INTERNATIONAL
APPLE CORE
TM

APPLE ORCHARD
Including CONTACT

IN THIS ISSUE:

APPLE *III* IS HERE!

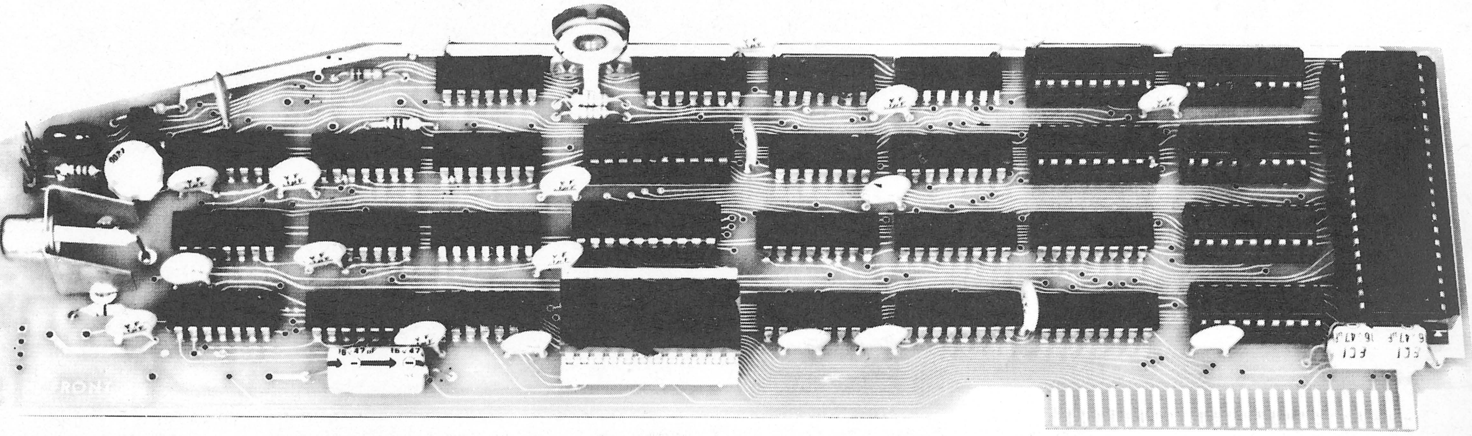
SEE APPLE "SOS" ON PAGE 29

AND ALSO

SPECIAL HI-RES GRAPHICS SECTION ON PAGE 7

FALL 1980 $3.50

SUP'R'TERMINAL™



SUP'R'TERMINAL IS AN 80 COLUMN BY 24 LINE PLUG-IN COMPATIBLE BOARD FOR THE APPLE II COMPUTER

SPECIFICATIONS & FEATURES

- 80 Columns by 24 lines, upper and lower case; all 128 ASCII characters.
- Upper and Lower case data entry using the APPLE II keyboard.
- Includes an Upper and Lower case 5x8 dot matrix ASCII character set, and inverse alpha characters.
- Expands existing keyboard for more ASCII characters
- Character set can be user definable
- Includes VBC™ (video balance circuit) which enables the use of displaying 80 columns on an inexpensive 8 MHz CRT monitor
- Works with LEEDEX monitor (version 2.2) and other inexpensive CRT monitors
- Shift Lock Feature
- KEYPRESS function for PASCAL programs supplied
- Works with APPLE PASCAL and APPLE BASIC
- Incorporates PASCAL and BASIC control characters
- Follows protocols of PASCAL and BASIC operating systems
- ALL monitor-type escapes are valid
- Compatible with ALL APPLE II peripherals.
- Effective baud rate greater than 10,000; fast scrolling and clearing
- Can be used with APPLE II communication interface board to act as self contained terminal for time-sharing or other applications. Terminal program supplied when used with a D.C. Hayes micromodem.
- 3K bytes of bank switched static ram
- 2K bytes of ROM
- The only board with continuous direct memory mapped screened ram.
- The only board that interprets VTABS by firmware (version 2.2)
- The only board with an adjustable scrolling window.
- The only 80 column board that is synchronous with the APPLE II
- Fully programmable cursor
- Conversion program supplied to modify existing APPLESOFT programs to work with SUP'R'TERMINAL (automatically converts HOME,CALL-936 and VTABS) (version 1.0)
- Works with the new Easywriter and APPLE PI word processors.
- Uses less current on the +5V supply than any other 80 column board
- Works with CORVIS hard disc system

PATENT PENDING

M&R ENTERPRISES
P.O. BOX 61011, Sunnyvale, CA 94088

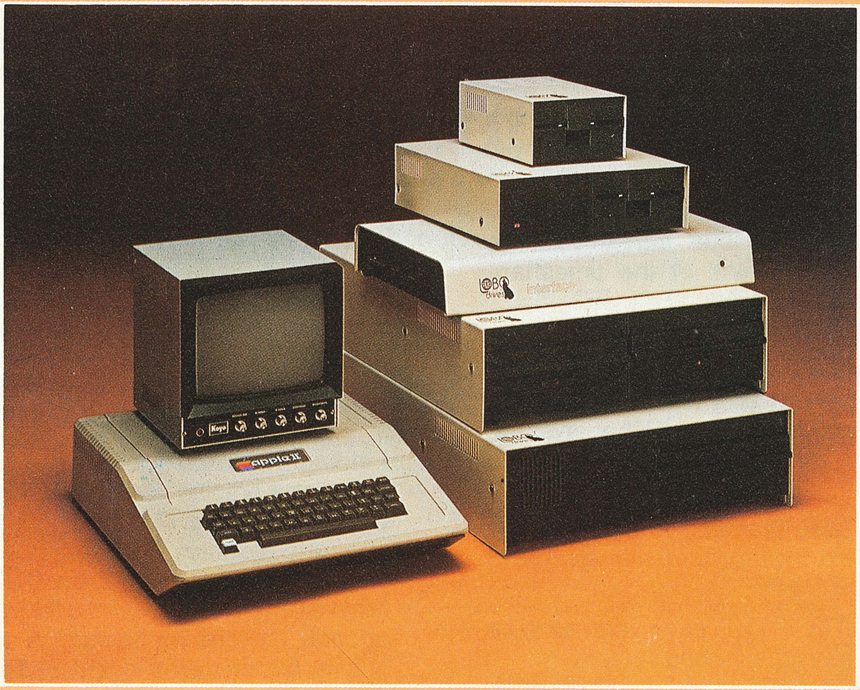
When It Comes To APPLE Add-on Memory...

LOBO Has It All.

LOBO DRIVES manufactures a complete line of APPLE compatible disk drives. All LOBO Subsystems are completely software compatible to both APPLE 3.1, 3.2, and 3.2.1 Disk Operating Systems (DOS), as well as APPLE Pascal Language Systems. All standard DOS commands i.e.; SAVE, LOAD, UNLOCK, DELETE, etc., will run on both the 5¼-inch and 8-inch drives. Any applications software designed for use with the APPLE will run on all LOBO Subsystems.

The simplicity of operation, high reliability, improved performance, and unique one-year warranty policy makes LOBO DRIVES Disk Memory Systems the ideal drives for the APPLE user who wants to develop or expand his own disk-based system.

Every LOBO DRIVES Disk Memory System comes complete with drive(s) power supply and chassis, cable, interface and controller card.



MODEL 1850 DUAL FLOPPY/FIXED DISK MEMORY SYSTEM

Up to 10 MBytes of high-speed, high-reliability Winchester technology hard disk storage, combined with a built-in 1.6 MByte Floppy for back-up, on/off load, etc. makes the 1850 ideal for small business and word processing applications.

- 5 or 10 MByte Fixed Disk Capacity
- 1.6 MByte Floppy Disk Capacity
- 70 Msec Average Access Time
- Software Compatible
- Sealed Environment with Winchester Reliability

See your nearest dealer, call, or write for the complete LOBO DRIVES story... find out just how competitively priced a quality drive can be.

LOBO drives

INTERNATIONAL

935 Camino Del Sur
Goleta, California 93017
(805) 685-4546
Telex: 658 482

MODEL 950 DUAL FLOPPY/FIXED DISK MEMORY SYSTEM

New 5¼-inch Hard Disk provides the storage capacity of more than 16 Floppy Disk Drives in one compact, easy to use cabinet. Built-in mini-floppy back-up.

- Up to 6.38 MBytes of High Speed Hard Disk Memory
- Up to 440 KBytes of Floppy Disk Back-up
- 170 Msec Average Access Time
- Sealed Environment Winchester Reliability

MODEL 390 5¼-INCH FLOPPY DISK MEMORY SYSTEM

A low-cost, high performance Mini-Floppy System fully compatible with APPLE Disk II.

- Up to 116 KBytes Capacity per disk
- Fast 200 Msec Access Time
- 156 Kbits/Sec Data Transfer Rate
- Individual File Write Protection

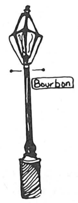
MODEL 800/850 DUAL FLOPPY DISK DRIVE MEMORY SYSTEM

Large capacity, with choice of single-sided, single or double density (850) configurations. LOBO's controller can support up to 4 daisy-chained drives.

- Up to 3.2 MByte Capacity
- Fast Access Time
- Intelligent Disk Controller
- APPLE Software Compatibility

Southeastern Software

6414 Derbyshire Drive • New Orleans, LA 70126
504/246-8438 504/246-7937



Southeastern Software 'NEWSLETTER' for APPLE II Owners
NOW IN THE THIRD YEAR OF PUBLICATION
10 Issues per year for $10.00
Back Issues available at $1.00 each
EXAMPLE:
Send $10.00 and receive next 10 Issues
Send $30.00 and receive 30 Issues beginning with #2

DATA CAPTURE 3.0 - $29.95

Is DATA CAPTURE 3.0 just another Smart Terminal program? NO! It is a GENIUS Terminal program for use with the Micromodem II™. It will 'capture' ANYTHING that appears on the screen of your CRT. ANY program or data. If you are using the Source you can even 'capture' CHAT. There is no need to create files in your file space on the other system to transfer data to your Apple. If you can list it you can capture it.

* You can then SAVE the data to disk, dump it to your printer or even do simple editing with DATA CAPTURE 3.0.
* You can use DATA CAPTURE 3.0 to compose text off line for later transmission to another computer. Think of the timeshare charges this will save you!
* Use DATA CAPTURE 3.0 with the Dan Paymar Lower Case Adapter and you can enter UPPER or lower case from the keyboard for transmission to another system. You can also capture UPPER/lower case data from another system.
* A program is also included to convert your programs to text files for transmission using DATA CAPTURE 3.0.
* DATA CAPTURE 3.0 will save you money if you are using any timesharing system.

Requires DISK II™, Applesoft II™

Add $64.95 to order the Dan Paymar Lower Case Adapter

BAD BUY DISKETTE - $9.99

Of course it's a bad buy. If you have issues #2 thru #11 of the NEWSLETTER you can type these programs in yourself. Includes a couple of bonus programs.

Requires DISK II™, Applesoft II™

We ship within 3 working days of receipt of order and welcome your personal check. We also accept Visa and Master Charge.

LCMOD for PASCAL - $30.00

Finally! DIRECT entry of UPPER/lower case into the Pascal Editor. Why pay hundreds of dollars for a terminal just to set lower case entry with Pascal? If you have the Paymar Lower Case Adapter you can use this program.

* Left and right curly brackets for comment delimiters.
* An underline for VARs, program names and file names.
* The ESCape key does the shifting and Control Q is used for ESCape. Have you ever typed in a page or two of text and lost it by hitting ESC accidentally? This won't happen with LCMOD.

Requires Language System and Paymar LCA

Add $64.95 to order the Dan Paymar Lower Case Adapter.

MAG FILES – $18.00

Finding it difficult to keep track of all those magazine articles you are reading? This program will help you do it. MAG FILES is Menu driven with separate modules for creating, editing, displaying and searching for your data. If you are using one drive a program is provided for transferring data to another diskette for backup. A sample data base of over 60 articles is included. The screen formatting and user orientation are what you have come to expect of Southeastern Software.

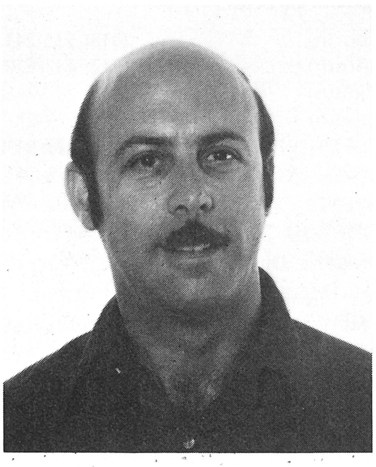
Requires DISK II™, Applesoft II™.

MAILER – $15.00

Don't let the low cost fool you. This is a single drive version of the program we use to maintain the NEWSLETTER subscriber list. Can be easily converted to 2.3 or 4 drives. Binary search and linear searches for finding any name in file. Sort on names and zip codes. Selective print by zip code or key. The separate modules are menu driven and will run on 32K system. There are 13 separate modules on the diskette for maintaining a mailing list. Sample data file included.

Requires DISK II™, Applesoft II™.

PRESIDENT'S CORNER



It was a small group of Apple users who met in San Francisco late in October of 1979. They represented some of the larger Apple Clubs in the United States.

The object of that meeting was to open channels of communications between Apple clubs and users. It was at this meeting the International Apple Core got its roots, and plans were made on the WHAT, HOW and WHEN of the IAC.

It is now not quite a year after that meeting and at the time of this writing, late July, the IAC has a membership of over 150 clubs, representing some 42 states and 13 countries, and some 13,000 Apple users. We are still receiving about three membership applications a week and requests for information about the IAC has kept our secretary very busy.

This unbelievable growth signifies to me that there is a need for just such an organization as the IAC. We will continue to grow as more clubs are formed and old ones find out about our services. The IAC is made up of clubs and it is up to the member clubs to guide the direction we take in the future. That idea is at the heart of the IAC — an organization responsive the needs of the membership. Your input to this effect should be via your area director. The director is familiar with his area's clubs and their particular needs. These directors are also elected to office by its area clubs.

Our prime concern is information transfer both from the manufacturers to the user and from the user to other users and back up to the manufacturers. We are trying many methods to accomplish this and will implement new ones as time passes.

The IAC is still very young, and like your club it is run by volunteers with regular 8 to 5 jobs. We are in need of more volunteers to help on committees and projects, so please be patient. If you would like to volunteer, please contact the IAC.

This is our second issue of "The Apple Orchard" with current plans for an issue each quarter. There has been over 80 pages of application notes sent to member clubs with more application notes being typed every week. In addition, IAC has already distributed free software to member clubs and will continue to do so as it becomes available.

Our SIG (Special Interest Groups) span from help to the handicapped to a Ham radio network of Apple users. At this time we are planning the FIRST — APPLE FEST. This will be the first personal computer faire dedicated to the Apple Computer. This means no other computer will be shown — only hardware and software for the Apple. The faire will be sponsored by the IAC and Boston/Apple and will be held in Boston on May 23 and 24 of 1981 where the IAC will hold its annual general meeting.

If you have any questions about the IAC please write us (include your phone number and we will endeavor to answer them). I hope to see many of you at the Fest in 1981.

Ken Silverman, President
International Apple Core

INTERNATIONAL APPLE CORE SPONSORING MEMBERS

I.A.C. sponsors are a special breed. They are the organizations who along with our advertisers, contribute to and support many I.A.C. activities. In addition, they will provide us with application notes concerning their products — notes that will benefit users by showing new and different ways to utilize their products or production/software modifications that have been made to upgrade their product. When considering a software or product purchase, we request that they be given special consideration.

Those organizations that would like to become sponsors or who would like additional information about the benefits and advantages of becoming a sponsoring member are urged to contact Michael Weinstock, Vice-President, International Apple Core, P.O. Box 976, Daly City, CA 94017.

A list of sponsoring members, current through the first of September, 1980, appears below.

Apple Computer, Inc.
10260 Bandley Drive
Cupertino, CA 95014
(408) 996-1010

Axiom Corp
5932 San Fernando Rd.
Glendale, CA 91202
(213) 245-9244

Bell & Howell, Inc.
7100 McCormick Road
Chicago, IL 60645
(312) 262-1600

Compuserve - Micronet
5000 Arlington Centre Blvd.
Columbus, OH 43220
(614) 457-8600

Custom Computing System, Inc.
122 2nd Ave. N.
Saskatoon, Sask.,
Canada S7K 2B2

Interactive Structures, Inc.
P.O. Box 404
Bala Cynwyd, PA 19004
(215) 667-1713

Malibu Electronics Corp.
2301 Townsgate Road
Westlake Village, CA 91361
(805) 496-1990

Nestar Systems, Inc.
430 Sherman Ave.
Palo Alto, CA 94306
(415) 327-0125

Peripherals Unlimited
3450 E. Spring St., Suite 206
Long Beach, CA 90806
(213) 595-6858

Programma International, Inc.
3400 Wilshire Blvd.
Los Angeles, CA 90016
(213) 384-1116

Siro-tech Software Products
6 Main St.
Ogdenstring, NY 13669
(315) 393-5151

Source Telecomputing Corp.
1616 Anderson Road
McLean, VA 22102
(703) 821-6660

Syntauri Ltd.
3506 Waverly St.
Palo Alto, CA 94306
(415) 494-1017

Verbatim Corp.
323 Soquel Way
Sunnyvale, CA 94086
(408) 245-4400

Xerox Retail Markets Div.
L-140, 24500 Industrial Blvd.
Hayward, CA 94545
(415) 786-5205

INTERNATIONAL APPLE CORE™

PRESENTS
THE APPLE ORCHARD

Vol. 1, No. 2 **Fall 1980**

Val J. Golding .... Editor
Ken Silverman .... Assistant Editor
Patricia Boner .... Editorial Assistant
Kathryn Hallgrimson .... Editorial Assistant
Larry Danielson .... Shipping Manager
Vic Warren Design .... Cover Design
Buck Evans .... Apple Orchard Postcard
Grawin Publications .... Production

ADVERTISING REPRESENTATIVES
Grawin Publications
1020 Lloyd Building
Seattle, WA 98101
(206) 223-0861

SUBSCRIPTIONS
Apple Orchard Subscriptions
P.O. Box 2227
Seattle, WA 98111
$10/year — Published Quarterly

INTERNATIONAL APPLE CORE

Officers

Ken Silverman	President	(415) 878-9171
Michael Weinstock	Vice-President	(516) 360-0988
Dave Gordon	Treasurer	(213) 384-0579
Joe Budge	Secretary	(919) 229-6037

Regional Directors

Jon R. Lawrence	(north)	(313) 534-2433
Harlan G. Felt	(north)	(312) 447-6267
Jerry Vitt	(south)	(214) 369-7660
Scott Knaster	(south)	(303) 355-2379
Bernie Urban	(east)	(301) 229-3458
Tony Cerreta	(east)	(914) 636-3417
Joe Alinsky	(west)	(213) 703-1894
Fred Wilkinson	(west)	(415) 585-2240

International Directors

Neil Bennett 55 Clarance St. Sydney, Australia 2000
Auby Mandell 409 Queen St. W. Toronto, Ont. Canada M5V 2A5
Wolfgang Dederichs Auf Drenhausen 2 4320 Hattingen, West Germany

Committee Chairman

Apple Fair	Bob Ramsdell	(617) 742-6100
Apple Orchard	Val J. Golding	(206) 932-6588
Constitution & Bylaws	Ken Silverman	(415) 878-9171
Education SIG	Ted Perry	(916) 961-7776
Ham Radio SIG	James E. Hassler, WB7TRQ	(307) 632-4934
Handicapped SIG	Bernie Urban	(301) 229-3458
I.A.C. Software	Neil Lipson	(215) 356-6183
Legal SIG	Butch Clayton	(803) 884-5370
Medical SIG	Dr. Larry L Stoneburner	(714) 953-9151
Newsletter Exchange	David Alpert	(312) 295-6078
Newsletter Library	Maj. Terry N. Taylor	
New Club Assistance	Randy Fields	(415) 775-7965
Standards	Mark Robbins	(303) 750-5813
Telecommunciations	Craig Vaughn	

IN THIS ISSUE

PRINT FRE (ed)

by Val J. Golding

One of the advantages of being an editor is that one can sit down at a typewriter and start banging away at the keys on almost any subject under the sun and be assured, within reasonable limits, that it will see print. Is it possible that within a three year period we can shed a nostalgic tear for the "good old days"?

We were fortunate to have been at the reins during the formative periods of two major forces, each of which in their own unique ways have become most influential in the world of Apple computing. Apple Pugetsound Program Library Exchange was among the pioneer Apple user groups that have subsequently matured to produce sophisticated software and a leading national magazine for its membership, an accomplishment still under way today.

International Apple Core, publishers of this magazine, the Apple Orchard, sprang from an idea to a nearly full blown operation in a matter of just a few months. Both organizations are slowly but surely overcoming the problems of growing pains. Both organizations are devoted to serving the needs of their respective memberships, but here the resemblance ends, and the goals of each, it will be seen, are widely divergent.

A.P.P.L.E. is a single user group composed of over 4000 individual members; I.A.C. is a group whose membership consists of over 150 different user GROUPS, scattered around the world. Its goals too, in the final analysis, are to serve the needs of individual Apple users, but through the medium of Apple user GROUPS. I.A.C. is structured to be responsive to individuals through their clubs, and through regional representation. Many of the I.A.C. services are either free or on a cost plus basis. Free software is provided to member clubs, which they in turn may distribute to their members on their own terms. Frequent mailings of application notes, furnished to IAC by Apple Computer, Inc. and others who manufacture/distribute Apple related products are made to member clubs. Again, the further dissemination of this information to their membership is at the discretion of the individual member clubs.

Through the pages of the Apple Orchard, the I.A.C. hopes to encourage readers and new Apple owners to join a local user group. To this end, you will find a list of names and addresses of current member groups elsewhere in this issue. Many of these local groups publish their own newsletters and offer other benefits such as group purchases of products and the opportunity to discuss technical and programming problems on a face to face basis.

Thanks to the efforts of the pioneer user groups, Original Apple Corps, San Francisco Apple Core, Apple Pugetsound and others, much of what we today recognize as "common knowledge" was not always the case. Many of the members of early groups literally spent hours of research, seeking out and publishing data that was not available in the early and skimpy documentation published by Apple Computer and others. The original Apple reference manual (before the "red book") was a mimeographed pamphlet of some 30 odd pages, a far cry from todays 200 page manual.

The pages of the Apple Orchard are a blend of three main categories, new material contributed by individuals and/or club members, material that has previously been printed in one of the low circulation club newsletters that is deserving of a much wider distribution, and material supplied by Apple Computer, Inc. in the areas of utility and reference material and promotional items. It should be emphasized also that the I.A.C., and in turn, the Apple Orchard, is under no obligation to Apple Computer or any other manufacturer, and in fact receives no direct financial support, other than that falling under the heading of sponsoring members.

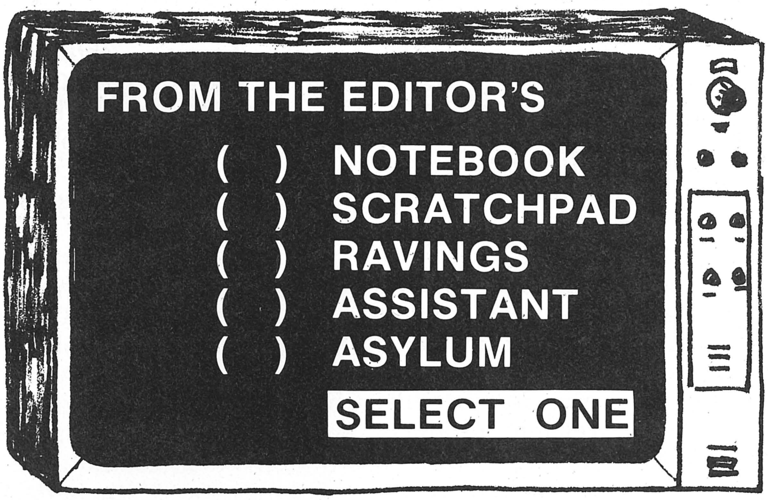
The entire premise upon which I.A.C. funds its various operations is through revenues created by sales of Apple Orchard magazines and advertising. Therefore, the I.A.C. must rely heavily on its member clubs to furnish us with suitable material, both original and reprint, that can be included in forthcoming issues of the Apple Orchard. Upon request, a modest page rate for published articles will be paid, but we also urge authors to consider their material as contributions.

What has happened between the "good old days" and the here and now? In 1977 the Apple II arrived on the scene among the Imsai, Sol, Southwest Technical and other micros, but with a difference. It heralded the beginning of an era where one could simply walk into a computer store, much as one goes to a television or appliance store, look over a few models, make a decision, take it home, plug it in and start using it. We believe its original purpose was a game machine, such as todays Atari and others. In fact, Steve Wozniak's original handwritten notes for Integer Basic called it "Game Basic".

But Woz and Steve Jobs never counted on the Apple II's achieving the tremendous success and popularity that it has. When Apple obtained Applesoft I from Microsoft, Inc., people first became aware of the tremendous potential of the Apple II as a games/household/business computer, and it took off, to be followed eventually by the Apple III, a moderately priced, sophisticated business computer, and which will shortly be followed by an Apple IV (although it will not be known by that name), a machine that in many respects may invite comparison with a 370.

Would we, if we had our

(continued on page 6)



By the Editor

From the fire to the frying pan, to invert a homily. We didn't know at first if we would really make it, this second issue of **International Apple Core's** "Apple Orchard". But if you, gentle reader, are reading this, then we indeed did!

There is a fierce competition today, a far cry from just a very few months ago, among computer magazines, and even more so, among those devoted to the Apple computer. There is no question but what, starting with a 45,000 circulation, that we are going to make our way to the head of the field. The Apple Orchard is/will be a *DIFFERENT* magazine, encompassing all areas of Apple Computing.

Included in this issue of the Orchard, and in succeeding issues, will be a separate section known as CONTACT, which will contain Application Notes and other material furnished to us by Apple Computer, Inc., to help you better understand and make use of your Apple. In addition, through the pages of CONTACT, you will learn of new Apple Computer, Inc. products and peripherals, even before they become widely available. And in this CONTACT, **Barry Yarkoni** offers a look at the Apple III and its "AppleSOS", while another article describes the whys and hows of RFI (Radio Frequency Interference), along with some suggested cures.

And that is just one section. Another section will be devoted to the International Apple Core. One of the features of this section will be to provide you with a listing of the name, address and phone number of each of the nearly 150 Apple user groups that are members of the **I.A.C.** This is a service that will help new owners find an Apple user group in their vicinity, plus there are a number of larger groups, national and international in stature, where membership can be of benefit. (These are indicated by a * in the listings.) And in future issues, there will be mention of computer shows, fairs, etc.

The wealth of information in just these two sections alone would make a subscription worthwhile, but we haven't even touched on the feature material. Look at this issue — starting on Page 7, there is a 14 page special High Resolution Graphics section — information not only on how it works, but actual applications, articles by **Bob Bishop, Loy Spurlock** and **Pete Rowe.**

Have you ever wondered just what goes on when you type "INIT HELLO, VI" and your disk drive starts up, sputters, burps and eventually hands you a fresh diskette you can use to store programs on? **I.A.C.** secretary **Joe Budge** takes a look at that process in **Inside Initialization,** and President **Ken Silverman** has whomped up a mess o' statistics showing the amount of current drawn by many Apple peripherals in **Don't Overload your Apple.**

And still there's more, but the Table of Contents has to serve some useful purpose. And if we haven't tempted you by now to whip out your checkbook and write out a ten dollar subscription check (there's a form on Page 26), then you better just hand this copy of the Apple Orchard back to your dealer. And while you're at it, better turn your Apple back in too, because it won't be of much use without the Orchard!

But don't touch that dial. . . before you go away, we want to take the time, and this space, to say a very special *THANK YOU* to our Editorial Assistant, **PATRICIA BONER.** Without Pat's miles of driving, untold hours of pasting up, repasting up and then pasting up some more, plus 1001 other services beyond the call of duty, neither the first Orchard nor this one could ever have become reality. Thanks, Pat. Love 'ya!

And in practically the same breath we want to welcome **Kathryn Hallgrimson** as our new assistant. Kathryn has been determinedly dogging Patricia's footsteps and will be contributing to the future success of the Orchard.

Val. J. Golding

PRINT FRE(ed) from page 5

"druthers", go back to the "good old days"? No indeed. For as much as we have already learned, we have barely scratched the surface of the Apple II. Every day, as we continue our exploration, we continue to learn. And all of this knowledge, in one form or another, will filter to you, the user, through the pages of the Apple Orchard and other magazines.

The readers are the real winners!

APPLE—II HI-RES GRAPHICS: RESOLVING THE RESOLUTION MYTH

by Bob Bishop
Apple Computer, Inc.

In the early part of 1977 the Apple—II computer was introduced, and it soon became one of world's most popular machines. One of the most exciting features of the computer was its high resolution (HI-RES) graphics capability. Early literature from Apple described the computer as being able to "...generate a high-resolution (280h x 192v) graphic display in four colors...." (The four colors were: black, white, violet, and green.) It wasn't long before two more colors, blue and orange, were added to the list. Now Apple—II could boast of having a resolution of 280 x 192 in six colors.

Because of the ambiguity of the wording, the literature soon became misunderstood as implying that the Apple—II could plot any of the six colors in any of the 280 horizontal positions. Unfortunately, such a capability would require more than the 8-K bytes of memory available to HI-RES, as can be easily calculated. Yet, even after three years since the Apple's introduction, this "280-point" myth still lives! Except now the story has been "fuzzed" a little to say things like "some points just can't be plotted in some colors," or that green "doesn't exist" at some points, or some such nonsense. (This "explains" (?) why the Applesoft program:

```
10 HGR
20 HCOLOR = 1
30 HPLOT 0,0 TO 10,150
```

draws five separate line segments instead of the one continuous line that we actually wanted.)

The real problem here is not in the hardware, but in the headware. The time has come when we must modify the "traditional" view of Apple—II HI-RES in order to reflect the true nature of the beast. The "modern" view realizes that, in actuality, the Apple—II doesn't have just one single HI-RES mode. It has two of them! And neither mode involves an alleged 280 points!!

Color HI-RES Mode

The first of the two HI-RES modes is the Color mode. Here, the screen resolution is 140 x 192 in six colors. If we were to clear the HI-RES screen and then "turn on" all the green dots that we could, we would find that there are only 140 dots in each line that would ever show up. A similar experiment with blue, orange, and violet would yield similar results. Well then, if there are only 140 color points per line, why try to pretend that there are 280? Let's just call a spade a spade and accept the fact that there only 140 color points. To anyone who still wants to pretend that there are actually 280 points (with "green-not-existing" at half of them), I can offer the equally valid counter-claim that there are really 2800 points with "green-not-existing" at ninteen out of twenty of them!

Black & White HI-RES Mode

The second of the two HI-RES modes is the Black & White mode. In this case, the "280-point" myth really sells the Apple—II short. It turns out that, on a Black and White display, we can actually achieve a horizontal resolution of up to 560 points!

In order to fully understand these two graphics modes we must take a closer look at the Apple—II's HI-RES capability.

At one time, the "hardest" part about HI-RES was handling the Y-coordinate. Because of the strange way in which the screen is mapped, an exotic "base calculation" routine was required to compute the absolute memory address of the beginning of the display line corresponding to Y. (See Figure 1.) Once this non-linear mapping was done, handling the X-coordinate was "easy." All we had to do was count over X bits starting with the LSB (Least Significant Bit) of the base byte, skipping the MSB (Most Significant Bit) of every byte. On a black & white display we would see the resulting dot in one of the 280 possible positions on the line. On a color display we would see the same dot in either violet/blue or green/orange, depending on whether the "skipped MSB" was a zero/one, and whether or not the X coordinate was even or odd, respectively. (See Figure 2.) Thus was the "280-point" myth born.

This "traditional" view of HI-RES has changed little in the three or so years that the Apple—II has been in existence. Although the "hard" job of doing the vertical coordinate base calculations is being replaced with table look-up methods (making it now the "easy" part), not much has been done to resolve the mish-mash about the (formerly "easy") horizontal component. I, personally, was never really happy with the 280-points/line philosophy except for black & white displays. For color graphics the 140 x 192 approach made more sense. (STAR WARS, ROCKET PILOT, APPLE-VISION, etc. were all written in accordance with the 140-point/line philosophy.)

So what is the "correct" way to view the HI-RES process? Well, let's see what really happens on a HI-RES screen, and then you be the judge.

Assume that we have an Apple computer with both a color display and a high resolution black & white display. If we enter the keyboard monitor and type the commands:

```
*2000:0
*2001<2000.3FFEM
*C050 C053 C057
```

we will see the blank HI-RES screen with four lines for text at the bottom. Typing:

```
*2000:1
```

produces a dot in the upper left-hand corner on both displays. (The color display's dot will be violet.) We now type:

```
*2000:81
```

and, as the RETURN key is hit, we see the dot on the black and white display move ever so slightly to the right. Looking at the color display we notice that the violet dot has become a blue dot. Continuing in a like manner we observe that:

***2000:2**

followed by:

***2000:82**

cause the dot on the black and white display to shift slightly to the right each time while the dot on the color display changes from blue to green and then from green to orange. If we were to continue on with:

***2000:4**
***2000:84**

•
•
•

***2027:C0**

we would cycle through the colors: violet, blue, green, and orange, 140 times as the dot slowly progresses from left to right across the top of the color display. The black & white display's dot would also be shifting to the right each time proving that there are actually 560 positions that it can occupy. (See Figure 3.)

Now lets try the following:

***2000:3**

This turns on dots 0 and 2 in the first line. Looking at the black & white display we see two dots very close together in the upper-left. But the color display shows a single white dot there. Next, type:

***2000:83**

Again we see a white dot there. (In the Apple literature these whites have been referred to as "White1" and "White2", while "Black1" and "Black2" represent their corresponding absences. More on this unfortunate business later.) So, white (the color) is created by simply turning on any pair of consecutive even or odd dots, while black, of course, is made by not turning them on. (It is also possible to create a "White1.5" by turning on an even dot in conjunction with an odd dot. However, this can only be done at the 20 places in each line where the two dots come from different bytes.) The fact that white can be made by combining a dot with either its preceeding or succeeding counterparts (e.g., green-violet or violet-green) serves as the only close claim to the much heralded "280-point" mode, and even this falls short by one! (There are only 279 pairs of dots across the screen, not 280.)

Let's return to the 560 x 192 Black & White mode for one last comment. It should be pointed out, if it isn't already obvious, that we do not have complete unrestricted access to all 560 dot positions on each line. Once a dot is plotted some of its neighbors become restricted in the sense that any later attempt to plot them will cause a one-position shift in some of the already plotted dots on the line. But then Apple users are already used to such plotting constraints. (For example, green lines cannot be plotted on orange backgrounds, etc.) And even in the worst case, the resolution obtained in the 560 mode is never worse than 280. (Again, that number!) So there is really very little reason to ever consider doing black & white plots in any other mode but the 560 mode. (Figure 4 is a listing of an Applesoft implementation of the 560 mode of plotting.)

Color Issues

The "traditional" problem of plotting one color near another and seeing a color change occur still exists, even in the "modern" view. Such "color conflicts" are not philosophical in nature, but are intrinsic to the hardware. For those "purists" who insist on color graphics without any "color conflicts" at all, we could postulate a third color graphics mode: 40 x 192 in six colors. But the utility of such a mode would probably be extremely limited due to the 3.5 fold decrease in horizontal resolution, a high price to pay for "purity".

The question of resolution becomes even more cloudy when we start to talk about HI-RES displays containing more than six colors. By a process known as "dithering",* the individual color dots in a HI-RES display can be viewed macroscopically as forming "mixed" colors. (For example, if we turned on only half of the blue dots on the screen in a "checkerboard" fashion we would see the color, dark blue. If we then changed all the remaining black dots into white the result would be light blue. Etc.) Depending on the order of the dithering and the exact nature of the algorithm used the resulting spatial resolution could be 70 x 192, 70 x 96, or even less. (This "color dithering" was the technique used in creating the COLOR SLIDE SHOW disks for both the Apple—II and the Apple—III.)

Black & White Issues

"Spin-offs" of the "280-point" myth are the "fictitious" blacks and whites, known affectionately as "Black1", "Black2", "White1", and "White2". The fact that these are true, unique color states of the HI-RES display is not in dispute. But what is disappointing is that most of the Apple's wares (soft and firm) require the users to actually specify which white or black is to be used. In practice most users don't care! All they want is Black or White, period. They aren't interested in the internals of how the graphics works. So why burden them with such needless details? The only difference between the "1" and "2" species of white and black is the state of the MSB in the byte. Since this bit only has an observable effect on the "colored" colors, a more sensible approach would have been to automatically set or clear the MSB only when dictated by the plotting of a "colored" color Plotting a white or black color would only set or clear the two "observable" bits and leave the "unobservable" MSB in its ambient state. Using this approach, only one black and one white would have been required instead of two. The result would have been a HI-RES package with a much cleaner human interface.

So, where do we go from here? Well, there are still some strange HI-RES anomalities that could be explored. Let's go back to our earlier experiments using the keyboard monitor. With a cleared HI-RES display we type the following:

***207F:40**

Nothing happens. But if we now type:

***2000:80**

(which simply turns on the MSB of location $2000), we see a phantom orange dot** appear in the upper-left corner at screen position: X=-1! Does this mean that the HI-RES

*See "Color 21" on page 21.

**See "The Mysterious Orange Vertical Line" on page 11.

screen has even more than 560 points per line resolution!? I'll leave that question for future investigators to answer.

Now that we have a more accu rate picture of the HI-RES process we are in a better position to utilize this powerful display capability of the Apple—II more effectively. Unfortunately, much of the "280-point" myth is cast in silicon and, as such, is frozen for all time. But that should not stop us from taking advantage of what we now know and applying it whenever we can. And above all, we should always try to determine for ourselves how things really are, and not simply accept the traditional explanations from the past. If we didn't, the earth would still be flat.

```
*847LL

0847-   A5 11       LDA   $11
0849-   0A          ASL
084A-   0A          ASL
084B-   29 1C       AND   #$1C
084D-   85 15       STA   $15
084F-   A5 11       LDA   $11
0851-   6A          ROR
0852-   6A          ROR
0853-   6A          ROR
0854-   6A          ROR
0855-   29 03       AND   #$03
0857-   05 15       ORA   $15
0859-   09 20       ORA   #$20
085B-   85 15       STA   $15
085D-   A5 11       LDA   $11
085F-   6A          ROR
0860-   29 E0       AND   #$E0
0862-   85 14       STA   $14
0864-   6A          ROR
0865-   6A          ROR
0866-   29 18       AND   #$18
0868-   05 14       ORA   $14
086A-   85 14       STA   $14
086C-   60          RTS
086D-   80          ???
086E-   AE 72 08    LDX   $0872
0871-   60          RTS
```

Figure 1: A typical HI-RES "base calculation" routine (from: ROCKET—PILOT, 1977)
The routine is entered at $847 with the Y-coordinate stored in $11. Upon leaving, the corresponding base address is stored in $14 and $15.

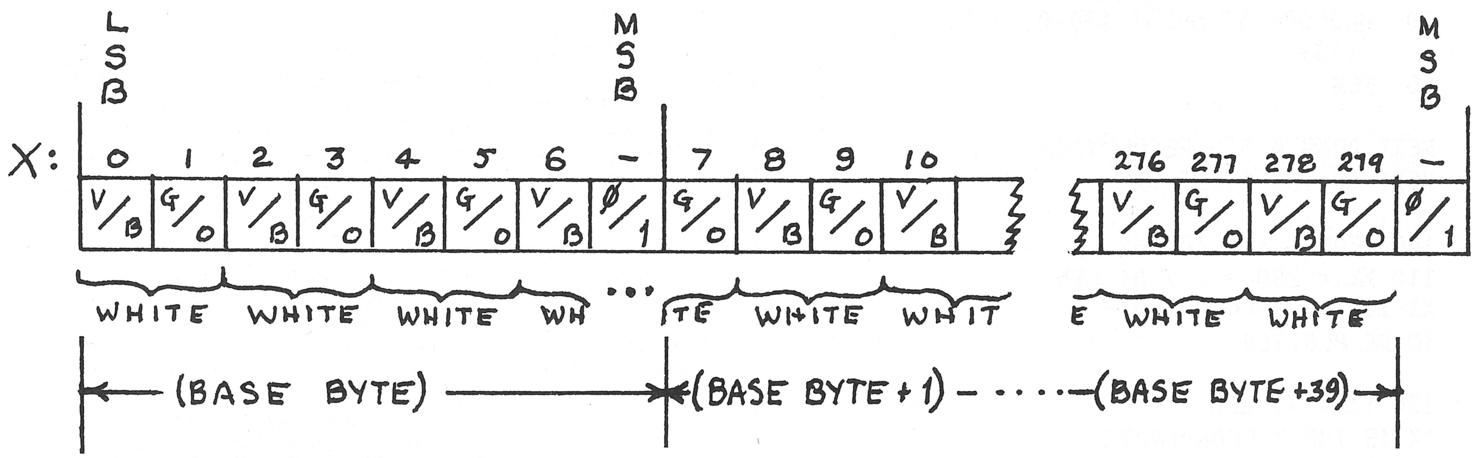


Figure 2: The "traditional" view of HI-RES
There is only one plotting mode, and it consists of 280 plot positions across the screen formed from 40 clusters of 7 bits. Each cluster represents one byte of display memory with the MSB determining the color set of the byte. (Notice that the bits are mapped "back-wards on the screen; the LSB shows up first followed by the remaining 6 bits in reverse order.)

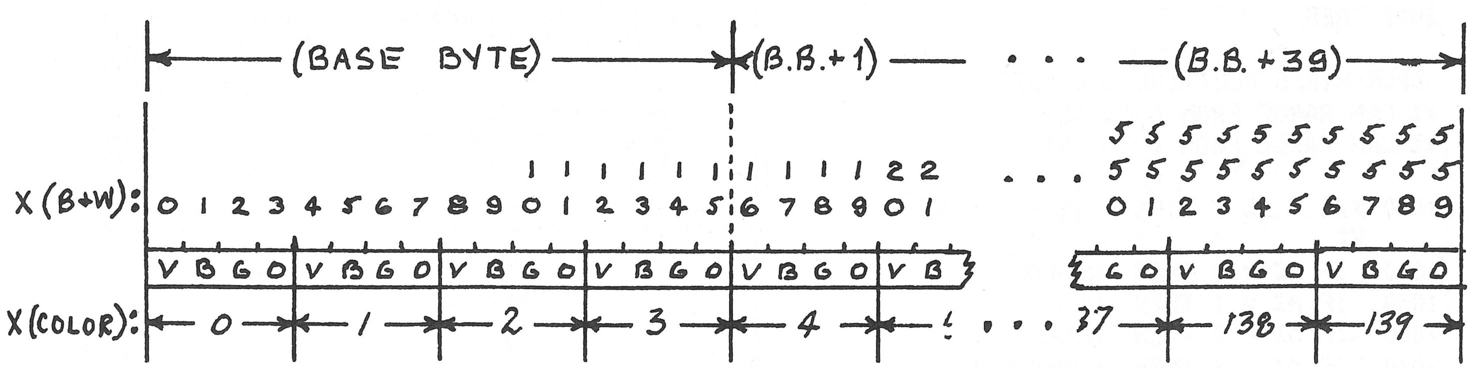


Figure 3: The "modern" view of HI-RES
There are actually two separate plotting modes: one for black & white displays, and one for color. The black & white mode consists of 560 points per line while the color mode consists of 140. Each color point is actually made up of the 4 consecutive B/W mode dots: V, B, G, and O, respectively. The "280-point" myth results from the attempt at combining these two distinct modes into one general purpose mode that is display-independent.

Figure 4: The 560 B/W mode can easily be demonstrated in Applesoft. This program first draws a steep vertical line in "traditional" 280-point mode. Then it draws an identical line in "modern" 560-point mode. Notice the smaller "stair-stepping" in the second line.

```
]
]LIST

10  HOME : HGR
20  REM

FIRST DRAW A LINE IN 280
RESOLUTION FOR COMPARISON

30  HCOLOR= 3: HPLOT 130,0 TO 140
     ,159
50  REM

NEXT DRAW A 560 RESOLUTION LINE
NEXT TO THE FIRST ONE

100  FOR Y = 0 TO 159
110 X% = 280 + Y / 8: REM
X% IS THE X-COORDINATE
TO BE PLOTTED

120 Y% = Y: REM
Y% IS THE Y-COORDINATE
TO BE PLOTTED

130  GOSUB 1000
140  NEXT Y
150  VTAB 22: END : REM

1000  REM

SUPER-HIRES PLOTTING SUBROUTINE
X% CAN RANGE FROM 0 TO 559
Y% CAN RANGE FROM 0 TO 191

1030 XX% = X% / 4:M% = X% - 4 * X
     X%
1040  IF M% = 0 THEN  HCOLOR= 2
1050  IF M% = 1 THEN  HCOLOR= 6
1060  IF M% = 2 THEN  HCOLOR= 1
1070  IF M% = 3 THEN  HCOLOR= 5
1080  HPLOT X% / 2,Y%: RETURN
```

The Mysterious Orange Vertical Line

by Pete Rowe
Computer-Advanced Ideas, Berkeley CA

On occasion you may have noticed that while displaying an Apple *Hires* picture, an orange vertical line of dots, one third the height of the screen, appears at the far left side of the picture. In fact, it appears to the left of the left-most *Hires* column. Also, this orange vertical line seems to appear and disappear at will, not under your control.

The *Hires* screens do *NOT* map to contiguous memory in the Apple. The first 40 bytes make up row zero (7 dots per byte times 40 bytes equal 280 dots horizontal *Hires* columns). The next 40 bytes are for row 64, followed by the next 40 on row 128. We have used only 120 bytes and the very next eight bytes are unused and do not map to the screen. The next set of three non-continuous lines goes to row 1, 65 and 129, again followed by 8 unused bytes. The following table shows the hex adddresses of the *Hires* page one unused bytes:

2078-207F	2878-287F	3078-307F	3878-387F
20F8-20FF	28F8-28FF	30F8-30FF	38F8-38FF
2178-217F	2978-297F	3178-317F	3978-397F
21F8-21FF	29F8-29FF	31F8-31FF	39F8-39Ff
2278-227F	2A78-2A7F	3278-327F	3A78-3A7F
22F8-22FF	2AF8-2AFF	32F8-32FF	3AF8-3AFF
2378-237F	2B78-2B7F	3378-337F	3B78-3B7F
23F8-23FF	2BF8-2BFF	33F8-33FF	3BF8-3BFF
2478-247F	2C78-2C7F	3478-347F	3C78-3C7F
24F8-24FF	2CF8-2CFF	34F8-34FF	3CF8-3CFF
2578-257F	2D78-2D7F	3578-357F	3D78-3D7F
25F8-25FF	2DF8-2DFF	35F8-35FF	3DF8-3DFF
2678-267F	2E78-2E7F	3678-367F	3E78-3E7F
26F8-26FF	2EF8-2EFF	36F8-36FF	3EF8-3EFF
2778-277F	2F78-2F7F	3778-377F	3F78-3F7F
27F8-27FF	2FF8-2FFF	37F8-37FF	3FF8-3FFF

Notice there are 64 sets of 8 bytes or 512 unused bytes in all for each *Hires* page.

The cause of the mysterious orange vertical line has been recently discovered in the unused bytes of the *Hires* screen: first, every row in the top third of the screen (rows 0 to 63) must have the first byte with the most significant bit on, that is, the first byte of a given top third row must contain orange, blue, black2 and/or white2. And second, the sixth bit must be on in the last byte of the eighth unused byte that follows the screen byte mentioned above in memory. The result: An orange dot will appear to the left of column zero of the given row. What's unusual are the facts that: (1) An apparently unused RAM cell is affecting our screen, and (2) the inter-byte culprit is not bit seven or zero, but bit six!

To try it for yourself, enter the Monitor and type:

```
*C050 C053 C057
*2000:0
*2001<2000.3FFEM (clears the
      screen)
*2000:C0    (puts a blue dot in
      column seven, first row)
*207F:40     (turns on bit six)
```

Notice the far left orange dot — the beginning of our vertical line.

```
*207F:0     (turns off bit six and
      the left orange dot goes away)
```

A similar experiment can be accomplished in BASIC:

```
10 POKE -16304,0:POKE -16301,
   0:POKE 16297,0
20 INPUT VALUE
30 FOR ADDRESS=8319 TO
   16383 STEP 128
40 POKE ADDRESS,VALUE
50 NEXT ADDRESS
60 END
```

RUN this program with *Hires* page one containing orange, blue, black2 and/or white2 at the far left of at least the top third of the screen. For VALUE, you will find that entering a number containing 64 (e.g., 64, 127, 255), will turn on the now-not-so-mysterious orange vertical line.

UNDERSTANDING HI-RES GRAPHICS

and how to include text in your Hi-res Graphics Programs

by Loy Spurlock

Reprinted from APPLESAUCE, Vol. 1, No. 7, Oct. 1979 and Call-Apple, January, 1980.

This article is about APPLE II's HI-RES graphics. It will cover three basic areas.

They are:

1. How the screen is formatted.
2. Which RAM is used to get the picture you want.
3. How the RAM is used to get the picture you want.

We will be discussing mainly how to put text into your graphics pictures. However, if you can understand how to put text in Hi-res, you will be able to define your own character set and put almost anything you want on the screen. We will do it with Integer Basic so that the majority of the beginners will be able to understand what is going on. If you know and understand assembly language, you will probably have no problem using the information.

The last page of this article is an Integer Basic program that will allow you to put text into your Hi-res pictures with just a few lines of BASIC.

There are six charts throughout this article that might be helpful in understanding the balance of the article.

They are:

Chart #1.	Full screen chart as it appears in the text mode.
Chart #2.	A blowup of the upper left corner of chart #1.
Chart #3.	A blowup of the upper left corner of chart #2.
Chart #4.	Total addressing of Hi-res page #1.
Chart #5,	Addressing and data charts in binary.
Chart #6.	Final breakdown of each bit in each byte.

At this point, I would suggest that you look the charts over and get familiar with them. Read the text that is with the charts so that you will know what that chart contains. After you do that, then come back and continue.

To start off, we will discuss memory locations used by the Hi-res graphics. There are two pages of Hi-res graphics. You can have a different picture on each one of them and flip from one to the other by doing the proper pokes as listed on page 30 of the red manual. The first page uses the RAM from location 8192 to 16383 and the second page uses RAM from 16384 to 24575. This means that if you have only a 16K machine, you cannot access page #2 because you will not have any RAM to operate it. So we will only be covering the use of page #1 in this article. If you wish to use page 2, you can use all this information by just starting at 16384 instead of 8192.

Each RAM location in the Hi-res area is continually analyzed by the hardware in the machine to determine what to put on the monitor screen. Each RAM location controls 7 dots on the screen the size of the period. If the proper value is in any given RAM location, all 7 dots will be turned on, creating a line two dots longer than the line at the top of the 'T'. Look at figure 1 on chart #5. That represents the 8 bits in every RAM location in the machine. Think of these bits as separate switches with which each can be turned on or off. The bit on the right represents the value of 1 when it is turned on. Box number 2 represents the value of 2 when it is on. The 3rd box is valued at 4, the 4th at 8, the 5th at 16, the 6th at 32, the 7th at 64 and the 8th at 128. To turn one or any combination of bits on, it is necessary for you to POKE the proper value into the location that you want to control. If you wanted to turn only the 1st bit on, you would POKE your location with the number 1, which is the value of the only bit that you want on. If you wanted only the 4th bit on, you would POKE the location with an 8 because that is the value of the 4th bit. Now let's light up 2 bits. To turn the 2nd and 5th bits on, you add the 2 values together. The 2nd bit is valued at 2 and the 5th bit at 16. 2+16=18, so, you would POKE your location with an 18, which is the value of the 2 bits that you want on.

Now, let's put this to use. Before we go into detail about the RAM formatting of the screen, we are going to play with turning bits on and off. Turn your APPLE on. While in the monitor mode, we will clear the Hi-res graphics page. With the '*' prompt showing, type '2000:0', then hit return. Do not type the apostrophies, only what is between them. Now type '2001<2000.3FFF M' and hit return. Now go to basic and type 'GR'. This will put you into LO-RES graphics. Now type POKE -16297,0 which will put you in HI-RES graphics. You should now be looking at a totally blank screen.

Before we get started with the experiments, I must tell you that the screen looks at the binary bits backwards from the way you use them in counting. What this means is that the left most bit is valued at 1 and the right most bit is valued at 128. Also, only 7 bits will show up on the screen. I am told that the 8th bit controls the additional colors on the newer machines. My APPLE is #86 and does not have the mod, so I have never used the 8th bit. Hopefully, someone out there who has extensive experience with the colors will submit a subsequent article.

At this point, you should be looking at the blank Hi-res screen. Memory location 8192 is the one that controls the first 7 dots in the upper left corner of the screen. So, POKE a 1 into 8192 by typing 'POKE 8192,1' and hit return. One dot in the corner should have come on. Now type 'POKE 8192,64'. The dot that was on should have gone off and another one come on. That's because 64 is the value of the 7th bit, which is the last bit that the graphics will use. If you want both dots on at the same time, add the 2 values together, 64+1=65, and type 'POKE 8192, 65'. Now, both dots should be on. This probably seems like a lot of work to get something on the screen. It is, if you do it all manually. However, when you can use formulas to figure out where to put the dots, the computer will do it all for you. Play with that for awhile. You can use any RAM location between 8192 and 8231 for the entire top line.

Now that you know how to control each RAM location, let's talk about the formatting of the screen so you can use the whole thing.

	A	B	C	D
A	1024	0	0	1063
B	1152	1	3	1191
C	1280	2	6	1319
D	1408	3	9	1447
	1536	4	12	1575
	1664	5	15	1703
	1792	6	18	1831
	1920	7	21	1959
	1064	8	1	1103
	1192	9	4	1231
	1320	10	7	1359
	1448	11	10	1487
	1576	12	13	1615
	1704	13	16	1743
	1832	14	19	1871
	1960	15	22	1999
	1104	16	2	1143
	1232	17	5	1271
	1360	18	8	1399
	1488	19	11	1527
	1616	20	14	1655
	1744	21	17	1783
	1872	22	20	1911
	2000	23	23	2039

CHART #1

This chart represents the screen as you see it in the text mode, with characters on it. It is broken into a lot of little boxes, (40 across and 24 down). Only one character at a time can be in each box. Notice the 'A' in the upper left corner? It will be on all the other charts also. This will help you to keep proper perspective as to the size of the portion of screen that we are looking at. In the text mode it takes only one memory location to operate each one of these boxes. In Hi-res it takes 8 bytes of RAM to control each box. Unfortunately the RAM locations do not follow through from one line to the next. In this chart you see 4 columns of figures. They are:

A— The first RAM location used for that line in the text mode.
B— The screen line number.
C— The line numbers as they fall in sequential RAM locations.
D— The last RAM location used for that line.

The screen line # will be the one referred to from this point unless otherwise noted.

Notice that line #1's first RAM location is 1024 and that the last is 1063. That is a total of 40 RAM locations for the 40 characters across on the first line. Also notice that line #2 does not have the next RAM location of 1064. It is on line #9. So, if following the RAM in sequence, it jumps to line #9 from line #1. The last RAM location on line 9 is 1103. 1104 is the first RAM location on line 17. The last RAM location of line 17 is 1143. Now, here is a more confusing part. There are 8 RAM locations that are not used (1144-1151). 1152 starts line #2. This continues until the entire screen is filled.

Check out chart #4. It contains all of the addresses of Hi-res page #1. By using this, you should be able to put a dot on the screen anywhere you want by manually poking them in. However, what we want to do is figure out a way to get the machine to calculate where to put the dots. What we have to do is find a definite pattern that the RAM uses so that we can build a formula to give the machine an X-Y coordinate.

First, notice that the first 8 lines all begin with a RAM location 1024 bytes from each other. That is definitely a pattern. The problem though, is that the 9th line goes back down. It goes down by 7040 bytes. Line #10, however, increases by 1024 bytes again and continues until line #17. How about that! All the way down the screen, the lines are broken into groups of 8 that increment by 1024 until the beginning of the next group. Let's find a pattern there. Line #9 decrements by 7040 from line #8. LIne #17 decrements 7040 from line #16. This keeps up until you hit line #64. That's another pattern for the top 1/3 of the screen. Line #64 decreases by 8024 from line #63 but then starts incrementing by 1024 again. In fact, the whole pattern starts over for the entire middle 1/3 of the screen. Line #128 decrements by 8024 from line #127 just like line #'s 63 and 64. That completes our entire pattern scheme. Let's recap the whole thing and see what we have. Starting at 8192, we increment by 1024 8 times, then decrease by 7040. Do this 8 times, then decrease by 8024. Call the above phase #1. Starting with the last address from phase one, you can now do phase #1 again. Now, starting with the last address from doing phase #1 the second time, do phase #1 a 3rd time. The screen is then broken into 3 main segments, which are broken into 8 smaller segments the size of a text line, which has 8 fine lines each. The addresses we just discussed are only the first byte, which controls only the first 7 dots in each line. To get to the other columns you would just add the X coordinate to the calculation of the Y coordinate (considering that X is across and Y is down). Here's a formula that will do the entire calculation. See if you can figure out what it is doing before you read the explanation. The variable 'L' is used for the final location address. The parameters of X & Y are: X (0-39 across), Y (0-24 down). Here's the formula: 'L=8192 + Y MOD 8 * 128 + Y / 8 * 40 + X'. Here's how it works. Let's say that the X-Y coordinates are 15-18, which would be the 15th column on the 18th line.

Since the multiplication and division of formulas is done from left to right before any addition is done, we will calculate all the multiplication and division first, then go back and do all the addition. To start with, if we convert all X's and Y's to the numbers that we chose, the formula would look like this: (L=8192 + 18 MOD 8 * 128 + 18/8 * 40 + 15). Y MOD 8 is the calculation that will give us the line number within the 1/3 of the screen that line Y (18) is in. The answer to 18 MOD 8 is 2. Next we multiply the 2*128. This gives us the top text line address within that 1/3 of the screen. The answer is 256. Now our formula looks like this: (L=8192+256+18/8* 40+15). The next multiplication and division is Y/8*40. The Y/8 will give us the 1/3 of the screen that we want. For example, 18/8=2. The 2 represents the 2nd 1/3 (middle) of the screen. We then multiply 2*40 to get the address of the middle section for an answer of 80. The formula now looks like this: (L=8192+256+80+15), or (location=start+line#+1/3 section+ column). The answer is: 8192+256+80+15=8543, which will be the top 1/8th of column 15 on text line #18. Remember that line #18 is really the 19th line of text because line #0 is the lst and line #1 is the 2nd. Our calculation comes out to the address of the 144th line of Hi-res + 15 bytes. The 15 bytes get us to the 16th column because remember the 1st column is column #0. You can check this out by referring to chart #4. After you learn how to create your own characters, you can put text or any other character of your choice anywhere on the Hi-res screen by giving this formula an X-Y coordinate and using it.

(text continued on page 17)

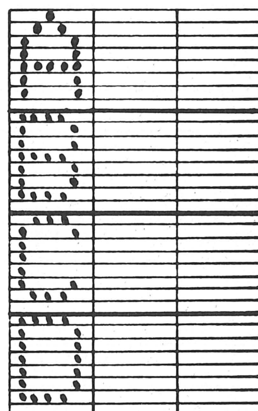
CHART ## 2

Notice that the 'A' is in the upper left corner as it is on chart #1.

The 2nd thing that you probably noticed is that some of the smaller boxes are divided into 8 sections. This is because it takes 8 RAM locations to control the same sized area as one controls in the text mode. There is another major difference, in that the RAM locations themselves are different. Instead of starting at 1024, we now start at 8192. Location 8192 is the location that controls the area that the single dot at the top of the 'A' is in. There are 40 RAM locations across the top of the screen, the same as in the text mode. However, each one controls only the top 1/8 of each box.

When the top line is completed with 40 bytes of RAM, it then jumps down to what would be the ninth line and controls the top 1/8 of what would be line #2 of the text mode. This is actually the 9th line of Hi-res which is where the top row of the 'B' is.

The 2 columns of numbers on the right side of the chart are:
1. The Hi-res line #.
2. The address of the first byte of that line.



0	8192
8	8320
16	8448
24	8576
32	8704
40	8832
48	8960
56	9088
64	8232
72	8360
80	8488
88	8616

0	8192				●				8193	8194
1	9216			●		●			9217	9218
2	10240		●				●		10241	10242
3	11264		●				●		11265	11266
4	12288		●	●	●	●	●		12289	12290
5	13312		●				●		13313	13314
6	14336		●				●		14337	14338
7	15360								15361	15362
8	8320		●	●	●	●			8321	8322
9	9344		●				●		9345	9346
10	10368		●				●		10369	10370
11	11392		●	●	●	●			11393	11394
12	12416		●				●		12417	12418
13	13440		●				●		13441	13442
14	14464		●	●	●	●			14465	14466
15	15488								15489	15490
16	8448			●	●	●			8449	8450
17	9472		●				●		9473	9474
18	10496		●						10497	10498
19	11520		●						11521	11522
20	12544		●						12545	12546
21	13568		●				●		13569	13570
22	14592			●	●	●			14593	14594
23	15616								15617	15618
24	8576		●	●	●	●			8577	8578
25	9600		●				●		9601	9602
26	10624		●				●		10625	10626
27	11648		●				●		11649	11650
28	12672		●				●		12673	12674
29	13696		●				●		13697	13698
30	14720		●	●	●	●			14721	14722
31	15744								15745	15746

CHART #3

Here we are looking at the upper left corner of the screen with the same 'A' and RAM locations as chart #2. You can now see that the box is divided 8 times down and 7 times across. Each line down is controlled by a different RAM location. Each RAM location within it stores 8 separate (bits) of information. Think of these bits as 8 separate switches that can be turned on or off by the value of the number POKEd into the RAM location. The bits as they work on the screen are counted from left to right.

In binary, the bits are counted from right to left. To learn more about binary, check the figures on chart #5.

To learn more about the address of the Hi-res lines, you can refer to chart #4.

0	8192- 8231	48	8960- 8999	96	8744- 8783	144	8528- 8567
1	9216- 9255	49	9984-10023	97	9768- 9807	145	9552- 9591
2	10240-10279	50	11008-11047	98	10792-10831	146	10576-10615
3	11264-11303	51	12032-12071	99	11816-11855	147	11600-11639
4	12288-12327	52	13056-13095	100	12840-12879	148	12624-12663
5	13312-13351	53	14080-14119	101	13864-13903	149	13648-13687
6	14336-14375	54	15104-15143	102	14888-14927	150	14672-14711
7	15360-15399	55	16128-16167	103	15912-15951	151	15696-15735
8	8320- 8359	56	9088- 9127	104	8872- 8911	152	8656- 8695
9	9344- 9383	57	10112-10151	105	9896- 9935	153	9680- 9719
10	10368-10407	58	11136-11175	106	10920-10959	154	10704-10743
11	11392-11431	59	12160-12199	107	11944-11983	155	11728-11767
12	12416-12455	60	13184-13223	108	12968-13007	156	12752-12791
13	13440-13479	61	14208-14247	109	13992-14031	157	13776-13815
14	14464-14503	62	15232-15271	110	15016-15055	158	14800-14839
15	15488-15527	63	16256-16295	111	16040-16079	159	15824-15863
16	8448- 8487	64	8232- 8271	112	9000- 9039	160	8784- 8823
17	9472- 9511	65	9256- 9295	113	10024-10063	161	9808- 9847
18	10496-10535	66	10280-10319	114	11048-11087	162	10832-10871
19	11520-11559	67	11304-11343	115	12072-12111	163	11856-11895
20	12544-12583	68	12328-12367	116	13096-13135	164	12880-12919
21	13568-13607	69	13352-13391	117	14120-14159	165	13904-13943
22	14592-14631	70	14376-14415	118	15144-15183	166	14928-14967
23	15616-15655	71	15400-15439	119	16168-16207	167	15952-15991
24	8576- 8615	72	8360- 8399	120	9128- 9167	168	8912- 8951
25	9600- 9639	73	9384- 9423	121	10152-10191	169	9936- 9975
26	10624-10663	74	10408-10447	122	11176-11215	170	10960-10999
27	11648-11687	75	11432-11471	123	12200-12239	171	11984-12023
28	12672-12711	76	12456-12495	124	13224-13263	172	13008-13047
29	13696-13735	77	13480-13519	125	14248-14287	173	14032-14071
30	14720-14759	78	14504-14543	126	15272-15311	174	15056-15095
31	15744-15783	79	15528-15567	127	16296-16335	175	16080-16119
32	8704- 8743	80	8488- 8527	128	8272- 8311	176	9040- 9079
33	9728- 9767	81	9512- 9551	129	9296- 9335	177	10064-10103
34	10752-10791	82	10536-10575	130	10320-10359	178	11088-11127
35	11776-11815	83	11560-11599	131	11344-11383	179	12112-12151
36	12800-12839	84	12584-12623	132	12368-12407	180	13136-13175
37	13824-13863	85	13608-13647	133	13392-13431	181	14160-14199
38	14848-14887	86	14632-14671	134	14416-14455	182	15184-15223
39	15872-15911	87	15656-15695	135	15440-15479	183	16208-16247
40	8832- 8871	88	8616- 8655	136	8400- 8439	184	9168- 9207
41	9856- 9895	89	9640- 9679	137	9424- 9463	185	10192-10231
42	10880-10919	90	10664-10703	138	10448-10487	186	11216-11255
43	11904-11943	91	11688-11727	139	11472-11511	187	12240-12279
44	12928-12967	92	12712-12751	140	12496-12535	188	13264-13303
45	13952-13991	93	13736-13775	141	13520-13559	189	14288-14327
46	14976-15015	94	14760-14799	142	14544-14583	190	15312-15351
47	16000-16039	95	15784-15823	143	15568-15607	191	16336-16375

CHART #4

This chart is a complete list of all 192 Hi-res lines as they appear on the screen. The two adjacent figures are the first and last bytes of RAM used to control that Hi-res line.

Now let's learn how to make our own characters. Look at chart #6, figure #1. Let's suppose that the letter 'A' is the character that you want to put on the screen. First get some graph paper and mark off a box that is 7 squares wide and 8 squares high. Now you can use this box to devise the character that you want. Remember that on the screen, there will be boxes butted right up next to the one you are working on, on all four sides. So, if you do not want other characters to touch your character, you have to leave the 1st and 7th columns and the bottom row empty. If you want to build a figure that will use two or more of this size box, you will then want to use these columns and rows to be sure that your characters are together with no gaps between them.

Each row of squares in your box will be controlled by different RAM location, so it is necessary to calculate each one of them separately. Remember, we learned to count in binary at the beginning of the article? It was mentioned that the screen looks at the bits in reverse, and that is what we have to do here. The 1st column from the left is valued at 1, the second column valued at 2, the 3rd valued at 4, the 4th at 8, the 5th at 16, the 6th at 32 and the 7th at 64. Let's take the top row first. Only the 4th bit needs to be on, so we give that row the value of 8. On the 2nd row, the 3rd and 5th bits need to be on, so we add their values together (4+16=20) to give the 2nd row a value of 20. The 3rd row needs the 2nd and 6th bits turned on, so add their values together (2+32=34) to give the 3rd row a value of 34. The 4th, 6th and 7th rows are exactly like the 3rd row, so we can give them all the same value. The 5th row needs the 2nd, 3rd, 4th, 5th and 6th bits turned on, so add them together (2+4+8+16+32=62) to give the 5th row a value of 62.

FIGURE #1

8	7	6	5	4	3	2	1
128	64	32	16	8	4	2	1

ALWAYS 0	PG #1-01 PG #2-10		FINE LINE IN COARSE LINE 0-7			COARSE LINE IN 1/3RD SECTION 0-7			CONTROLS 1/3RD SECTION & COLUMN						
1	2	1	3	2	1	3	2	1	7	6	5	4	3	2	1
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

FIGURE #2

128	64	32	16	8	4	2	1								
32 767	16 384	8 192	4 096	2 048	1 024	512	256	128	64	32	16	8	4	2	1
8000	4000	2000	1000	800	400	200	100	80	40	20	10	8	4	2	1

FIGURE #3

CHART #5

Figure #1 is a simple chart to help you convert decimal to binary and back again. The chart is representative of an 8 bit number because there are 8 positions that can be used to make up the desired number. Think of each of these boxes as a separate switch that can be either on or off. The numbers in the top portion represents the bit number. The numbers in the bottom part represent the value of that bit. Let's imagine that the location of our chart is memory location 800 in our machine. If we were to POKE 800 with a 1, we would have just turned bit #1 on in that location. All the rest would be off. Now, suppose we wanted to turn the 2nd and 6th bits on. We would have to add the value of the 2nd and 6th bits together (2+32) for a total of 34. So, if we POKE 800 with a 34, we would have just turned on the 2nd and 6th bits. All the rest will be off. Naturally if you just want to store a number in a memory location, you could care less which bits are on or off, just as long as the number will still be there when you need to use it. However, when putting a Hi-res picture on the screen, it becomes very important which bits are on and how to control them at your will.

Figure 2 shows the 16 bits representing the address of the Hi-res pages. The top row explains what those bits within the address control. The top number in the bottom row is the bit number within the range of the above explanation. The number on the bottom of the bottom row is the bit number of the entire 16 bit address.

Figure #3 shows the values of the bits in the entire 16 bit address. It takes 2 bytes to make a 16 bit address, so the top row represents the bits as they would be counted in the second byte. The middle row shows the value of the bits in decimal. The bottom row shows the value of the bits in hexadecimal.

Now that we have assigned each row a value, we need to put it into the program. Look at the program listing at the end of this article and I'll show you how I did it there. First look at line # 193. 193 is ASCII for the 'A', so I put it on line #193 to make it easy to access by using the ASCII number to calculate where to go. Now, look at the data on that line. It says A=8, which is the value of row #1. Then, it says B=20,which is the value of row#2. C=34, which is the value of row #3. Then D=C, F=C and G=C because we want them all to be the same. We also have E=64, which is the value of the 5th line. The only row we did not do is row 8 because in text all row 8's are = to 0, which is a blank line. If you want to use that last row in your own characters, you just have to give it a value and put it in. Now look at line #10 in the program. It has the formula to find the location using the X/Y coordinates, and then a series of pokes. The first POKE is into location L with the value that we gave the 1st row. The second POKE is into location L+1024. If you remember back when we were discussing the formatting of the screen, we found that in going from one Hi-res line to the next, the address would increase by 1024 for 8 lines. That is why each of the subsequent POKEs on line 10 will increase by 1024 each time. Notice the last POKE uses the value of 0. That is for the bottom row which is blank. If you are using the bottom row in your characters, you will have to POKE its value, instead of 1, here.

One last thing to mention before closing, is the fact that you may want to put your characters someplace besides exactly within the boxes mentioned. Take a look at figure #2 on chart #6. This represents a graph line that may have been plotted on the screen. Now you want to put a character out of the normal boundaries. You have two problems. #1 You have to have a formula that will shift the character and calculate the new values from those received from the data tables. #2 On the bottom left

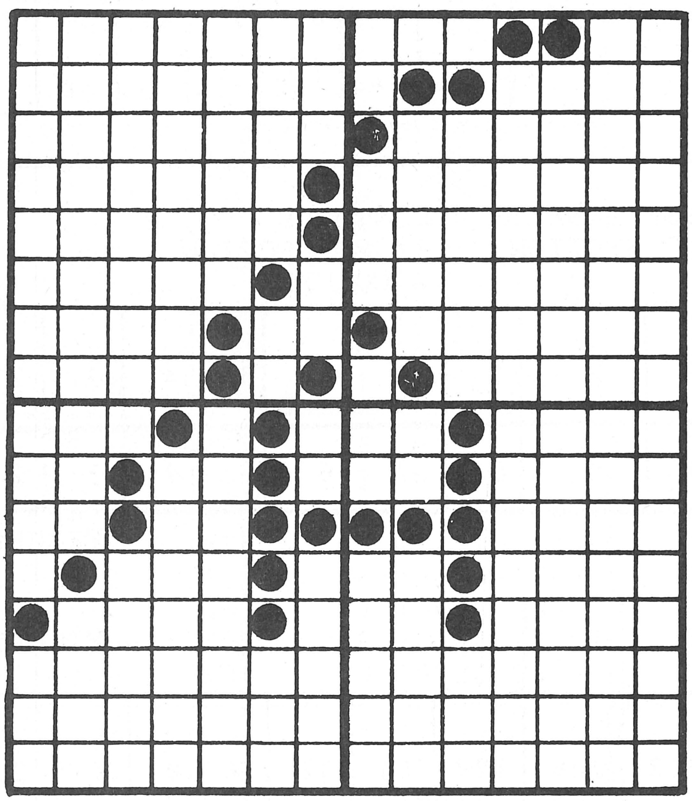


FIGURE # 2

CHART #6

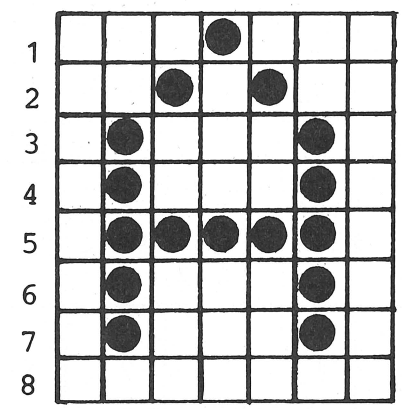


FIGURE #1

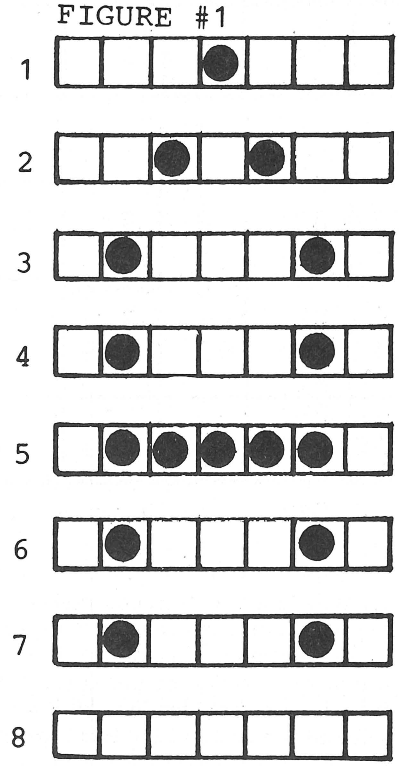


Figure #1 shows the letter 'A' as it would appear on the screen in Hi-res. Notice that the 1st and last columns are not used. The dots as they appear on the screen are so close together sideways that it takes 2 spaces to separate the characters. The dots are already far enough apart up and down so that only one space is necessary to separate them. Each row of the character uses a different byte of RAM in Hi-res, unlike the one character to a byte in text. Therefore, you see below the 'A', each row is separated from the rest. Each box in the row is representative of the 7 bits used in Hi-res to control the screen. By using information on chart #5 we can figure out what the value of each row of the 'A' is. Remember though, the Hi-res screen looks at the bits in reverse order, so that bit #1 is on the left instead of on the right. Try to calculate what you think the value of each row is before reading further. The values of the rows are as follows:

Row # 1=8, # 2=20, # 3=34, # 4=34, # 5=62, # 6=34, # 7=34 and # 8=0.

Figure #2 shows 4 areas, each the size of a normal character. The area could have come from anywhere on the screen. It depicts a portion of a graph with the letter 'A' beside it. You would not normally put a letter this close to a graph because the letter actually touches it. This is one way to demonstrate that sometimes you may want a letter to be in a place other than the normal squares where they usually go. I do not have the time or space to go into detail, however, there are some clues in the text of this article on how to do this.

box, you see that it is necessary to add the values of the graph to the values of your character. I haven't got the time or room here to give you the necessary information to do this. However I will give you some clues. 1. To shift any row to the left or right, you just divide or multiply by 2 as many times as you want to shift. 2. To get the other side of the character that you shifted out of the box, you have to shift the same data in the opposite direction for 7 minus the number of times you shifted the first time. 3. If you only POKE the values of the character in, you may lose the graph if it is within the same area.

I would appreciate any comments or constructive criticism. It is very difficult to try to put this type of information in very basic terms. I sometimes forget to mention something that I may take for granted, that a beginner may not know, and the fact that I am not a writer makes it very difficult to find the proper words and still stay within the allotted space.

THE PROGRAM

This program will print text characters in Hi-res. Line #'s 160 through 223 are data lines. They are the lines that have the information for the program to use to put the characters on the screen. The line number that the data is on is also the "Apple" ASCII value of the character that the data will create. For example, ASCII for the 'A' is 193, so you will find the data for the letter 'A' on line #193. Now, let's analyze this program one line at a time.

Line 100 sets up the routine to clear Hi-res screen #1. The CALL 468 is a built in routine in the APPLE to move data from one place to another in RAM. The first POKE puts a blank spot in the first screen location. The later POKEs set the pointers up to move that blank space to all the other screen places.

Line 110 sets up the machine to look at Hi-res page #1, then goes to line 1000.

Line 120 prints the character on the screen using the formula to figure out just where by using the X-Y coordinates. The 8 POKEs that follow the formula are the actual commands to put the data from the data line on the screen. Notice that each POKE is 1024 higher than the previous one. That's because each Hi-res line down is 1024 bytes higher than the one above, within that 8 line area.

Line 1000 gets the data from line 223 which has the data for the '_ '. It then goes to line 120 to print it on the screen. The '_' is what I used for the prompt sign.

Line 1010 strobes the keyboard for an input of an ASCII character. If any key but the back arrow (ASCII 136) is hit, the program will go to 1020. If it is hit then GOSUB 160 to get the data for a blank space. GOSUB 120 to print this space where the '—' was and decrement X.

Line 1015 checks to see if you are already on the left edge of the screen. If you are, it will then decrement the Y coordinates by one and increment the X coordinates by 40. This will put you on the line above, all the way to the right of the screen. It then checks to see if you are already at the top of the screen. If you are, it will then increase the Y by 24, which will put you at the bottom of the screen. GOTO 1000.

Line 1020 checks for a carriage return (ASCII 141). If not, then GOTO 1030. If yes, then increment the Y coordinates by 1 to move you down 1 line and then set the X coordinates to 0 for the far left column. It will now check to see if you are already at the bottom of the screen and if so, set Y to 0 to put you at the top of the screen.

Line 1030 checks to be sure that any remaining characters are a legitimate character (not a control character). If not good then go back to get another character on line 1010. If OK then GOSUB Z (Z=the negative ASCII value of the character) and get the data for the character. then GOSUB 120 to print the character.

Line 1050 increments X to the next column. If already on column. If already on column 39 then increment Y by 1 and decrement X by 40. If Y is already at the bottom, then decrement Y by 24. GOTO 1000.

```
10 REM CREATING HI-RES CHARACTERS
20 REM        BY  LOY SPURLOCK
30 REM       THE COMPUTER FORUM
100 POKE 8192,0: POKE 60,0: POKE
    61,32: POKE 62,255: POKE 63
    ,63: POKE 66,1: POKE 67,32:
    CALL -468
110 POKE -16297,0: POKE -16302,
    0: POKE -16304,0: GOTO 1000
120 L=8192+Y MOD 8*128+X+Y/8*40
    : POKE L,A: POKE L+1024,B: POKE
    L+2048,C: POKE L+3072,D: POKE
    L+4096,E: POKE L+5120,F: POKE
    L+6144,G: POKE L+7160,0: RETURN
160 A=0:B=A:C=A:D=A:E=A:F=A:G=A: RETURN
161 A=8:B=A:C=A:D=A:E=A:F=0:G=A: RETURN
162 A=20:B=A:C=A:D=0:E=D:F=D:G=D: RETURN
163 A=20:B=A:C=62:D=A:E=A:F=A:G=A: RETURN
```

```
164 A=8:B=60:C=10:D=28:E=40:F=30:G=A: RETURN
165 A=6:B=38:C=16:D=8:E=4:F=50:G=48: RETURN
166 A=4:B=10:C=B:D=A:E=42:F=18:G=44: RETURN
167 A=8:B=A:C=A:D=0:E=D:F=D:G=D: RETURN
168 A=8:B=4:C=2:D=C:E=C:F=B:G=A: RETURN
169 A=8:B=16:C=32:D=C:E=C:F=B:G=A: RETURN
170 A=8:B=42:C=28:D=A:E=C:F=B:G=A: RETURN
171 A=0:B=8:C=B:D=62:E=B:F=B:G=A: RETURN
172 A=0:B=A:C=A:D=A:E=8:F=E:G=4: RETURN
173 A=0:B=A:C=A:D=62:E=A:F=A:G=A: RETURN
174 A=0:B=A:C=A:D=A:E=A:F=A:G=8: RETURN
175 A=0:B=32:C=16:D=8:E=4:F=2:G=A: RETURN
176 A=28:B=34:C=50:D=42:E=38:F=B:G=A: RETURN
177 A=8:B=12:C=A:D=A:E=A:F=A:G=28: RETURN
178 A=28:B=34:C=32:D=24:E=4:F=2:G=62: RETURN
179 A=62:B=32:C=16:D=24:E=B:F=34:G=A: RETURN
180 A=16:B=24:C=20:D=18:E=62:F=A:G=A: RETURN
181 A=62:B=2:C=30:D=32:E=D:F=34:G=28: RETURN
182 A=56:B=4:C=2:D=30:E=34:F=E:G=28: RETURN
183 A=62:B=32:C=16:D=8:E=4:F=E:G=E: RETURN
184 A=28:B=34:C=B:D=A:E=B:F=B:G=A: RETURN
185 A=28:B=34:C=B:D=60:E=32:F=16:G=14: RETURN
186 A=0:B=A:C=8:D=A:E=C:F=A:G=A: RETURN
187 A=0:B=A:C=8:D=A:E=C:F=C:G=4: RETURN
188 A=16:B=8:C=4:D=2:E=C:F=B:G=A: RETURN
189 A=0:B=A:C=62:D=A:E=C:F=A:G=A: RETURN
190 A=4:B=8:C=16:D=32:E=C:F=B:G=A: RETURN
191 A=28:B=34:C=16:D=8:E=D:F=0:G=D: RETURN
192 A=28:B=34:C=42:D=58:E=26:F=2:G=60: RETURN
193 A=8:B=20:C=34:D=C:E=62:F=C:G=C: RETURN
194 A=30:B=34:C=B:D=A:E=B:F=B:G=A: RETURN
195 A=28:B=34:C=2:D=C:E=C:F=B:G=A: RETURN
196 A=30:B=34:C=B:D=B:E=B:F=B:G=A: RETURN
197 A=62:B=2:C=B:D=30:E=B:F=B:G=A: RETURN
198 A=62:B=2:C=B:D=30:E=B:F=B:G=B: RETURN
199 A=60:B=2:C=B:D=B:E=50:F=34:G=A: RETURN
200 A=34:B=A:C=A:D=62:E=A:F=A:G=A: RETURN
201 A=28:B=8:C=B:D=B:E=B:F=B:G=A: RETURN
202 A=32:B=A:C=A:D=A:E=A:F=34:G=28: RETURN
203 A=34:B=18:C=10:D=6:E=C:F=B:G=A: RETURN
204 A=2:B=A:C=A:D=A:E=A:F=A:G=62: RETURN
205 A=34:B=54:C=42:D=C:E=A:F=A:G=A: RETURN
206 A=34:B=A:C=38:D=42:E=50:F=A:G=A: RETURN
207 A=28:B=34:C=B:D=B:E=B:F=B:G=A: RETURN
208 A=30:B=34:C=B:D=A:E=2:F=E:G=E: RETURN
209 A=28:B=34:C=B:D=B:E=42:F=18:G=44: RETURN
210 A=30:B=34:C=B:D=30:E=10:F=18:G=B: RETURN
211 A=28:B=34:C=2:D=A:E=32:F=B:G=A: RETURN
212 A=62:B=8:C=B:D=B:E=B:F=B:G=B: RETURN
213 A=34:B=A:C=A:D=A:E=A:F=A:G=28: RETURN
214 A=34:B=A:C=A:D=A:E=A:F=20:G=8: RETURN
215 A=34:B=A:C=A:D=42:E=D:F=54:G=A: RETURN
216 A=34:B=A:C=20:D=8:E=C:F=A:G=A: RETURN
217 A=34:B=A:C=20:D=8:E=D:F=D:G=D: RETURN
218 A=62:B=32:C=16:D=8:E=4:F=2:G=A: RETURN
221 A=62:B=48:C=B:D=B:E=B:F=B:G=A: RETURN
222 A=0:B=A:C=8:D=20:E=34:F=A:G=A: RETURN
223 A=0:B=A:C=A:D=A:E=A:F=A:G=28: RETURN
```

```
1000 GOSUB 223: GOSUB 120
1010 Z= PEEK (-16384): IF Z<128 THEN
     1010: POKE -16368,0: IF Z#136
      THEN 1020: GOSUB 160: GOSUB
     120:X=X-1
1015 IF X<0 THEN Y=Y-1: IF X<0 THEN
     X=X+40: IF Y<0 THEN Y=Y+24:
      GOTO 1000
1020 IF Z#141 THEN 1030: GOSUB 160
     : GOSUB 120:X=0:Y=Y+1: IF Y>
     23 THEN Y=0: GOTO 1000
1030 IF Z<160 THEN 1010: GOSUB Z:
      GOSUB 120
1050 X=X+1: IF X<40 THEN 1000:X=
     X-40:Y=Y+1: IF Y<24 THEN 1000
     :Y=Y-24: GOTO 1000
```

EDITOR'S NOTE

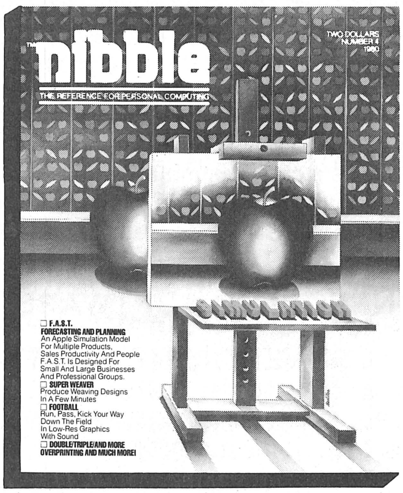
Readers may be interested in expanding Loy Spurlock's program to include lower case letters by defining their own shapes for the balance of the ASCII character set, decimal 225-254.

A lower case letter may be printed by first hitting the "escape" key, letting the program find it and set a flag (LC), then going back and reading the keyboard to get another character, which will then be converted to its lower case ASCII value by adding 32 to the original ASCII value.

An even more sophisticated modification may be attempted by setting "escape" to signify a single *UPPER* case character, two "escapes" as an upper case shift lock, with a single "escape" to unlock.

```
]LIST

5  REM       " COLOR TWENTYONE "

        BY  DARRELL ALDRICH

10  GR : HGR : HOME : PRINT "TWEN
     TY-ONE COLORS"
20  DATA  GREEN,VIOLET,WHITE,BLAC
     K,ORANGE,BLUE
30  FOR I = 1 TO 6: READ A$(I):
     NEXT

40  FOR A = 1 TO 6: FOR B = A TO
     6
50  VTAB 23: PRINT A$(B)"-"A$(A)"
     "
60  FOR C = 29 TO 119 STEP 2
70  HCOLOR= A: HPLOT 40,C TO 240,
     C
80  HCOLOR= B: HPLOT 40,C + 1 TO
     240,C + 1
90  NEXT C,B,A: GOTO 40
```

"NIBBLE® IS TERRIFIC" (For Your Apple)



NIBBLE IS: *The Reference for Apple computing!*

NIBBLE IS: One of the Fastest Growing new Magazines in the Personal Computing Field.

NIBBLE IS: Providing Comprehensive, Useful and Instructive Programs for the Home, Small Business, and Entertainment.

NIBBLE IS: A Reference to Graphics, Games, Systems Programming Tips, Product News and Reviews, Hardware Construction Projects, and a host of other features.

NIBBLE IS: A magazine suitable for both the Beginner and the Advanced Programmer.

Each issue of NIBBLE features significant new Programs of Commercial Quality. Here's what some of our Readers say:

– *"Certainly the best magazine on the Apple II"*
– *"Programs remarkably easy to enter"*
– *"Stimulating and Informative; So much so that this is the first computer magazine I've subscribed to!"*
– *"Impressed with the quality and content."*
– *"NIBBLE IS TERRIFIC!"*

In coming issues, look for:

☐ Numeric Keypad Construction Lab ☐ Assembly Language Programming Column
☐ Pascal Programming Column ☐ Data Base Programs for Home and Business
☐ Personal Investment Analysis ☐ Electronic Secretary for Time Management
☐ The GIZMO Business Simulation Game

And many many more!

NIBBLE is focused completely on the Apple Computer systems.

Buy NIBBLE through your local Apple Dealer or subscribe now with the coupon below.

Try a NIBBLE!

nibble No. 4

Box 325, Lincoln, MA. 01773 (617) 259-9710

I'll try nibble!
Enclosed is my $15 (for one year).
☐ **check** ☐ **money order**

(Please allow 4 to 6 weeks for delivery of 1st issue)
BACK ISSUES of NIBBLE are available for $2.00 + .50 postage and handling.

Name ______________________
Address ______________________
City ______________________
State ______________ Zip ______________

NOTE:
First Class or Air Mail is required for all APO, FPO and all foreign addresses with the following additional amounts.
— USA, Canada, Mexico, APO, FPO $7.50
— Central and South America $9.00
— Europe $12.00
— Asia and elsewhere $15.00

INTERNATIONAL APPLE CORE MEMBER CLUB ROSTER

A description of the International Apple Core and many of its functions may be found elsewhere in this issue. If you have already read that description, then you know IAC is composed of more than 140 Apple user groups representing over 12,000 individuals.

User groups serve their members in many ways, but in general, their purpose is to assist their members by providing product reviews, programming hints and instructional material. The IAC member clubs in addition, have access to the IAC APNOTES, covering a multitude of modifications and improvements, and information about existing products and software. These notes are made available to the member clubs at no cost, above and beyond the nominal membership fee, and may be freely reprinted in club newsletters or accessed through the club library.

This roster of member clubs is directed primarily at APPLE ORCHARD readers who either do not currently belong to any club, or are looking for additional sources of information. The roster is arranged alphabetically by state and foreign country so the reader may look for his/her own state and find a local user group. In addition, those groups that fall into one or more of the following categories:

1. membership in the several hundreds
2. national or international in nature
3. publish an above average newsletter

are flagged by an asterisk (*). It is suggested that readers may wish to also contact these groups and request a sample newsletter. Some of the clubs make no charge for this service, but we recommend enclosing a check for two dollars to cover their postage and handling costs.

Dig in!

ALABAMA

APPLE CORPS OF BIRMINGHAM
2931 Pahokee Trace
Birmingham, AL 35243
Phone - 205-967-4261

NEWTON'S TREE APPLE USER GROUP
3714 Lakewood Circle
Huntsville, AL 35811
Phone - 205-852-0537

ARKANSAS

LITTLE ROCK APPLE ADDICTS
P.O. Box 55215 Hillcrest Sta.
Little Rock, AR 72205
Phone - 501-568-5059

ARIZONA

ADAM II
P.O. Box 34056
Phoenix, AZ 85206
Phone - 602-248-4595

CALIFORNIA

APPLE FOR THE TEACHER
5848 Riddio St.
Cirtus Heights, CA 95610
Phone - 916-961-7776

APPLE SAC
8074 Ruthwood Wy.
Orangeville, CA 95662
Phone -916-381-4166

*SAN FRANCISCO APPLE CORE
3673 Bassett Ct.
So. San Francisco, CA 94080
Phone - 415-878-5382

*ABACUS USER GRP
2850 Jennifer Dr.
Castro Valley, CA 94546
Phone - 415-538-2431

MIDWAY COMPUTER CLUB
506 Ridgewood Dr.
Vacaville, CA 95688
Phone - 707-448-8430

SILICON APPLE PROGRAMMERS SOC.
18138 Bancroft Ave.
Monte Sereno, CA 95030
Phone - 408-354-6120

L.A. APPLE USERS GROUP
9513 Hindry Pl.
Los Angeles, CA 90045
Phone - 213-649-1428

SANTA CRUZ APPLE GROUP
P.O. Box 1428
Santa Cruz, CA 95061
Phone - 408-335-8750

APPLE P.I.E.
333 Escuela Ave. #316
Mountain View, CA 94040
Phone - 415-968-7851

S.P.A.C.E.
4546 El Camino Real
Los Altos, CA 94022
Phone - 415-493-8330

APPLE MUG
c/o Med Logic Systems
2030 East 4th St. #133
Santa Ana, CA 92705
Phone - 714-953-9151

THE 'PITS' OF SANTA BARBARA
3835 Connie Way
Santa Barbara, CA 93101
Phone - 815-969-5607

APPLE PEELERS
391 Shipley
Daly City, CA 94015
Phone - 415-878-0789

*ORIGINAL APPLE CORPS
12804 Magnolia
Chino, CA 91710
Phone -

HESEA APPLE COMPUTER CLUB
21111 Dolores #146
Carson, CA 90745
Phone - 213-549-9664

APPLE JACKS
3681 Cranford Ave. #44
Riverside, CA 92507
Phone - 714-886-6838

APPLE CREEK
1815 Ygnacio Valley Rd.
Walnut Creek, CA 94598
Phone - 935-6502

LERC ACES
P.O. BOX 551
Burbank, CA 91520
Phone - 213-899-2323

APPLE PI
c/o Computerland-Marion Clarke
171E. Thousand Oaks Blvd. - Ste 104
Thousand Oaks, CA 91360
Phone - 805-495-3554

JPL COMPUTER CLUB/JPL APPLE CLUB
24575 Spartan St.
Mission Viejo, CA 92691
Phone - 213-354-7009

APPLEPICKERS
P.O. Box 4208
Santa Rosa, CA 95402
Phone - 707-544-4783

HFEA APPLE COMPUTER USERS GROUP
417 Meadowbrook Pl.
Anaheim, CA 92801
Phone - 714-776-6384

COLORADO

*APPLE PI USERS GROUP
P.O. Box 17467
Denver, Co 80217
Phone - 303-355-2379

CONNECTICUT

APPLE USERS OF WESTPORT
1439 Post Road East
Westport, CT 06880
Phone - 203-227-6854

APPLELIST
55 Pardee Place
New Haven, CT 06515
Phone -

NEW LONDON APPLE USERS GROUP
130 Jefferson Ave.
New London, CT 06320
Phone - 203-447-1079

D.C.

*WASHINGTON APPLE PI
P.O. Box 34511
Washington, D.C. 20034
Phone - 202-332-9102

DELAWARE

GRAPE
P.O. Box 8904
Newark, DE 19711
Phone - 302-738-6365

FLORIDA

A.C.E.S.
P.O. Box 9222
Coral Springs, FL 33065
Phone - 305-941-7252

MAUG
2300 N.W. 135 Street
Miami, FL 33167
Phone - 305-595-8728

SCAT
21 Clearwater Mall
Clearwater, FL 33516
Phone - 813-961-5705

APPLE USERS CORE
307 Tarpon Rd.,
Mary Esther, FL 32569
Phone - 581-0002

SUN COAST COMPUTER ASSN
P.O. Box 15294 Southgate P.O.
Sarasota, FL 33579
Phone - 813-485-2564

HAWAII

HONOLULU APPLE USERS SOC.
98-1451-A Kaahumanu St.
Aiea, HI 96701
Phone - 808-261-3733

IOWA

THE GREEN APPLES
4417 N. Zircon Ln. Lot 129
Cedar Falls, IA. 50613
Phone - 319-268-0572

GLITCH KICKERS COMPUTER CLUB
3711 Douglas
Des Moines, IA 50310
Phone - 515-265-6266

IOWA CITY APPLE USERS GROUP
134 Ravencrest Dr.
Iowa City, IA 52240
Phone - 319-353-3170

IDAHO

A.B.U.G.
1505 Ressigue
Boise, ID 83702
Phone - 208-345-7149

P.I.N.E. APPLES
Alameda Plaza
Pocatello, ID 83201
Phone - 208-232-1960

ILLINOIS

*NW SUBURBAN APPLE USERS
1300 S. Elmhurst Rd.
Mt. Prospect, IL 60056
Phone - 312-593-2709

C.A.C.H.E.
359 Lawton Rd.
Riverside, IL 60546
Phone - 312-447-6267

APPLE PI COMPUTER CLUB
11630 S. Nagle Ave.
Worth, IL 60482
Phone - 312-448-6548

DUPAGE APPLE USER'S GROUP
10 S 592 Windjammer
Naperville, IL 60540
Phone - 312-420-8505

COMSAT
12 Crossroads Plaza
Granite City, IL 62040
Phone - 618-452-1860

INDIANA

THE APPLE PICKERS
1742 E. 52nd St.
Indianapolis, IN 46205
Phone - 317-251-5181

KANSAS

APPLEBUTTER
10049 Santa Fe Dr.
Overland Park, KS 66212
Phone - 913-884-8529

KENTUCKY

L.A.U.G.H.S.
8207 Pipilo
Louisville, Kentucky 40222
Phone - 502-426-3815

LOUISIANA

B.R.A.N.C.H.
324 W. Parker Blvd. #35
Baton Rouge, LA 70808
Phone -504-766-62265

CRESCENT CITY APPLE CORE
72 Old Hickory Ave.
Chalmette, LA 70043
Phone - 504-246-8438

CENLA APPLE
Box 1564,
England AFB, LA 71301
Phone -

MASSACHUSETTS

*N.E.A.T.
25 Emerson St.
Medford, MA 02155
Phone - 603-742-3703

APPLESEED
17 Saxon Rd.
Worcester, MA 01602
Phone - 607-755-2126

APPLE CORE OF BERKSHIRE COUNTY
32 Deborah Ave.
Pittsfield, MA 01201
Phone - 413-442-4759

APPLE USERS - BOSTON COMPUTER SOC.
P.O. Box 59
Rockport, MA 01966
Phone - 617-742-6100

APPLESAUCE
118 Brookhaven Dr.
East Longmeadow, MA 01028
Phone -

MARYLAND

MARYLAND APPLE CORPS
13A Allegheny Ave.
Towson, MD 21204
Phone - 301-256-3560

MICHIGAN

*MICHIGAN APPLE COMPUTER CLUB
P.O. Box 551
Madison Heights, MI 48071
Phone - 313-353-7648

K.A.C.U.S.
455 W. Michigan Ave.
Kalamazoo, MI 49007
Phone - 616-381-6476

GRAND RAPIDS APPLE
3268 Coach Lane #2A
Kentwood, MI 49508
Phone -

ANN ARBOR APPLE
P.O. Box M-1047
Ann Arbor, MI 48106
Phone -

MINNESOTA

MINI'APP'LES
13516 Grand Ave. S.
Burnsville, MN 55337
Phone - 612-890-5051

MISSOURI

APPLE JACKS
P.O. Box 8452
St. Louis, MO 63132
Phone - 314-567-0321

MICRO/PERSONAL COMPUTER CLUB
41 Roland Dr.
Ballwin, MO 63011
Phone - 314-227-6702

APPLE SQUIRES OF THE OZARKS
c/o Milton Rhoads
1904 E. Meadowmere
Springfield, MO 65804
Phone - 417-862-6500

APPLE BITS
1811 W. 43rd Street
Kansas City, MO 64111
Phone - 913-831-3199

NORTH CAROLINA

*CAROLINA APPLE CORE
P.O. Box 31424
Raleigh, NC 27622
Phone - 919-781-3755

GREEN APPLES
218 N. Elm St.
Greensboro, NC 27401
Phone - 919-275-2983

APP-LE-KATIONS
6525 Springfield Dr.
Charlotte, NC 28212
Phone - 704-554-8709

NEBRASKA

APPLESAUCE OF OMAHA
7435 Pacific St.
Omaha, NE 68124
Phone - 402-391-3737

NEW JERSEY

APPLE GROUP — NJ
1411 Greenwood Dr.
Piscataway, NJ 08854
Phone - 201-968-7498

SOUTHERN NJ APPLE USERS GROUP
106 Ashbrook Rd.
Cherry Hill, NJ 08034
Phone - 609-428-4429

NEW MEXICO

NEW MEXICO COMPUTER SOC.
6609 Orphelia Ave. N.E.
Albuquerque, NM 87109
Phone - 505-821-7418

MESILLA VALLEY ORCHARD
P.O. Box 114
Las Cruces, NM 88001
Phone - 505-526-4218

NEVADA

APPLE CORPS OF S. NEVADA
6325 Portola Rd.
Las Vegas, NV 89108
Phone - 702-647-6502

NEW YORK

SUFFOLK APPLE COMPUTER SOCIETY
64 Pinedale Rd.
Hauppauge, NY 11787
Phone - 516-360-0988

BIG APPLE USERS GROUP
55 A Locust Ave.
New Rochelle, NY 10801
Phone - 914-636-3417

APPLE POWER
21 Ridgedale Ave.
Farmingville, NY 11738
Phone - 516-248-8080

U.A.U.G. C/O UPSTATE COM.
629 French Rd.
New Hartford, NY 13413
Phone - 315-399-1139

MID HUDSON MICRO USERS
Imperial Plaza
Wappingers Falls, NY 12590
Phone - 914-297-1223

TSAUG APPLE CLUB
216 Cherry Rd.
Syracuse, NY 13219
Phone - 315-468-4262

APPLE C.I.D.E.R.
c/o Jim Berube
1435 Tudor Way
Victor, NY 14564
Phone - 716-924-7705

CAMS — APPLE USERS GROUP
Box 348 Ridge Rd. - R.D. #1
Scotia, NY 12302
Phone -

OHIO

APPLE—DAYTON
4819 Leafburrow Drive
Dayton, OH 45424
Phone -

NEO—APPLE CORE
1646 Higby Dr.
Stow, OH 44224
Phone - 216-261-5325

APPLE—SIDERS
5707 Chesapeake Way
Fairfield, OH 45014
Phone -

CENTRAL OHIO APPLE COMPUTER HOBBY
1357 Bernard Rd.
Columbus, OH 43227
Phone - 237-3380

OKLAHOMA

TULSA COMPUTER SOC—APPLE USERS
P.O. Box 1133
Tulsa, OK 74101
Phone - 918-835-3926

OKC APPLE USERS GROUP
3600 N.W. 39
Oklahoma City, OK 73112
Phone - 405-755-1260

OREGON

CORVALLIS APPLE CLUB
2013 N.W. Monroe
Corvallis, OR 97330
Phone - 503-757-7496

A.P.P.L.E. PORTLAND
1915 N.E. Couch
Portland, OR 97232
Phone - 503-283-8361

PENNSYLVANIA

APPLE USER OF PA
29 S. New Ardmore Ave.
Broomall, PA 19008
Phone - 215-356-6183

KEYSTONE APPLE CORE
4640 Carlisle Pike
Mechanicsburg, PA 17055
Phone - 717-652-6655

ARG
16 Laurel Lane
Glen Riddle, PA 19037
Phone -

SOUTH CAROLINA

SCAPPLE
1610 Longview Rd.
Mt. Pleasant, SC 29464
Phone - 803-554-9171

TENNESSEE

APPLE CORPS OF MEMPHIS
627 S. Mendenhall
Memphis, TN 38117
Phone - 901-761-4743

MUSIC CITY APPLE CORE
765 McMurray Dr. Apt. #04
Nashville, TN 37211
Phone - 615-331-2287

TEXAS

APPLE CORPS
P.O. Box 5537
Richardson, TX 75080
Phone - 214-324-2050

APPLESEED
6812 San Pedro
San Antonio, TX 78216
Phone - 512-657-3210

*HAAUGG
12502 Bexley
Houston, TX 77099
Phone - 713-469-5805

MICRO APPLE CORE
3920 Caruth Blvd.
Dallas, TX 75225
Phone - 691-6140

RIVER CITY APPLE CORPS
12404 Split Rail Parkway
Austin, TX 78750
Phone - 512-258-5486

A.I.D.E.
5700 Dixon
Amarillo, TX 79109
Phone - 806-352-3563

HOBBY COMPUTER INFORMATION EXCHANGE
6718 Spring Haven
San Antonio, TX 78249
Phone - 512-699-0146

UTAH

APPLE SLICE
P.O. Box 536
Bountiful, UT 84010
Phone - 292-4555

VIRGINIA

APPLE ORCHARD OF SE VIRGINIA
117 Cardinal Dr.
Hampton, VA 23664
Phone - 804-850-0626

APPLE T.A.R.T.
1706 Hanover Ave.
Richmond, VA 23220
Phone - 804-320-2260

APPLE WORMS
3307 Indigo Rd.
Chesapeake, VA 23325
Phone - 804-545-6404

WASHINGTON

*APPLE PUGETSOUND PROGRAM LIBRARY EXCHANGE
304 Main Ave. S. — Suite 300
Renton, WA 98055
Phone - 206-271-4514

KITSAP APPLE USERS GROUP
P.O. Box 1194
Silverdale, WA 98383
Phone -

WISCONSIN

ADAM & EVE APPLE GROUP
11 S. Hancock St.
Madison, WI 53703
Phone - 608-256-5306

WYOMING

THE APPLE NET
129 Park Ave. — Orchard Valley
Cheyenne, WY 82001
Phone - 307-632-4934

FOREIGN

APPLE USERS CLUB
8 Leemon St.
Condell Park-NSW, Australia 2200
Phone -

TAS APPLE USERS CLUB
422 Elizabeth St.
North Hobart, Tasmania
Australia 7000
Phone - 349616

APPLE USER CLUB AUSTRIA
P.O. Box 51
A-1181 Wien, Austria 0
Phone - 0043-222-652795

BRAZIL APPLE CLUBE
Rua Maestro Pena 90,
Porto Alegre, Brazil 90 000
Phone - 0512-23-0577

APPLE CORE CANADA
409 Queen St. W.
Toronto-Ontario, Canada M5U 2A5
Phone - 868-1315

CLUB APPLE DE MONTREAL
10;265 Hamelin
Montreal, Canada H2B 2E7
Phone -

APPLE B.C. USERS SOCIETY
#101-2044 West Third Ave.
Vancouver, B.C. Canada
Phone 604-731-7886

OEDIP — APPLE
8 Place Ste Opportune
Paris, France 75001
Phone - 1-5084621

APPLE CLUB ROEDINGHAUSEN
Wehmerhorstsr.110,
Roedinghausen, Germany D-4986
Phone -

HONG KONG APPLE
15 Cumberland Rd Rear Portion,
Kowloon Tong, Hong Kong
Phone -

APPLE USERS — DUBLIN
51 Lower Camden St.
Dublin 2, Ireland X
Phone - 751484

YEDA—NIKUV COMPUTERS
12 Karlibach St.
Tel-Aviv, Israel 0
Phone -

BAKED APPLE
Miyadaira Apts No. 1; 1575 Sugao
Takatsu-Ku, Kawasaki-Shi
Kangawa, Japan
Phone -

NZ GROUP OF APPLE USERS
90 Washington Av.
Brooklyn 2, New Zealand *
Phone - 894800

EMU
Box 3143
G.P.O.
Sydney 2001, NSW Australia
Phone -

APPLE BUGS
20-29 Banchi Yamate-Cho 1 Chome,
Suita City, Osaka
Japan
Phone -

APPLE — EDEN
10 Seton Terrace
Glasgow, Scotland G31 2HU
Phone - 041-554-3664

APPLE OF EUROPE
P.O. Box 4068
Hattingen, W. Germany D-4320
Phone - 02324/67472

APPLE CLUB FRANKFURT
Schweizer Str. 92,
Frankfurt/M.70, W. Germany D-6000
Phone - (0611)-61-45-12

APPLE CLUB ZAGREB
Ruzmarinka 3/11,
41000 Zagreb, Yugoslavia 0
Phone - X

Since this roster was prepared in August, more than 15 Apple user groups have joined the International Apple Core. Data on these groups will be printed in the Winter 1980-81 issue of The Apple Orchard.

INTERNATIONAL APPLE CORE TM

APPLE ORCHARD SUBSCRIPTIONS

P. O. BOX 2227 SEATTLE, WASHINGTON 98111, USA

The International Apple Core will make individual subscriptions to "The Apple Orchard" available commencing with Volume I, Number 2 to be published in September, 1980.

NAME ____________________

STREET ____________________

CITY ____________ STATE ____________ ZIP ____________

COUNTRY ____________________

Annual Subscription Rate: $10.00 per year
First Class Postage: $5.00 per year additional (required for Canada, Mexico, APO, and FPO addresses)
Overseas and other foreign air mail postage (required): $10.00 per year additional

TOTAL REMITTANCE ENCLOSED: $(USA) ____________

Make check or money order payable to "International Apple Core" and return with this form to:
Apple Orchard Subscriptions
P.O. Box 2227
Seattle, Washington, USA 98111

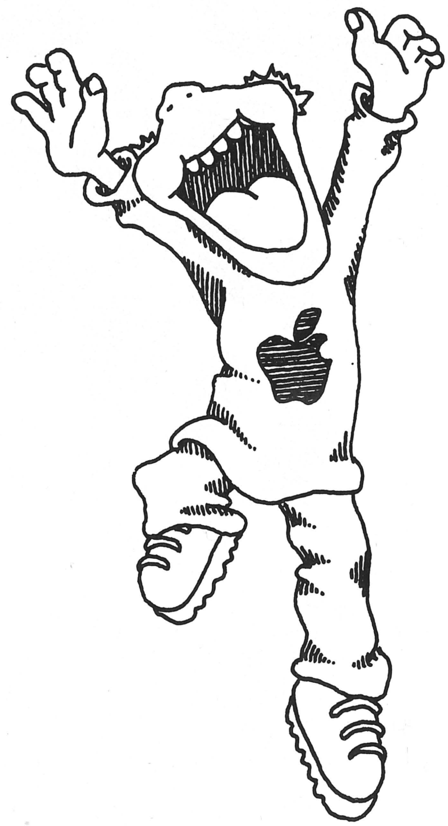
7/7/80

$$ WIN $$

Sirius Software announces
TWO CONTESTS

1. STAR CRUISER

Does something unusual when you score over 9999. Be the first to describe what happens and win $100 cash. Earliest postmarked correct entry wins. Send your entry with the face sheet from the package or other proof of purchase to Sirius Software.



2. E-Z DRAW

We're looking for the most creative use of E-Z Draw. Judging will be done by members of the Apple Sac Club based on originality, artistic merit, and practicality. First prize will be $100 cash and ten runners up will each receive their choice of either E-Z Draw, Starcruiser or both barrels.

Entries for both contests must be postmarked before 12/31/80. Star Cruiser entries need only send proof of purchase with a written description of what happened when they scored over 9999. E-Z Draw entries must be submitted on apple compatible diskette. Entries must include name, address & phone number. Diskettes will be returned to the contestants but we cannot guarantee safe return and we will not assume responsibility beyond the cost of a blank diskette for any work submitted. Winners of the contests will be contacted directly and the names of all winners will be announced in advertisements in this magazine after January 1, 1981. Contest void where prohibited by law. Employees or family of Sirius Software or their distributors are excluded from participation.

SIRIUS SOFTWARE is proud to announce that SYNERGISTIC SOFTWARE is now a distributor for these products. Contact your local dealer or call SYNERGISTIC SOFTWARE at (206) 641-1917.

SIRIUS SOFTWARE
1537 Howe Avenue #106
Sacramento, CA 95825
(916) 920-8981

SYNERGISTIC SOFTWARE
5221 120th Ave. S.E.
Bellevue, WA 98006
(206) 641-1917

contact

apple computer inc.

the user group newsletter

A Look Inside the Apple III

by Barry Yarkoni
Apple Computer, Inc.

By now, you have probably read about the Apple III, and possibly seen one at your local computer dealer. Most of the attention has been focused on the features of the Apple III, like improved graphics, increased memory capacity, and the processor design. Surprisingly little emphasis has been placed on one of the most advanced aspects of the Apple III; that is, its operating system, SOS.

What is SOS? How does it work? How do you use it? Let's take a look. SOS separates the Apple III programmer from the detailed aspects of the machine operation without sacrificing flexibility or capability. To do this, SOS operates at many different levels. We'll look at SOS from the top level, the external interfaces. Most applications programmers will never even see SOS. They will utilize its capabilities through various commands provided by high level languages. For those programmers, this will provide some understanding of what's going on underneath the languages.

The operating system presents five sets of interfaces to external programs, the File Management Interface, the Device Management Interface, the Memory Management Interface, the User Interrupt Interface, and the Utilities Interface.

SOS File Management Interface

The File Management interface provides the caller with a view of the system based on files. You can look at almost any device on the system as a file (whether or not it is actually a disk file). Files can contain various amounts of information. A disk file may contain many thousands of bytes of data in some known quantity, while an RS232 Port contains an unknown number, but an ordered set of bytes. Files can also be "acceptors" of information, as is the case with a printer. A file is simply a "container" for some quantity of information.

Two types of files are known to SOS, local files and directory files. SOS doesn't know (and doesn't care) what's in an local file. On the other hand, SOS is intimately familiar with the contents and meaning of a directory file. A directory file contains the names and locations of various local files in the system. At any particular time, the system may have many local files and also many directory files! But it may contain only one Master Directory. All the files in an Apple III system are organized in a "tree" structure, with the Master directory as the "root". The Master directory may contain local and directory files. Those directory files may contain other local and directory files, and so on.

A file name must be unique only within a directory file. That is, you can have a file with the same name in some other directory. How does the Apple III know which file you are talking about if two files have the same name? Simple. When you specify the file your are looking for, you "prefix" it with the names of the directories it is contained in.

Prefixes can get pretty long if you choose to build up a hierarchy of directories. For convenience, SOS will keep track of the prefix for the directory you are using at a particular time. So if you have a file named "X" contained in directory "Y" which is contained in directory "Z" you could specify the file as "Z/Y/X" or you can tell SOS that the prefix is "/Z/Y/" and simply refer to file "X". Sound complicated? Well, you'll just have to try it. You'll like it. All languages on the Apple III access files through the SOS file system, so you'll only need to learn it once. Some languages have minor differences from the general file specification described here. Pascal, for instance, uses ":" instead of "/".

So much for how to specify files, now what can you do with them? SOS can be asked to perform the following functions: CREATE either a file or a directory, DESTROY a file or a directory, RENAME a file, OPEN, CLOSE, READ, or WRITE to a file, plus calls to set the currrent position in a file or end of file. . . You can SET and READ PREFIX, ask for file INFO, and FLUSH. . . a file buffer, that is.

In short, the SOS file management interface provides a set of high level capabilities accessable directly via assembly language, or at a very high level through the languages on the Apple III. Each language will have its own convention for using these capabilities, but SOS allows a very high level of consistency.

Device Management Interface

SOS also provides a lower level access to devices. Although this level of control is more primitive, it is useful where more performance is required. At this level, the file concepts are stripped away, and the caller has immediate access to the device drivers for any devices configured into the operating system. As a matter of fact, the high level calls just described are implemented using these calls!

SOS is a configurable operating system. Each device on the system is supported by a set of routines that can be added to or removed from the operating system using the System Configuration Program supplied with SOS. Every SOS "boot" disk contains a file named SOS.DRIVERS. The driver contains the necessary information to control a particular kind of device.

All devices known to the operating system are controlled by a device handler and if necessary, an interrupt handler. The device handler for a device contains up to six distinct modules: OPEN, CLOSE, READ, WRITE, CONTROL, and STATUS. The modules associated with each device are directly callable by the SOS user through the Device Management System.

Memory Management Interface

Many have asked the inevitable question: "How does the Apple III address more than 64K bytes of memory with a 6502." The answer encompasses both hardware and software on the Apple III.

Apple III memory hardware provides two capabilities toward this end: bank switching and zero page switching. Thus, the amount of RAM memory in the system can extend beyond 64K bytes, and each code module or set of modules can have its own zero page.

Apple III has three bank switches, a main bank switch, an I/0 bank switch, and a ROM bank switch. The main switch selects the chunk of RAM that will sit in the 8K-40K address range. The I/0 switch can select between the hardware I/0 bank or a 4k RAM bank, and finally, the ROM bank switch selects between a 4k ROM bank or another 4k RAM bank.

SOS's role is to manage both the bank switching and zero page select in a manner that reduces the apparent complexity of the memory as the operating system user views it. This is provided in the form of a bookkeeping mechanism for allocating and deallocating chunks of memory. The REQUEST SEG system call asks SOS to request a specific portion of memory. If the user doesn't care where the memory is located, he can simply FIND SEG. SOS will search for a free chunk of memory of the specified size, and assign it. The user can later expand this chunk of memory. When a program has finished using its assigned memory, it assigns it back to the free space pool using the RELEASE call.

Several other calls are provided to obtain information on the allocation of memory. The memory management system may seem to provide more capability that most programmers need. However, by allocating memory through the operating system, a programmer will never have to worry about another module in the system interfacing with his memory space, almost as if the Apple III had "virtual" memory.

Interrupt Management

One of the most complex aspects of writing I/0 intensive programs is the handling of interrupts. SOS relieves the program of this difficult task by providing two types of interrupt management. The first is a peripheral interrupt. These interrupts are handled entirely by SOS and its constituent device drivers. These device drivers must be designed with great care since they are literally part of the operating system itself and because of danger of data overrun due to critical timing constraints. Fortunately, device drivers with thoroughly tested interrupt handlers are provided with SOS for all the built-in devices on the Apple III.

A new kind of user software interrupt mechanism has been provided to allow user programs some asynchronous capabilities without interfering with the ability of SOS to deal with true peripheral interrupts. These are, of course, at a lower priority level than the peripheral interrupts.

A user software interrupt usually corresponds to a similar peripheral interrupt. For example, a user software interrupt on the CONSOLE device would probably correspond to an interrupt on the Apple III keyboard.

Associated with each distinct user software interrupt is a portion of the user code called an interrupt handler which processes the interrupt. Each interrupt is also assigned a priority level from 0 to 255 used to determine the order that simultaneous interrupts are handled.

A number of SOS calls are provided to take advantage of the user interrupt facility. These calls allow the user to install and remove various interrupt handlers, set interrupt priority level, and exit from the interrupt routine back to the location where the interrupt occurred.

Utility Interface

The Apple III contains several handy features that can be neatly accessed through SOS calls. DATE allows the user to read (and set) the current year, month, day, date, day of the week, and time. The JOYSTICK call reads the current value of the analog and switch inputs for either of the two joystick ports.

Summary

Although this is by no means a complete description of SOS (or the features of the Apple III) I hope it has provided the reader with at least the flavor of the power and sophistication of the Apple III operating system. Complete documentation of the operating system and system calls will appear in the Apple III Technical Reference Manual, which will be available at your dealer in a few months. All the capabilities described here, however, are available to the programmer through the languages on the Apple III and the device drivers already provided with the Apple III.

ASCII, EBCDIC, and the Apple

by John Crossley
Apple Computer, Inc.

The following routine allows the Apple to selectively either convert its output to EBCDIC or convert incoming EBCDIC to ASCII. The converter resides at $800, so precautions must be taken to protect any Applesoft programs that are in memory or Integer Basic variables that have been defined.

Before loading an Applesoft program type:

POKE 103,1
POKE 104,9
POKE 2304,0

then load and run the program as usual.

Before running an Integer Basic program type

LOMEM:2304

The converter must be entered into memory before it can be used. The first time you must enter the monitor (CALL -155) and type in the information in the listing. The parameters to save the routine are:

Disk: **BSAVE EBCDIC CONVERTER,A$800,L$FF**
Tape: **800.8FFW** (from the monitor)

The next time you want to use the converter, you can load it with:

Disk: **BLOAD EBCDIC CONVERTER**
Tape: **800.8FFR** (from the monitor)

To actually use the routine,
PR#slot : **CALL 2048** and/or
IN#slot : **CALL 2075**

All subsequent transactions will be converted from ASCII to EBCDIC and back as required.

Another PR#slot or IN#slot will disable the converter and revert to normal ASCII.

```
*800.8FF

0800- 20 36 0B A0 01 B1 2A 8D
0808- 74 08 88 B1 2A 8D 73 08
0810- A9 46 85 36 A9 08 85 37
0818- 4C EA 03 20 36 08 A0 03
0820- B1 2A 8D 76 08 88 B1 2A
0828- 8D 75 08 A9 59 85 38 A9
0830- 08 85 39 4C EA 03 38 AD
0838- E7 03 E9 6E 85 2A AD E8
0840- 03 E9 00 85 2B 60 8E 72
0848- 08 29 7F AA BD 80 08 AE
0850- 72 08 6C 73 08 6C 75 08
0858- 60 20 55 08 8E 72 08 A2
0860- 7F DD 80 08 F0 06 CA 10
0868- F8 A2 3F 8A 09 80 AE 72
0870- 08 60 00 00 00 00 00 00
0878- 00 00 00 00 00 00 00 00
0880- 00 01 02 03 37 2D 2E 2F
0888- 16 05 25 0B 0C 0D 0E 0F
0890- 10 11 12 FF 3C 3D 32 26
0898- 18 19 3F 27 22 FF 35 FF
08A0- 40 5A 7F 23 5B 6C 50 7D
08A8- 4D 5D 5C 4E 6B 60 4B 61
08B0- F0 F1 F2 F3 F4 F5 F6 F7
08B8- F8 F9 7A 5E 4C 7E 6E 6F
08C0- 7C C1 C2 C3 C4 C5 C6 C7
08C8- C8 C9 D1 D2 D3 D4 D5 D6
08D0- D7 D8 D9 E2 E3 E4 E5 E6
08D8- E7 E8 E9 FF E0 FF FF 6D
08E0- FF 81 82 83 84 85 86 87
08E8- 88 89 91 92 93 94 95 96
08F0- 97 98 99 A2 A3 A4 A5 A6
08F8- A7 A8 A9 C0 6A D0 A1 07
```

Yes! There Is A Fix For APPEND In DOS 3.2 (and 3.2.1)!

The problem with APPEND in DOS 3.2 is that DOS doesn't write an End Of File marker on the disk when you close a file. DOS normally fills new sectors with EOF markers, so the newly APPENDed information usually has an EOF after the last character. However, when the last character of the file falls exactly at the end of a sector, DOS doesn't find a new sector to fill with EOF markers. The next time DOS does an APPEND it can't find the EOF marker and defaults back to the beginning of the file.

The fix is to write out an EOF marker before closing the file after each write. Here is a five byte routine that will supply an EOF. It can be moved to any address if you are already using 768 to 772.

```
10 LET D$ = CHR$( 4)
20 POKE 768,169
30 POKE 769,0
40 POKE 770,76
50 POKE 771,237
60 POKE 772,253
70 REM NOW TO USE IT!
80 PRINT D$;"APPEND FILE"
90 PRINT D$;"WRITE FILE"
100 PRINT "THIS IS DATA"
110 PRINT "SO IS THIS"
120 CALL 768: PRINT: REM THIS
   IS IT!
130 PRINT D$;"CLOSE FILE"
140 END
```

Using this method, one need never worry about APPEND overwriting the start of a file.

RFI: The F.C.C. and Your Apple

INTRODUCTION

What Interference Is

Radio and TV sets operate by "receiving" electro-magnetic waves. They all have antenna's which convert the electromagnetic wave into voltage which is in turn converted to sound, picture or both. The antenna may be built into the set or be remotely connected. (Very remote in the case of cable TV hook-ups.)

Interference, in the sense we'll use here, is a voltage appearing to the receiving set as though it were from the antenna (as it may well be) which is not intended to be sent by the broadcaster. Interference is always present to some extent. Interference from outer space, sunspots, random collisions of electrons, etc. is called "random noise" and on a TV set appears as "snow". Snow is a result of receiving electromagnetic waves which are not at all related to each other in time and wavelength. "Noise" is used to describe interference which is substantially "snow-like".

Noise from automotive ignition systems is another kind of interference. It is transmitted as an electromagnetic wave packet of short but high intensity, wherein an enormous number of wavelengths are represented. Because of this, it will be received everywhere in the spectrum, appearing as bright dots on almost all channels. This form of wide band interference was very quickly recognized when the TV set was introduced and subsequently automatic ignition systems were designed to reduce the radiated energy to acceptable levels.

Interference generated by computers is similar to both automotive ignition noise and snow in the sense of being very wide band. It is made up of short intense packets. The packets are not nearly as intense as ignition noise, but they occur much more frequently and the rates at which the packets are generated are related to each other. That is to say, there is a pattern in time to the packets. Consequently, the visual effect of computer generated interference is almost snow with squiggles and bars moving about. In general, the existence of patterns in interference makes it more objectionable than the purely random snow and for the same levels of interference, computer generated interference is much more obnoxious than snow.

General Comments re FCC

Briefly (and consequently at the risk of being inaccurate), we will comment on the role of the Federal Communication Commission. The FCC has among its many duties that of keeping the air waves pure. Like any environmental protection agency, it is asked to mediate conflicts between public interest groups when they arise: such as the conflict between computer users and TV watchers. Clearly, these groups have both rights and responsibilities. And despite the enormous technical and political complexities, the Commission is dealing with the problem.

The Commission has decided that some pollution from computers is unavoidable. In keeping with their tradition they believe that, since the computer is the new kid on the block, most of the burden shall fall upon the manufacturer and user of computers. In office, commercial and industrial environments, substantial pollution will be allowed since the TV set is not likely to be used and then only at substantial distances from the computer itself. In residential environments, the TV set is ubiquitous and likely to be closer to the computer and consequently, the allowable radiated pollution will be markedly lower.

The Commission also has taken the attitude that the residential computer user may very well interface with his own TV set. The user has the choice of turning off either the computer or the TV set. The possible conflicts between spouse or between grandpa and the kids is not the FCC's concern. But the computer user must not interfere with his neighbor's TV. In fact, the Commission puts the full burden on the residential computer user to avoid such "harmful" interference" right up to the point of ceasing to operate computer. This is so — regardless of whether or not the manufacturer's equipment complies with the applicable rules and regulations.

If you are interfering with your own TV set, it is possible that your neighbor is having problems as well. You should check. (the distance from the computer to your neighbor's antenna is a key parameter. Buying a troubled neighbor a new TV antenna may be a more palatable solution than restricting your computing times to the wee hours of the morning.

Whence Cometh the Radio Frequency Interference?

Computers today operate at radio frequencies and the bursts of voltages and currents that take place when information is latched in a memory IC (for example) have significant energies in the frequency range from 30 MHz to several hundred MHz. Since the voltages are relatively low, we'll be talking mostly about the currents, but this is for convenience and you should remember that both are inevitably present and both create a field and that field will radiate a wave.

An antenna is simply a device designed to radiate a wave. By pleasant symmetry, an antenna is equally good (or bad) whether radiating or receiving a wave. The TV antenna seen on roof-tops is, by and large, a pretty good device; it will receive (or radiate) a high percentage of the electrical energy delivered to it at the wavelength for which it is designed. Transmission lines are electrical devices intended not to radiate: that is, if the current in one conductor is exactly equal and opposite to the current in the other conductor (both spatially and in time) then the net radiated field is zero. A transmission line must be properly used (and loss free) for this to be true. The twin-lead leading from the roof-top antenna to your TV set is an example.

Rabbit ears are a mis-matched transmission line, more or less. The vagaries of rabbit ears are well known. Just keep in mind that mis-matched transmission lines will both radiate and receive — with surprising efficiency at times. A "mis-matched" transmission line is one that is improperly installed and thus violates the "equal and opposite current" criteria outlined above. A solitary wire suspended above the earth is also a transmission line with the return current coming back through the earth just as the current leaves thru the wire. These wires radiate a great deal. Remember that at the short wave lengths we're talking about, one does not need a direct connection to enable current flow. Any two conductors in space have capacitance between them which allows current to flow whenever there is a rapidly time-varying voltage between them. To get an intuitive picture of the transmission line radiating, just imagine the field produced by the current of first one of the conductors, then the other. Subtract the two fields and if the result is not zero, then radiation will take place.

The reason for discussing antennas is that we want to give our TV receiver the very best and our computer the worst we can arrange. In the real world, one seldom needs both — but it happens.

GETTING THE TV SIGNAL TO THE TV SET

When interference occurs, whether it's your set or your neighbor's, the first efforts should be to check the installed antenna system. First, to convice yourself that the interference is really arriving at the antenna terminals of the TV set, disconnect the antenna at the set and check the TV picture. You should see nothing but snow — good clean snow — on all the lower channels (2 through 13). TV sets manufactured these days have pretty good shielding and modestly effective line filters. It is only rarely that a local station will be so powerful as to sneak into a set without an antenna. Secondly, convince yourself that the observed interference is actually due to the computer (and/or accessories) by simply turning it on and off. Remember that there are other sources of airway pollution — arc welders, cash registers, blenders, hair dryers and even clock radios.

After you are sure that the interference is getting to the TV set antenna terminals, you must now examine the entire system by which the TV signal, converted to a tiny voltage by the TV antenna, is carried to the TV set. You should look for bad connectors, broken wires, impedence mis-matches, faulty power splitters etc... anything that degrades overall antenna performance.

On outdoor antennas, the weather and chimney fumes will convert metals into insulators. The connections will have to be scraped clean, reconnected and then varnished or otherwise coated to keep the elements away. Often, the wind will cause the lead-in wire to flex and, over a period of years, the wires will break inside the insulation and present you with an invisible but very bothersome open circuit.

Twin-lead rarely shorts because the wires are widely separated, but coax may, if connectors are improperly installed or the cable has been crushed. Careful visual inspection is useful. Coaxial cable will be terminated at both ends with "Balun Transformers". This is frequently mounted in a little box with a short piece of twin-lead coming out of one end, a coax connector at the other end. The antenna end must also have a Balun.

Unfortunately, there are many little gadgets sold that claim to improve TV reception. In fact, most are carelessly designed and constructed. Interference suppressors, band splitters, color enhancers, ghost suppressors etc..., very frequently deteriorate antenna performance, mis-match the lead-in transmission line or otherwise cause grief. It is good practice to remove all such devices when an RFI problem is under investigation. For simplicity, ignore installed lead-in systems and temporarily run 300 ohm twin-lead directly from the antenna to the TV set in question. Twist the twin-lead at the rate of one or two turns per foot. You can later back up and reinstall as neat a system as you like. It is imperative that there be no mysterious little boxes (installed by the previous owner) hidden in walls or attics.

An apartment or house will often have multiple TV antenna "outlets" installed when the dwelling was built. This is a problem from two standpoints. Each time the central antenna is "tapped", a power splitter is installed which cuts your signal in half. Not nice! In addition, your house is wired with all these "outlets" that become antennas, picking up interference and conducting it back through the splitters, right into your TV set. If "min-loss pads" have been used instead of real power splitters, then you are losing 75% through each one. Min-loss pads are valuable when there is a powerful antenna "distribution anplifier" that boosts the received signal up to a level where the loss in the pads can be tolerated. All too often, systems once designed for a distribution amplifier break down, the amplifier is removed and you are left with a passel of min-loss pads and almost no signal. In any case, first try a temporary direct connect system. Afterwards, try terminating every unused TV outlet with a 300 ohm resistor.

Antenna orientation is important. If a particular channel presents a problem, point the antenna in the direction of the station. Improvements of an order of magnitude are common. Most "log periodic" antennas have a very good "front to back ratio" and simply pointing the antenna away from the source of interference is very effective.

A very useful, but inconvenient, step in the analysis of an interference problem, is to carry a portable TV set right up to the TV receiving antenna and connect it directly to the antenna, via a foot or so of twin-lead. With all other leads removed, check for interference. If interfence persists, then the radiation from the computer will have to be reduced. If the interference has gone away, then the lead-in system will have to be fixed somehow.

If the interference is determined to be from the computer, then the following procedure should help in cutting it down. *NOTE: This procedure may void your warranty.*

Required parts (Apple Kit Part #652-0152)

1- Two 0.1 uf capacitors- Apple part #132-8101
2- Two solder lugs- Apple part #517-0009
3- Six ferrite beads (Toroids)- Apple part #159-0001

MOTHERBOARD REMOVAL

1- Power off.
2- Remove Apple lid and any peripheral card plugged in.
3- Turn Apple upside down and rest keyboard on protective foam pad.
4- Remove six flat-head screws from three outside edges of flat portion of Apple base. (See number 6 on figures page).
5- Remove four round-head screws and lock washers from front of base; (number 7).
6- Grasping both base and housing, turn Apple right side up.
7- Gently lift front of housing slightly off base and unplug keyboard connector from location A7 at front of motherboard; (number 1).
8- Lift housing off base and set aside.
9- Pinch sides of power supply plug at location K1. Release and lift it out; (number 2).
10-Unplug speaker connector at location B8 on motherboard; (number 3).
11-Remove 5/16 inch nut and lockwasher in middle of motherboard (do not forget to reinstall it later!); (number 4).
12-Push in on flanges with screwdriver, or needle-nose pliers, to release four stand-offs at corners of board and two stand-offs between I/0 connectors 4 and 5; (number 5). Lift board up and out. Lay gently to the side.

CAPACITOR INSTALLATION

1- Feed one end of each capacitor through small hole (from uplifted side) of each lug respectively.
2- Solder lead to lug and cut off excess.
3- Remove back left and right stand-offs by removing screw from underside.
4- Scrape paint off top side of base plate around mounting holes of above mentioned stand-offs.
5- Place lugs, uplifted side pointing up, one each between base plate and above mentioned stand-offs. reinstall standoffs.
6- Reinstall motherboard.
7- Scrape solder mask off motherboard next to back left and right mounting holes.
8- Solder other leads of capacitors to just scraped areas of motherboard and cut off excess leads.
9- Reinstall motherboard to base.
10- Add a ferrite bead to the keyboard cable before reinstalling it into location A7 on motherboard; (number 1).
11- Reassemble and test Apple.

FERRITE BEAD INSTALLATION

1- Power off.
2- Take one end of monitor cable and thread as many turns as possible through it, then reconnect it.
3- Repeat procedure at other end of cable.
4- Add a ferrite bead to any peripheral cable going out of the Apple. One or two turns should be adequate.

(continued on page 35)

***SYNTAX ERR

Corrections to "Applesoft Internal Entry Points", Apple Orchard Vol. I, No. 1

In HIRES GRAPHICS SUBROUTINES, entry addresses should be corrected as shown below.

ROUTINE	WAS	SHOULD BE
HGR2	F3D4	F3D8
HGR	F3DE	F3E2
HCLR	F3EE	F3F2
BKGND	F3F2	F3F6
HPOSN	F40D	F411
HPLOT	F453	F457
HLIN	F530	F53A

HPLOT: The entry conditions are:
Horizontal = Y,X
Vertical = A

Zero Page locations:
ROT $F9
SCALE $E7

Omitted under STRING UTILITIES:

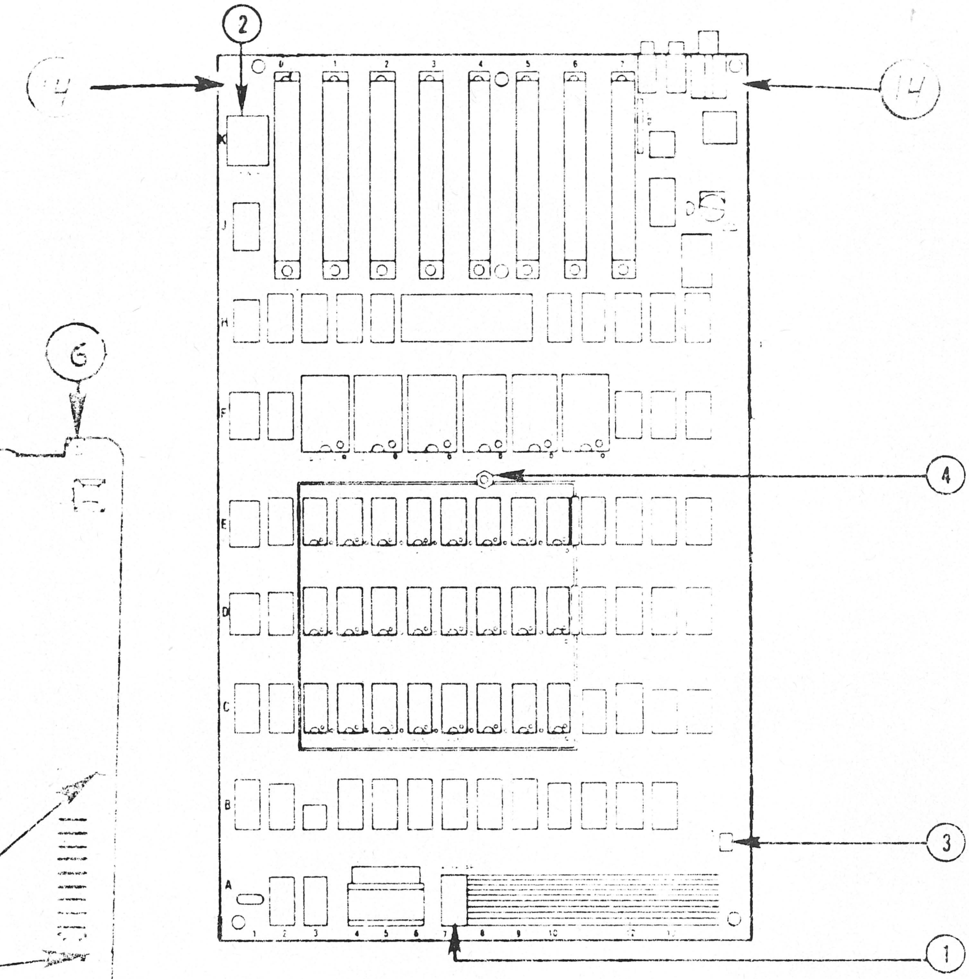
FREFAC E600 (58880)
Frees temporary descriptor pointed to by FAC.

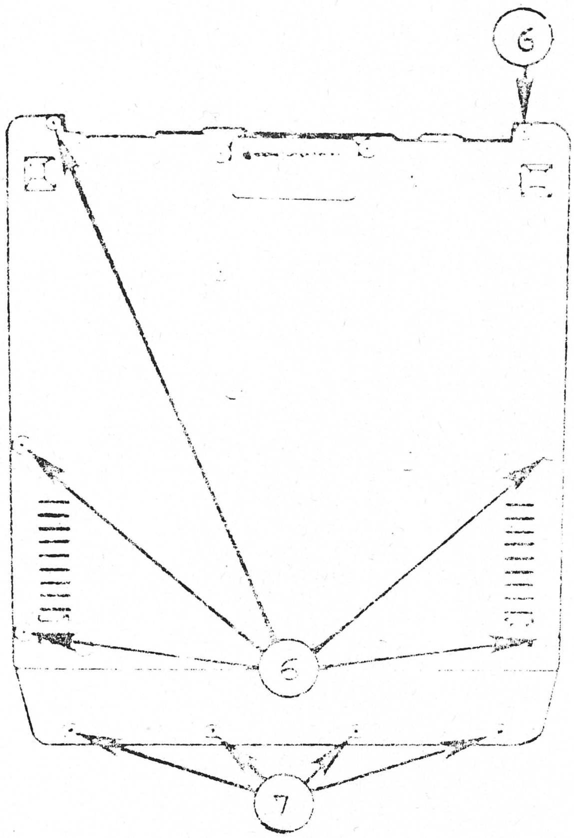
FLASH CARDS -Page 90
Line 8020 RETURN was omitted.

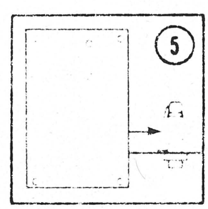
5- Power on and test Apple.

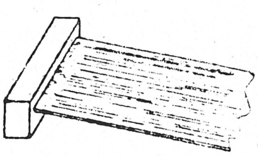
NOTE:

Number 8 through 13 on figures page show how to thread a cable through a ferrite bead.
The same method can be applied when threading keyboard cable.
Number 15 shows where the capacitors go on the mother-board.

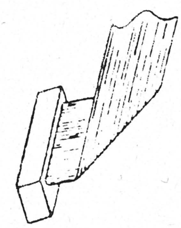




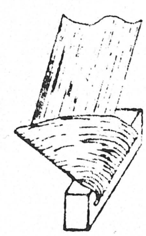




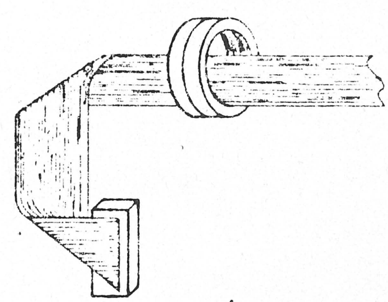
8



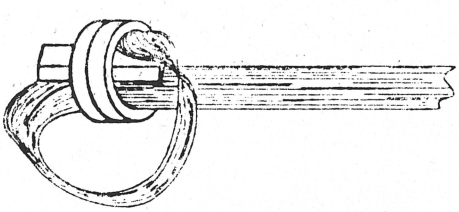
9



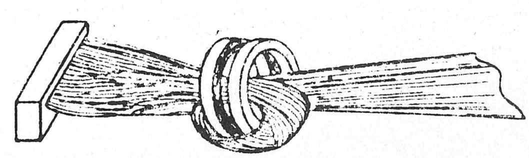
10



11



12



13


The man, the lig

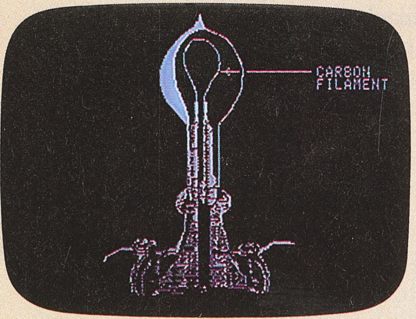
ht and the Apple.

If you could talk to Thomas Edison, he'd tell you what it was like to turn the lights on in 1879. You could tell him about some bright ideas of the 20th century… particularly, a technological phenomenon that can handle everything from solar heat control to lighting your home via voice command. The Apple personal computer.

Expand your own inventiveness with the always-expandable Apple.

Take a look inside your local computer store. There's a range of Apple systems for you… whether you want expansion capabilities of four or eight accessory slots… or memory expandable to 64K bytes or 128K bytes. With this kind of flexibility, the possibilities for creating your own computer system are endless.

Want to add an A to D conversion board? Apple makes it happen. Want to plug into time sharing, news and electronic mail services? Apple does it all. Because Apple is the most popular personal computer with the least complicated interface, over 100 companies supply peripherals for the Apple family… including an IEEE 488 bus for instant control.



With Apple, Edison could've written a program to determine why some filaments burned longer than others.

Disk drives, a tool kit and creativity in color.

Apple was one of the first to use disk drives for increased performance and application versatility. Today, our $5\frac{1}{4}''$ disk drive offers high density (143K bytes), high speed and low cost. No wonder this drive is the most popular on the market.

But now Apple goes one better with the DOS Tool Kit. A series of utility programs, it gives you the freedom to easily design 280h x 192v graphic displays in a palette of living color… depending on your choice of Apple system.

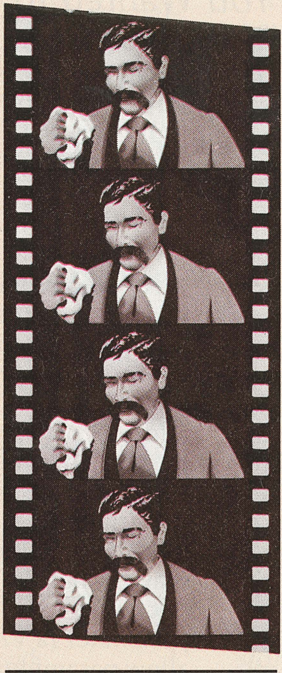
Edison was first with the movie camera and projector. Now, with Apple's DOS Tool Kit, you can be first to work wonders with colorful creative animation.

Imagine the broadest line of software programs ever.

Apple's broad line of peripherals is equalled only by the most extensive line of software you'll find in the personal computing world. Since more than 170 companies offer software for the Apple family, you can have one of the most impressive program libraries ever.

When you write your own programs, your Apple speaks creatively in BASIC, Pascal, FORTRAN, PILOT and 6502 assembly language. Use these languages to score a sonata. Apple will play back your musical masterpiece on its built-in speaker.

Edison listened to his voice on a revolutionary phonograph in the 1800s… now you can listen to the sounds of today with Apple's inventive family of personal computers.



Edison had the first movie camera… and Apple has the DOS Tool Kit that takes you into the colorful world of animation.

Where to find even more illuminating Apple experiences.

There's always something new being invented at Apple to set your imagination soaring. And there's always an expert to tell you all about it in detail. Your Apple dealer. If you already own an Apple, there's a whole future ahead to challenge man, mind and machine.

If you're considering a personal computer, stop by the computer store and compare. Apple's reliability, proven performance and recognized technological leadership will help you see the light. Don't let history pass you by. Visit your nearest Apple dealer, or call 800-538-9696. In California, 800-662-9238.

apple® computer

PASCAL OPERAND FORMATS
or, The Secret Life of a Variable

(or, Everything You Wanted to Know About Pascal Variables, But Couldn't Get Through To The Hotline To Ask)

by
Jo Kellner

The Apple hotline has received numerous calls about the internal structure of Pascal variables. This information can be very useful when sending data (especially complex data formats such as strings) to an assembly routine from a Pascal host program. This article describes a few of the more commonly used variable types. For a complete description of the more complex variables, including records and arrays, see pages 202 through 204 of the Apple Pascal reference manual.

Machine language (assembly) routines are commonly used when speed is critical, and when the code must access other assembly routines such as PROMs or I/O drivers which can't be re-assembled as part of the program. Also, most single-bit operations are much easier to do in assembly than in Pascal.

In the USCD Pascal system, it's a fairly simple matter to create short assembly programs which can be linked into a Pascal host program. In some cases, it may be sufficient to merely call the assembly routine; however, most routines require data in order to be useful. The means by which data is passed to or from these routines is called a "parameter".

A parameter is a temporary variable created by Pascal for the purpose of passing data to or from a subroutine. The term "formal parameter" implies that the address of the actual variable is passed to the subroutine as a parameter instead of its value.

Certain types of variables may be passed by value, but any variable may be passed by name by simply declaring it to be a formal parameter (a VAR). Pascal does not allow parameters of variable length (with the exception of certain sets and long parameters) to be passed on the CPU stack, since this could exceed the stack capacity and crash the operating system, so these parameters are automatically used as formal parameters. A good explanation of the various ways of passing parameters may be found in Peter Grogono's book, "Programming in Pascal".

Before delving into the details, let's define some terms and conventions which we'll use later on:

BIT = a binary digit (0 or 1). A bit is the smallest unit of information which can be stored in a computer.
NYBBLE = 4 bits (half a byte). A hexadecimal digit is one nybble (pronounced "nibble").
BYTE = 8 bits (2 nybbles). This is the unit of storage which the 6502 processor uses.
WORD = 2 bytes (16 bits). A word is the unit of information which Pascal uses.
LSB = least significant bit
MSB = most significant bit

(See Figure 1)

This diagram of memory structure will be used in describing the variable formats. Usually, when you write down a number, you write it from left to right. However, Pascal reads data from memory from right to left starting at the least significant byte.

INTEGERS

Integers, in UCSD Pascal, are whole numbers in the range of -32768 to +32767. They are stored in one word (2 bytes). Negative integers are represented in "two's complement", which means that they appear to have positive values (>32767). By subtracting this positive value from 65536, the negative integer is revealed. Similarly, large positive integers are stored as a complementary negative numbers (remember Integer Basic?). The sign bit (MSB) is 0 if positive, 1 if negative.

(see Figure 2)

Example: the number 3 is represented in binary as:

```
MSB                             LSB
0 0 0 0 0 0 0 0   0 0 0 0 0 0 1 1
```

However, -3 shows up as

```
MSB                             LSB
1 1 1 1 1 1 1 1   1 1 1 1 1 1 0 1
```

which also reads as 65533 (or 65536-3)!

Integers may be passed by value or as formal parameters.

REALS

Real numbers, in USCD Pascal, are floating point numbers in the range of +/- 1.17550E-38 to +/-3.402 82E+38. Real numbers use four bytes (2 words). The binary representation is similar to the proposed IEEE standard for floating numbers:

(See Figure 3, next page)

```
              decimal 65535                          0
          hexadecimal FFFF <.......memory.....> 0000 addresses
                       MSB                         LSB
```

FIGURE 1

```
  <.........byte.......>  <........byte......>
 15  14 . . . . . 8  7 . . . . . . . 0         <== 16 bits
 SIGN          INTEGER VALUE
```

FIGURE 2

```
  31     30 . . . . . .23      22 . . . . . .. . . 0      <== 32 bits
 SIGN           EXPONENT                 MANTISSA
```

FIGURE 3

"Mantissa" is the name given to the decimal portion of a number which is expressed in scientific (exponential) notation. The "exponent" indicates the power to which the mantissa is raised. In decimal, the number 3 x 10^2 can be seen as a mantissa of 3, an exponent of 2, in base 10 (decimal).

The sign bit refers to the sign of the mantissa, and is 0 if positive, 1 if negative. The exponent is "offset" by 127; that is, a value of 127 in the exponent field corresponds to an exponent of 0. Similarly, if the value is 1, the exponent is –127, and if the field is 254, the exponent is +127. A value of 0 indicates that the real number is 0.

The mantissa of the real number is stored in normalized format in bits 0-22. "Normalizing" a number means adjusting it so that the highest bit is significant (a 1). The exponent indicates how many times (and in which direction) the value was shifted during normalization.

Notice that the MSB of The Mantissa of any non-zero number which has been normalized is always a one. The number zero can be treated as a special case by simply setting the exponent to zero. So, to gain additional precision, the mantissa has an implied "1" which is not stored, resulting in a functional 24-bit mantissa, even though only 23 bits are actually used. This gives slightly more than 6 decimal places (single precision) accuracy.

To make this clearer, let's look at some examples:

(See Figure 4)

```
Real number = 1
  MSB 0            01111111          00000000000000000000000 LSB
  Exponent = 127 (2 ^ 0)       Mantissa = 1 (the implied 1 isn't stored)

Real number = –9.9
  MSB   1          10000010          00111100110011001100110   LSB
  Exponent = 130 (2 ^ 3)         Mantissa = 99000015
```

FIGURE 4

In the second example, the real number (in binary) appears as 1001.1110011 etc... During normalization, the decimal point is moved to the left 3 times (incrementing the exponent), and the most significant bit becomes implied. The sign bit is 1, indicating that the number is negative.

Real numbers may be passed by value, or may be defined as formal parameters and passed by address.

CHARACTERS

Characters, by ASCII definition, are simply integers in the range of 0 to 255. Characters take up one word of storage. The ASCII value of the character is stored in the least significant byte. The most significant byte is not used by Pascal and should be ignored.

```
15  . . . 8   7  . . . 0 < ==16 bits
  unused        ASCII code
```

FIGURE 5

EXAMPLE: the character "A" has an ASCII value of 65 (hexadecimal 41). Represented in binary, this would be:

```
MSB x x x x x x x x 0 1 0 0 0 0 0 1 LSB
 ........ not used .......    4 (hex)    1
```

Characters can be passed as either actual parameters (pass by value) or formal parameters (pass by address).

STRINGS

A string is a packed array of characters which can be from one to 256 bytes long. The first byte of a string always contains a number from 0 to 255 which indicates the length of the string. One character is stored per byte, and the string ends on a word boundary; that is, if the last character in the string is the first byte of a new word, the other byte of the word is also reserved, but is not used by the string.

Each character of the string can be accessed in a packed array of characters; however, you cannot access the length byte (the 0th element). Doing so will promptly generate the message: "Value Range Error".

EXAMPLE: The string "ABCD" would look like this:

(See Figure 6)

S[4]	S[3]	S[2]	S[1]	S[0]
MSB 01000100	01000011	01000010	01000001	00000100 LSB
"D"	"C"	"B"	"A"	4

FIGURE 6

Pascal always passes strings as formal parameters, since the length may vary.

POINTERS

Address pointers are UNSIGNED integers which occupy 1 word of storage. The format is the same as for integers, except that the values range from 0 to 65535.

EXAMPLE: The address of AN0 (one of the annunciator ports is hex C058 (49240 decimal)). This would be stored as:

```
MSB 1 1 0 0 0 0 0 0   0 1 0 1 1 0 0 0 LSB
  <-------> <-------> <-------> <------->
      C         0         5         8
```

FIGURE 7

Pointers, like integers, may be passed by value or by name (formal parameter).

LONG INTEGERS

Long integers are a special type of variable which was first defined at UCSD as part of their extensions to the Pascal language. They are primarily used to handle calculations involving numbers which cannot be represented accurately in floating point (real) format and are too large to store in integer format.

Long integers are stored in BCD (binary coded decimal), one digit per nybble. One entire word is reserved for the sign of the long integer, and the variable must end on a word boundary. Four digits can be contained in one word, so the smallest definable long integer takes up two words of memory. The numbers are padded with leading zeros when necessary to fill up the last word. The sign will be 0 if positive and 255 if negative (one byte is used).

To illustrate this, let's take a specific example: the long integer

–123456 will take 3 words: one for the sign, and two for the digits, since they are stored in multiples of 4. The format will look like this:

```
<...... each digit is one nybble ........>
MSB 6 5 4 3 2 1 0 0 0 0 F F LSB
  < word-> <-word-> sign word
```

FIGURE 8

Long integers must always be passed by address because they have a length which depends on their definition.

BOOLEANS

The Boolean, or binary, variable can have two values: TRUE and FALSE. This is most commonly used in determining yes/no conditions such as equality or set inclusion. This variable is stored in one word, although only the LSB (least significant bit) is used. TRUE is indicated by a 1, and FALSE shows as a 0.

```
MSB 15 . . . . . . 8 7 . . . . . . . 0 LSB
                                boolean
```

Booleans are most efficient in packed arrays, where each bit of the word is utilized. DRAWBLOCK is probably the best-known example of this use. For an excellent example of the use of boolean packed arrays, look at the program GRAFDEMO on the Apple Pascal diskette APPLE3.

Boolean variables may be passed by value or by address.

OTHER TYPES

In addition to the previously mentioned standard types, Pascal allows the programmer to define a wide variety of non-standard variable types. Probably the most popular example of this is the set.

A set is an arbitrary collection of elements, where each element is assigned an ordinal position (that is, represented by a number). Each element of the set is represented by a name which can be any word of your own choosing (except for Pascal reserved words or other variable definitions already in use). Each name is then associated with one bit in the data definition beginning with bit 0. The set is stored in memory as a series of bits which are identified by the ordinal position of the element in the type definition. A set must end on a word boundary, so, for example, 17 elements would take up 2 words, even though only one bit of the second word is actually used.

EXAMPLE:
TYPE COLOR=RED, GREEN, BLUE, YELLOW, BLACK, WHITE
is a set of colors. Red occupies position 0, and white is position 5.

```
   <................one word..............>
MSB 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 LSB
                        W B Y B G R
                        h l e l r e
                        i a l u e d
                        t c l e e
                        e k o   n
                            w
```

FIGURE 9

Sets may be passed either by name or by value, with certain restrictions. See page 203 of the Pascal reference manual for details.

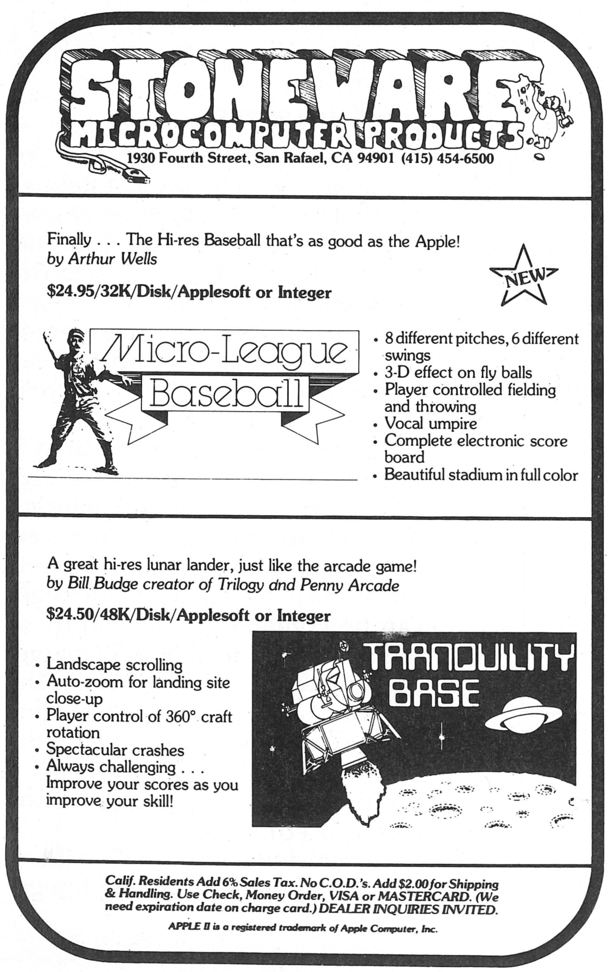
In general, complex record types consist of one or more standard types which are stored as described. For the last word on Pascal data types, read Niklaus Wirth's Report in "User Manual and Report" by Jensen and Wirth.

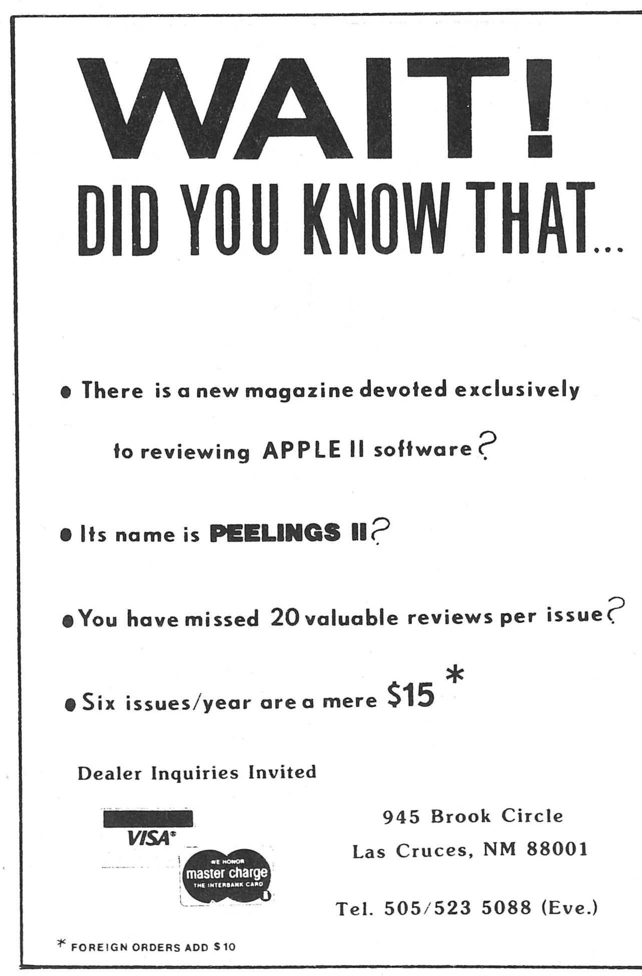
REFERENCES

Apple PASCAL Reference Manual, by Apple computer Inc. 1979.

Programming in Pascal, by Peter Grogono, Addison Wesley, 1978.

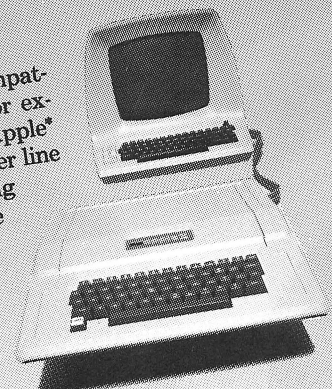
User Manual and Report, by Kathleen Jensen and Niklaus Wirth, Springer-Verlag, 1974.





Why not kill two birds with one stone?

If you have an Apple* and you want to interface it with parallel and serial devices, we have a board for you that will do both. It's the AIO.™



Serial Interface.

The RS-232 standard assures maximum compatibility with a variety of serial devices. For example, with the AIO you can connect your Apple* to a video terminal to get 80 characters per line instead of 40, a modem to use time-sharing services, or a printer for hard copy. The serial interface is software programmable, features three handshaking lines, and includes a rotary switch to select from 7 standard baud rates. On-board firmware provides a powerful driver routine so you won't need to write any software to utilize the interface.

Parallel Interface.

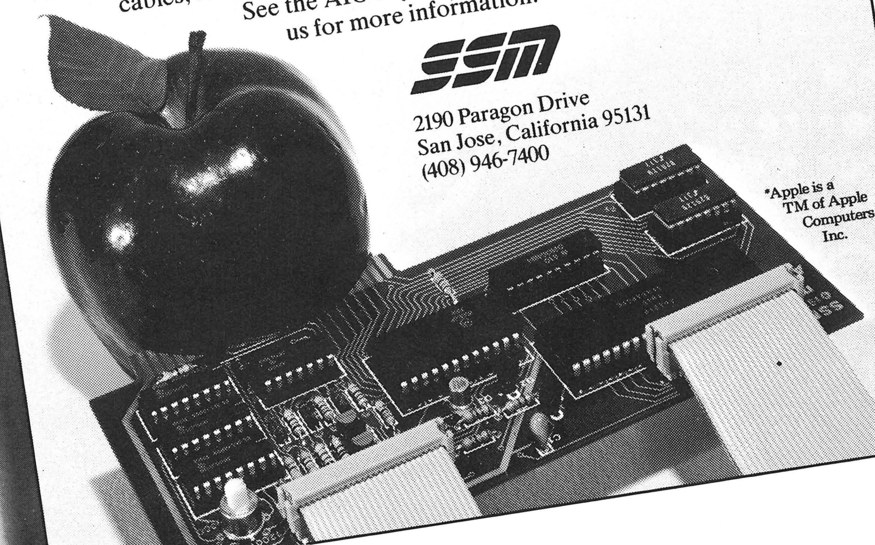
This interface can be used to connect your Apple* to a variety of parallel printers. The programmable I/O ports have enough lines to handle two printers simultaneously with handshaking control. The users manual includes a software listing for controlling parallel printers or, if you prefer, a parallel driver routine is available in firmware as an option. And printing is only one application for this general purpose parallel interface.

Two boards in one.

The AIO is the only board on the market that can interface the Apple to both serial and parallel devices. It can even do both at the same time. That's the kind of innovative design and solid value that's been going into SSM products since the beginning of personal computing. The AIO comes complete with serial PROM's, serial and parallel cables, and complete documentation including software listings. See the AIO at your local computer store or contact us for more information.

SSM

2190 Paragon Drive
San Jose, California 95131
(408) 946-7400



*Apple is a TM of Apple Computers, Inc.

Maybe we can save you a call.

Many people have called with the same questions about the AIO. We'll answer those and a few more here.

Q: Does the AIO have hardware handshaking?
A: Yes. The serial port accommodates 3 types—RTS, CTS, and DCD. The parallel port handles ACK, $\overline{\text{ACK}}$, BSY, STB, and $\overline{\text{STB}}$.

Q: What equipment can be used with the AIO?
A: A partial list of devices that have actually been tested with the AIO includes: IDS 440 Paper Tiger, Centronics 779, Qume Sprint 5, NEC Spinwriter, Comprint, Heathkit H14, IDS 125, IDS 225, Hazeltine 1500, Lear Siegler ADM-3, DTC 300, AJ 841.

Q: Does the AIO work with Pascal?
A: Yes. The current AIO serial firmware works great with Pascal. If you want to run the parallel port, or both the serial and parallel ports with Pascal, order our "Pascal Patcher Disk."

Q: What kind of firmware option is available for the parallel interface?
A: Two PROM's that the user installs on the AIO card in place of the Serial Firmware PROM's provide: Variable margins, Variable page length, Variable indentations, and Auto-line-feed on carriage return.

Q: How do I interface my new printer to my Apple using my AIO card?
A: Interconnection diagrams for many popular printers and other devices are contained in the AIO Manual. If your printer is not mentioned, please contact SSM's Technical Support Dept. and they will help you with the proper connections.

Q: I want to use my Apple as a dumb terminal with a modem on a timesharing service like The Source. Can I do that with the AIO?
A: Yes. A "Dumb Terminal Routine" is listed in the AIO Manual. It provides for full and half duplex, and also checks for presence of a carrier.

Q: What length cables are provided?
A: For the serial port, a 12 inch ribbon cable with a DB-25 socket on the user end is supplied. For the parallel port, a 72 inch ribbon cable with an unterminated user end is provided. Other cables are available on special volume orders.

The AIO is just one of several boards for the Apple that SSM will be introducing over the next year. We are also receptive to developing products to meet special OEM requirements. So please contact us if you have a need and there is nothing available to meet it.

SSM

SSM Microcomputer Products
2190 Paragon Drive
San Jose, California 95131
(408) 946-7400

AUTO-RUN APPLE WITHOUT DOS

Some applications require that an Apple start running an Applesoft program from power-up without human interference. This is easy with the disk and Auto-Start ROM. Simply initilize the diskette with the desired program in memory and the disk will boot and run it when the power comes on. But sometimes a disk drive is undesirable, especially where there is only one program to run and cost or people who don't know disks from Frisbees are involved. So here is a way to have a card that will load and run an Applesoft program automatically on power-up in any Apple II Plus that it is plugged into.

I will assume the use of a card like the Mountain Computer ROM+ that has a 256 byte "control ROM" and room for some larger ROMs for storing the Applesoft program. On the ROM+ this is a bank of up to six 2716 type EPROMs. Using EPROMs has the advantage that you can change your Applesoft program later by erasing the EPROMs and reprogramming and the disadvantages of higher cost and using more power from the Apple's power supply. In most Apples that aren't filled with cards, the power comsumption of the EPROMs won't be a problem.

The software in the control ROM is required to do five things:

1) Pretend that it's a disk controller card so that the Auto-Start ROM will execute its code.
2) Initialize Applesoft.
3) Move an image of the Applesoft program down from the ROMs into the proper area of RAM.
4) Set up the required Applesoft pointers for the end of the program.
5) RUN the program.

All that's needed to convince the Auto-Start that there's a disk controller card out there is to have a ROM whose first four odd bytes match the Apple P5 or P5A PROM. If the monitor finds a ROM that matches that in slot number n it will do a jump to $Cn00. The routine that does this is at $FAA6 on page 144 of the Apple II reference manual. So the first eight bytes in the control ROM will be

24 20 24 00 24 03 24 3C

Note that by having the even numbered bytes equal $24, (BIT Page zero) when the code is executed starting at $Cn00, nothing will happen until the byte following the $3C.

The easiest way to initialize Applesoft is to jump to $E000. Unfortunately this entry into Applesoft falls into the normal command level routine. To regain control so that the control ROM can load a program we can use the same trick that DOS uses. As soon as Applesoft reaches its command level it outputs a prompt, ']' and waits for the user to type in a command. Since all input and output in the Apple is handled through two pointers in RAM, we can divert, say, the input routine to point back into the control ROM. This will leave several levels of subroutine on the 6502 stack, but Applesoft will re-initialize the stack anyway so it doesn't matter.

Now the question becomes what address to put into the pointer. The control ROM's address will change with which peripheral slot that it's plugged into. The low byte is just the offset from the start of the ROM since the address always starts on a 256 byte boundary, but the high byte could be anything from $C1 to $C7. When the Auto-Start ROM was looking for a disk controller card it saved the high order byte in $7F8. This location is meant to allow an interrupting device driver to restore the Apple's I/O vectors to where they were before returning. However, here it just makes the job easier. So the contents of $38 and $39, the input pointer, become the offset and the contents of $7F8. Then we can jump to $E000 to initialize Applesoft, confident that we will regain control when it's done.

For the next step, a copy of the image of the Applesoft program needs to be programmed into the EPROMs. The program starts at the address pointed to by $67 and $68 and ends at the address pointed to by $AF and $B0. The end address, $AF and $B0, will also be needed in the next section of the control ROM. When control comes back through the input routine pointer, Applesoft has already initialized $67 and $68. So the next step is to move the image of the Applesoft program down to where it originally came from. How this is done will depend on the hardware of the ROM card and the length of the Applesoft program. If you use the Apple firmware card you will have to address the soft switch to select the firmware card and then address the switch again to reselect the Applesoft ROMs. With the ROMPLUS it could be as simple as using the monitor move routine, $FE2C, to move a program of less than 2 kilobytes long.

There is a little more initialization to be done before the Applesoft program can be RUN. The end of program pointer mentioned earlier must be put into $69 and $6A and one more Applesoft routine must be called. Unfortunately this one also drops into Applesoft's command mode so we have to modify the input pointer again to point to a third part of the control ROM. Once this is done the final initialization can be done with a jump to $D4F2.

And now the final part, we need to reset the input pointer so that the program can input normally from the keyboard and actually RUN the program. To make things easier there is a routine in the Auto-Start ROM that will set the input pointers to the keyboard at location $FE89. Then all that's left to do is jump to $D566 which will run the program.

For a bit of finesse, if in this last part of the control ROM code we put a $80 into location $D6, the user will not be able to list the program. In fact any attempt to do any Applesoft command except LOAD from cassette will cause the program to RUN. Also, by changing the reset vectors in the Auto-Start ROM to point to the RUN routine, the program will become very difficult to stop or modify. (To change the reset vector, load memory starting at $3F2 with 66 D5 71; see page 36 and 37 of the Apple II Reference manual for more details).

THE LISTING

This listing is all done relative to the start of the ROM, so all add-

(continued on page 43)

INITIALIZING APPLE PERIPHERALS WITH POKES

by John Crossley
Apple Computer, Inc.

Neither the PR# or IN# commands in Applesoft and Integer Basic initialize the interface to which they refer. This can cause problems for the user who needs to modify the parameters of the interface for his application because he must send a character through the interface before poking in the new parameters. The following lists are the POKEs needed to initialize the memory locations used by the various interfaces. Please refer to the manual for the interface for more information on what each POKE will do.

Included for each interface is a list of POKEs that will *replace the PR# and IN# commands.* These POKEs must be used to reap the benefits of the previous POKEs. The CALL 1002 should be used if you will be doing DOS commands while the interface is enabled. However, if speed is of the essence, don't use the CALL 1002 until after the data transfer has been made since DOS does slow down I/0. These POKEs must all be on one command line separated by colons to work in command mode. They can have separate line numbers in a program.

The normal way to reset the I/0 to the Apple video and keyboard is:

```
D$="": REM CTRL-D
PRINT D$;"PR#0"
PRINT D$;"IN#0"
```

However, this will only work after a PRINT and will be ignored after a GET or PRINT terminated with a comma or semicolon. To avoid having to do the extra PRINT you can use:

```
CALL -375 : REM THIS IS IN#0
CALL -365 : REM THIS IS PR#0
CALL 1002 : REM THIS
  RECONNECTS DOS
```

SPECIAL NOTE: All of these interfaces have the option of echoing to the Apple's video while outputting and your program or variables will suffer if you don't disable the video output while outputting past the 40th column.

In all the lists the letter "s" should be replaced by the slot number.

Parallel Printer Interface

POKE 1400+s,80	(Carriage Width)
POKE 1656+s,0	character counter
POKE 1784+s,137	(set command prefix to ctrl-I)

The following are for the centronics card only.

POKE 1912+s,0	(no video)
or ,1	(enable video)

The following are for the general purpose card only.

Poke 1912+s,0	(no video, no linefeed)
or ,1	(no video, enable linefeed)
or ,128	(enable video, no linefeed)
or ,129	(enable video, enable linefeed)

POKE 54,2	PR#s
POKE 55,192+s	
CALL 1002	

Communications Interface

POKE 1784+s,32	(lower case, page 17)
POKE 1912+s,0	(video echo, pg 17)
POKE 2040+s,17	(STAT, page 27)
POKE -16242+s*16,3	(reset ACIA, page 27)
POKE -16242+s*16,17	(status, page 27)

POKE 54,5	PR#s
POKE 55,192+s	
POKE 56,7	IN#s
POKE 57,192#s	
CALL 1002	

Serial Interface Card

POKE 1144+s,64	(BRATE, page 21)
POKE 1272+s,2	(STBITS, page 21)
POKE 1400+s,7	(STATUS, page 22)
POKE 1528+s,0	character counter
POKE 1784+s,80	(PWDTH, page 23)
POKE 1912+s,9	(NBITS, page 23)
POKE 2040+s,129	(FLAGS, page 24)

POKE 54,7	PR#s
POKE 55,192+s	
POKE 56,5	IN#s
POKE 57,192+s	
CALL 1002	

APPLEWRITER MODIFICATION FOR LOWER CASE DISPLAY

This note is taken from a letter from Lou Rivas of Canoga Park, California. It allows Applewriter to be used with the Paymar Lower Case Adapter. The few ASCII codes not available on the Apple keyboard are also supported. This modification still uses normal Applewriter files, only the display routines were changed.

```
UNLOCK TEDITOR
BLOAD TEDITOR

CALL -155

811:8D 10 C0 4C 48 18
AE6:20 64
AE8:18
1549:20 6B 18
1848:C9 81 F0 01 60 AD 00 C0
1850:10 FB C9 AF D0 06 A9 DC
1858:8D 10 C0 60 C9 AD D0 F8
1860:A9 DF D0 F4 20 78 18 91
1868:28 C8 60 C9 A0 90 06 20
1870:01 15 20 78 18 4C F6 FD
1878:C9 E0 90 02 49 40 C9 C0
1880:90 02 09 20 C9 40 B0 08
1888:C9 20 B0 02 09 40 09 80
1890:60
```

to check your typing, enter

```
811.816 AE6.AE8 1549.154B
  1848.1890
```

which should duplicate the above information.

Now save the editor with:

```
BSAVE TEDITOR,A$803,L$10F8
LOCK TEDITOR
```

The extra characters are "_", " ", and "|" and can be entered into a file with:

"|" <ctrl-A> /
"\" <ESC> <ctrl-A> /
"_" <ESC> <ctrl-A> -

<ctrl-A> - will display a small white block on the screen but nothing will be printed.

The total character set is now:

!"#$%&'()*+,-./ 0123456789:; <=> ?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmno pqrstuvwxyz {|} ~

With some printers, some of the characters may be defined differently.

AUTO-RUN from page 42

resses are given as one byte. If you use an assembler, simply orgin the code in RAM and when it is programmed into the control ROM it will work just fine. The routines are not arranged in order of execution so that the move routine will be at the end, since the length of the move routine will vary with the hardware requirements. Just be sure that the three byte jump is inserted behind the move routine.

(listing on page 44)

AUTO-RUN LISTING from page 43

```
* THE FIRST 4 BIT INSTRUCTIONS LOOK LIKE A DISK
BUT DON'T DO ANYTHING
*
00: 24 20          BIT $20
02: 24 00          BIT $00
04: 24 03          BIT $03
06: 24 3C          BIT $3C
*
* THIS PART INITIALIZES APPLESOFT AND GETS
CONTROL BACK AT ENTRY2
*
08: A9 1A          ENTRY1         LDA # ENTRY2
0A: 85 38                         STA KSWL
0C: A5 F8 07                      LDA $07F8
0F: 85 39                         STA KSWH
11: 4C 00 E0                      JMP COLDSTART
*
* RESET INPUT TO THE KEYBOARD AND RUN THE
PROGRAM
*
14: 20 89 FE       ENTRY3         JSR SETKBD
17: 4C 66 D5                      JMP RUN
*
* FINISH INITIALIZATION AND MOVE THE PROGRAM
TEXT DOWN FROM THE ROMS
*
1A: A9 14          ENTRY2         LDA # ENTRY3
1C: 85 38                         STA KSWL
1E: A9 11                         LDA # LENGTHL
20: 85 69                         STA LOMEML
22: A9 hh                         LDA # LENGTHH
24: 85 6A                         STA LOMEMH
INSERT YOUR MOVE ROUTINE HERE FOLLOWED BY
  : 4C F2 D4     JMP INIT PART 2
```

DOS TOOLKIT
Selected Aids For The Apple II Programmer

Apple's DOS Toolkit is a collection of programs and subroutines designed to aid the Apple II user in the development of Applesoft BASIC and 6502 Assembly Language programs. The Toolkit simplifies program development by providing a number of handy features that make programming easier.

Included are an assembler and source editor for use under DOS on Apple II or Apple II Plus systems, as well as an assembly language program that renumbers, merges, and deletes remarks from Applesoft BASIC programs. In addition, there are two special high resolution graphics programs in the Toolkit — one that helps you create and edit high resolution character sets, and another that lets you display characters on the high resolution graphics screen. Also included are three graphics demo programs, and character sets for editing high resolution characters.

If you're a programmer familiar with Applesoft BASIC and/or machine language, Apple's DOS Toolkit contains a number of unique programming aids that will prove invaluable to you.

BENEFITS

Apple's DOS Toolkit. . .

- Reduces programming time, by providing the user with such powerful program editing capabilities as character search, line search, and string replace...
- Allows the user to assemble arbitrarily large source files, because its disk-based operation requires that only the symbol table be held in RAM. . .
- Makes the assembler easier to learn, since it is fully compatible with 6502 syntax. . .
- Increases programming flexibility, because its text files feature provides a degree of compatibility with other assemblers. . .
- Simplifies the creation of relocatable modules by providing the user with a relocating loader.
- Speeds program development, because its multiple applications assist users in accomplishing many time-consuming and difficult programming tasks. . .
- Aids in the design of high resolution graphics characters through the use of a special graphics editor.

THE DOS TOOLKIT — A CLOSER LOOK

The four programs and subroutines that make up the DOS Toolkit were designed to meet a variety of programming needs.

The Editor/Assembler is an integrated assembler and source editor designed for the creation of 6502 assembly language programs. After accessing the Editor/Assembler from the Toolkit diskette, you can create and edit source code files in RAM; store and retrieve programs as text files; assemble disk source files into disk object files; and create your own symbol table summary. The Editor/Assembler program also features relocatable or absolute code output, as well as a relocating loader.

The Hi-Res Character Generator is an assembly language program for displaying text on the high resolution graphics screen. Using the Generator, you can mix text with high resolution graphics; write text over an existing background; automatically downshift alphabetic characters for displaying lower case text; and animate figures. The Generator also allows alternate character sets for user-defined characters, and features a text wraparound within the text window. Additionally, it provides examples of graphic implementation through three graphics-oriented demos and several alphabetic fonts.

Animatrix (Character Editor) is a special Applesoft BASIC program which makes it easy for you to create and edit character sets for the Hi-Res Character Generator.

Applesoft Programmer's Assistant is an assembly language program that helps you write your own programs in Applesoft BASIC. The Assistant can determine program length, renumber and merge several programs, and delete remarks. Its automatic line numbering feature makes program entry easier, and — since it allows you to cross reference variables — takes some of the confusion out of programming. The Assistant also provides you with the use of three, non-standard keys: underscore, left bracket, and backslash. In addition, it will print non-visible characters when listing a program.

SYSTEM CONFIGURATION

To use the DOS Toolkit, you will need:

- Apple II (with Applesoft Firmware Card) or Apple II Plus, each with 48K of memory; or
- Apple II or Apple II Plus with Apple Language System;
- Apple Disk II with controller and 16-sector PROMs;
- Video monitor or television.

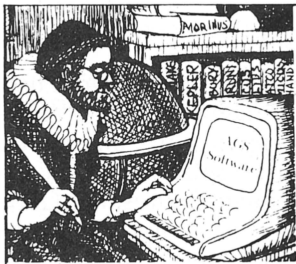
TECHNICAL SPECIFICATIONS

Language: DOS Toolkit is written in Applesoft and Machine Language.

THE DOS TOOLKIT PACKAGE —
Order No. A2D0029

With your DOS Toolkit order, you will receive:

- DOS Toolkit diskette;
- 6502 Assembler/Editor instruction manual;
- Applesoft Toolkit instruction manual.



The Electronic Astrologers™

cast an accurate birth chart for any date, time and place from 1880 to 2000, *then tell you what it means!* They give personalized astrological consultations of 1500 words or more, based not just on your Sun sign, but on the unique relation of ALL the planets at your birth moment.

ASTRO-SCOPE

delineates your character, its strengths and weaknesses, and touches on many areas of life such as relationships, finances, career and life goals. Text is by Steve Blake, psychology-oriented astrologer and popular lecturer, and Robert Hand, pioneer in astrological microcomputing and author of four bestselling astrology books...... $30*

SEX-O-SCOPE

tells you things your astrologer would blush to reveal! John Townley, author of *Planets in Love*, an editor of *Sexology Today*, and a student of *all* forms of sexual behavior, uncovers your tastes and turn-ons.......... $30*

*In Mass. add 5% sales tax.

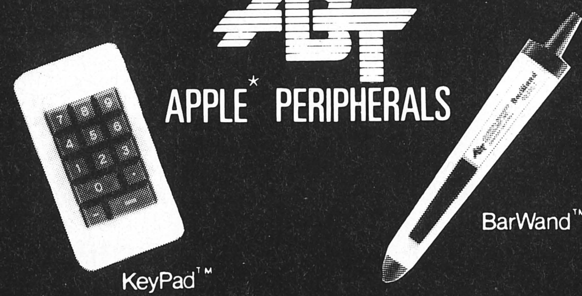
For Apple II† with Applesoft, 32K RAM, disk drive, or TRS-80, 32K RAM, dual drives.**

†TM of Apple, Inc. **TM of Tandy Corp.

Dealers: for information, call 617/255-0510.

AGS Software
217 Rock Harbor Rd., Orleans, Ma. 02653

ABT APPLE* PERIPHERALS



IMPROVE YOUR DATA ENTRY

BarWand™

- For UPC, Paperbyte,** LabelCode, and others.
- Sells Point-of-Sale Inventory Systems
- Libraries, Factories, Security

KeyPad™

- Essential to business applications
- Accountants Keyboard Layout
- Permits relaxed arm position

– Available from local Apple dealers –

ADVANCED BUSINESS TECHNOLOGY INC.
12333 Saratoga-Sunnyvale Road, Saratoga, CA 95070 (408) 446-2013

★ TRADEMARK OF APPLE COMPUTER, INC.
★★ TRADEMARK OF McGraw Hill

CRAE

A fast co-resident Applesoft editor for Applesoft programmers. Now perform global changes & finds to anything in your Applesoft program. Quote (copy) a range of lines from one part of your program to another. A fully optimized stop-list command that lists your program to the screen with no spaces added and forty columns wide. Append Applesoft programs on disk to program in memory. Formatted memory dump to aid debugging. Powerful renumber is five times faster than most available renumber routines. Auto line numbering. CRAE need be loaded only once and changes your Applesoft program right in memory. 48K Apple II or Plus & Applesoft ROM & disk.

MCAT

MCAT is a binary program which creates a master catalog report. The first list is sorted by file names and the second by volume number with sectors used indicated. provisions for duplicate volume numbers. 600 file names capacity on 48K system. 200 for a 32K system.

THE WIZARD

HIRES Adventure-like game using over 100 pictures. Requires 48K, Applesoft ROM, Disk.

CRAE on Disk With 16 Page Manual $19.95
MCAT on Disk With 10 Page Manual $14.95
CRAE And MCAT On One Disk With Manuals $29.95
The Wizard on Disk $19.95
CRAE/MCAT Manuals Include Instructions For Making A Backup Copy.

SEE YOUR LOCAL DEALER OR SEND CHECKS TO

HIGHLANDS COMPUTER SERVICES
14422 S.E. 132nd
Renton, Washington 98055 (206) 228-6691

Washington residents add 5.3% sales tax. Applesoft and Apple registered trademarks of Apple Computers Inc.

GRAPPLE

T.M.

The Original Inexpensive paper GRAPHICS PAD for the APPLE II and BELL & HOWELL MICROCOMPUTERS*

ART
ARCHITECTURE
BUSINESS
DISPLAY
DESIGN
EDUCATION
LAYOUT
GRAPHS
SCIENCE
STATISTICS
ELECTRONICS
TECHNICAL
TRAINING
. . . AND FUN!

Available at your local dealer . . . Ask to see the Demonstration Software

ANOTHER CREATION FROM
SEMPL×SCIENCE

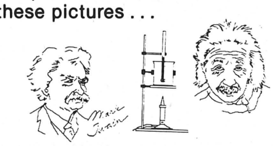
TEXT / LO-RES PAD
Formats either the TEXT or LO-RES screen. Features AUTOMATIC TEXT CENTERING.

HI-RES PAD
HI-RES screens without expensive Graphics Tablet Includes most-used Graphics commands.

TRUE screen proportions . . . NOT just graph paper. EXCELLENT for precision applications.

EASY TO USE
Effective even at the elementary school level.

Simple HPLOT statements made these pictures . . .



GRAPPLE PADS:
$3.00 per 50 sheet pad

DEALER INQUIRIES INVITED

Distributed by:
SOLUTIONS, INC.
3740 Colony Drive
San Antonio, Texas 78230
(512) 690-1017

*APPLE II is a registered TRADEMARK of Apple Computer Co.
*BELL & HOWELL MICROCOMPUTER is a registered TRADEMARK of BELL & HOWELL, INC.
*GRAPPLE is a TRADEMARK of SOLUTIONS, INC.

INTERNATIONAL APPLE CORE TM

APPLICATION FOR MEMBERSHIP

Name of Organization: ______________________

Mailing Address

Street: ______________________

City: ______________ State: ______________ Zip: ________

Country: ______________________

(If the above is a post office box, please supply a street address below where parcels may be sent.)

____

Officers	NAME	PHONE
President:		
Treasurer:		
Editor:		
Other:		

Terms Expire: ______________

Number of Members: ______________

Copies of "The Apple Orchard" desired
No. 1 @ $1.00 each ______________
No. 2 @ $1.50 each ______________

Total Remittance Enclosed: $______________

Return application and remittance to the International Apple Core, P. O. Box 976, Daly City, California 94017, USA.

Check appropriate categories below:

____ FULL MEMBERSHIP is available only to Apple User's Groups. A $50.00 initiation fee must accompany this form.

____ ASSOCIATE MEMBERSHIP applicants are asked to provide evidence that they are non-profit institutions. There is no membership fee. Club Orchard rates do not apply.

____ SPONSORS: Please indicate the name, position, and telephone number of the person in your organization responsible for liaison with the IAC. The Sponsoring Membership fee is $200.00

____ Please add 15% to your fees if your organization is overseas and you would like all material sent International Air Mail.

INSIDE INITIALIZATION

by Joseph H. Budge
Carolina Apple Corps

There are many sections of the Apple II's disk operating system (DOS) which seem especially designed to keep you waiting for eternity. My pet bugaboo for a long time has been the file handling routines. Since I couldn't do anything about them, I began looking for other places in the DOS to improve disk speed. The most obvious start was the "INIT" command. Fortunately it yielded to improvements, which I will try to describe.

This article applies to DOS 3.2.1 in a 48K machine. Addresses mentioned may be converted to 3.2 or 3.1 by subtracting four bytes from the given address. Users with less memory must subtract the appropriate amount of memory (1K + 1024 bytes).

Before describing the modifications I made, you should know how the "INIT" command works, then you will understand what the changes do. It is convenient to think of commands in the DOS as operating at two levels. The first is the command level with deals with interpretation of the "INIT" command you type. Most command level operations aren't worth modifying unless you plan to rewrite about 10K of machine language DOS. That is why file handling is difficult to improve. The second level is machine level. This mainly involves a routine called Read/Write Track and Sector (or RWTS) which does all of the disk access. RWTS, fortunately, is well documented in the original Wozpak. Most time during INIT is spent at the machine level in RWTS's disk formatter. For the purpose of this explanation we will assume that the INIT command has been properly entered and decoded by command level DOS. We shall enter at the point where RWTS is being called with the command to format a diskette:

After turning on the disk drive, RWTS calibrates the drive's read/write head to track zero. Calibration is accomplished by stepping the read head through 127 tracks. Since there are only 35 tracks in the first place, the head hits a stop at track 0 and halts there, generating the jackhammer-like noise you frequently hear. Once calibrated, RWTS steps through each track, erasing and formatting as it goes. To erase, RWTS writes over everything with one byte, $FF, in a loop that repeats 7,000 times per track. This will write over each track several times before the track is formatted. The erasure cleans off any previous data that may be on the disk, at least most of the time. Sometimes the manufacturer may put a test signal on the disk that's too strong to erase. In that case a large number of I/O ERRORs on each disk in the batch tell you to go find a bulk eraser. Audio recording tape erasers do a good job.

After erasure, formatting a track proceeds sector by sector, with each sector essentially identical. To format a sector, RWTS first writes a number of timing bytes onto the disk. These bytes are $FF's with a special spacing on the disk, and there must be at least 16 of them before each sector. Next RWTS writes an address block for the sector. The block contains four elements. Three starting marks ($D5, $AA and $B5) begin the block; they tell the DOS that data is to follow. Then comes the address data itself. Volume number, track number, and sector number are written in encoded form. Next comes the checksum. Finally three end of data marks bring up the rear ($AA, $EB, and $FF).

It is helpful to know that data isn't put onto the disk in the same form as the DOS receives it from your programs or the keyboard. All data sent to the DOS arrives there in the standard Apple format of eight-bit words (bytes). But the data can't be stored on the disk that way for a variety of reasons. Instead, the RWTS routines go through a complicated procedure to recode the data. In brief: the data is split into five bit nybbles*, data from each nybble is combined with data from other nybbles, creating a pointer to a table. RWTS gets a byte from the table and sends that value to the disk. Believe it or not, the value sent to the disk is once again an eight-bit word. But it now only represents five bits of your information which are all scrambled. This encoded byte is sent to the disk hardware, which stores each bit on the diskette in serial form, one bit at a time. Naturally the read section is smart enough to figure out all this encoding to extract the information when you want it.

The upshot of all this encoding is that no data is stored on the disk as one byte. The sector number 0, for example, is written as "$AA, $AA." The volume number, track number, and sector number each take two bytes. One sector of data, normally 256 bytes, becomes 410 bytes when encoded. Timing marks, beginning marks, and ending marks are the only single bytes ever put on the disk.

The last two paragraphs of diversion help explain what INIT does next. You will recall that we left RWTS just as it finished formatting a sector's address block. Now, ordinarily the address block would be followed by a few timing bytes and then the sector's data block. Like an address block, the data block contains beginning marks, encoded data, a checksum, and ending marks. The beginning marks for a data block are: ($D5, $AA and $AD), while its end marks are: ($DE, $AA and $EB). These are different from the beginning and ending marks in an address block. But the formatter in RWTS is lazy. Since no data needs to be written during the formatting

(continued on page 51)

Find Your Way Around The New Apple® DOS With The Dakin5® Programming Aids 3.3©

Dakin5 Corporation, a Colorado software house, is making available to the public 12 utility programs on one 16 sector diskette, utilizing the new Apple DOS 3.3, which provides 23% more storage.

These menu-driven utilities will facilitate the development of your own microcomputer programs.

All of the **Dakin5 Programming Aids 3.3** programs are also compatible with the Corvus Disk Drive system.

This 12-in-1 set of utility programs accomplishes the following:

The Lister sends BASIC programs to the printer to be listed, utilizing the full line capacity of the printer. Pagination and page headings, including program name and date, are also provided as additional options.

The Line Cross Reference produces a display or a printed listing of all lines referenced by GOTO, THEN, GOSUB, LIST or RUN statements in an Applesoft BASIC program. Cross-referencing of most programs is done in a few seconds. An option allows you to print only the line numbers referenced in GOSUB statements.

The Variable Cross Reference creates a display or a printed listing of all variable names used in an Applesoft BASIC program, showing all line numbers where a given variable name is used.

The Peeker displays or prints either all or selected records from a text file.

The Patcher allows you to display any sector of a given file or program, and then to update any data within that sector. Another option permits you to specify the sector you wish to update such as directory sectors and sectors occupied by DOS.

The Copier copies absolutely ANY type of file or program on a normally formatted diskette from one diskette to another. The name of the program or file is the ONLY information needed.

The Calculator adds, subtracts, multiplies and divides very large numbers using numeric string data. The Calculator subroutine (using twenty place accuracy) is written in Assembler code, and runs much faster than an equivalent BASIC subroutine.

Apple is a registered trademark of Apple Computer Inc.

The Diskette Copy is a diskette-to-diskette copy program that does more than just copy. First, the program verifies the input. Then it formats an output disk, copies each track, and checks that the output matches the input. Additional options allow you to either initialize a diskette without DOS, or to create a copy without DOS, thereby increasing storage by 32 sectors. You may even create a copy with a different volume number than the original.

The Array Editor is a simple word processor that allows you to create, modify, print and save your own text or EXEC files.

The Screen Printer permits contents of the text screen to be sent to the printer at any time the keyboard is active (i.e. the cursor is visible). This Screen Printer program remains in effect until you press RESET or "reboot" the system.

The Prompter is a data entry subroutine that handles both string and numeric data. You have the option of using commas, decimal points and leading zeros with right-justified numerics. Alphanumeric data is left justified with trailing spaces added as required. With the Prompter you are also able to specify maximum field length to prevent overflow in both numeric and alphanumeric fields. You can even define your own set of valid characters.

The Cruncher removes REM statements, unreferenced (dead) code, and compresses code in Applesoft programs. This will increase the speed of your programs; memory and disk space savings could be more than 45%.

Many of these utility programs have been developed and tested for in-house use while producing The Controller™ business package for Apple Computer Inc.

Suggested retail price for **Dakin5 Programming Aids 3.3** is $70.00.

Each programming aids package includes a program diskette and very complete documentation, all attractively packaged in a padded, blue print vinyl 3-hole notebook with silver lettering. An identifying tab separates each program for convenient reference.

See your Apple dealer or contact Dakin5 Corporation, P.O. Box 21187, Denver, Colorado 80221. Telephone: 800-525-0463. Visa or MC welcome.

DAKIN5 CORPORATION

The Controller is a registered trademark of Dakin5 Corporation.

stage of INIT, why bother with the data block at all? Instead, RWTS simply fills enough disk space with $FF's to allow room for a data block. Then RWTS proceeds on to the next sector. Since the fake data block just written has no beginning marks, the DOS won't be able to find data in the sector. Therefore an initialized sector can't be read (I/O ERROR) until properly written to. As far as DOS is concerned, there's absolutely nothing there.

Once a sector gets formatted, RWTS goes on to format the next sector. As you might expect by now, sectors aren't put on the disk in a straightforward linear fashion (0, 1, 2, 3...). Instead they are interleaved (0, 10, 7, 4, 1, 11, 8, 5, 2, 12, 9, 6, 3) to give the computer thinking time between sectors, DOS uses this time to handle data encoding and decoding, as well as to find out what sector to process or command to interpret next. Thus sector 2 is actually four sectors behind sector 1. Reading an entire track actually takes four revolutions of the diskette rather than one (.13 sec vs. .03 sec).

Once each sector has been formatted on a track, RWTS's formatter performs a quick check to insure that the next sector coming past the read head is really sector zero. If sector zero wasn't there, then RWTS must have gotten its timing wrong, so it adjusts the number of timing bytes between sectors and starts over. When sector zero can be found on cue, the timing is correct. Then RWTS moves on to the next track.

After formatting all tracks, INIT returns to command level DOS. There, the DOS copies itself out onto the diskette, writes out a catalog, a Volume Table Of Contents, or "VTOC", and executes a "SAVE" command for the Hello program. The entire initialization takes two to three minutes or more, depending on the length of your Hello program.

There are several places in the format routine to cut corners and get away with it. The first place that springs to mind is the routine which calibrates the read/write head to track 0. Adventurous souls can try POKEing new numbers into 48801 ($BEA1) to see what happens; just keep the number above 35. I've seen too many instances where the drive's head only hit the stop once or twice to muck about here. On most systems the maximum time savings would only be about five seconds, anyway.

Another, safer number to change is the number of timing bytes which begin each sector. When first formatting track 0, RWTS puts 60 bytes between each sector. This is clearly too much to fit onto one track causing the last sectors to overwrite sector 0. An error handling routine decrements the number of timing bytes by one and tries again. If the number of timing bytes falls below 16, the error handler will quit with an error message. Normally the routine finds a number of timing bytes that works and sticks with that number for the rest of initialization. A well-adjusted drive will have about 40 timing bytes between each sector, so you can speed things up by starting the countdown with a lower number. Try POKE 48817,48 ($BEB1:30). Forty-eight bytes will work with any disk drive within 30 speed units of zero as measured on Apple's Disk Speed Adjust program. This change will shave about ten seconds from the beginning of INIT.

Both changes so far have just nibbled at the speed problem. Now get ready for the Big Fix. Since the timing bytes between sectors together with the sectors themselves fill up each track, there's no need to erase the whole track beforehand. Rather, one need only erase enough to cover the gap between the end of the last sector (sector 3, remember?) and the start of sector 0's timing marks. POKE 48821,4 does the trick ($BEB5:04). My own trials showed that any number less than four yields flakey disks that can't be trusted. This change, together with the preceeding one, cuts initialization time from two minutes to 45 seconds.

Another change to the DOS can speed up disk access, although it doesn't help speed INIT. The change must be effected prior to initializing a diskette for the new diskette to work faster, so I'll include it with the rest of this discussion on INIT.

Remember our old friend, sector interlacing? Well, it turns out that RWTS doesn't need all of that extra time to think after all. If you POKE 48998,7 ($BF66:7) before initializing a diskette you will change the interleaving sequence from every fourth sector to every other sector. This effectively doubles machine level access to the disk. You won't believe how fast a disk created this way will boot, for example. Unfortunately much of the DOS itself is a very slow program, so you can't double the speed of everything. This interleaving change will help machine language access via RWTS tremendously. Access time through command-level DOS will only be improved 10%, however. Curiously, the alternate interleaving scheme seems to be the scheme originally intended by Apple. The formula given in the original Wozpak documentation support this interleaving scheme. However, the actual scheme used is the slower one described above.

All is not lost if you still would like to read programs and data off the disk at a reasonable speed. Mark Pump (ABBS Illinois Microcomputers, Inc.) has discovered that by POKEing 48998,2 ($BF66:02), file read time can be speeded up by almost 30%. Boot time wil be slightly increased from normal, however. This interleaves the sectors such that any two consecutively numbered sectors are 180 degrees apart on the disk. Evidently this interleaving synchronizes just right with the delays in command-level DOS. Any slave diskette created by a DOS containing the above changes will retain those changes for initializing its offspring.

Everyone knows that Language System disks pack data with higher density. There are many misconceptions about what changes were made. Since I'm discussing disk formatting, I might as well clear some up. Many people think the Language System achieves higher density by going from 256 byte to 512 byte sectors. The 512 byte blocks used in Pascal actually are two 256 byte sectors grouped together by the Pascal operating system. On disk the sectors still represent 256 bytes. Unlike DOS 3.2.1, however, each track has 16 sectors instead of 13. Therefore the data in each sector must be denser.

(continued on page 52)

Indeed it is. The boys in the back room at Apple found a way to have each byte on the disk represent 6 bits of your information as opposed to the old DOS's 5 bits. Thus each sector only needs 336 bytes to encode data, which leaves more room on the track for more sectors.

In addition to changing the data density, Apple made a few other format changes on Language System diskettes which bear mention. To speed operations up, sectors are interleaved using the alternating scheme described above. In addition, the beginning marks for both address and data blocks have been changed to ($D5, $AA, $96). This means that the old DOS would never find anything on the disk no matter how hard it tried. The disks are also designed to boot differently, with the boot program jumping to $800 instead of $300. Presumably this allows loading a larger second-stage boot routine at that address.

The changed boot program is contained in one of the Disk Controller Card ROMs which come with the Language System. The other controller card ROM is called the "state machine ROM." Actually the latter ROM is a complex logic gate which controls the format and timing of bit transfer between the Apple and its disk hardware. The original state machine ROM works well with 13 sector disks, but isn't reliable enough for 16 sectors. A few simple changes fix the reliability problem, and in addition allow a denser form of data encoding. A new state machine ROM with the necessary changes is included with the Language System.

The improved density provided with the Language System is very desirable. So desirable, in fact, that Apple is now introducing DOS 3.3 which uses the improved 16 sector format. DOS 3.3's most obvious benefit will be greater disk storage: 140K vs. 114K. A program supplied will reformat your diskettes for you. The changes involved in the new format require extensive revision of the RWTS routines and at least some modifications to the main DOS. For compatability reasons the command level DOS will be the same as in all previous DOS's, however, so don't look for any dramatic increases in speed.

* A true nybble is four bits. ...ed.

LOCKESMYTHE AND THE DEDICATED PROGRAMMER (or) WRITING USER-PROOF INTERACTIVE CODE

by Scot Kamins
San Francisco Apple Core

A note on the style: I use "s/he" and "hir" throughout the article to do away with the sexist use of "he" and "his" when personal pronouns are called for.

In an ideal world somewhere all computer users always follow directions on computer screens and their fingers never hit wrong keys. This article is for those programmers who know that such a world exists only theoretically.

The average computer user will, in response for specific information, supply data not only irrelevant to the question but if at all possible fatal to the program. S/he will type in 256 characters when the program can handle only 255, will type a numeric when only an alpha will do, will press the RETURN or ENTER key without entering other data, will type a six-digit number when a one-character letter is needed, and so on.

Whenever you call for the user to touch the computer, you can save yourself a lot of grief by setting up a buffer zone betweeen the input statement and the CPU. The buffer zone would consist of conditional statements (IF...THEN...) checking against parameter violations. We'll deal with some of the most common violations and give suggestions and examples of how to deal with them.

LENGTH VIOLATIONS

THE EMPTY INPUT

One usual source of grief is the "empty" input. The screen prompts for an input and the user (for various reasons, most of them bizarre) immediately presses the RETURN key. In most cases, this won't do. If your input variable is a real or integer (which you shouldn't be using anyway, the reason for which will soon become apparent) then your computer will probably put an error message on the screen. At best this messes up your pretty screen formatting; at worst (depending on your version of BASIC) your program goes down in flames. If you are using a string variable then your program probably won't crash; but you'll have a null string to contend with as well as a field containing incorrect data (ie, none).

Luckily there is a simple way out of this problem — we'll use BASIC's LEN function. LEN tells us how many characters are contained in a string. Enter the following program:

```
100 VTAB 10
200 HTAB 1
300 PRINT "NAME:...................
    <STOP";
400 VTAB 10
500 HTAB 7
600 INPUT"";IN$
650 REM
680 REM
710 REM
740 REM
999 END
```

(The empty REM statements are place holders that we'll fill in a few minutes.)

If in response to this request for a "name" our user presses RETURN without typing anything else, then the name entered will be "". While that name takes up very little memory space, we must assume that it's inaccurate — most likely, an accident. So we reject its entry by adding a new line:

```
610 IF LEN (IN$) = 0 THEN GOTO
    100
```

This says that if there aren't any characters in the input then go back and try again. It goes back to line 100 and NOT, as you might expect, to line 600 because we want to show the input prompt consisting of the dots, the word STOP, and so on.

NOT ENOUGH CHARACTERS

We can use that line's basic form to deal with more specific problems. Sometimes we want at least N characters in an input — maybe for a code. For our example, we'll assume that 8 characters are needed: we modify the line to read

```
620 IF LEN (IN$) = > 8 THEN
    GOTO 650
```

Translation: if there are at least 8 characters in the string then branch to the next section. If not then do the next line which says

```
630 GOSUB 1000
```

We add this subroutine beginning at line 1000:

```
1000 REM NOT ENOUGH CHARS
1010 PRINT"PLEASE TYPE AT
     LEAST 8 CHARACTERS"
1020 RETURN
```

And we add the clincher that sends the program back for the input again:

```
640 GOTO 100
```

The subroutine is necessary so that the user knows why hir input has not been accepted. Without this message, s/he might type the same less-then-8-characters all day and never get anywhere! Of course, proper form would have demanded that we had already displayed somewhere on the screen that the user needed to type at least this many characters; we didn't put it in so that we could provide this object lesson in the real nature of interactive programming.

Wasn't that thoughtful?

TOO MANY CHARACTERS

The other side of the not-enough-characters problem is the too-many-characters problem. If we have set up our program to accept fields of up to 20 characters and it suddenly must deal with 25, it most likely will stop working correctly (data fields have strong unions). In the example just used, the careful graphic delimiters in line 300 (the dots indicating available spaces and the "STOP" message) give adequate instruction to the user, but is not enough to protect the program from human error (or perversity, depending on your worldview). We need something to make sure the user doesn't violate the limit we've established:

```
650 IF LEN (IN$) =<20 THEN
   GOTO 680
```

Translation: if the input string has no more than 20 characters then branch to the next section. If it has more then 20 then do the next line — which is

```
660 GOSUB 2000
```

Being the quick learner that you are, you've already figured out that the subroutine beginning at line 2000 will be something like

```
2000 REM TOO MANY CHARS
2010 PRINT "SORRY - ONLY 20
      CHARACTERS PER
      CUSTOMER"
2020 RETURN
```

And, of course, we would add

```
670 GOT 100
```

in order to get the name again.

Here's what your program should look like now:

```
100 VTAB 10
200 HTAB 1
300 PRINT "NAME....................
   <STOP";
400 VTAB 10
500 HTAB 7
600 INPUT"";IN$
610 IF LEN (IN$) = 0 THEN GOTO
   100
620 IF LEN (IN$) = >8 THEN
   GOTO 650
630 GOSUB 1000
640 GOTO 100
650 IF LEN (IN$) = <20 THEN
   GOTO 680
660 GOSUB 2000
670 GOTO 100
680 REM
710 REM
740 REM
999 END
1000 REM NOT ENOUGH CHARS
1010 PRINT "PLEASE TYPE AT
      LEAST 8 CHARACTERS
1020 RETURN
2000 REM TOO MANY CHARS
2010 PRINT "SORRY - ONLY 20
      CHARACTERS PER
      CUSTOMER"
2020 RETURN
```

If you want to get technical, line 610 is now redundant since line 630 guarantees that there be not just 1, but 8 characters in the input.

Many variations on this theme are possible, of course. If our needs were for *EXACTLY* 13 characters, we would say

```
IF LEN (IN$) = 13 THEN GOTO
NEXTPHRASE
```

followed by a direction to the appropriate subroutine containing the invective

```
PRINT"I NEED EXACTLY 13
CHARACTERS                "
```

We could guard against too few and too many characters at the same time (assuming we needed at least 1 and not more than 20) with

```
IF LEN (IN$) > 1 AND LEN (IN$)
<21 THEN GOTO NEXTPHRASE
```

and then the direction to the message

```
PRINT "I NEED BETWEEN 1 AND
20 CHARACTERS             "
```

We here at Interaction Central like really responsive computers that give really specific messages in response to really specific user errors. We wouldn't combine messages like the last example does; but we're fanatics. As long as you let the user know completely and clearly what hir error is then you're ok.

THE WRONG CHARACTER

Line 300 specifically calls for a name, and names usually consist of alphabetics. This is not universally true of course; in stores and factories the names of many parts and products are in fact often numbers. But let's assume that our example is part of a name-and-address data base. What if the user gets ahead of hirself and starts typing in an address? Not an uncommon experience for even the most sophisticated user, especially if s/he has been typing in names and addresses all day long. You can prevent this kind of "bad format error" by adding the following lines:

```
680 IF LEFT$ (IN$, 1) => "A" AND
   LEFT$ (IN$, 1) =<"Z" GOTO
   710
690 GOSUB 3000
700 GOTO 10
3000 REM WRONG CHAR
3010 PRINT "FIRST CHARACTER
      MUST BE A LETTER"
3020 RETURN
```

The LEFT$ function begins at the left-most character of IN$ and looks at N characters — in this case, 1 (if it said LEFT$ (IN$, 3) it would look at 3 characters and so on). The effect of the new code is to say "if the first character entered is not a letter then reject this input and try again."

A MOMENTARY DIVERSION: ASCII CODE

Actually, line 680 is pretty strange. It says "if the first character is equal to or greater than an 'A' as well as being equal to or less than a 'Z' then the input is ok". At first glance, this construction looks both ludicrous and baffling: how can a letter of the alphabet be "less than" or "greater than" anything?

As it turns out, computers are odd things that see the world in terms of numbers only. Micros see the keyboard world (characters entered from your computer's keyboard) in terms of the ASCII CODE. ASCII consists of up to 256 numbers from 0 to 255. Different computers use ASCII differently, but certain of the codes are standard, including the numbers 0 - 9 (ASCII code 48 - 57) and the letters A - Z (ASCII code 65 - 90). The ASCII number 66 is always "B", ASCII 67 is always "C" and so on.*

* In the case of Apple, Applesoft recognizes ASCII characters in the range 0 - 127, while Integer Basic recognizes the same characters as 128-255. The Apple video display recognizes the same "negative" ASCII as Integer, and values less than 128 are interpreted as either INVERSE or FLASHing. See the new Apple II reference manual, page 15. ...ed.

When we say "=>A" we're actually saying "equal to or greater than ASCII number 65". So what we're saying in the new line is "if the first letter of the input falls between the ASCII numbers 65 and 90 then it must be a letter and is therefore acceptable." Virtually all computer manuals have copies of the ASCII chart in them, usually in an appendix. We strongly recommend that you check out your own computer's ASCII.

NUMBER, PAH-LEEZ.

If line 300 had called for an address usually beginning with a number then we could use the same basic format as line 680 changing the parameter values to reflect "0" as the lowest acceptable input and "9" as the highest:

IF LEFT$ (IN$,1) => "0" AND LEFT$ (IN$,1) =< "9" THEN GOTO NEXTPHRASE

Naturally, you'd change the subroutine to reflect that what's wanted is a number rather than a letter.

It's easy to include exceptions in the line. Let's say that the address could begin with the word BOX, as in a post office box:

IF LEFT$ (IN$, 3) = "BOX" OR (LEFT$ (IN$,1) =>"0" AND LEFT$ (IN$, 1) =<"9") THEN GOTO NEXTPHRASE

This line first checks the beginning three characters of the input to see if they spell b-o-x. If they do, then the program branches to whatever lines follow the correction routine, as in the previous examples. If they don't then the first character is checked to see if it is a number. If it is a number, then the program again branches to NEXTPHRASE. But if the input line conforms to neither of these specifications then the program loops back to the beginning for another try (after delivering an appropriate message, the code for which you can write on your own).

Notice that, except where we check for the presence of the word BOX, we only check the first character in the field. If we knew that the entire field was to contain ONLY letters or only numbers to conform to the necessary format, then we might put the "verifying" line in a for-next loop. We could then check every character. Here's an example checking for just letters (we'll skip line numbers here):

FOR X = 1 TO LEN (IN$)
IF MID$ (IN$, X, 1) < "A" OR MID$ (IN$, X, 1) > "Z" THEN GOTO REJECT
NEXT X

REJECT is the name we made up summarizing those lines that would give a "rejection" notice, send the program back for another input and so on. In the case of the APPLE II computer the lines would have to include some "housekeeping" function to clear the prematurely-exited FOR-NEXT loop (known as "Fornextus Interruptus" to the Cognizenti). Something as simple as "X = LEN (IN$) would do nicely.

Note that we use MID$ here instead of LEFT$. The LEFT$ function lets us check the first N characters beginning at the leftmost character, while MID$ lets us check N characters from any position in the string. Since we want the position to move ahead one character each time through the loop, and since we want to check only on one character, we switch to MID$.

If we change the parameter values by substituting "0" for "A" and "9" for "Z" then the loop will reject an input that is not all *NUMBERS.*

"SPECIAL"PARAMETERS

ON USING STRING INPUTS

Sometimes the inputs we want are more specialized. Rather than wanting just numbers or just letters, we want a certain *RANGE* of numbers or letters. For example, we want the user to type a number between nine and fifteen (9-15). We simply use the format established in line 75 (except now we'll look at the whole string instead of just the first character) and plug in the proper values:

IF IN$ <"9" OR IN$ > "15" THEN GOTO REJECT

The same format can be used for any range of individual characters or strings.

The observant reader (you clever person, you!) will note that even in those cases where we want a number we use a string input. There are two basic reasons for this apparantly cavalier use of strings. First, strings are easier to manipulate. LEFT$-RIGHT$-MID$ functions don't work on integers and reals, and these string functions are the most convenient ones in BASIC to use for error trapping. And secondly, many BASICs will give a *HARD* rejection if a non-numeric character is entered when a numeric variable is used to accept an input. That means, friend, that your program might crash and burn. A string variable, on the other hand, usually will accept anything.

LEADING/TRAILING SPACES

There is a negative side to this lack of discrimination on the part of string variables. Since they accept whatever is typed, they will accept inadvertant hits of the space bar. The string "gobble" is *NOT* the same as the string " gobble". When the computer looks at "gobble" the first thing it sees after the quote mark is ASCII 71 — what we see as a G. But when it looks at " gobble" the first thing it sees is ASCII 32, the space. That means that the name " Martha Mayno" will *NEVER* be found in a data base searching for "Martha Mayno". Goodness knows we don't want to lose poor Martha!

Since the example we've been using is supposed to be part of a name and address data base, we're concerned about extraneous spaces. So we'll add the following lines:

710 IF LEFT$ (IN$, 1) < > " " THEN 740
720 IN$ = RIGHT$ (IN$, LEN (IN$) - 1)
730 GOTO 710

This code eliminates leading spaces from a string. It keeps on looping until all leading spaces are gone (just in case more than one got in). You'll also want to check for trailing spaces (since "Martha Mayno " will be lost forever) which can be done using similar code. The only real change is the substituting of BASIC's RIGHT$ function for LEFT$ (which works the same way, except backwards — see your BASIC manual for details). The following two lines of code are based on the way APPLESOFT handles the "truth" of IF-THEN statements: if the condition is false the program goes to the next line, as opposed to the next statement. Thus we can write:

710 IF LEFT$(IN$, 1) = " " THEN IN$ = RIGHT$ (IN$, LEN (IN$) - 1) :GOTO 710
720 IF RIGHT$ (IN$, 1) = " " THEN IN$ = LEFT$ (IN$, LEN (IN$) - 1) : GOTO 720

We keep on Looping in lines 710 and 720 until all leading and trailing spaces are gone, just in case the space bar got hit more than once.

These new lines are different from the other "error checkers" we have seen in that they automatically correct the error rather than making the user do it.

Here's the Program in its final form:

```
100 VTAB 10
200 HTAB 1
300 PRINT "NAME:...................
      < STOP";
400 VTAB 10
500 HTAB 7
600 INPUT " " ;IN$
610 IF LEN (IN$) = 0 THEN
      GOTO 100
620 IF LEN (IN$) = > 8 THEN
      GOTO 650
630 GOSUB 1000
640 GOTO 100
650 IF LEN (IN$) = < 20 THEN
      GOTO 680
660 GOSUB 2000
670 GOTO 100
680 IF LEFT$ (IN$, 1) = > "A"
      AND LEFT$ (IN$, 1) =<
    "Z" THEN GOTO 710
690 GOSUB 3000
700 GOTO 10
710 IF LEFT$ (IN$, 1)<>" "THEN
    740
720 IN$ = RIGHT$ IN$, LEN (IN$)
    -1)
730 GOTO 710
740 IF RIGHT$ (IN$, 1)< >" "
      THEN 999
750 IN$=LEFT$ (IN$, LEN (IN$) -
      1)
999 END
1000 REM NOT ENOUGH CHARS
1010 PRINT "PLEASE TYPE AT
       LEAST 8 CHARACTERS"
1020 RETURN
2000 REM TOO MANY CHARS
2010 PRINT "SORRY - ONLY 20
       CHARACTERS PER
       CUSTOMER"
2020 RETURN
3000 REM WRONG CHAR
3010 PRINT "FIRST CHARAC
       TER MUST BE A LETTER "
3020 RETURN
```

SUBROUTINES AND CODER'S CRAMPS

"My Goodness, that's an awful lot of code to write just to check a simple input" we hear you say (excellent ears here on the Coast; has to do with listening for earthquakes). "Will I have to write all this code *EVERY* input (whimper, whimper)???"

Gracious, no, faithful programmer. You need write this code only once by putting the various sections in subroutines. We'll take the lines dealing with minimum input length (lines 620 - 640, 1000 - 1020) as an example. All we need do is substitute a variable for the number in line 1010, assign the value of the variable before we branch to the subroutine, and we're in business!

```
625 N = 8
1010 PRINT "PLEASE TYPE AT
       LEAST "; N; " CHARAC
       TERS"
```

It's as easy as that. The code dealing with leading and trailing spaces can itself constitute a subroutine; all you need do is make sure that IN$ equates to the input string before branching (just like N = 8 in the last example).

As it turns out, writing "bullet - code" doesn't really take all that much extra work or extra memory. Subroutines are the key.

THE NUMERIC MENU: A CRUMMY TYPIST'S BEST FRIEND

To paraphrase an old friend of ours, the input highways are indeed fraught with maurauders. And a route especially vulnerable is the one with crossroads — we ask the user to make a selection from a list of alternatives.

Let's assume that you have written a program that tells the retail prices of all models of five major auto manufacturers — Ford, Chrysler, Chevrolet, Volkswagon and Packard (We're classicists). Your menu *COULD* look like this:

MAJOR AUTOMOBILES
Please type the line you want to see:

FORD
CHRYSLER
CHEVROLET
VOLKSWAGON
PACKARD

Which one?

Our user, an energy-conscious buyer and an especially poor typist, enters "VULKSVAGEN". How do you protect against this menace? The examples we have used so far won't do; the input passes all the tests. We need a spelling checker! So we all hop ~~into~~ a rented Vulksvagen bus and head for the nearest Artificial Intelligence lab (M.I.T. on the East Coast, Stanford on the West Coast, and we don't know about the middle).

The next simplest way is to have a line like:

```
IF IN$< >"FORD"OR IN$< >
"CHRYSLER" OR IN$< >...
```

You get the idea. This "crunch it out" method is tedious at best. And what happens if there are 25 or 300 choices!!

Ascending the better-way-to-solve-it stairway to that Ultimate Algorithm, we find the ARRAY method. Put all the choices in an array and compare the input to the array elements. The "checker" section would look like this:

```
FOR X = 1 TO CHOICES
IF CHOICE$(X) = IN$ THEN
  GOTO EXITLOOP
NEXT X
GOTO REJECT
EXITLOOP   X = CHOICES
NEXT X
GOTO FOUNDIT
```

(An explanation: CHOICES = total number of lines; CHOICE$ (X) is the name of the array element being compared to the auto name typed in. If the program can't find it, it goes to the REJECT routine and seeks another auto name from the user. Making the control value X equal CHOICES and saying "NEXT X" in the EXITLOOP routine is a semi-required housekeeping task in our APPLE that pops the FOR-NEXT stack).

While this certainly works better than the previous method (inelegant and klutzy though it may be) in that the program won't get fouled up by the bad input, the user is still left with hir lousy typing problem. S/he may end up keying in hir choice four or five times before getting it right!

Luckily there is something we can do for the user short of teaching hir how to type — we can reduce the *QUANTITY* of typing s/he'll have to do.

THE AXIOM OF MINIMUM CONTACT states that the fewer keys the user must press the lower the likelihood of an error. In the above example we can reduce the likelihood of error by asking the user to type in a NUMBER instead of an entire name. On screen it will look like this:

MAJOR AUTOMOBILES

Please type the number of the line you want to see:

1) FORD
2) CHRYSLER
3) CHEVROLET
4) VOLKSWAGON
5) PACKARD

Which number?

Now all we have to check for is the validity of the number:

IF IN$ <"1" OR IN$ > "5" THEN GOTO REJECT

Here's what the whole program would look like, exclusive of the subroutines containing the model-price information (references to which you would insert at line 210):

```
10 REM AUTO PRICING
   PROGRAM
20 REM RUNS ON APPLE II
30 REM LINES 40, 60, 160,170
   ARE SCREEN FORMAT
   COMMANDS
40 HOME
50 I$ = "MAJOR AUTOMO
   BILES"
60 HTAB 21 — LEN (I$) / 2
70 PRINT I$
80 PRINT : PRINT
90 PRINT "PLEASE TYPE THE
   NUMBER OF THE LINE YOU
   WANT TO SEE:"
100 PRINT : PRINT
110 FOR X = 1 TO 5
115 READ CHOICE$(X)
120 PRINT TAB (10) X; " )";
    COUNT$ (X)
130 PRINT
140 NEXT X
150 PRINT
160 VTAB 22
170 HTAB 5
180 INPUT "WHICH NUMBER?";
    IN$
190 IF IN$ < "1" OR IN$ > "5"
    THEN GOSUB 1000 : GOTO
    160
200 CHOICE = VAL (CHOICE$)
210 ON CHOICE GOSUB N1,
    N2... NN
999 END
1000 REM WRONG CHOICE
1010 PRINT "NOT ON MY LIST.
     NUMBERS 1 - 5 ONLY."
1020 RETURN
2000 DATA FORD, CHRYSLER,
     CHEVROLET, VOLKS
     WAGON, PACKARD
```

There are other input errors that users can make, of course. S/he can hit the RESET key or kick out the plug — in fact, there is no end to the creativity of the truly dedicated naive computer user. All we can do is protect against the errors we know and constantly strive to devise more careful user-proof code.

Either that or go back to CB radios.

HIGHER GRAPHICS II
by Robert Clardy

Improved Higher Graphics II performs the high-resolution graphics screen creation functions of hardware graphic devices at a twentieth of the cost. Both allow you to sketch lines and points, color areas of the screen, move the screen up, down, left, or right, etc., creating detailed high-res pictures.

Producing beautiful high-res screens, however, is just one of the many features provided by Higher Graphics II. You can now also create and edit shapes and shape tables; scale, rotate, and color shapes on your screen; type text directly to the screen to label your graphs, figures, program logos, game boards, etc. Predefined shape tables come with the disk. Improved high-res machine language routines make working with graphics faster and easier than ever.

Higher Graphics II is easy to use and will teach you all you need to know about high-res graphics, animation, and using sophisticated graphics in your own programs.

(Integer/Applesoft - 48K)

DIRECTORY MANAGER
by Dennis Brown

Turn your disk catalog into a menu. Execute disk commands (LOAD, SAVE, DELETE, LOCK, UNLOCK, EXEC, BLOAD, BRUN, etc.) with just 2 keystrokes. Use the menu for a variety of other useful disk functions:

—UN-DELETE, Resurrect files that have been deleted but not yet overwritten.
—Transfer programs, binary files, or text files to other disks without needing to know the address or length parameters.
—Rearrange or sort part or all of your catalog, organizing programs and files into more logical groupings or sequences.
—Rename any file using upper or lower case; normal, inverse, or flashing video.

All commands require only a few keystrokes. All operations performed in seconds by machine language routines.

(Integer/Applesoft - 48K)

DOOM CAVERN & BLACK HOLE

DOOM CAVERN is a high-resolution graphics version of the classic "Dungeons & Dragons" type board games. Set up the persona (strength, intelligence, wisdom, constitution, etc.) of your players with dice rolls, then venture forth into the dungeons of Hammardoom castle. With perseverance, some luck, and reasoning you can win treasures and explore to the lowest depths of the dungeon.

BLACK HOLE is a high-resolution graphics game of logic and deduction. Fire rays into the uncharted black hole. By observing the exit angle of the rays, deduce the location of the hidden targets. A challenging mind teaser.

Both games provided on 1 disk.

(Integer - 48K)

SYNERGISTIC'S SOFTWARE

ODYSSEY: THE COMPLEAT APVENTURE
by Robert Clardy

Odyssey is the ultimate adventure game for the Apple. Explore desolate islands of the dread Sargalo Sea. Learn how to enter the deserted castles, tombs, ruins, and other buildings in search of their treasures. Use your gold to buy weapons and supplies you need for your quest. With enough gold, you can buy a ship and set sail. Face pirates, monsters, storms, demon haunted dungeons, bandits, warlocks, sea serpents, and hundreds of other hazards before you try for the ultimate prize, the High One's vacant throne.

Odyssey utilizes the full capabilities of the Apple with its 3 interlocking programs; detailed and colorful high-res maps, sound effects, and varied animation effects.

(Integer - 48K)

MAILING LIST DATABASE
by Chris Anson & Robert Clardy

This fast, easy to use, mailing list program produces mailing labels or lists of customers, suppliers, patients, or friends and relatives. Search 2 disks of names and addresses (up to 1700) for just those that are required for your next mailout Search or sort by name, address, city, state, zip code, phone number, company name, comment, or code characters. Features include:

—Single keystroke commands for editing, display, deletion, and printing of records.
—Fast machine language routines for searches and sorts.
—Menu driven, all required inputs prompted by the program.
—One, two, three, or more labels adjacent as required.

(Applesoft - 48K)

NOW AVAILABLE

		CASS	DISK
GAMES			
DUNGEON CAMPAIGN	(I/A, 32K)	$15.00	$17.50
WILDERNESS CAMPAIGN	(I/A, 48K)	$17.50	$20.00
WILDERNESS & DUNGEON	(I/A, 48K)		$32.50
DOOM CAVERN/BLACK HOLE	(I, 48K)		$25.00
BOTH BARRELS	(A, 48K)		$25.00
ODYSSEY	(I, 48K)		$30.00
UTILITIES			
HIGHER TEXT	(I/A, 32K)		$35.00
PROGRAM LINE EDITOR	(I/A, 32K)		$40.00
DIRECTORY MANAGER	(I/A, 48K)		$30.00
HIGHER GRAPHICS II	(I/A, 48K)		$35.00
E-Z DRAW & VIDEO MARQUEE	(A, 48K)		$35.00
BUSINESS			
MAILING LIST DATABASE	(A, 48K)		$40.00
MODIFIABLE DATABASE	(A, 48K)		$79.50

AVAILABLE AT YOUR LOCAL DEALER OR SEND CHECK OR INQUIRY TO
SYNERGISTIC SOFTWARE, 5221 - 120th AVE. S.E., BELLEVUE, WA 98006, (206) 641-1917
WA residents add 5.3% sales tax.

LINKING MACHINE LANGUAGE ROUTINES TO APPLESOFT PROGRAMS

Adding binary information or programs that load with an Applesoft program is fairly easy. This note will show how to hide a binary program that will follow the Applesoft program but won't show up on a LIST. One use for this techique is loading a program and it's shape table for creating Hi-Res images.

When Applesoft lists a program it continues listing until it finds three hexadecimal zeros in a row. However, when the program is saved, Applesoft looks at the end of program pointer, $AF,B0.

So to save machine language programs you need to:

1. LOAD the Applesoft program
2. Enter the monitor (CALL -151)
3. Load the binary starting at the address pointed to by $AF,B0
4. Change $AF,B0 to point to the end of your binary program
5. Re-enter Applesoft
6. SAVE the combined program as a normal Applesoft program
7. Reload the program before running it

This works both with tape and disk. See Applesoft Renumber for an example.

Using the machine language program is a little harder because it moves around in memory as you modify the Applesoft program. This means that the 6502 program should be relocatable. Here is one way to find out where the binary program is at a given time. Form a pointer from the contents of $AF,B0 (175,176 decimal) and subtract the length of the binary program. For example, if the binary program was 100 bytes long then we could

```
100 BI = PEEK (175) + PEEK (176)
  * 256 - 100
110 CALL BI
```

This same technique will work for shape tables where line 110 would be replaced by

```
110 POKE 232,BI-INT(BI/256)
  *256
120 POKE 233,BI/256
```

NOTE: Renumbering a program after adding binary information won't work and might destroy the program or at least kill the binary information.

CREATING COMMON ACCESS "SOURCE" FILES

It is my opinion that all users should be aware of how to share information in their files with others. You can control the information you want to share; the systems manuals tell you the gory details (DATA SYSDOC). Here's the lowdown for the most common case:

Suppose you are TCA123 and your password is XYZ. You have an ASCII file you made with the editor that you want anybody to be able to read; it is called NOTICE. Here we go:

>PASSWD You are going to remove the password protection from "non-owner" access to your files.

Old Password: XYZ You type old password.

New Password: XYZ, Restate your password. The comma tells the system that the second ("non-owner") password is non-existent.

Enter it again: XYZ, The system is just making sure.

At this point, you have eliminated the password requirement. You still have to tell the system just what files you want to make public.

>PROTEC NOTICE 7 1 This allows you (the owner) all rights (that is, read, write, and delete) to NOTICE, but non-owners can only read.

That does it. Now anyone on your system (SYS10 or SYS11) can say TY TCA123>NOTICE and see your file. If you change your mind about access, just say PROTEC NOTICE 7 0 and your file is private again. To make other files available, you only need to use the PROTEC command; no further PASSWD work is ever necessary.

If you do a FILES command, the files that other can read will appear with a lower-case "r" at the right-hand edge of the line.

This method of "unprotecting" your files makes them accessible to other on the same system (10 or 11) as you are. Users on the other system can't get to them. As far as I know, the best way for a SYS11 user to see a public SYS10 file is this:

1. CHAT a user who is signed on to SYS10, and ask him/her to help you for a few minutes.
2. Ask your correspondent to MAIL the file to you. In case he does not know how, tell him to type out this file (TY TCD728>SHARE). If the SYS10 correspondent is, say, CL0987, you are TCH555, and you want to see TCA123>NOTICE, C<0987 returns to command level and goes:

   >MAIL SEND
   To: TCH555
   Subject: File request
   Text:
   .LOAD TCA123 > NOTICE
    42 lines loaded
   .SEND
   TCH555 -- Sent
   >
3. The file will appear more or less immediately in your mailbox. Of course, if you want to save it for future reference, you can use the SAVE disposition when reading your mail.

I realize this procedure is rather cumbersome, but at least it gets the message through.

The Microsoft Z-80 SoftCard. Leading a Whole New Lineup for Your Apple II.

Plug the new Microsoft Z-80 SoftCard into your Apple II™ and start using all of the system and application software written for Z-80 based computers. Software that you could never use before on your Apple II.

The SoftCard actually contains a Z-80 processor and lets you switch between the Apple's 6502 and the Z-80 with simple commands, so you can use software written for either processor.

Starting with Two Software Standards. Versatile CP/M,® the most widely used microcomputer operating system ever, is included on diskette in the SoftCard package, ready to run on your Apple II.

You get Microsoft's 5.0 BASIC too, the most powerful version to date of our famous BASIC interpreter.

PRINT USING, 16-digit precision, CALL, and CHAIN and COMMON are just some of the major BASIC features you'll add. Applesoft's graphics extensions are still included.

More Power Down the Line. You can get even more programming power and versatility by adding Microsoft's FORTRAN, COBOL, BASIC Compiler and Assembly Language Development System. All are available separately to run with the SoftCard system.

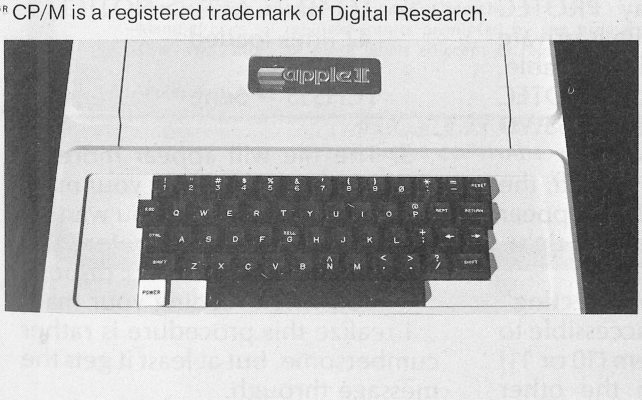
And the whole host of CP/M-based business, scientific and educational applications can be easily transferred to your Apple with SoftCard.

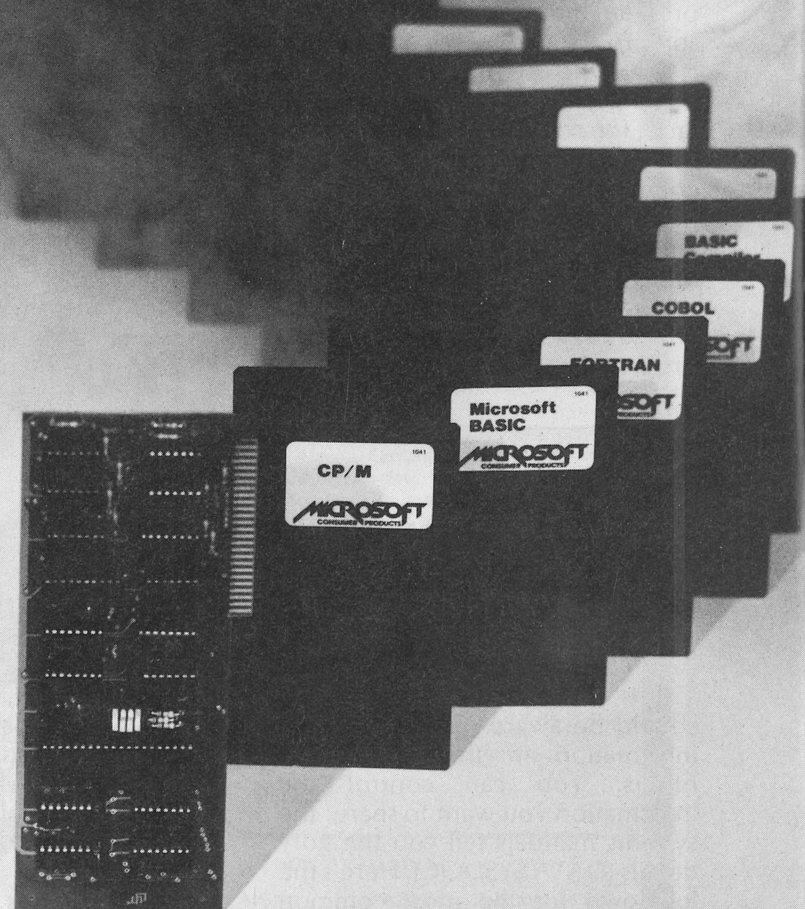
The Microsoft Z-80 SoftCard is compatible with most every Apple product from the Apple II to the Apple II Plus, Language Card and peripherals. Independent peripherals for the Apple are supported as well. The SoftCard package requires a system with 48K and a disk drive.

Line up a SoftCard demonstration at your Microsoft Consumer Products dealer today. They'll be glad to show you how the Z-80 SoftCard and your Apple computer combine to form a system that can't be beat for either practicality or pure pleasure by any personal computer available today. Or give us a call, 206/454-1315, for more information.

But act quickly. At the low price of $349 for SoftCard, CP/M, Microsoft BASIC and complete documentation, you may have to stand in line to get one!

™Apple II is a trademark of Apple Computer, Inc.
® CP/M is a registered trademark of Digital Research.

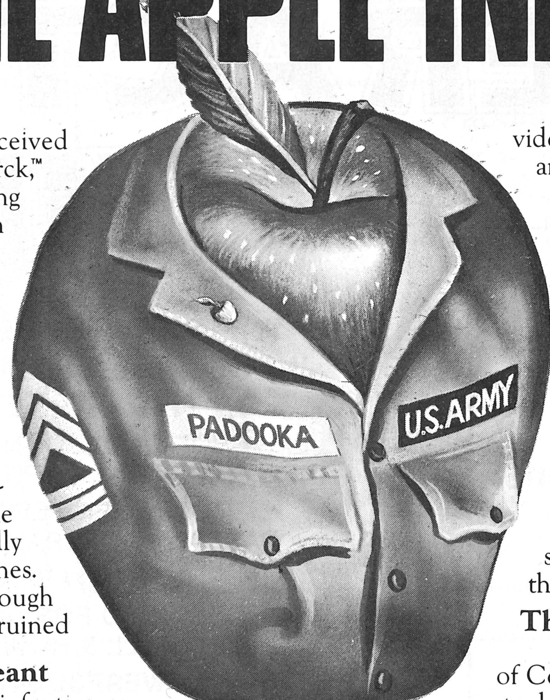




MICROSOFT CONSUMER PRODUCTS

10800 Northeast Eighth, Suite 507
Bellevue, WA 98004
(206) 454-1315

JOIN THE APPLE INFANTRY!



Judging by the letters we've received from buyers of Computer Bismarck,™ home computer historical wargaming is a great mind-stretching recreation to uncramp the old synapses after a few hours of trying to cram 54K of code into 48K of memory. But before you read any further, let us warn you that our *new* game, Computer Ambush,™ is more gut-wrenching than mind-stretching.

Strategy versus Tactics

Computer Bismarck is a "strategic" wargame, casting you in the role of a British or German admiral coolly deploying fleets of ships and planes. Computer Ambush is "tactical"...tough and dirty street fighting in a half-ruined French town.

You're a Sergeant

You command a squad of ten infantrymen (either American or German). Each man has a name, rank, and such individual combat skills as footspeed, strength, intelligence, endurance and marksmanship...all of which affect the success of every move you order. Your squad is armed with grenades, rifles, automatic weapons, plastic explosives, bayonets, and even garottes. You fight with carefully-aimed shots, area bursts, explosions, and hand-to-hand combat. They can result in wounds or deaths, depending on time, distance, the individual skills of each soldier, and your ability as a squad leader.

Battlefield

Street fighting is the most challenging tactical command situation in modern warfare. Using "Higher Text", a character generator, the computer displays a map showing buildings (your plastic explosives can turn them into rubble during the game), walls, hedges, doors, windows (nasty sniper positions), and each of your men by name. The enemy is usually hidden.

Play the Computer

The computer plays the German squad leader (*Feldwebel* Kurt Reich) to perfection. It defends the town with sniping, machine guns, grenades, and finally, with hand-to-hand combat.

You're Sergeant Buck Padooka. You maneuver your men and fire at revealed and probable German positions. If you kill all the Germans before they get you, the town is yours. But the computer's a tough, experienced squad leader, so don't expect to win very often.

Play a Friend

You take turns examining the video map display, moving your men, and firing weapons. Your options are limited by casualties, wounds, physical exhaustion, ammo supplies, terrain, and the individual skills of each of your men. The same is true for your opponent. And every action takes precious time, even the flight of a grenade or bullet. (Remember, time is life or death on the battlefield *and* in Computer Ambush!) After each turn, the computer displays the movements and weapons fire of both squads as tracks on the video map...just once, so watch carefully to figure out where the enemy is, or was.

The Sweat and Death of War

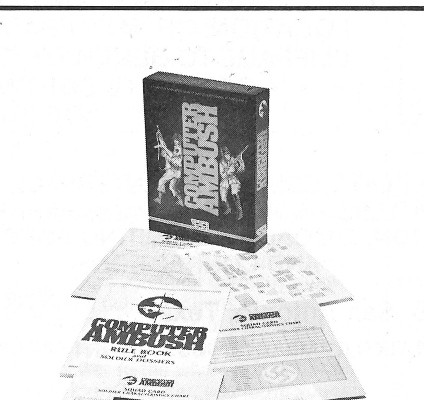
The time pressure and complexity of Computer Ambush create the stress of actual combat command. Your palms sweat as you watch PFC Chuck Lawson get blown away by that damned Kraut machine gun you forgot when you ordered him to sneak across the alley. If you can imagine a game that's more complex than chess, requires much faster decision-making, rewards courage and cruelly punishes foolhardiness...that's Computer Ambush!

$59.95 and an Apple

If you've got an Apple II Plus (or an Apple III or an Apple II with Applesoft Firmware ROM Card) with 48K memory and a 5¼ inch mini-floppy disc drive, you can be playing Computer Ambush in a few days. For $59.95, you get the game program disc; 2 mapboard charts (for plotting strategies in grease pencil while your opponent is at the computer); 2 squad leader's data cards; and a rule book. You also get a game selection card which tells you how to set up any of seven wargames: NCO Training, Ambush or Raid against the computer; and Patrol, Ambush, Strongpoint, or Free Form against a human opponent.

Call **800-648-5600 (toll free)**, and ask Operator 181 to charge Computer Ambush (or Computer Bismarck) to your VISA or MASTER-CHARGE. In Nevada call 800-992-5710. To order by mail, send your check to Strategic Simulations Inc., Dept. AO, 450 San Antonio Road, Suite 62, Palo Alto, CA 94306.

With our 14-day money back guarantee, your satisfaction is assured. So come and join our Apple Infantry!



STRATEGIC SIMULATIONS INC.

COMPUTER AMBUSH™... You've got a war on your hands.


MPI ANNOUNCES
THE INFLATION FIGHTER
mpi

DON'T OVERLOAD YOUR APPLE II

by Ken Silverman

As you shop for that next attachment for your Apple to turn it into an APPLEMATIC, that will liquefy, chop, and dice your programs — *BEWARE* you don't overload the Apple power supply.

There is such a proliferation of plug-in equipment now made for the Apple II that it has become difficult to make a decision as how to use the eight slots to best advantage.

If, for instance, you were in the market for a serial interface card and found that two of the ones you looked at would handle the job at about the same price, which one would you purchase? All operating criteria being equal, you might see which one uses the least amount or current. This information is not normally given in the manufacturers operating manuals, but if you took the time to write them, they would most likely give you the information.

This area of current drain can be very important from the point of view of overloading your supply. The Apple Reference Manual (A2L0001A), on page 92, specifies the limits of the switching power supply.

Full load power output:
+5v: 2.5 amp
-5v: 250ma
+12v: 1.5 amp*
-12v: 250ma

*This +12v can supply 2.5 amp intermittent load if not run for more than 20 minutes and is followed by 10 minutes at normal load.

The power supply has a built in protection circuit and if you short it or have no load on it, the built in oscillator will stop and cut all output. It will try and restart the oscillations about every half second and when the impairment is removed it will start up again. This also happens when an overload condition is present on the supply. The reference manual states that this cycle can continue indefinitely without damage to the power supply. In some cases the oscillator might not start again which could be caused by a faulty supply, or the fuse FU1 might be blown (located inside the supply). If this happens, it must be fixed by an authorized repair center.

The majority of peripheral cards use the +5v and +12v outputs, and it is in this area you should configure your system so that the limits are not exceeded.

As a starting point, for the +5v, the manual specifies 2.5 amp is available. This figure is the amount of current the supply can deliver to the Apple and any additional plug-in cards. First you have to determine what the motherboard of the Apple is using before any I/O devices are plugged in. This information is on page 104 of the manual, which states that 1.5 amp is consumed by the motherboard (with 48K). This now leaves only 1.0 amp for your I/O devices. I did check several Apples and found the average to measure 1.46 amp, but in a few cases, using low power RAMS, the system used only 1.2 amp for only a savings of 300 ma. I would advise using the 1.5 amp figure unless you can get a qualified technician to measure the motherboard for you. (The current meter goes in series with pin 3 for +5v measurement — see drawing on page 104 of the reference manual). The +12v supplies 400 ma to the motherboard which leaves 1.1 amp in the normal mode and up to 2.1 amp for an intermittent load.

Using CHART A and adding up the current drain of each card you should be able to keep within the limits specified. Some cards do a power down when not in use or not addressed and this should be taken into account when adding the drains. The following method should be used when adding up the power:

1. Add those currents in which there is no difference between ON & OFF.
2. Some cards are turned on for system use — like the Applesoft Firmware card. These also should be added as if always in the ON condition.
3. Take the current for the rest of your system and add the OFF drain.
4. Total the above.
5. Add the DIFF column to the total for any one item from instruction 3 which this should be the maximum drain at any one time.

A typical system might consist of an Applesoft firmware card, Disk controller, Apple High Speed Serial Card (for a printer) and maybe a D.C. Hayes modem. Referring to CHART A for the +5v column and using the instructions:

CARD +5v	OFF	ON	DIFF
Applesoft Card	381ma	411ma	33ma
Disk Controller	180ma	275ma	95ma
Apple Serial	38ma	166ma	128ma
Modem	184ma	194ma	10ma

In this example the firmware and the modem would be ON at the same time with either the disk or serial. This could give a maximum drain at any one time of 951 ma (with serial card in use). This same procedure should be used for the +12v supply.

I attempted to test as many cards as possible and those that were made available to me by friends and local stores. If you have a card that is not listed, please write me and I will try to get the measurement.

I discovered while working on this article that some systems, even though they were within current limits, had a tendency to crash. This crash, or the system going out to lunch, is not a power supply problem. The Apple could have a thermal (heat) problem inside the Apple case. With just a few peripherals, the system could be dissipating from 15 to 25 watts inside an almost closed box. Although there are vents in the casing, the air is stagnant and the internal temperature can rise. All that's needed is a RAM or ROM that is heat sensitive and a crash occurs. The normal fix is to turn off the Apple until it cools down, or take the top off so the air doesn't get stagnant. You might consider adding a fan to your system to cool it.

SMALL SAVINGS WITH LANGUAGE SYSTEM

If you use the Language System (A2B0006) in your Apple you can save approximately 70 ma of +5v current. In this configuration there is no way you can address ROM F8 on your motherboard. If you remove this ROM (monitor chip) you will save the current it normally uses.

(continued on page 69)

Dont Overload. . . from page 68

CHART A

The currents measured (note 1) were from off the shelf products and could have small variations from one unit to the next. The figures should be used to give an approximate total.

All values on the chart are in MA (Milliamps, 1000 ma = 1 amp).

		+5 VOLT			+12 VOLT			
EQUIPMENT	**MODEL**	**NOTE**	**OFF**	**ON**	**DIFF**	**OFF**	**ON**	**DIFF**
APPLE								
Serial	A2B0005		38	166	128	-	-	-
Communications	A2B0003		82	196	114	-	-	-
Parallel	A2B0002		60	135	75	-	-	-
Applesoft	A2B0009		381	411	30	-	-	-
Language System	A2B0006		168	168	-	-	-	-
Disk Controller	A2M0004 one drive	2	180	275	95	7	422	415
Disk Controller	A2M0004 two drives	2	249	351	102	14	422	408
Graphics Tablet	A2M0029		201	201	-	24	24	-
Silentype	A2M0036	3	295	295	-	-	580	580
MT. HARDWARE								
Clock	MHP-X003		58	58	-	32	32	-
ROM +	MHP-X007		186	186	-	-	-	-
Super Talker	MHP-X006		57	116	59	14	25	11
Introl/X-10	MHP-X016		37	108	71	-	-	-
CALIFORNIA COMPUTER SYSTEMS								
PTM-1	7440A		75	75	-	-	-	-
GPIB Interface	7490A		350	405	55	-	-	-
Parallel	7720AB		52	245	193	-	-	-
A/D Converter	7470A		30	30	-	50	50	-
Serial	7712A		105	298	193	17	17	-
Arithmetric	7811A		62	62	-	48	48	-
80 COLUMN BOARDS								
VIDEX	Videoterm	4	475	475	-	40	40	-
Computer Stop	DOUBLEVISION		603	603	-	-	-	-
M&R Enterprises	SUP'R'TERMINAL		390	390	-	200	200	-
MISCELLANEOUS								
SVA DISK	8" Controller		510	530	20	35	35	-
ALF	Music		107	107	-	16	16	-
Micro Music	K-1002-4(A)		65	65	-	-	-	
SSM	AIO		170	340	170	-	-	-
DC HAYES	Modem II	5	184	194	10	375	375	-
HURESTIC	Talker		171	171	-	63	63	-
W CAI	Video		111	236	125	-	-	-
TRENDCOM	Mod A2	6	140	140	-	-	-	-
TRENDCOM	Mod A2G	6	95	95	-	-	-	-
Corvus	Hard Disk		165	175	10	-	-	-
SYMTSC	Lightpen		170	180	10	-	-	-
CMP Keypad	79A002-C		109	225	116	-	-	-
Microworks	Videocard DS65		355	335	-	46	46	-
WestSide Elec	APT-1 clock		40	40	-	25	25	-
Microproducts	NP77101		122	122	-	-	-	-
Malibu Printer	Card		225	410	185	-	-	-

NOTES

1. The tests were performed using a Hewlett Packard HP970A Probe Multimeter with a Current Shunt/ Bench Cradle attachment. Specified accuracy of plus or minus 2.5% of reading. A standard 50 pin, 1" high, standoff plug with pin number 25 (+5v) and number 50 (+12v) lines cut and series wires attached to be used with the meter (figure 21 on page 106 of reference manual show connector pinout).

2. When the disk drive first turns on, the starting torture of the motor causes the unit to draw about 700 ma of +12V current for a second of two, then it draws the 422 ma while running. The values for one or two drives are measurements using one controller card.

3. The current for the +12v supply varies from 200 to 580 ma when the unit is printing.

4. The figures for the Videoterm were measured without a graphic EPROM (2708) in the unit.

5. The D.C. Hayes draws an extra amount of +5v current, about 10 ma when in the dialing mode.

6. The model A2 is without graphics and the model A2G is with graphics.

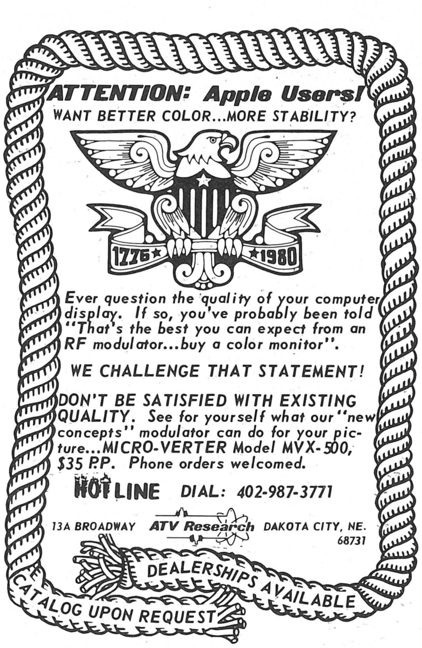
WHAT IS A USER GROUP

by Dan Buchler
President, *Mini'app'les*
Minneapolis, Minnesota

What is a User Group? That was the title of the editorial in the first edition of Apple Orchard. In that editorial Val Golding stated that the primary accent was on software. I do not dispute the statement, but would like to offer a slightly different point of view.

Our user group was formed over 2 years ago and had grown to over 50 in the first 6 months and currently has over 200 members with a growth rate of about 5 to 10% per month. Our members are not all software addicts. Some are; others are educators; some like to play games; others use their Apples in Business or Industry; some are Ham radio operators. Most do little or no programming although most want to learn as much as possible. The fact is that a typical user is not much of an expert in programming. That person wants to do his thing with the apple, but has not the time to learn to become an expert. Many will try to make small changes to programs, but most would hesitate to get very involved.

What am I trying to say? I am telling you that we user group organizers often cater to the wrong audience. Sure, I enjoyed the article on Applesoft Internal Entry Points by John Crossley in the first edition of Apple Orchard. In fact I used ideas from there to improve the text editor used to write this article. But, to how many readers does such an article appeal? 2%? 3%? Maybe 5% at most. Obviously, you can't please all the people all the time. But let us not forget who are the majority. The user group is for the benefit of those of many backgrounds. Let us all spend more time and effort to help those others enjoy their Apples.



TO ALL PROGRAMMERS

from Neil D. Lipson
Software Chairman, I.A.C.

One of the more important functions of the I.A.C. is to provide free quality software to the end user. We rely on any of the thousands of Apple users to provide us this software. However, we have received very little from the user groups within the I.A.C. We encourage any of the clubs to contribute *SOME* software, with the documentation on the disk so we can continue to give away a disk every month. Keep in mind, the more you submit, the more is distributed. Please help out, and everyone will benefit. Thanks!

ADVERTISERS'S INDEX

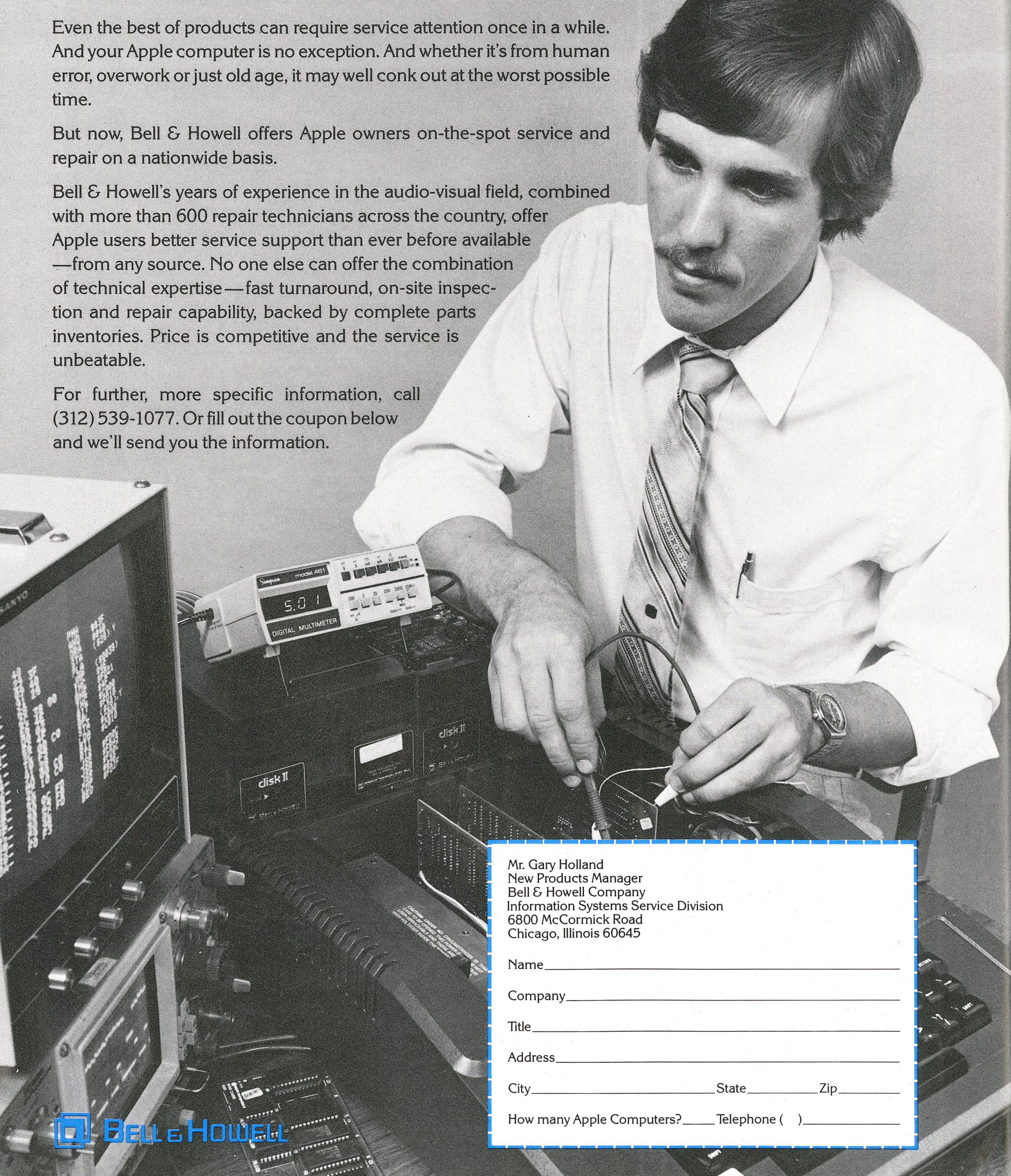
BELL & HOWELL SERVICE CAN GET TO THE CORE OF YOUR APPLE PROBLEMS.

Even the best of products can require service attention once in a while. And your Apple computer is no exception. And whether it's from human error, overwork or just old age, it may well conk out at the worst possible time.

But now, Bell & Howell offers Apple owners on-the-spot service and repair on a nationwide basis.

Bell & Howell's years of experience in the audio-visual field, combined with more than 600 repair technicians across the country, offer Apple users better service support than ever before available —from any source. No one else can offer the combination of technical expertise—fast turnaround, on-site inspection and repair capability, backed by complete parts inventories. Price is competitive and the service is unbeatable.

For further, more specific information, call (312) 539-1077. Or fill out the coupon below and we'll send you the information.



Mr. Gary Holland
New Products Manager
Bell & Howell Company
Information Systems Service Division
6800 McCormick Road
Chicago, Illinois 60645

Name____________________

Company____________________

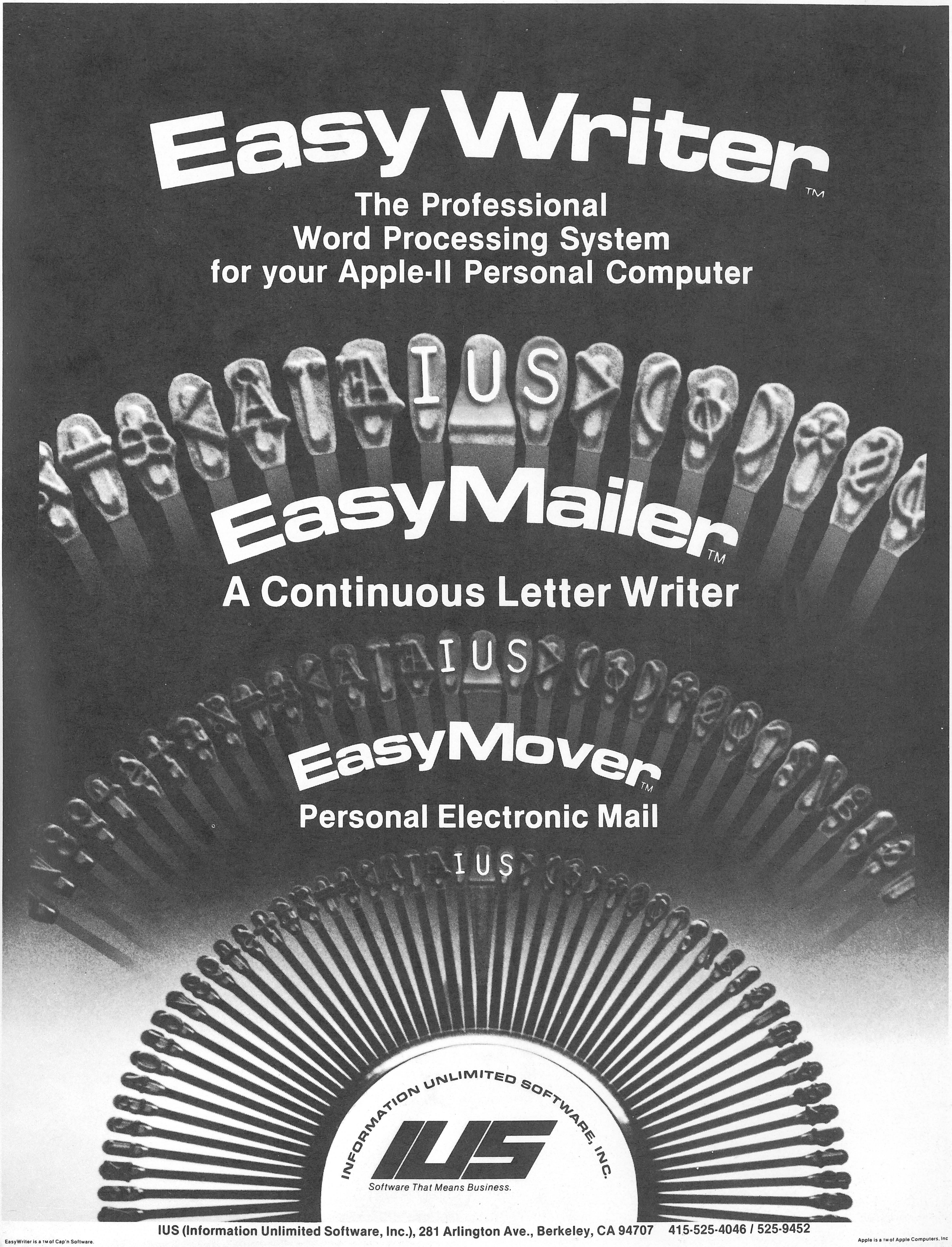
Title____________________

Address____________________

City__________ State_____ Zip_____

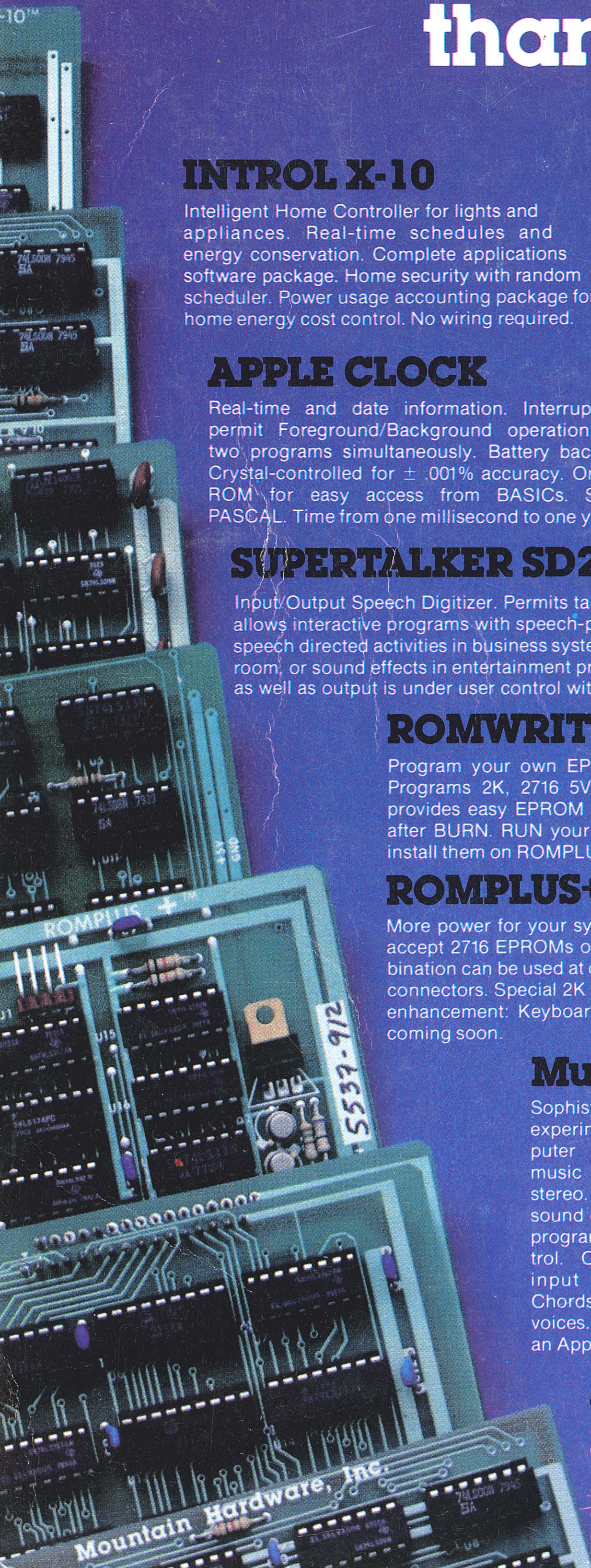
How many Apple Computers?____ Telephone (   )________

BELL & HOWELL


EasyWriter™
The Professional
Word Processing System
for your Apple-II Personal Computer
IUS
EasyMailer™
A Continuous Letter Writer
IUS
EasyMover™
Personal Electronic Mail
IUS
INFORMATION UNLIMITED SOFTWARE, INC.
IUS
Software That Means Business.

Mountain Hardware makes more peripherals for the Apple Computer than Anybody.

and . . .
a place to put them



INTROL X-10

Intelligent Home Controller for lights and appliances. Real-time schedules and energy conservation. Complete applications software package. Home security with random scheduler. Power usage accounting package for home energy cost control. No wiring required.

APPLE CLOCK

Real-time and date information. Interrupts permit Foreground/Background operation of two programs simultaneously. Battery back-up. Crystal-controlled for ± .001% accuracy. Onboard ROM for easy access from BASICs. Supports PASCAL. Time from one millisecond to one year.

SUPERTALKER SD200

Input/Output Speech Digitizer. Permits talking programs. I/O capability allows interactive programs with speech-prompted inputs. Use output for speech directed activities in business systems, announcements in a control-room, or sound effects in entertainment programs. Easy to use because input as well as output is under user control with special software operating system.

ROMWRITER

Program your own EPROMs. Create your own firmware. Programs 2K, 2716 5V EPROMs. Disk software package provides easy EPROM programming. EPROMs are verified after BURN. RUN your programs from on-board socket or install them on ROMPLUS+.

ROMPLUS+

More power for your system through firmware. Six sockets accept 2716 EPROMs or ROM equivalents. Six or any combination can be used at once. Scratch-pad RAM and two TTL connectors. Special 2K ROMs available for powerful system enhancement: Keyboard Filter ROM—COPYROM—Others coming soon.

MusicSystem

Sophistication previously available only on experimental mini and mainframe computer synthesizers. Digital instrumental music synthesizer system. 16 voices in stereo. Instrument definitions simulate the sound of real instruments—and more. Fully programmable waveforms. Envelope Control. Composition system—sheet music input using standard music notation. Chords and multi-part scoring up to 16 voices. A true instrument that anyone with an Apple can play.

A/D+D/A

16 channels analog to digital input. 16 channels digital to analog output. Eight bit resolution. Super-fast 8μ sec. conversion time. Monitor and output to the real world. All on one card.

EXPANSION CHASSIS

By popular demand! Eight more slots for your Apple. Attractive sturdy enclosure. Its own heavy duty power supply. Easy to use. Address cards in Expansion Chassis the same way as in your Apple. Only one additional command to specify in Apple or in Expansion Chassis. Compatible with all Apple peripherals.

MOUNTAIN HARDWARE has the most comprehensive line of Apple peripherals available. Anywhere. From anybody. We know the Apple inside and out and are committed to providing the most innovative and unique products to expand and enhance its capabilities and use. After all, we were the first company to make an Apple peripheral—except Apple Computer.

The message is simple. If you have an Apple, you need to know MOUNTAIN HARDWARE.

Available at Apple Dealers worldwide.

Mountain Hardware
Leadership in Computer Peripherals
A Division of Mountain Computer, Inc.
300 Harvey West Blvd.
Santa Cruz, CA 95060 (408) 429-8600

MORE PERIPHERALS? Send me information.

NAME ____________________

ADDRESS ____________________

CITY __________ STATE __________ ZIP ______