with an even better scheme, and added a 'hook' to the base .CONSOLE driver that will load as many background program modules as a user's SOS.DRIVER file can find space to put them.

When installing the new POWER KEYS DM+ .CONSOLE driver with the SYSTEM CONFIGURATION PROGRAM of SYSTEM UTILITIES (by the time you read this D.A. DataSystems should have the self-install program shipping with the drivers), the user defines a pathname to a directory where the POWER KEYS macro files and driver modules are stored. At boot time, the basic .CONSOLE module goes to the defined directory and loads any macro file defined as DEFAULT.KBD and all of the POWER KEYS driver files in the MODULES sub-directory.

As you buy new modules for the POWER KEYS system, all you have to do is add the module to the MODULES sub-directory. That makes things really simple! All modules are sold separately, but are very reasonably priced.

The base module itself costs $49.95. Other modules presently available are:

QUICKCAT - Sick of applications where that lack a disk CATALOG feature. Tired of having to boot SYSTEM UTILITIES everytime you need to find a particular file? The QUICKCAT module (command GC) will catalog any directory for you by simply entering a pathname. If you just press 'RETURN' at the request QUICKCAT will catalog the POWER KEYS macro and note pad file directory. This module is a must. Cost: $19.95

DISKMGR - Tired too of booting SYSTEM UTILITIES to copy files and format diskettes? A companion to QUICKCAT, DISKMGR allows you to format a 140K floppy diskette from within ANY /// program WITHOUT the need of the ///'s format drivers. DISKMGR can also copy any disk file from one system disk to another. My only complaint is that you cannot format diskettes in any drive but the internal (.D1) drive. Wish I could use it on the occasional Micro-Sci A143 diskette. Still, this is one module I highly recommand. Cost: $19.95.

SHOWTIME - Displays your computers current time and date at a keystroke (command GT). A nice feature but I wish you didn't have to enter the POWER KEYS program to use it. Maybe something like a temporary display in the upper right hand corner upon pressing CONTROL and the POWER key. Cost: $19.95.

NOTEPAD - Have a $3000.00 computer system and still using a pencil and paper to take notes on scratch paper? No more. NOTEPAD (command GN) gives you a on line mini-word processor/scratch pad to quickly type up, save and print your important notes. Can also be used for calandar purposes. I find this module a must for myself. Cost: $29.95.

ASCITBL - Displays the ASCII table and currect system FONT loaded in your system, including special control characters. Very useful for those occasions you need to remember 'some blamed ASCII character/decimal number/hex number' for programming or printer set-up. This module is a must. Cost: $19.95.

PRINTMGR - Allows you to send complex setup codes directly to your printer or printer driver. A BIG help for those programs that cannot send printer code set-up codes. Cost: $19.95.

DIALER - This module, when used with a modem, can be used to dial telephone numbers for you using POWER KEY macros. For example, you can load in a particular macro/phone number file, press a couple of POWER KEYS and dial your number through your modem. If your modem has a speaker, you can hear when the other party answers. A very useful module for salespersons working from a data base. Cost: $29.95.

QUICKSCREEN - Successor to the 'CUT and PASTE' module of the old POWER KEYS, this module allows the printing of any screen to a printer or disk file of your choice. The former is a great way to grab a quick bit of data, and the latter is a great way to save data for use in other programs. This module is a must. Cost: $29.95.

Other modules are in the works, including a few NOT produced by D.A. DataSystems itself. In fact, if you would like to write modules for the D.A. DataSystems interface, D.A. DataSystem will be glad to help you! A few of the present modules under development are:

REMOTE /// - A module I am waiting for! REMOTE /// will allow your /// to be operated via telephone modem by a remote terminal just as though you are at the keyboard! The terminal itself itself doesn't even have to be a ///, just an ASCII terminal! One of the reasons that we have not yet set up THE /// MAGAZINE BBS is because we are going to use this module- you will actually log on to a CATALYST menu from your terminal, and be able to play with different programs languages, etc.! Let's see an IBM PC do this!

BATCH /// - Familiar with BUSINESS BASIC's EXEC command? BATCH /// will allow you to set up an ASCII text file that the module will execute as though the text of the file were keyboard commands, and it will work with ANY program. Got an all night job to do? Tell BATCH /// to do it, then go home.

MOUSEMGR - Allows you to set up your MOUSE up in any program as the input for the ARROW and RETURN or ESCAPE keys.

DRVRLOAD - Allows you to load an individual SOS.DRIVER in any program as needed. Great for developers and testing when you have 'forgot' to add a driver to your SOS.DRIVER file. DYNALOAD will do the same thing dynamically from CATALYST.

XMODMMGR - A fully implimented Xmodem protocol terminal on line at any time.

Only a couple of problems with POWER KEYS have been reported. In the pre-lease version of POWER KEYS sent to us we did have some trouble with with QUIKSCREEN. It would occasionally 'lock' when sending data to a printer. D.A. DataSystems reports that this problem has been corrected. The POWER SCREEN menus cannot be accessed from graphics screens like the drawing screens of DRAW ON ///. This is because graphics screens reside in a different part of /// memory than the text screen. To view the POWER KEYS screens you must return to a text mode.

Another problem exists for programs like DRAW ON /// and WORD JUGGLER that make use of both 'apple' keys, making the practical use of POWER KEYS impossible in those programs. It is too bad the POWER KEY designation is not left up to the user, but must be one of the 'apple' keys.

The last problem is one of memory limitations. While the present POWER KEYS system only takes up 18K of memory space, memory fragmentation and future drivers can cause crashing problems for /// systems running under CATALYST with memory hogs like DRAW ON ///. The only solutions are a separate boot diskette or a 512K memory upgrade. If you want the features of a system under POWER KEYS, I strongly suggest the 512K upgrade.

The POWER KEYS system can be ordered from D.A. DataSystems, 3792 Windover Drive, Hamburg, NY 14075. (716) 648-2492.

# BITS AND PIECES by Frank W. Moore

On the facing page is a picture of your favorite editor/publisher. No, I did not put it in for vanity's sake- after all, it makes me look a bit fatter than actually I am! Rather, it is a demonstration of a digital picture printed by a ///. Draw On /// author Mel Astrahan used a new digitizer board on a Apple ][, changed the file over to a SOS foto format, and finished it up with Draw On ///. The digitizer board works with any kind of video input- T.V. output, video camera or VCR- and converts the input into a graphics file. It is a great way to take 'pictures' for printing on documents or for archive storage on diskettes. Mel is considering writing a /// SOS.DRIVER for the digitizer board, but wants to know what kind of a market for such a board exists before he begins. Interested parties should write to THE /// MAGAZINE and we will forward the requests to Mel. I estimate the cost of an installed board with driver to be about $150-$200.

* * *

A new /// SIG/Club has reported in. The Calgary Apple Corps (Apple /// SIG) is looking for contact with other clubs. Write for more information to David Curwen at P.O. Box 6684 STN D, Calgary, Alberta, CANADA T2P 2E6. (403) 288-4544.

* * *

If you are a user of the FANASTIC OMNIS 3 data base (see our upcoming article in June), but having trouble setting your system up, you might consider the services of THE OMNIS MAN. Michael Reilly has developed a variety of super OMNIS 3 turnkey systems capable of running a transmission shop or an entire insurance company, invoicing to payables. Best of all, you can modify and update your system to fit your changing needs. For more information, contact Michael Reilly at 38 A Pearl Street, San Francisco, CA 94103 (415) 861-4018. His rates are VERY reasonable, probably more cost effective than doing it yourself!

* * *

My gosh, talk about bad timing. A new tax program for the Apple /// just came to our attention on March 15th! Office Data Systems' SMARTTAX will calculate and print your Federal and State Income Tax and forms. The program is perfect for the individual, family and small businessperson taxpayer. We will PERSONALLY vouch for its abilities- it did our taxes better than we did them! The program is available for the /// and MS-DOS computers, and will be updated each year. The cost is only $89.95. Contact Steve Wozenski, 351 Olema Road, #15, Fairfax, CA 94930. (415) 454-4357. If your tax income preparation needs are pretty standard, this is a program you should consider using. Great for projections, too!

* * *

A common complaint with IBM game programs is that the IBM random number generator is not truly random. The IBM program shown below should generate a complete random graphics display of screen dots, as does the /// program. Instead, the IBM will generate a set of bars on the screen. Something to throw in the face any egotistical 'Big Bluers' you meet. Challenge them to a duel!

```
APPLE ///                          IBM

10 HOME                            10 SCREEN 2
20 INVOKE"bgraf.inv"               20 X=RND*640
30 PERFORM grafixmode (%2,%1)      30 Y=RND*200
40 PERFORM grafixon                40 PSET (X,Y)
45 PERFORM fillport                50 GO TO 20
50 x%=RND(1)*559
60 y%=RND(1)*191
70 PERFORM dotat(%x%,%y%)
80 GOTO 50
```

* * *

We like to give some consideration to those makers of Apple ][ software who remember the ///. I just got a brochure from a company called Software Programming that makes a data base program called THE DATA REPORTER. And guess what? They state that it will run on the Apple /// under emulation. Nice to hear that on occasion! Impressive looking data base, too. For more information contact Software Programming at 4418 East Chapman, Suite 156, Orange, CA 92669. (714) 730-0666.

* * *

E-COM LIVES! The E-COM electronic mail division of the U.S. Postal service closed down last year. Old-time subscribers will remember that we sent our former weekly, THE /// NEWSLETTER, via E-COM for hard copy delivery. The E-COM format allowed a user to transmit via phone modem as many as 3,000 letters an hour (at 1200 baud) to the Postal Service for hard copy printing, stuffing and carrier delivery. E-COM was a great way to send out loads of mail quickly and cheaply. Heck, I could write and transmit THE /// NEWSLETTER to our thousands of subscribers in one evening! Well, several companies have bought up the old E-COM equipment, and plan to offer the same services formerly offered by E-COM. Two of the first to go on line are Business Corporation of America's ZIPNET service (Attn: Alice Gordon, 185 Alewife Brook Parkway, Cambridge, MA 02138-1185. (617) 491-4000) and NETWORD (Attn: Gary W. Gilbert, 6801 Kenilworth Avenue, #420, Riverdale, MD 20737-1331). Letters cost as little as 38 cents for one page, .43 for two pages, hand delivered to your customers.

If you need to send a great deal of mail FAST on a regular basis, this is a service you should consider! And, we have improved and re-released our program E-COM CALL as THE ELECTRONIC MAILMAN (see below) to service the needs of Apple /// owners interested using the E-COM format carriers, which is likely to become the standard format for hard copy electronic mail transmissions.

* * *

A tip from reader John Lomartire: "Anyone making up cables with solder-type pin connections runs the risk of heating the pins just a little too long. If this happens, the plastic in which the pins are embedded will soften, and alignment of the pins may be affected. To prevent this, always use a pair of connectors (male and female) fitted together during the soldering operation. I keep old connections on hand just for this purpose."

* * *

Several new products available from PAIR SOFTWARE (See page 27 for ordering details. Please note and use the NEW Order #'s):

By popular need and demand, we have begun to carry several lines of hardware peripheral of interest to Apple /// owners. The DATA SPEC RS232 and CENTRONICS switch boxes provide /// users an alternative to the dangerous practice of pluging and unplug RS232 and Centronics devices to and from your ///. Just switch to the required peripheral (second printer, modem, etc.) as needed. And, check out the prices for the quality switch you are getting! Hard to beat. Our prices range from $59.95 to $149.95.

We now carry the tough and inexpensive STAR printer line. We presently use our STAR SG-15 to do not only the office work but the type setting for THE /// MAGAZINE! Another STAR at my other office has had its fourth birthday of daily work without a visit to the repairperson (it has outlived two Imagewriters and an Olympia SW3000). These printers are workhorses, and even have a letter quality mode! Our prices range from $299.00 (9" SG-10) to $365.00 (15" SG-15).

And, for you users hungry to go color, we now offer the TEKNIKA MJ22 RGB/COMPOSITE COLOR MONITOR. The MJ22 is an excellent replacement for the Monitor ///, has great color and good 80 column text display. You can use the color composite mode on most ///'s for great color in the Apple ][ emulation mode. Our price for the MJ-22 is $349.00.

While we did not plan to get into the peripheral market, the demands of ///r's tired of searching for peripherals that meet THEIR needs made us decide to at least provide the items most in demand. Cables and shipping costs are extra.

In software, we have added FONT MAKER /// by John Cisar, a SUPER, easy to use FONT design program. FONTS are something that many ///r's unfortunately ignore- specially designed FONT sets can be great tools in program design and screen layout. #5900, $15.00. Also by John Cisar is LABEL MAKER ///, a very easy to use printer label creator and formatter. #5903, $15.00.

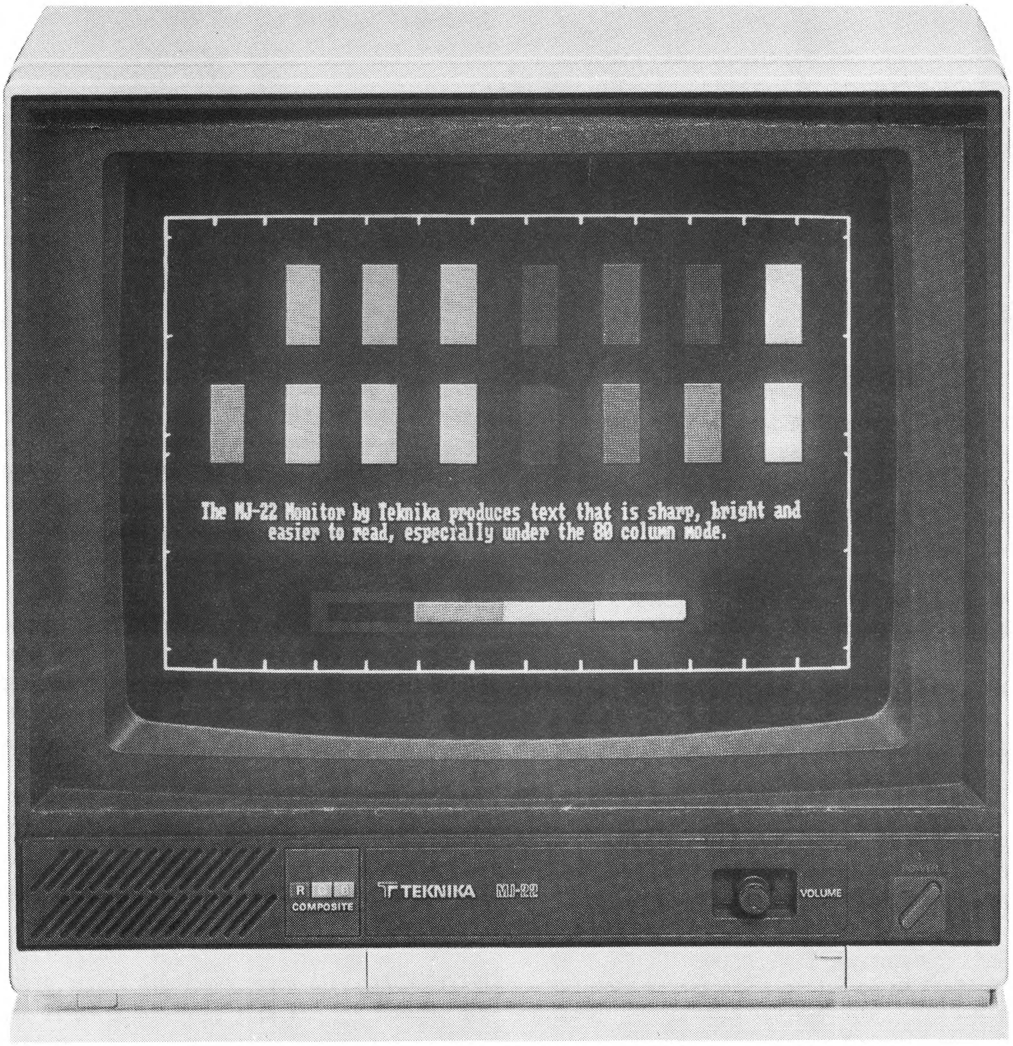
PASCAL MODULES #4 includes another set of PASCAL program modules to help developers quit re-inventing the wheel. Included on this diskette are the GENERAL UTILITY and SOS TIME modules. #5909, $15.00.

CHECKBOOK /// is a super checkbook ledger program for business or home accounting. It will maintain up to 10 different accounts, up to 100 codes for allocation of dollar amounts, and up to 3000 checkbook entries per account. It also has multiple report formats, merge file capabilites for interim period entry and reconciliation of data, and much more. We were so impressed with the program we hope to port it over to the MS-DOS and ATARI worlds for additional sales, but it runs best on the machine it was designed for- The ///! #6300, $49.95

THE ELECTRONIC MAILMAN lets you format your disk based word processor letter files and address lists into the E-COM format for transmission to the new E-COM format carriers. Send mass mail like the big boys when you NEED to get it out! #6302, 100.00.

# COMPARE!

## COMPARE TEKNIKA'S MJ-22 RGB/COMPOSITE MONITOR WITH ANY OTHER AND YOU'LL SEE WHY IT'S THE BEST INVESTMENT ANY COMPUTER OWNER CAN MAKE!



The MJ-22 Monitor by Teknika produces text that is sharp, bright and easier to read, especially under the 80 column mode.

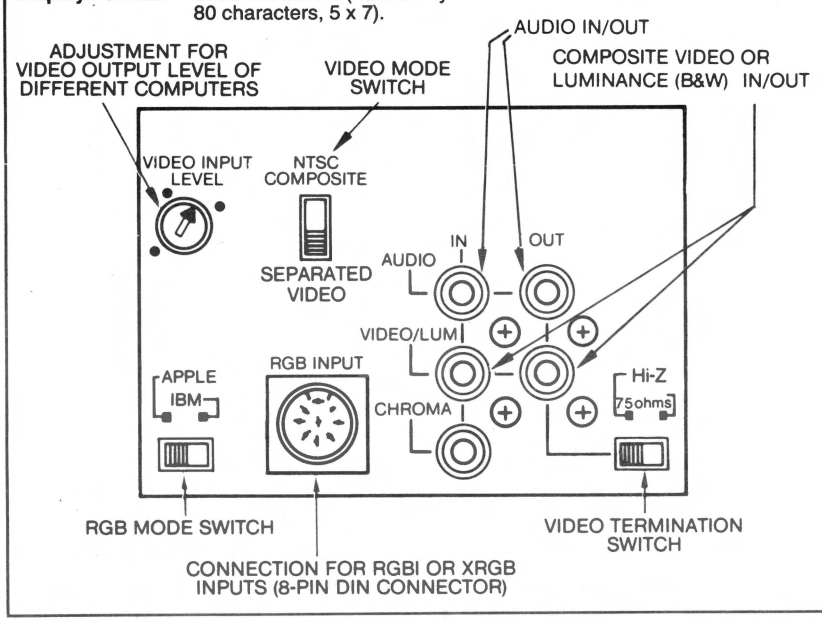
Teknika was the first to bring high quality component television to the market. And now we bring our engineering expertise to the computer field. The result? The astonishing new MJ-22 13″ RGB/composite color display monitor. A monitor with superior resolution, readability, color and flexibility. A monitor superior in every way to any other medium resolution monitor.

The MJ-22 is the best monitor investment for any computer user because it is the only monitor you will ever need. A monitor that gets the best out of the computer you own now...or the higher quality computer you might own in the future.

**COMPARE THE TEKNIKA MJ-22....**

## SPECIFICATIONS

| | |
|---|---|
| **CRT:** | 13″ 90-degree in-line, 0.5mm slot pitch, dark face, high resolution. |
| **Input Signals:** | RGB/NTSC composite/ separated video (luminance, chroma).<br>Horizontal, vertical sync: positive or negative combined (h/v). |
| **Input Levels:** | RGB, TTL level comp: 0.4-2.5 Volts peak-to-peak. |
| **Display Format:** | 2000 characters (25 lines by 80 characters, 5 x 7). |
| **Colors:** | 32-color palette of 16 colors in ROM for IBM and Apple. |
| **Resolution:** | RGB: 506 x 240 pixels; composite: 320 lines; separated video: 530 lines. |
| **Video Bandwidth:** | RGB: 18 MHz; composite: 3 MHz; separated video: 5 MHz. |
| **Audio:** | Input: 0.4 Volts RMS, 47K Ohms; Output: 0.8 Watts. |
| **Cabinet:** | Beige with 0-8 degree tilt stand. |
| **Dimensions:** | 14″ high x 14.5″ wide x 15.2″ deep. |
| **Weight:** | 26.4 pounds |
| **Front Controls:** | Horizontal Position, Vertical Hold, Contrast, Brightness, Color, Tint, RGB/Composite Push Switch. |
| **Rear Controls:** | Video Input Level, NTSC/ Separated Video Switch, Apple/IBM RGB Switch, Video termination Switch, Vertical Linearity, Vertical Size, Horizontal Hold. |



## ACCESSORIES

| | |
|---|---|
| **MA-19** | Cable set for the IBM PCjr. |
| **MA-20** | Cable set for Commodore, Atari, TI, Timex-Sinclair, etc. (8-Pin DIN) |
| **MA-21** | Cable for IBM PC (must have IBM Color/Graphics Monitor Adapter installed).<br>Will also work with IBM PCjr when an IBM Color Display Adapter Cable is installed. |
| **MA-23** | Cable for Apple II, Apple II+, Apple IIe (MA-RGB-7 board required) or Apple III. |
| **MA-RGB-7** | RGB Board for Apple II, II+, IIe. |
| **6510** | 140-channel Remote Control Television Tuner. Allows the MJ-22 to be used as a studio quality remote control TV. |

Specifications subject to change without notice.

# TEKNIKA, THE QUIET COMPANY WITH SOLID TECHNOLOGY

TEKNIKA is a subsidiary of THE GENERAL CORPORATION, Japan, a major manufacturer of a wide range of electronic products and electrical home appliances. THE GENERAL CORPORATION was established in 1936.

TEKNIKA is partly owned by C. Itoh & Co., Ltd., a $60 billion corporation founded in 1854. We are not one of those importer-distributors who may not have the full support of the manufacturer. TEKNIKA itself was founded in 1977.

The MJ-22 is our first display monitor and will be followed by two others: the MJ-10 and the MJ-40. The MJ-22 was developed as a result of extensive market research and the application of solid technology harnessed by the General Corporation over the years.

Visit a nearby computer dealer and take a look at the MJ-22. Compare it with any other color display monitor in its price range. You will be convinced that all our claims for the MJ-22 are well justified.

PAIR SOFTWARE

3201 MURCHISON WAY
CARMICHAEL, CA 95608
(916) 485-6525

See Page 27 for ordering details

# MACSTUFF FOR THE /// by Frank W. Moore

The MacIntosh computer has received a lot of press on how 'easy to use' its human interface is for new users. There is no real big secret to the interface- it is merely a graphics screen menu system that is able to post very large selections of menu options by movement of the Mac's MOUSE device and pressing of the MOUSE button. Still, for first time users the ease of using the interface does go a long way to overcoming computer phobias.

Several companies have adapted this MOUSE driven graphics interface menu system to other computers, including the ATARI and IBM (GEM interface). Apple has even taking to claiming that 'copying' of the MacIntosh type system, that is, graphics display systems that give another computer the 'look and feel' of the MacIntosh, is a violation of their copyright! (Personally, it appears to me that the MacIntosh system is itself a copy of the excellent, but now defunct, Xerox Star system.)

Now, Apple /// programmers can add a MacIntosh style system to their PASCAL programs with a new unit called MACSTUFF. MACSTUFF mimics most of the features of the MacIntosh system and gives the user an easy to use, direct interface with a /// program. It works with a MOUSE and driver or JOYSTICK ///.

Now, note that I said 'Apple /// programmmers'. MACSTUFF cannot be added to a compiled commercial PASCAL program. It must be integrated as a unit in your PASCAL SYSTEM.LIBRARY and called by the source text of your PASCAL program. Still, MACSTUFF can help you create a great user interface and, because of the procedures native to the unit, save you the trouble of 're-inventing the wheel' for a variety of program needs.

Once installed in your PASCAL library for the program, the procedures of the unit, well documented in the user manual, can be called as needed. Quite a few procedures calls are needed to create a typical screen, but the procedures themselves are very straightforward and easy to use, mostly requiring you only to pass label and position data to the procedures for their creative work.

Much of the grunt work for creating a menu screen has been eliminated by MACSTUFF. It will center screen labels, set foreground and background colors, create and remove windows and command box menus, etc., without any special coding on the programmer's part. I set up one modification of the SKELETON.CODE that come with the program to flip the graphics menu from one graphics mode to another (ala ATARI 520 ST) while keeping the menu screen labeling in the correct layout for the graphics mode used!

Also included are procedures to re-draw screens, show choices, highlight choices, catalog a disk, and even return file names (e.g., <file>.CODE) for execution!

This is not to say the MACSTUFF unit interface is without its limitations. First of all, as I noted above, a graphics menu interface is ideal for a the new or low level user, but experienced computer users may find the combination of MOUSE/MOUSE BUTTON/KEYBOARD a less efficient method than the keyboard alone. Since it runs in the ///'s graphics mode, the unit itself runs a bit slow, although not a lot slower than the standard MacIntosh interface.

While the unit works fairly quickly and flawlessly in the Black and White graphics modes, I found it less satisfactory in the color modes, requiring a lot more graphics screen restore calls and operations to keep the screen layout clean, i.e., correct color overwrites and screen changes. Of course, these restore operations take time, and make the unit operation even more slowly. This is unfortunate, as the color modes are really neat and to me offer far more potential for user interfaces.

MACSTUFF itself does not directly support MacIntosh/GEM screen identification icons (like hands, pointers, trashcans, etc.), although these could be created and displayed via a new system FONT. For example, any of the +128 ASCII FONT control characters could be made into icons or parts of a larger icon with a FONT generator program. If you are a programmer using the unit, creating a new system FONT for your program is not going to be a project.

All in all, MACSTUFF is a very interesting program unit that can provide a very easy to use program interface for your PASCAL programs, especially if you need to write programs for the inexperience or occasionally user. Licensing arrangements for use of the unit in your commercially released programs are available. Contact Apollo Software, Attn: Steven C. Robertson, P.O. Box 6434, Kent WA 98064-6434.

# THE RIGHT TO ASSEMBLE

by Stephen J. Street

I hope that the little dabbling we did with SOS calls last time has whetted your appetite because this month you're going to learn how to really put SOS to work! You should have a copy of the February issue of The /// Magazine handy as space limitations prevent repeating much of what has already been covered on SOS call parameter lists.

One complaint that many Basic programmers have is the difficulty in writing keyboard input routines that are "idiot proof". They don't want the programs that they write to "blow up" just because the user entered something unexpected from the keyboard. But writing keystroke edit routines that effectively trap and reject incorrect entries is tedious at best.

Yes, the public domain libraries are full of useful and very clever input routines that have been written entirely in Basic. But, rarely are they general purpose enough to use "as is" in an application you've written (not to mention the difficulty of inserting them into a program due to line number and variable name conflicts). And, the more general purpose and comprehensive they are, the slower they run.

Assembly language to the rescue! You've already seen in previous articles how fast your /// really can run when you use an invokeable module. Even a rather inefficient machine language program should have no difficulty keeping up with a fast typist. So that's what we're going to do: write an assembly language invokeable module to quickly do the main work of keystroke editing.

I admit that the sample module included in this article still isn't an "all purpose" input edit routine. But with the techniques you've been learning in this series and the powerful SOS calls we'll talk about this time, you'll be able to take the sample program and turn it into your own personal version of the "ultimate" input edit routine. Just the thing to give your Business Basic programs that "crisp" keyboard response that is characteristic of assembler programs.

As you look over the sample invokeable module, the first thing you'll notice is that it makes extensive use of assembler macros. The BBSUB, BNK2EXT, POP and PUSH macros are the same as the ones we've used in previous programs. So you can use the Pascal Editor's Copy File facility to copy them from a previous program. Or, if you've started to build a macro library (a real time saver), you can use the assembler .INCLUDE directive to insert them. The "Apple /// Program Preparation Tools" manual describes this technique in the assembler section. Remember that .INCLUDE is one of the few statements that you are not allowed to put a comment on.

Since it isn't good programming practice to ignore any errors that may occur with a SOS call (or, for that matter, any other part of a program - Basic or assembler), the SOS macro has been enhanced to include an optional third parameter. This parameter is the label to branch to if the SOS call returns a non-zero code in the A-register. Adding this parameter to the macro simplifies and reduces the amount of coding that has to be done in the program for each SOS call.

The new SOS macro has one other wrinkle: the "BEQ *+5" statement. This is called a "relative branch" since the branch-to location is relative to the branch instruction itself rather than to a specific location. It is used here to avoid any assembly errors caused by duplicate local ($xxx) labels. Its best to use this technique only within macros (or short sections of code that never change). Otherwise, later inserting or changing of an instruction can cause a bug that is very hard to find. Briefly:

```
    To branch forward, use "*+n"
      where n = 2 + number_of_bytes_skipped_over

    To branch backward, use "*-n"
      where n = number_of_bytes_skipped_over
```

Notice that it is BYTES skipped and not instructions skipped!

Caution: Many 6502 instructions can be 2 or 3 bytes long depending upon the addressing mode used (ie. a regular or a zero page operand); and a few (ASL, LSR, ROL and ROR) can even be 1, 2 or 3 bytes long depending upon usage! Be especially careful using relative branches in macros where, depending upon parameters, an instruction can be different lengths. This is one reason why the BNK2EXT macro still uses local labels rather than relative branches. If you're at all unsure about this, stick with local labels until you've had a chance to experiment some with relative branches.

The DCONTROL and WRITE macros are new this time so you should look them over carefully to understand how they work. As written, they are not necessarily general purpose enough to want to include them in your macro library. Rather, they were specifically designed to reduce the amount of typing (and the associated potential for errors) in this particular program. A clever assembler programmer (such as you've become) could easily make these more general purpose. Notice how both of these macros have a SOS macro inside of them ("nested macros" is the technical term); and, how the fourth parameter of the DCONTROL/WRITE macro becomes the third (%3) parameter of the SOS macro.

Like it or not, errors will occur when making SOS calls. Sometimes it may be a mistake in coding a SOS call parameter list. Other times your program is perfect but the user does something unexpected (such as removing a diskette that your program is trying to read). In any case, good programmers anticipate errors and attempt to handle them "gracefully" and not just "bomb". What's needed is an easy and convenient way to handle all manner of possible SOS call errors without having to write a lot of code to interpret and pass errors back to the caller. Once again, the Business Basic interpreter has a service routine to make life easier on us assembler programmers.

"SError" (SOS Error) is Business Basic service routine number fifteen ($F). It translates the most common SOS call errors into their corresponding Business Basic error number (and message) and then returns to the Basic program.

```
   Input:
     - A-Register is the error number returned by a SOS call.
```

This is very convenient since SOS calls always put any error codes in the A-register.

Output:

- Business Basic error message and number corresponding to the SOS call error code.

Or:

- The general Business Basic error 22 ("SOS CALL ERROR") if the code isn't one that can be translated into a more informative (?) Business Basic message.

Errors: None

Notes:

- This service routine always returns directly to the Basic program's ON ERR routine and does NOT come back to the invokeable module.

A big advantage of using this service routine (other than not having to write and debug a lot of complicated assembler code) is that it allows the Basic program's ON ERR routine to trap and handle any errors that occur in the invokeable module. And, it is a standardized approach to error handling which also simplifies the Basic program.

A second error handling service routine is "Error"; number 14 ($E). It is similar to SError but it is just for Business Basic error codes.

Input:

- X-Register (not the A-Register) is the Basic error code.

Output:

- Error message and number corresponding to the Business Basic error code.

Errors: None

Notes:

- This service routine always returns directly to the Basic program's ON ERR routine and does NOT come back to the invokeable module.
- Only works correctly for X-register values between 1 and 41 (decimal). The Basic program's ON ERR routine is triggered for any X-register value but the Business Basic error messages are incorrect if the X-register is greater than 41 (such as 255 which is "Break/CTL-C"). So you will probably want to have an ON ERR routine to interpret the error codes.

There are seven (count 'em, seven!) new SOS calls in this month's invokeable module and they are divided into two general categories:

File Oriented Calls (Open, Close, Read and Write)
Device Oriented Calls (Get Device Number, Device Status
and Device Control).

The File Oriented SOS calls are functionally very similar to their Business Basic counterparts of the same name. The purpose of the SOS Open call ($C8) is to create an "access path" to a pathname (the main input parameter for Open). It does this by: making an entry in the SOS File Control Block (FCB), allocating input/output buffers and assigning a file "Reference Number". This reference number is analogous to the Business Basic file number and it uniquely identifies the access path (not the file itself). The Open call must be done before any of the other file oriented SOS calls since all of the others need the Reference Number to know which access path to use. After doing the Open, your program must put the reference into the parameter lists for any Read, Write or Close SOS calls. For .CONSOLE (and all other character devices), the Pathname and the Device Name are the same. This is not true for block devices since there can be many pathnames on the same device (for example, many files can be on one disk).

The Close call ($CC) is the opposite of Open and undoes everything what Open did. Plus, if the access path was to a block device, it writes out any partial blocks ("flushes" the buffers) and updates the directory. The file reference number is the only input parameter for Close.

As you might expect, the SOS Read ($CA) and Write ($CB) calls transfer data to and from a user buffer and the SOS input/output buffers. Both of these calls, of course, use the reference number to know which access path to use. And, both have a pointer to the user data buffer (in two byte LLHH format since SOS takes care of any necessary bank switching). The Request Count tells SOS how many bytes to transfer. Read has one additional parameter, the Transfer Count, which tells how many bytes actually were transferred. The Transfer Count can be less than the Request Count if the end of the file or the termination (Newline) character is encountered during the Read. This invokeable module can ignore the Transfer Count value since the termination character has been set to "none" and character devices don't ever get end of file.

The second general category of SOS calls, the Device Oriented calls, don't have any direct Basic counterparts since Business Basic deals primarily with files and has few facilities for directly controlling devices. These calls do not use the Reference Number. Instead, they use a Device Number. Each device has its own unique Device Number that is assigned when your system is booted. These numbers do not change unless the SOS.DRIVER file is changed and the system is re-booted. And how does one find out this number for a specific device? Easy. Just ask SOS by using the Get Device Number ($84) call. This call takes a pointer to the Device Name as input and returns the Device Number.

Once you have the Device Number, you can do the Device Status and Device Control calls. Device Status ($82) returns information about the internal status of the device (another SOS call, Device Info, returns external status information). The Status call is valid for both character and block devices but the purpose of individual status codes may be different for different drivers. In our invokeable module, console status code 1 is used to save the current settings of all of the console options. This makes it easy to later use a Device Control call to put everything back the way it was before returning control to the Business Basic program.

Device Control ($83) is used to send control information to a particular device or driver. Like Device Status, the purpose of individual control codes may be different for different drivers.

However, Device Control is only valid for character devices. This invokeable module uses two control codes (2 and 11) to set console options, one code (17) to download a new bit pattern for the underline character and one code (1) to restore the settings of all the console options.

All of the console Status and Control codes are explained under "Advanced Techniques" in the console driver chapter of the Standard Device Drivers Manual. This chapter also describes the unique capability of the .CONSOLE driver to do many of the Device Status and Device Control functions by writing special control characters to the console. I haven't used this approach since I wanted to demonstrate the more general and common technique of using the SOS Device Control/Status calls.

To summarize, this diagram shows the difference between the File and Device oriented calls and shows the relationship between Pathnames, Device Names, File Reference Numbers and Device Numbers:

```
          +-----------+                +------------+
          ¦   Path    ¦                ¦   Device   ¦
          ¦   Name    ¦                ¦    Name    ¦
          +-----------+                +------------+
                V                            V
          +-----------+                +------------+
          ¦   Open    ¦                ¦ Get Device ¦
          +-----------+                ¦   Number   ¦
                ¦                      +------------+
                V                            V
          +-----------+                +------------+
          ¦ Reference ¦                ¦   Device   ¦
          ¦  Number   ¦                ¦   Number   ¦
          +-----------+                +------------+
                V                            V
      +---------+-----------+          +-----+------+
      ¦         ¦           ¦          V            V
      V         V           V      +--------+  +---------+
  +------+  +-------+  +-------+   ¦ Device ¦  ¦ Device  ¦
  ¦ Read ¦  ¦ Write ¦  ¦ Close ¦   ¦ Status ¦  ¦ Control ¦
  +------+  +-------+  +-------+   +--------+  +---------+
```

For more detailed information about these and all the other SOS calls, order a copy of the two volume "SOS Reference Manual" from Sun Data ($30; call 1-800-821-3221 to order).

The Business Basic program demonstrates how to use the invokeable module as part of a simple data entry application.

Like all good assembler programs, the invokeable module has lots of comments and the the macros should make it fairly easy to read and follow the program logic flow. One "tricky" part is the way that various errors are handled. Since the Basic program and the invokeable module are "sharing" the use of the console, it is very important that all errors be completely "cleaned up" and that no "loose ends" (ie. .CONSOLE left open, underline character changed, etc.) are left to cause problems for the Business Basic program. This diagram shows the different clean up steps that are taken depending upon where in the invokeable module the error occurs:

```
+----------------------+  If   +---------------------+
¦ After .CONSOLE OPENed ¦------>¦ Restore Status Table ¦
¦ and After Status     ¦ Error ¦ and Download normal ¦
¦ Table is preserved   ¦       ¦ Underline character ¦
+----------------------+       +---------------------+
                                      ¦             ¦
                                   And then         ¦
                                      V             ¦
+----------------------+  If   +---------------+    ¦
¦ After .CONSOLE OPENed ¦------>¦ Close .CONSOLE ¦    ¦
¦ but Before Status    ¦ Error +---------------+    ¦
¦ Table is preserved   ¦          ¦        ¦        ¦
+----------------------+         And      If       If
                                Then    Error    Error
                                  V        V        V
+----------------------+  If   +---------------------------+
¦ Before .CONSOLE OPEN ¦------>¦ "BBSUB SError" and exit to ¦
+----------------------+ Error ¦ Basic program ON ERR code ¦
                               +---------------------------+
```

Variations

By now you're probably already thinking about variations you'd like to try in order to create the world's greatest input edit module. Here are a few ideas I had. The first two should be easy to do but the rest will involve a bit of work.

- Use "tab" or another character to indicate end-of-input
- Have more than one end-of-input character.
- Check for a minimum or exact number of characters.
- Allow only valid numbers as input (ie. one plus/minus sign in first position, one decimal point, etc.)
- Various combinations of automatic upper/lower case to simplify and speed up data entry. I used the "Proper" function from last November's Right To Assemble for this.

Or, how about trying different "cursor" characters? Appendix G in the "Standard Device Drivers Manual" has some information on how to do this. But I found the instructions to be a little confusing, so here is an example of setting up a blinking "?" character:

```
                                                      Set "X"=1 and
                                                      Space=0. Then
        Draw the character      Invert the            flip the bits
        using graph paper        character            left to right
        +-+-+-+-+-+-+-+-+       +-+-+-+-+-+-+-+-+     ---- ---- ---
Byte 0  ¦ ¦ ¦X¦X¦X¦ ¦ ¦ ¦       ¦X¦X¦ ¦ ¦ ¦X¦X¦ ¦     1110 0011=$E3
Byte 1  ¦ ¦X¦ ¦ ¦X¦ ¦ ¦ ¦       ¦X¦ ¦X¦X¦X¦ ¦X¦ ¦     1101 1101=$DD
Byte 2  ¦ ¦ ¦ ¦ ¦X¦ ¦ ¦ ¦       ¦X¦X¦X¦X¦ ¦X¦X¦ ¦     1110 1111=$EF
Byte 3  ¦ ¦ ¦ ¦X¦ ¦ ¦ ¦ ¦  ==>  ¦X¦X¦X¦ ¦X¦X¦X¦ ¦ ==> 1111 0111=$F7
Byte 4  ¦ ¦ ¦ ¦X¦ ¦ ¦ ¦ ¦       ¦X¦X¦X¦ ¦X¦X¦X¦ ¦     1111 0111=$F7
Byte 5  ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦       ¦X¦X¦X¦X¦X¦X¦X¦ ¦     1111 1111=$FF
Byte 6  ¦ ¦ ¦ ¦X¦ ¦ ¦ ¦ ¦       ¦X¦X¦X¦ ¦X¦X¦X¦ ¦     1111 0111=$F7
Byte 7  ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦       ¦X¦X¦X¦X¦X¦X¦X¦ ¦     1111 1111=$FF
        +-+-+-+-+-+-+-+-+       +-+-+-+-+-+-+-+-+     ---- ---- ---
         0 1 2 3 4 5 6 7         0 1 2 3 4 5 6 7      7654 3210 Hex
            B i t s                 B i t s           ^  Bits
                                                      ¦
  A "1" here causes blinking when inverse ------------+
```

Here are some other cursor characters to try:

|  | Inverse non-<br>blinking "?" | Normal non-<br>blinking "?" | Normal<br>Basic | Blinking<br>Block |
|---|---|---|---|---|
| Byte 0 | $1C | $63 | $00 | $80 |
| Byte 1 | $22 | $5D | $00 | $80 |
| Byte 2 | $10 | $6F | $00 | $80 |
| Byte 3 | $08 | $77 | $00 | $80 |
| Byte 4 | $08 | $77 | $00 | $80 |
| Byte 5 | $00 | $7F | $00 | $80 |
| Byte 6 | $08 | $77 | $00 | $80 |
| Byte 7 | $00 | $7F | $00 | $80 |

Note that when you download a new character definition, all occurrences of that character on the screen change even if they were displayed prior to the change.

The technique of using Device Control to download full or partial character sets to the Apple /// video generator can be used to do some fairly fancy high speed animation. I believe that this is how the Apple /// "Running Horses" demonstration was done. If there is interest, this is a possible topic for a future article.

Until then, remember that SOS is your friend and you have The Right To Assemble.

---

Suggestions for future issues of The Right To Assemble should be sent to Stephen Street in care of The /// Magazine. Questions are also welcome and will be answered as space and time permit.

### ***   BASIC PROGRAM   ***

```
100   REM ********************************************************
105   REM *                                                      *
110   REM *    "TEST Read Console"                               *
115   REM *                                                      *
120   REM *    Business Basic Main Program to demonstrate using  *
125   REM *    the "PERFORM RdConsole" invokeable module to      *
130   REM *    control user keystrokes in a simple data entry    *
135   REM *    form                                              *
137   REM *                                                      *
140   REM *    Stephen J. Street    2/17/86                      *
145   REM *                                                      *
150   REM ********************************************************
155   REM
160   INVOKE"RdCons.INV","Proper.INV"
165   Upper.Case$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
170   Lower.Case$="abcdefghijklmnopqrstuvwxyz"
175   Digits$="0123456789"
180   REM
185   REM  Display the data entry form
190   REM
191   ON ERR GOTO 700
195   HOME
200   VPOS=1:HPOS=00:INVERSE:PRINT"Last Name:":NORMAL
205   VPOS=1:HPOS=33:INVERSE:PRINT"First Name:":NORMAL
210   VPOS=1:HPOS=61:INVERSE:PRINT"Initial:":NORMAL
215   VPOS=3:HPOS=00:INVERSE:PRINT"Address:":NORMAL
220   VPOS=3:HPOS=41:INVERSE:PRINT"City:":NORMAL
225   VPOS=3:HPOS=58:INVERSE:PRINT"State:":NORMAL
230   VPOS=3:HPOS=68:INVERSE:PRINT"ZIP:":NORMAL
235   VPOS=5:HPOS=00:INVERSE:PRINT"Telephone Number:":NORMAL
240   VPOS=5:HPOS=34:INVERSE:PRINT"Social Security Number:":NORMAL
245   VPOS=7:HPOS=00:INVERSE:PRINT"Birthdate (mm/dd/yy):":NORMAL
250   VPOS=7:HPOS=32:INVERSE:PRINT"Salary:":NORMAL
255   REM
260   REM  Get "Last Name" (maximum 20 characters; alphabetics
265   REM  only)
267   REM
270   VPOS=1:HPOS=12:Length%=20:Allow$=Upper.Case$+Lower.Case$
275   GOSUB 600:Last.name$=A$
280   REM
285   REM Get "First Name"  (maximum  15  characters;  alphabetics
287   REM only)
290   REM
295   VPOS=1:HPOS=45:Length%=15:Allow$=Upper.Case$+Lower.Case$
300   GOSUB 600:First.name$=A$
305   REM
310   REM Get "Initial" (1 character; upper case alphabetics only)
315   REM
320   VPOS=1:HPOS=70:Length%=01:Allow$=Upper.Case$
325   PERFORM RdConsole(@Initial$,%Length%,@Allow$)
326   IF LEN(Initial$)>0 THEN Initial$=Initial$ +".":VPOS=1
      :HPOS=71:PRINT".";
330   REM
335   REM Get "Address" (maximum 30 characters; alphabetics,
340   REM digits, period, comma and space allowed)
345   REM
350   VPOS=3:HPOS=10:Length%=30:Allow$=Upper.Case$+Lower.Case$
      +Digits$+"., "
355   GOSUB 600:Address$=A$
360   REM
365   REM  Get "City" (maximum 10 characters; alphabetics only)
370   REM
375   VPOS=3:HPOS=47:Length%=10:Allow$=Upper.Case$+Lower.Case$
380   GOSUB 600:City$=A$
385   REM
390   REM Get "State" (maximum 2 characters; upper case
392   REM alphabetics only)
395   REM
400   VPOS=3:HPOS=65:Length%=02:Allow$=Upper.Case$
404   PERFORM RdConsole(@State$,%Length%,@Allow$)
405   IF LEN(State$)<>2 THEN PRINT CHR$(07);:GOTO 404
410   REM
415   REM  Get "Zip" (maximum 5 characters; digits only)
420   REM
425   VPOS=3:HPOS=73:Length%=05:Allow$=Digits$
426   GOSUB 600:Zip$=A$
435   REM
440   REM Get "Telephone Number" (maximum 14 characters; digits,
445   REM space, parenthesis and dash allowed)
450   REM
455   VPOS=5:HPOS=19:Length%=14:Allow$=Digits$+"()- "
460   GOSUB 600:Phone$=A$
465   REM
470   REM Get "SS Number" (maximum 11 characters; digits and dash
472   REM only)
475   REM
480   VPOS=5:HPOS=58:Length%=11:Allow$=Digits$+"-"
485   GOSUB 600:Social.Secur$=A$
490   REM
495   REM Get "Birth Date" (maximum 8 characters; digits and slash
      only)
500   REM
```

```
505   VPOS=7:HPOS=23:Length%=08:Allow$=Digits$+"/"
510   GOSUB 600:Born$=A$
515   REM
520   REM  Get "Salary" (maximum 10 characters; digits, period,
525   REM                  comma and dollar sign allowed)
530   REM
535   VPOS=7:HPOS=40:Length%=10:Allow$=Digits$+".,$"
540   GOSUB 600:Salary$=A$
545   REM
550   VPOS=10:HPOS=1:PRINT
555   END
600   REM
601   REM  Input Subroutine
603   REM
605   VP= VPOS:HP= HPOS
610   PERFORM RdConsole(@A$,%Length%,@Allow$)
611   IF LEN(A$)=0 THEN PRINT CHR$(07);:GOTO 610
612   PERFORM Proper(@A$)
615   VPOS=VP:HPOS=HP:PRINT A$;
620   RETURN
700   REM
701   REM  Error Handling Routine
702   REM
710   VPOS=10:HPOS=1:PRINT"Error number "; ERR;" occurred."
720   END:REM of program
```

*** ASSEMBLER MODULE ***

```
        .TITLE  "Read Console Invokable Module"
        .NoPatchList
        .NoMacroList
;*******************************************************************
;*                                                                 *
;*      Read Console Invokable Module                              *
;*                                                                 *
;*      Do keyboard input and editing using SOS calls and          *
;*      Business Basic interpreter service routines.               *
;*                                                                 *
;*      To use from Basic:                                         *
;*                                                                 *
;*        INVOKE "RdCons.INV"                                      *
;*        PERFORM RdConsole(@Keys$,%Length%,@Allow$)               *
;*                                                                 *
;* Keys$ is the edited keystrokes returned to the Basic program    *
;* Length% is the maximum number of keystrokes (255 maximum)       *
;* Allow$  is a string of the keyboard characters allowed          *
;*                                                                 *
;*      S. J. Street   2/15/86                                     *
;*                                                                 *
;*******************************************************************
;
;     Macro Definitions
;
        .MACRO  BBSUB
        PHA                             ; Save A-Register temporarily
        LDA     #%1                     ; Get subroutine number
        STA     0E7                     ;  and set it
        PLA                             ; Reload A-Register contents
        JSR     0E4                     ; Go do Business Basic subroutine
        .ENDM
;
        .MACRO  POP
        PLA                             ; Pull low byte off of stack
        STA     %1                      ;  and save it
        PLA                             ; Pull high byte off of stack
        STA     %1+1                    ;  and save it
        .ENDM
;
        .MACRO  PUSH
        LDA     %1+1                    ; Get high byte and
        PHA                             ;  put it on the stack
        LDA     %1                      ; Get low byte and
        PHA                             ;  put it on the stack
        .ENDM
;
        .MACRO  SOS
        BRK                     ; Cause hardware interrupt
        .BYTE   %1              ; %1 is the SOS call number
        .IF     "%2"<>""
        .WORD   %2              ; %2 is the Req. Parameter List pointer
        .IF     "%3"<>""
        BEQ     *+5             ; Continue if no SOS call error
        JMP     %3              ; Branch if SOS call error
        .ENDC
        .ENDC
        .ENDM
;
        .MACRO  WRITE
;        Buffer,Length,Write_List,SOS_Err
        LDA     %1                      ; Set
        STA     %3+2                    ;  Write
        LDA     %1+1                    ;  Buffer
        STA     %3+3                    ;  Address
        LDA     %2                      ; Set Request
        STA     %3+4                    ;  Count
        SOS     0CB,%3,%4               ; Do SOS Write
        .ENDM
;
        .MACRO  DCONTROL
;        DC_Code,DC_List,DC_Address,SOS_Err
        LDA     #%1                     ; Set
        STA     %2+2                    ;  Control Code
        LDA     %3                      ; Set
        STA     %2+3                    ;  Control
        LDA     %3+1                    ;  List
        STA     %2+4                    ;  Address
        SOS     083,%2,%4               ; Do D_Control
        .ENDM
;
        .MACRO  Bnk2Ext
        LDA     0FFEF           ; Get current bank number
        AND     #0F             ; Bank number is just the low 4 bits
        BEQ     $010            ; Branch if bank zero
        TAX                     ; Otherwise, save bank number in X-reg.
        LDA     %1+1            ; Get HH part of Address
        CMP     #0A0            ; Is Address greater than $9FFF?
        BCS     $010            ;  Yes, its in High S-Bank
        SEC
        SBC     #20             ; Address = Address - $2000
        BMI     $010            ; Address is less than $2000
        BEQ     $005            ; Address is between $2000 and $20FF
        STA     %2+1            ; Address is between $3000 and $9FFF
        TXA                     ; Get the bank number and
        JMP     $015        ;  go turn on extended addressing bit
$005    LDA     #80         ; Convert Address so it is
        STA     %2+1        ;  between $8000 and $80FF
        DEX                 ; Bank# = bank# - 1
        TXA                 ; Get the bank number and
        JMP     $015        ;  go turn on extended addressing bit
$010    LDA     #0F       ; Special XByte for S-Bank and Bank zero
$015    ORA     #80       ; Turn on extended addressing bit
        .ENDM
;
;    Zero Page pseudo registers for Extended Indirect Addressing,
;    miscellaneous equates and other Zero page locations
;
;                          Business Basic Interpreter Subroutines
NotNow   .EQU   13.             ;  Get String Data
BError   .EQU   14.             ;  Translate Basic error and exit
SError   .EQU   15.             ;  Translate SOS error and exit
StrCp    .EQU   33.             ;  String Copy
InpCom   .EQU   34.             ;  String Assign
Allow    .EQU   0EA         ; Address of ALLOW$ string descriptor
AllowX   .EQU   1601+Allow ; X-Byte of ALLOW$ string descriptor
FacMO    .EQU   06C             ; General String pointer in Fac
FacMOXB  .EQU   1601+FacMO      ; X-byte for FacMO pointer
Strng1   .EQU   07B             ; Input pointer for StrCp
Strng1XB .EQU   1601+STRNG1     ; X-byte for Strng1 pointer
InpOut   .EQU   057             ; Output pointer for InpCom
InpOutXB .EQU   1601+INPOUT     ; X-byte for InpOut pointer
Index    .EQU   035             ; General Purpose Pointer
IndeXB   .EQU   1601+Index  ; X-byte for General Purpose Pointer
;
        .PROC   RdConsole,3 ; Three words passed to Read Console
;
        JMP     Start       ; Skip over parameter lists, etc.
;
;      Temporary storage
;
RtnAdr  .WORD   00          ; Return Address
Length  .BYTE   00          ; Value of LENGTH% parameter (LSB only)
AllowLn .BYTE   00          ; Length of ALLOW$ string data
KeysD   .WORD   00          ; Address of KEYS$ string descriptor
KeysX   .EQU    1601+0E8    ; X-Byte of KEYS$ string descriptor
Count   .BYTE   00          ; Input character count
InputA  .WORD   Input       ; Address of Input characters
Input   .BLOCK  255.        ; Input characters (255 maximum)
;
;      Parameter lists for various SOS calls
;
Open    .EQU    0C8         ; SOS call number for Open
OpenLst .BYTE   04          ; Open has four parameters
        .WORD   PathNam     ; Address of Pathname to Open
OpenRef .BYTE   00          ; Reference Number (filled in by SOS)
        .WORD   00          ; No optional
        .BYTE   00          ;  parameter list needed
PathNam .BYTE   8.          ; Length of Pathname for Open
        .ASCII  ".CONSOLE"  ; Pathname for Open
;
Close   .EQU    0CC         ; SOS call number for Close
ClosLst .BYTE   01          ; Close only has one parameter
ClosRef .BYTE   00          ; Reference Number (fill in after Open)
;
GetDNum .EQU    084         ; SOS call number for Get Device Number
DNumLst .BYTE   02          ; Get Device Number has two parameters
```

```
        .WORD   PathNam    ; Address of Device Name
DevNmbr .BYTE   00         ; Device Number (filled in by SOS)
;
DStatus .EQU    082        ; SOS call number for Device Status
DSList  .BYTE   03         ; Device Status has three parameters
DSDNmbr .BYTE   00         ; Device Number (filled in later)
DSCode  .BYTE   01       ; Device Status Code (Preserve Status)
STableA .WORD   StatTbl  ; Address of the Status Table
StatTbl .BYTE   90.      ; Status Table is 90 decimal bytes long
        .BLOCK  90.      ; Console Status Table
;
DCntl   .EQU    083      ; SOS call number for Device Control
DCList  .BYTE   03       ; Device Control has three parameters
DCDNmbr .BYTE   00       ; Device Number (filled in later)
DCCode  .BYTE   00       ; Device Control Code (filled in later)
CListA  .WORD   00       ; Address of the Control List (filled in
;                          later)
C2LstA  .WORD   C2List   ; Address of Control Code 2 List
C2List  .BYTE   00       ; Line Status = $00 (no termination
;                          character)
        .BYTE   00       ; Null Termination Character
;
C11LstA .WORD   C11List  ; Address of Control Code 11 List
C11List .BYTE   00       ; Echo Status = $00 (no echo, etc.)
;
NewUL   .WORD   Blink    ; Address of Control Code 17 List
Blink   .BYTE   01       ; Download one character
        .ASCII  "_"      ; Replace Underline with Blinking
;                            underline
        .BYTE   07F             ; Character row 0 (top row)
        .BYTE   07F             ; Character row 1
        .BYTE   07F             ; Character row 2
        .BYTE   07F             ; Character row 3
        .BYTE   07F             ; Character row 4
        .BYTE   07F             ; Character row 5
        .BYTE   07F             ; Character row 6
        .BYTE   0FF          ; Character row 7 (blinking bit on)
;
OldUL   .WORD   NoBlink  ; Address of Control Code 17 List
NoBlink .BYTE   01       ; Download one character
        .ASCII  "_"     ; Replace Blinking Underline with Underline
        .BYTE   00      ; Character row 0 (top row)
        .BYTE   00      ; Character row 1
        .BYTE   00      ; Character row 2
        .BYTE   00      ; Character row 3
        .BYTE   00      ; Character row 4
        .BYTE   00      ; Character row 5
        .BYTE   00      ; Character row 6
        .BYTE   07F     ; Character row 7 (blinking bit off)
;
SWrite  .EQU    0CB  ; SOS call number for Write
WrList  .BYTE   03   ; Write has three parameters
WrRef   .BYTE   00   ; Reference Number (filled in later)
WrBuffA .WORD   00   ; Write Data Buffer Address (filled in later)
WrLen   .WORD   00   ; Write Request Count (filled in later)
;
WCurA   .WORD   WCursor           ; Write Cursor Buffer Address
WCursor .BYTE   18.,"_",17.,8     ; Cursor Buffer: Inverse, UL,
;                                      Normal, BS
WCurL   .EQU    *-WCursor         ; Write Cursor Buffer Length
;
WBlkA   .WORD   WBlank            ; Write Blank Buffer Address
```

```
WBlank  .BYTE   " ",8,8," ",8   ; Blank Buffer: Space, BS, ,BS,
                                ; Space, BS
WBlkL   .EQU    *-WBlank        ; Write Blank Buffer Length
;
WBeepA  .WORD   WBeep           ; Write Beep Buffer Address
WBeep   .BYTE   07              ; Beep Buffer
WBeepL  .EQU    *-WBeep         ; Write Beep Buffer Length
;
Read    .EQU    0CA      ; SOS call number for Read
RdList  .BYTE   04       ; Read has four parameters
RdRef   .BYTE   00       ; Reference Number (filled in later)
RdBuffA .WORD   RdBuffr  ; Read Data Buffer Address
RdLen   .WORD   0001     ; Read Request Count = 1
RdXfr   .WORD   00       ; Read Transfer Count (filled in by SOS)
RdBuffr .BYTE   00       ; Read Buffer (only 1 byte needed)
;
;      Start of Invokeable Module code
;
;      Pull  Return Address and parameters off of the stack.
;
Start   POP     RtnAdr   ; Return Address is first thing on the
;                           stack
        POP     Allow    ; Get address of ALLOW$ string descriptor
        PLA              ; Pull MSB of LENGTH%
        STA     Length   ;  and save it
        PLA              ; Pull LSB of LENGTH% and ignore it
;                          (remember Bus. Basic integers are HHLL)
        POP     KeysD    ; Get address of KEYS$ string descriptor
;
;   Just return if LENGTH% is zero (ie. no keystrokes requested).
;
        LDA     Length   ; Get LENGTH% value
        BNE     LenOK    ; Continue if its not zero
        JMP     Return   ; Just Return if it is zero
;
;      Use Business Basic "NotNow" subroutine to get the address
;      of the ALLOW$ string data and its length.
;
LenOK   LDA     Allow           ; Input to NotNow is address
        STA     Index           ;  of string descriptor
        LDA     Allow+1
        STA     Index+1
        LDA     AllowX          ; Including the X-Byte
        STA     IndeXB
        BBSUB   NotNow        ; Get data address and string length
;
;    On return from "NotNow", pointer to data is in Index, Index+1
;      and IndeXB. The length of the string is in the A-Register.
;
;     Just return if ALLOW$ data length is zero (ie.  nothing
;     allowed).
;
        TAX                     ; Is length greater than zero?
        BNE     AllowOK         ;  Yes, continue
        JMP     Return          ;  No, just Return
AllowOK STA     AllowLn         ; Save length of ALLOW$ data
        LDA     Index           ; Replace the zero page
        STA     Allow           ;  ALLOW$ Descriptor Address
        LDA     Index+1         ;  with the ALLOW$
        STA     Allow+1         ;  Data Address
        LDA     IndeXB          ;  (Including the X-Byte)
        STA     AllowX
```

```
;
;     OPEN the console and then fill in the parameter
;      lists for CLOSE, READ and WRITE.
;
        SOS     Open,OpenLst,Error1    ; Open .CONSOLE
        LDA     OpenRef        ; Copy the Reference Number in to:
        STA     ClosRef        ; Close parameter list
        STA     WrRef          ; Write parameter list
        STA     RdRef          ; Read parameter list
;
;     Get the Device Number of the console and then fill in
;      the parameter lists for Device Status and Device Control.
;
        SOS     GetDNum,DNumLst,Error2 ; Get the Device Number of
                                       ; .CONSOLE
        LDA     DevNmbr          ; Copy the Device Number in to:
        STA     DSDNmbr          ; Device Status parameter list
        STA     DCDNmbr          ; Device Control parameter list
;
; Use the SOS Device Status call to save all of the current
; console settings so they can be restored later by a Device
; Control call.
;
        SOS      DStatus,DSList,Error2 ; Preserve the Console
                                       ; Status Table
;
; Use SOS Device Control calls to set the console parameters
; to what we want for this invokeable module.
;
        DCONTROL 02.,DCList,C2LstA,Error3  ; Set No Termination
                                           ;  Character
        DCONTROL 11.,DCList,C11LstA,Error3 ; Set No Echo, etc.
;
; Use SOS Device Control call to Download A Partial Character Set
;
        DCONTROL 17.,DCList,NewUL,Error3   ; Download Blinking
;                                            Underline
;
        LDA     #00
        STA     Count            ; Input character count = 0
;
; Main program loop:
; Display blinking underline cursor
; Read one character from the keyboard
; Check character for carriage return (end of input)
; If not end of input, see if character is backspace
; If not end of input or backspace, see if character is allowed
; Beep if it isn't allowed; save it if it is allowed (unless
; Length% characters have already been accepted)
;
WriteC  WRITE   WCurA,#WCurL,WrList,Error3 ; Display Blinking
;                                            Underline
;
DoRead  SOS     Read,RdList,Error3 ; Read one keystroke character
        LDA     RdBuffr            ; Get character that was read
        CMP     #03.               ; Is it Control-C?
        BNE     ChkCR              ; No, go see if was Carriage
;                                    Return
        LDX     #255.          ; Yes, set Basic error code,
        LDA     #00            ; indicate not a SOS call error,
        JMP     Error3         ; and go process error
ChkCR   CMP     #13.           ; Is it Carriage Return?
```

```
        BNE     ChkBS           ; No, go see if it was Backspace
        JMP     Finish          ; Yes, go finish up and return
ChkBS   CMP     #08.            ; Is it Backspace?
        BNE     ChkChar         ; No, go see if it is allowed
;
;       Process a Backspace input character
;
        LDA     Count           ; Get the input character count
        BNE     Backup          ; If not zero, backup 1 char
Beep    WRITE   WBeepA,#WBeepL,WrList,Error3 ; If zero, Beep the
;                                                   speaker
        JMP     DoRead          ; Continue if Write went OK
;
Backup  DEC     Count           ; Char Count = Char Count - 1
        WRITE   WBlkA,#WBlkL,WrList,Error3    ; Blank out last
;                                                   input character
        JMP     WriteC          ; Continue if Write went OK
;
;       Check if input character is one of those allowed
;
ChkChar LDY     AllowLn  ; Get length of ALLOW$ data
        DEY              ; Convert Length to an Offset
ChkNext LDA     @Allow,Y ; Get a character of ALLOW$ data
        CMP     RdBuffr  ; Compare it with the character that was
;                          read
        BEQ     Found    ; Branch if they match
        DEY              ; Otherwise, move to next ALLOW$ character
        BPL     ChkNext  ; Continue if more ALLOW$ characters to
;                          check
        JMP     Beep     ; All checked and no match - go Beep
;                          speaker
;
Found   LDX     Count    ; Get count of characters so far
        CPX     Length   ; Is it equal to LENGTH% already?
        BEQ     Beep     ; Yes, don't accept character
        LDA     RdBuffr  ; No, get the input character
        STA     Input,X  ; and save it.  Then
        INC     Count    ; Char Count = Char Count + 1
;
        WRITE   RdBuffA,#01,WrList,Error3 ; Echo the input
;                                           character
        JMP     WriteC                ; and go get more input
;
;       After carriage return keyed, load Input characters
;        received into KEYS$ using Business Basic subroutines.
;
;
;       First, convert address of Input string to Extended form and
;        setup input pointer for StrCP Business Basic subroutine.
;
Finish  LDA     #" "                    ; Load buffer with
        STA     RdBuffr                 ;  a Blank and
        WRITE   RdBuffA,#01,WrList,Error3 ;  erase the Blinking UL
        Bnk2Ext InputA,Strng1       ; Convert InputA to Extended
;                                     form
        STA     Strng1XB            ; Set Extended Address X-Byte
        LDA     InputA              ; Set low byte of
        STA     Strng1              ;  StrCP input
;
;       Create temporary string descriptor and set
;        FacM0, FacM0+1, and FacM0XB as pointers to it.
;
        LDY     Count          ; Get length of Input data
```

```
        BBSUB   StrCP         ; Create temporary string descriptor
;
; Setup output pointer for InpCom Business Basic subroutine.  Then
; assign temporary string to Business Basic string variable KEYS$.
;
        LDA     KeysD           ; InpCom input is pointer to
        STA     InpOut          ;  KEYS$ string descriptor
        LDA     KeysD+1         ;  (FacM0 already points to
        STA     InpOut+1        ;   Input string descriptor)
        LDA     KeysX
        STA     InpOutXB        ; Don't forget the X-Byte!
        BBSUB   InpCom          ; KEYS$ = Input
;
; Use SOS Device Control call to restore the Console Status Table
;
        DCONTROL 01.,DCList,STableA,Error2 ; Restore Console
;                                            Status Table
;
; Use SOS Device Control call to Download A Partial Character Set
;
        DCONTROL 17.,DCList,OldUL,Error2 ; Download normal
;                                          Underline
;
;       Finally, CLOSE the console.
;
        SOS     Close,ClosLst,Error1       ; Close the Console
;
;       Normal Exit. Put Return Address back on to
;        the stack and then Return to the caller.
;
Return  PUSH    RtnAdr          ; Put Return Address back on stack
        RTS                     ; Return to Business Basic program
;
;       Error Exits if any SOS call or Basic errors occur.
;
;        For SOS call errors, code is in A-Register.
;        For Basic errors, code is in X-Register and A-Register is
;        zero.
;
;        Error1: Error occurred while OPENing the console
;        Error2: Error occurred after console was open but before
;                Status Table was preserved
;        Error3: Error occurred after console was open and after
;                Status Table was preserved
;
;
Error3  PHA                                 ; Save error code on
;                                             the stack
        DCONTROL 01.,DCList,STableA,GetCode ; Restore Console
;                                             Status Table
        DCONTROL 17.,DCList,OldUL,GetCode   ; Download normal
;                                             Underline
        PLA                          ; Reload the error code
Error2  PHA                          ; Save error code on the stack
        SOS     Close,ClosLst        ; Close the Console
GetCode PLA                          ; Reload original error code
;
Error1  TAY                     ; Non-zero code in A-Register?
        BNE     SOSErr          ; Yes, SOS call error occurred
        BBSUB   BError          ; No, must have been a Basic error
SOSErr  BBSUB   SError          ; Translate SOS code into Basic error
        .END
```

# THE /// HELP PAGE

## /// USER GROUPS

### ARIZONA

Indp. Comp. Consultants Assoc.
Attn: Mike Diross
P.O. Box 32115
Phoenix, AZ 85064
(602) 892-3270

### CALIFORNIA

A.T.U.N.C.
Attn: Chuck Schreiber
220 Redwood Highway, #184
Mill Valley, CA 94546
(415) 934-4175

Fresno Apple /// User Group
Attn: Chairperson
4175 North Blackstone
Fresno, CA 93726
(Unknown)

International Apple Core
Attn: Apple /// SIG
908 George Street
Santa Clara, CA 95054
(408) 727-7652

L.A.-South Bay Apple /// Users
Attn: Gary Fortune
P.O. Box 432
Redondo Beach, CA 90277
(213) 316-7738

Orange County Apple /// Users
Attn: Steven Blythe
22501 Eloise Avenue
El Toro, CA 92630
(714) 951-1231

Sacramento Apple /// User Grp.
c/o Chuck Davidson
2906 Tioga Way
Sacramento, CA 95821
(916) 487-9830

Stanislaus Apple Group
Attn: Chairperson, /// SIG
1124 Brady Avenue
Modesto, CA 95350
(Unknown)

Trinity Solutions
Attn: Chairperson
5340 Thornwood Drive
San Jose, CA 95123
(408) 578-3162

### CANADA

Apple British Columbia Computer
Attn: Chairperson, /// SIG
P.O. Box 80569
Burnaby, British Columbia
CANADA V5H 3X9
(Unknown)

Canadian Apple /// User Group
Attn: Murray Lambert
80 Antibes Drive, #2805
Willowdale, Ontario
CANADA M25R 3N5
(416) 665-3622

R.A.M.
Attn: Robert Leger
P.O. Box 21
St. Jean, Quebec
CANADA J3B 6Z1

### COLORADO

Apple /// User Grp. Colorado
Attn: Carol Chrzanowski
P.O. Box 3155
Englewood, CO 80112
(303) 779-4003

### CONNECTICUT

Apple /// Soc. of South Conn.
Attn: John Lomartire
34 Burr School Road
Simsbury, CT 06880
(203) 226-4198

Hartford Users Group Exchange
Attn: Richard Johnson
29 Ridge Road
Simsbury, CT 06070
(203) 658-6692

### FLORIDA

Sarasota Apple /// Users Group
Attn: Chairperson
909 South Tamiami Trail
Nokomis, FL 33555
(813) 484-4021

### GEORGIA

Atlanta /// Society
Attn: Steve Harrison
385 Saddle Lake Drive
Roswell, GA 30076
(404) 992-3130

### IDAHO

Computer Farmer
Attn: Kelly Klaas
Route 1, Box 4133
Twin Falls, ID 83301
(208) 733-4251

### ILLINOIS

Third Apple Users Group (TAU)
Attn: Lavona Rann
1113 Wheaton Oaks Drive
Wheaton, IL 60187
(312) 665-6319

### INTERNATIONAL

Apple /// Owners and Users Int.
Attn: H. Joseph Dobrowolski
PSC Box 913
Langley AFB, VA 23665
(804) 764-7109

Apple User Group Europe e.V.
c/o Box 11 01 69 D-4200
Oberhausen 11
WEST GERMANY 0049-6195-7 3917

BAUD Apple Computer Club
Attn: Thomas E. Linders
P.O. Box 991
APO NY, NY 09098
(Unknown, Americans in Europe)

British Apple Sys. User Group
Attn: Chairperson, /// SIG
c/o Box 174, Watford Herts
ENGLAND WD2 6NF
0727 73390/72728

Copenhagen Computer Club
Attn: Frank Buchan
Ravnsborggade 19
Copenhagen, 2000 N.
DENMARK

Le Club Apple
Attn: Sophie Heuze
43, Avenue De La Grande-Armee
75116 PARIS
FRANCE

Personal Computers Users Soc.
Attn: Osvaldo Briceno
P.O. Box 56, Barcelona
VENEZUELA 6001-A

### KANSAS

Kansas City Apple /// Users Group
Attn: Steven Chernoff
5533 Granada
Roeland Park, KS 66205
(913) 588-6060

### KENTUCKY/INDIANA

Kentucky/Indiana Personal Comp.
Attn: Robert Hastings
P.O. Box 3564
Louisville, KY 40201
(502) 589-0254

### MAINE

Southern Maine Apple Users Group
Attn: Bill Harkins
Casco Street
Freeport, ME 04033
(207) 865-4761

### MARYLAND

Washington Apple Pi
Attn: Charlene Ryan
8227 Woodmont Avenue, #201
Bethesda, MD 20814
(301) 654-8060

### MASSACHUSETTS

Applesauce
Attn: Chairperson, /// SIG
24 Dickinson Street
Amherst, MA 01002
(Unknown)

### MICHIGAN

Grand Rapids Apple Group
Attn: Howard Bultman
4444 Bonnie Street S.E.
Grand Rapids, MI 49508
(Unknown)

### MINNESOTA

Minnesota Apple Corp. User Group
Attn: Dave Rasmussen
P.O. Box 796
Hopkins, MN 55343

### NEBRASKA

Big Red Apple Club
Attn: John Wrenholt, Ed.
1105 South 13th Street, #103
Norfolk, NE 68701
(402) 379-4680

### NEW JERSEY

American Assoc. Micro. Investors
Attn: /// SIG Chairperson
P.O. Box 1384
Princeton, NJ 06542
(609) 921-6494

North Jersey Apple /// Users
Attn: Roger T. Richardson
P.O. Box 251
Allamuchy, NJ 07820
(201) 852-7710

### NORTH CAROLINA

North Carolina Apple /// Users
Attn: Chairperson
2609 North Duke Street, #103
Durham, NC 27704

### NORTH DAKOTA

North Dakota Apple /// Users
Attn: Chairperson
Harwood, ND 58042
(Unknown)

### OHIO

Apple Dayton
Attn: Chairperson, /// SIG
P.O. Box 1666
Fairborn, OH 45324-7666
(513) 879-5895

Cincinnati Apple /// Users
Attn: Chairperson
7960 Shelldale Way
Cincinnati, OH 45252
(513) 542-7146

### OREGAN

Portland Apple /// Users
Attn: Milton Johnson
15291 S.E. Thornton Drive
Milwaukie, OR 97222
(503) 225-1623

### TEXAS

Apple Corps of Dallas
Attn: Chairperson, /// SIG
P.O. Box 5537
Richardson, TX 75080
(Unknown)

Houston Area Apple Users
Attn: Roger Schermerhorn
P.O. Box 610150
Houston, TX 77063
(713) 480-5690

River City Apple Corps
Attn: Chairperson, /// SIG
P.O. Box 13349
Austin, TX 78711
(512) 454-9962

### VIRGINIA

Charlottesville Apple /// Users
Attn: Prof. Richard Martin
213 Turkey Ridge Road
Charlottesville, VA 22901
(804) 924-4981

Greater Tidewater /// Users
Attn: Daniel Whiting
Route 2, Box 216
Hayes, VA 23072
(804) 642-5655

### WASHINGTON

Apple Puget Sound Pro. Lib.
Attn: Dick Hubert
290 S.W. 43rd Street
Renton, WA 98055
(206) 251-5222

Central Washington RBBS (C/PM)
Attn: Patrick E. O'Farrell
P.O. Box 538
Sehal, WA 98942
(509) 697-7296

Computer Users Federation
Attn: Chairperson, /// SIG
P.O. Box 23483
Milwaukee, WI 53223
(Unknown)

## HELP LINE

THE /// HELP LINE is a service sponsored by THE /// MAGAZINE for the benefit of all Apple /// owners. /// users with problems can call a number within their area to get assistance. The persons answering the phones are fellow ///r's who have volunteered to help others with their /// questions at no charge to you. Please remember the time zone differences, and be SURE to observe the listed calling hours- we don't want to wake anyone out of a sound sleep! We are always looking for additional names for the list, and if you would like to help other ///r's, please write us and state your areas of expertise and times available.

| Subject | Code | Subject | Code |
|---|---|---|---|
| Accounting | AC | Graphics | GR |
| Agriculture | AG | Micro-Sci | MI |
| Assembly Language | AL | Modems | MD |
| Business Basic | BB | Pascal | PA |
| Catalyst | CT | Profile | PR |
| Cobol | CO | Quark | QU |
| CP/M | CP | SOS | SO |
| Data Base | DB | Spreadsheets | SS |
| Education | ED | Telecommunications | TC |
| Financial | FI | Word Processor | WP |
| Fortran | FO | Emulation | AE |
| General | GE | /// EZ Pieces | EP |

| NAME | STATE | TELEPHONE | DAYS | HOURS | SUBJECTS |
|---|---|---|---|---|---|
| EASTERN STANDARD TIME | | | | | |
| Coville Woodburn | NH | (603) 863-5590 | M,Tu,Th,F | 7-8 pm | CT,GE |
| Ken Johnson | MA | (413) 253-2298 | Su-Sa | 6-9 pm | BB,PA,MD,MI,WP |
| Don Loosli | MI | (313) 626-3848 | M-F | 9 am-5 pm | WP,SS,DB,WP |
| Richard F. Malley | CT | (203) 232-9505 | M,Tu,W,F | 6-9 pm | GE,SO,WP,CT,PR |
| Harry T. Hanson | NJ | (201) 467-0712 | M-F | 6-9 pm | GE,PA,BB,CT |
| Edward N. Gooding | VA | (804) 747-8751 | Su-Sa | 6-9 pm | CO,SS,MD,CT,PR |
| Al Johnson | FL | (904) 739-1042 | M-F | 9 am-6 pm | GE |
| Daryl Anderson | NY | (716) 648-2462 | M-F | 9 am-5 pm | AL,BB,CT,SO |
| CENTRAL STANDARD TIME | | | | | |
| J. Donald Glenn | NE | (402) 291-9177 | Su-Th | 7-10 pm | GE |
| Jim Ferencak | IL | (312) 599-7505 | M-F | 10 am-5 pm | SO,WP |
| Neil Quellhorst | IL | (217) 434-8727 | Su-Sa | 7-9 pm | AL,BB,TC,GR,PA |
| MOUNTAIN STANDARD TIME | | | | | |
| Terri Wiles | CO | (303) 850-7472 | Su-Sa | 10 am-6 pm | PA |
| PACIFIC STANDARD TIME | | | | | |
| Pat Holwagner | CA | (415) 433-2323 | M-F | 10 am-6 pm | GE,SS,SU,AE,EP |
| Vincent F. Latona | CA | (818) 703-0330 | M-F | 9 am-5 pm | GE,WP,BB,SS,AE |
| Frank W. Moore | CA | (916) 485-6525 | M-F | 6-9 pm | BB,GE,WP |
| Dennis R. Cohen | CA | (818) 956-8559 | Su-Th | 10 am-10 pm | GE |
| Tim Harrington | CA | (415) 257-1220 | M-F | 6-9 pm | AL,BB,GE |

## BBS

### EAST COAST

Three's Company
(804) 747-8752
Access Times: After 6 pm EST
Protocols: 7 bits
Duplex: Full
Parity: Odd
Transmission Speeds: 300-1200
Cost: Free
(Largest /// private BBS in the world)

The Lost Caverns of Tsojcanth (Third Dimension)
Phone: (201) 891-5699
Access Time: After 6 pm EST
Protocols: 7 or 8 bits
Duplex: Full
Parity: Odd or Even
Transmission Speeds: 300-1200
Cost: Free
(Technical/Game Board)

Washington Apple Pi
(301) 986-8085
Access Times: After 6 pm EST
Protocols: 7 or 8 bits
Duplex: Full
Parity: Odd or Even
Transmission Speeds: 300
Cost: Free

### INTERNATIONAL

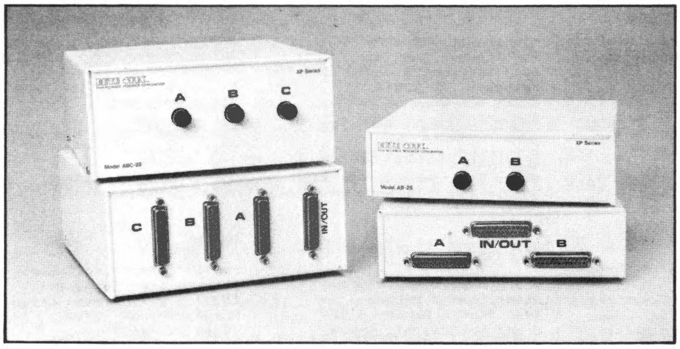
Compuserve (MAUG: GO APPLE3)
Phone: Various (See manual)
Access Times: 24 hour
Protocols: 7 or 8 bits
Duplex: Half or Full
Parity: Odd or Even
Transmission Speeds: 300-2400
Cost: $6.00+/Hour depends on time/area
(World's largest Apple BBS)

### WEST COAST

Doug Thom's BBS (CP/M)
Phone: (408) 253-1309
Access Times: Varies
Protocols: 7 bits
Duplex: Full
Parity: None
Transmission Speeds: 300-1200
Cost: Free

A.T.U.N.C.
Phone: (415) 673-9571
Access Times: After 6 pm PST
Protocols: 7 or 8 bits
Duplex: Full
Parity: None, odd or even
Transmission Speeds: 300
Cost: Free

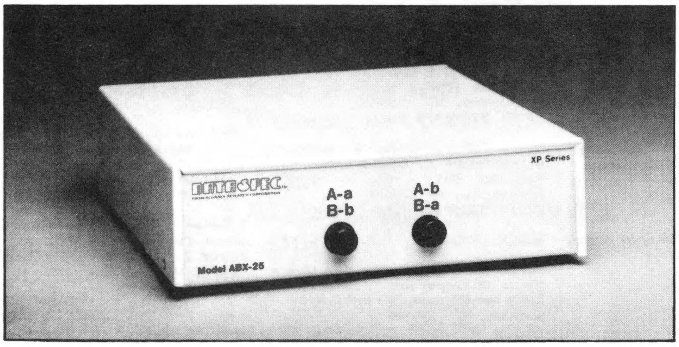
## 25 PIN (EIA RS232) SWITCHES



Models: AB-25, ABC-25

- All 25 pins switched. Can be switched with IBM PC parallel port.
- DB25F (female) connectors with gold plated contacts.

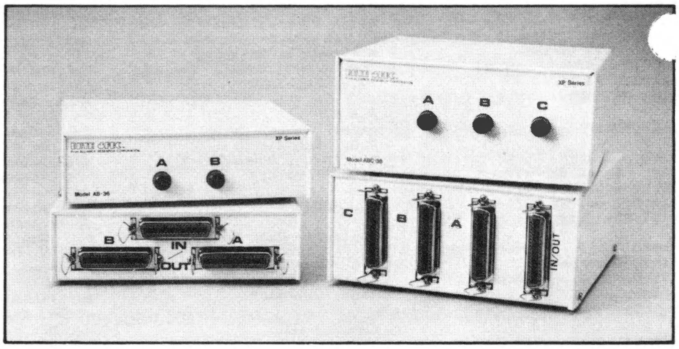
## 25 PIN (CROSS MATRIX) SWITCH



Model: ABX-25

- Enables the alternate use of 2 computers and 2 peripherals.
- All 25 pins switched.
- DB25F (female) connectors with gold plated contacts. (Note: Cross matrix switch also available in 36 pin model).

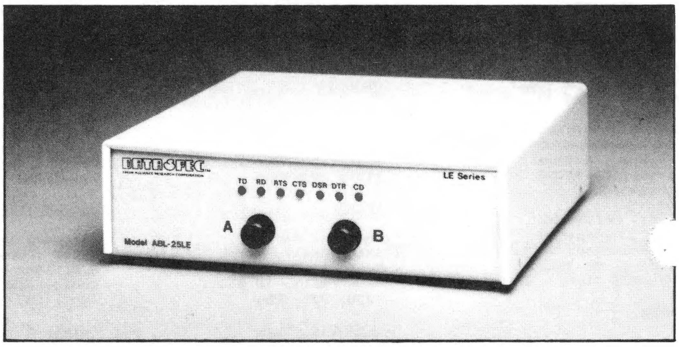
## 36 PIN (CENTRONICS TYPE) PARALLEL SWITCHES



Models: AB-36, ABC-36

- All 36 pins switched.
- Female centronics connectors with gold plated contacts.

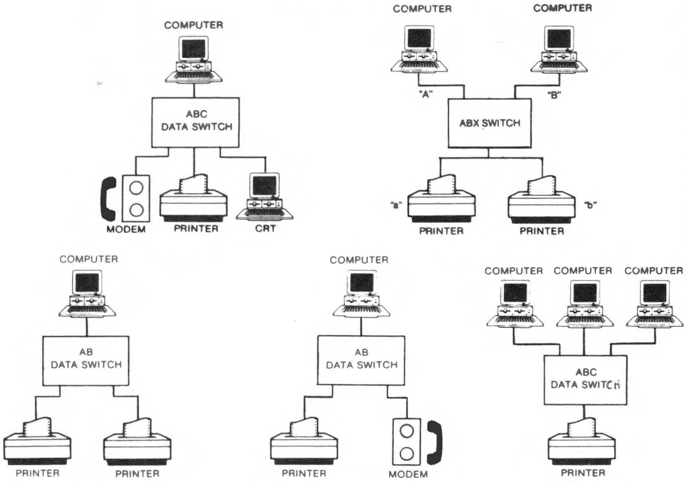
## 25 PIN (L.E.D) SWITCH



Model: ABL-25LE

- Features L.E.D. functional indicators.
- All 25 pins switched.
- DB25F (female) connectors with gold plated contacts.

## Typical Applications for DATA SPEC® Data Switches:



| MODEL | DESCRIPTION | ORDER # | PRICE | S/H COSTS |
|---|---|---|---|---|
| AB-25 | A/B Switch | 8000 | 59.95 | 4.00 |
| ABC-25 | A/B/C Switch | 8001 | 104.95 | 5.00 |
| ABX-25 | Cross-Matrix | 8002 | 89.95 | 5.00 |
| ABX-36 | Cross-Matrix | 8003 | 145.95 | 6.00 |
| AB-36 | A/B Switch | 8004 | 99.95 | 5.00 |
| ABC-36 | A/B/C Switch | 8005 | 149.95 | 6.00 |
| ABL-25LE | LED Switch | 8006 | 129.95 | 6.00 |

PAIR SOFTWARE

3201 MURCHISON WAY
CARMICHAEL, CA 95608
(916) 485-6525

DATA SPEC®

"Prices do not include tax or shipping."

# NEW /// PRODUCTS FROM PAIR SOFTWARE

| ITEM | ORDER # | COST | S/H |
|---|---|---|---|
| ASSEMBLY LANGUAGE | | | |
| Jeppson Dissassembler (with manual) | 5000 | 15.00 | 1.00 |
| BASIC PROGRAMS | | | |
| Appleseed (basic program modules to build your own) | 5100 | 10.00 | 1.00 |
| Basic Input Module (full text input control) | 5101 | 10.00 | 1.00 |
| Basic Tools (various utilities written in basic) | 5102 | 10.00 | 1.00 |
| Basic Text Processors (word processer program frame) | 5103 | 15.00 | 1.00 |
| Chartmaker (for Apple plotters) | 5104 | 10.00 | 1.00 |
| CP/M | | | |
| CP/M Public Domain Pk. #1 (various programs) | 5200 | 10.00 | 1.00 |
| FORTH and XLISP Languages | 5201 | 10.00 | 1.00 |
| Small-C Language | 5202 | 10.00 | 1.00 |
| COMMUNICATIONS | | | |
| Access /// Interpreter (upgrade w/command language) | 5300 | 15.00 | 1.00 |
| Access /// 1.1 Program (upgrade, with 1.1 notes) | 5301 | 15.00 | 1.00 |
| Access /// 1.1 Program (with manual and 1.1 notes) | 5302 | 35.00 | 2.00 |
| Codefile Transmitter (for transmission of any file) | 5303 | 10.00 | 1.00 |
| DATA SPEC RS232 AND CENTRONICS SWITCH BOXES | | | |
| *AB-25 RS232 Two Way Switch Box | 8000 | 59.95 | 4.00 |
| *ABC-25 RS232 Three Way Switch Box | 8001 | 104.95 | 5.00 |
| *ABX-25 Cross-Matrix Switch Box | 8002 | 89.95 | 5.00 |
| *ABX-36 Cross-Matrix Switch Box | 8003 | 145.95 | 6.00 |
| *AB-36 Centronics Two Way Switch Box | 8004 | 99.95 | 5.00 |
| *ABC-36 Centronics Three Way Switch Box | 8005 | 149.95 | 6.00 |
| *ABL-25LE RS232 Two Switch Box (w/LED readouts) | 8006 | 129.95 | 6.00 |
| DISK UTILITIES | | | |
| Backup /// (for all hard disk users) | 5400 | 10.00 | 1.00 |
| Blocread (block read/save utility) | 5401 | 15.00 | 1.00 |
| Confidence Program (checks /// system) | 5402 | 10.00 | 1.00 |
| Disk Maker Utility (volume copier for diskettes) | 5403 | 15.00 | 1.00 |
| Dealer Diagnostics (96K/128K /// only) | 5404 | 10.00 | 1.00 |
| Dealer Diagnostics (256K /// & ///+ only) | 5405 | 10.00 | 1.00 |
| Fixer ('repairs' Catalyzed disks for individual use) | 5406 | 10.00 | 1.00 |
| DIF Print (print DIF files in format) | 5407 | 15.00 | 1.00 |
| Disk Check (Repair Damaged volume directories) | 5408 | 15.00 | 1.00 |
| File Backup Utility (individual file copier) | 5409 | 15.00 | 1.00 |
| Grabit (Converts ASCII text files to Quick File) | 5410 | 10.00 | 1.00 |
| List It (list PASCAL text files formatted) | 5411 | 15.00 | 1.00 |
| Mail List Manager Utilities (for MLM) | 5412 | 15.00 | 1.00 |
| Pascal File Converter (PASCAL text to ASCII text) | 5413 | 10.00 | 1.00 |
| Reformatter /// (Interchange DIF/ASCII/Qk. File files) | 5414 | 20.00 | 2.00 |
| System Utilities 1.1 (w/device sub-type changer) | 5415 | 10.00 | 1.00 |
| System Utilities 1.2 (latest version) | 5416 | 10.00 | 1.00 |
| Textgraphics (ASCII text to FOTO file converter) | 5417 | 15.00 | 1.00 |
| FINANCE | | | |
| VisiCalc Templates Disk #1 (various templates) | 5500 | 15.00 | 1.00 |
| LISTS | | | |
| Apple /// Cust. List (on 140K Disk /// diskettes) | 5600 | 150.00 | 5.00 |
| Apple /// Cust. List (on 572K Micro-Sci diskettes) | 5601 | 150.00 | 2.00 |
| /// Magazine Customer List (latest list) | 5602 | 50.00 | 1.00 |
| MANUALS AND OTHER PUBLICATIONS | | | |
| Service Manual w/256K Diagnostics Disk | 5700 | 70.00 | 5.00 |
| /// MAGAZINE 1985 (issues Magazine/Newsletter 1985) | 5701 | 50.00 | 1.00 |
| "How to use your Apple /// in Ten Easy Video Lessons (VHS or BETA)" | 5702 | 69.95 | 2.00 |
| NEW USER PACKAGE - Includes OWNERS and STANDARD DEVICE DRIVERS Manuals, SYSTEM UTILITIES 1.2 and CONFIDENCE | 5703 | 40.00 | 3.00 |
| OPERATING SYSTEM AND LANGUAGES | | | |
| Bus. Basic 1.23 (upgrade, no 64K program limit) | 5800 | 10.00 | 1.00 |
| CONSOLE and BOOT Source Text (disassembed info) | 5801 | 10.00 | 1.00 |
| FORTRAN-77 (Pascal upgrade, 1977 standard) | 5802 | 35.00 | 3.00 |
| PASCAL 1.2 (upgrade) | 5803 | 20.00 | 2.00 |
| PASCAL 1.2 Dir. Patch (put PASCAL 1.2 on sub-direct.) | 5804 | 10.00 | 1.00 |
| PAS. 2.0 Compiler (upgrade w/manual, for PASCAL 1.2) | 5805 | 20.00 | 2.00 |

| ITEM | ORDER # | COST | S/H |
|---|---|---|---|
| PASCAL PROGRAMS | | | |
| *Font Maker /// (Generate new FONT files) | 5900 | 15.00 | 1.00 |
| FreeForm (form based data base w/source code) | 5901 | 20.00 | 1.00 |
| GrafDraw (graphic drawing program) | 5902 | 15.00 | 1.00 |
| *Label Maker /// (creates mail list labels) | 5903 | 15.00 | 1.00 |
| p-Code Disassembler (Op-Code creator only) | 5904 | 15.00 | 1.00 |
| Pascal Menu Program (one menu runs many PASCAL prgs.) | 5905 | 15.00 | 1.00 |
| Pascal Modules #1 (various programmer modules) | 5906 | 10.00 | 1.00 |
| Pascal Modules #2 (various programmer modules) | 5907 | 15.00 | 1.00 |
| Pascal Modules #3 (various programmer modules) | 5908 | 15.00 | 1.00 |
| *Pascal Modules #4 (various programmer modules) | 5909 | 15.00 | 1.00 |
| Pascal X Reference (find variables in source text) | 5910 | 15.00 | 1.00 |
| Pascal Utilities #1 (volume directory sorter) | 5911 | 15.00 | 1.00 |
| Pascal Utilities #2 (includes Font Capture) | 5912 | 15.00 | 1.00 |
| Sign Maker (makes banner signs) | 5913 | 15.00 | 1.00 |
| PERIPHERALS | | | |
| *STAR SG-10 NLQ Dot Matrix printer | 8101 | 299.00 | 8.00 |
| *STAR SG-15 NLQ Dot Matrix printer | 8102 | 465.00 | 9.00 |
| *Apple /// Printer Connector (all modules) | 8100 | 20.00 | 2.00 |
| *TEKNIKA MJ22 RGB/COMPOSITE Monitor | 8201 | 349.00 | 12.00 |
| *Apple /// Monitor Cable (all models) | 8200 | 35.00 | 2.00 |
| PUBLIC DOMAIN PACKAGES (Various Programs) | | | |
| Public Domain Pk. #1 (with Elisa AI program) | 6000 | 10.00 | 1.00 |
| Public Domain Pk. #2 (with Star Trek program) | 6001 | 10.00 | 1.00 |
| Public Domain Pk.#3 (w/Hec-Dec Converter) | 6002 | 10.00 | 1.00 |
| Public Domain Pk. #4 (with Mortage rate program) | 6003 | 10.00 | 1.00 |
| Public Domain Pk. #5 (with file type Changer program) | 6004 | 10.00 | 1.00 |
| Public Domain Pk. #6 (with Mac-Unit program module) | 6005 | 15.00 | 1.00 |
| Public Domain Pk. #7 (w/Basic Modem program) | 6006 | 15.00 | 1.00 |
| SYSTEM DRIVERS | | | |
| RAT //e .Mouse driver (any /// language or system) | 6100 | 15.00 | 1.00 |
| SOS Drivers Upgrade (latest known) | 6101 | 15.00 | 1.00 |
| WORD PROCESSING | | | |
| Applewriter WPL Pk. #1 (WPL shorts) | 6200 | 10.00 | 1.00 |
| Ink Well Upgrade (upgrade, added features) | 6201 | 15.00 | 1.00 |
| Super Applewriter (upgrade v4.1) | 6202 | 15.00 | 1.00 |
| PAIR SOFTWARE COPYRIGHTED ITEMS (Not Public Domain) | | | |
| *CHECKBOOK /// (checkbook ledger program) | 6300 | 49.95 | 2.00 |
| Data Base /// (super data base manager) | 6301 | 100.00 | 4.00 |
| Electronic Mailman (E-Mail formatter) | 6302 | 100.00 | 4.00 |
| *Modula-2 (Coming Soon!) | 6303 | ---.--- | --.--- |
| Post Master (super mail list manager) | 6304 | 200.00 | 3.00 |

* = New Items

ALL ITEMS MARKED UPGRADE MUST HAVE YOUR ORIGINAL DISKETTE SUBMITTED WITH YOUR ORDER. ALL SALES FINAL. SOME PROGRAMS MAY NEED WORK TO BE USED IN YOUR APPLICATION. CP/M ITEMS REQUIRE CP/M CARD. BUSINESS BASIC PROGRAMS REQUIRE VERSION 1.23. PASCAL PROGRAMS REQUIRE VERSION 1.2. CALIFORNIA RESIDENTS MUST INCLUDE 6% SALES TAX. PLEASE BE SURE TO INCLUDE SHIPPING/HANDLING COSTS. OVERSEAS BUYERS PAY 3 TIMES LISTED AMOUNT. CHECK OR MONEY ORDER ONLY! A $10.00 PER PACKAGE SURCHARGE REQUIRED IF YOU ARE NOT A SUBSCRIBER TO THE /// MAGAZINE. ALLOW FOUR WEEKS FOR DELIVERY.

SEND YOUR ORDER TO PAIR SOFTWARE, 3201 MURCHISON WAY, CARMICHAEL, CA 95608 (916) 485-6525.

# Of all the great features found on this Apple /// the most attractive is its price.

## FREE REPLACEMENT GUARANTEE!

Quantity Discounts Available

The Apple ///. A computer that was built to take the kind of treatment dished out in the business environment. Easy to use day in and day out. Just ask someone who already owns one or two.

Our systems come complete with monitor and these computer features:

- 128K or 256K options
- Easy-to-use keyboard (typewriter style)
- Numeric keypad (calculator style)
- Built-in disk drive
- 4 expansion slots
- Full 30-day Sun Warranty

SUN SYSTEMS REMARKETING
P.O. BOX 4059 Logan, UT 84321

**ORDERLINE**
**1-800-821-3221**
**For Orders Only**

**CUSTOMER SERVICE**
**801-752-7631**
**Questions—Service**

*Includes computer and monitor. Unless otherwise noted equipment is reconditioned with a 30-day unconditional warranty. Manuals are not supplied with reconditioned equipment but are available seperately at extra cost. A complete copy of Sun Data's Warranty is available on request.